Hypothesis Aware Science Autonomy

P.Michael Furlong

November 4, 2017

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Nearly Divine Personage, Chair
Someone else
Yet another person
Someone from a strange and faraway land

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2017 P.Michael Furlong
Keywords: Science Autonomy, Exploration-Exploitation, Active Learning, Productivity of Information
Abstract

Exploration by autonomous agents requires both opportunistic and deliberative decision-making. Opportunistic decision making chooses to investigate events of phenomena which could not have been anticipated from prior knowledge but which may yield valuable scientific data. Deliberative decision making plans sequences of observations based on prior knowledge that increase collected scientific information. The approaches to deliberative and opportunistic science autonomy presented in this thesis have in common the use of hypotheses to direct robot behaviour. The thesis of this work is that the performance of the robot will improve when the robot considers the hypothesis linking the input observations \( X \) to the resultant observations \( Z \).

The work addresses three questions, each of which is answered by an algorithm. Those questions are:

1. When should an agent stop or restart sampling objects of particular, discrete classes?
2. How can an agent be confident that a value it is monitoring has changed?
3. How can an agent select samples that best falsify competing hypotheses?

Effective answers to these questions help shepherd resources that are scarce in state-of-the-art autonomous exploration missions.

The first algorithm, information foraging, addresses when to stop and start sampling classes of objects. Information foraging moves science autonomy algorithms beyond simply responding to matched templates or anomalous data. The work integrates the hypothesis about distributions underlying discrete classes into the decision making process.

The second algorithm addresses how to effectively conduct prospecting without relying on either arbitrary thresholds or responding to anomalies. The change detection algorithm developed in this thesis encodes a level of confidence that a change has occurred, considering the data that has been collected by the agent.

The third algorithm moves autonomous science into the realm of falsification, the gold standard of scientific enquiry, and chooses the samples necessary to prove one hypothesis (or family of hypotheses) more credible than others. Where prior approaches to trajectory planning for science considered the diversity data that are being collected this work considers they hypotheses that are being tested, which will give artificial scientists considerable more autonomy than they have had to date.

The three algorithms presented in this document will work towards liberating human and robot scientists from wasteful micromanagement and increase the productivity of future science missions.
Acknowledgments
Contents

1 Introduction .......................... 1
   1.1 What is This Thesis About? ........................................ 1
   1.2 Motivation ...................................................... 1
      1.2.1 What is Science Autonomy? ..................................... 1
      1.2.2 Why Automate Science? ........................................ 3
   1.3 Document Structure .............................................. 4

2 Opportunistic Sampling ............. 7
   2.1 Problem Statement .............................. 9
   2.2 Preliminaries .............................................. 9
      2.2.1 Background ............................................... 10
      2.2.2 The Secretary Problem .................................... 10
      2.2.3 Multi-armed bandits ....................................... 10
      2.2.4 Optimal Foraging .......................................... 11
      2.2.5 Science Autonomy .......................................... 12
   2.3 Experiments .............................................. 13
      2.3.1 Method ..................................................... 13
      2.3.2 Algorithms ............................................... 13
      2.3.3 Experiments ............................................... 15
   2.4 Results .............................................. 16
      2.4.1 Experiment 1 - Underlying Distribution .................... 16
   2.5 Conclusion .............................................. 20

3 Prospecting .......................... 25
   3.1 The Mojave Volatiles Prospector Project ..................... 26
   3.2 Related Work .............................................. 27
      3.2.1 Inadequacies of the Secretary Problem Setting .............. 29
   3.3 Algorithms .............................................. 30
      3.3.1 Detecting Changes with a Threshold .......................... 30
      3.3.2 Detecting Changes with a Threshold and Memory ............ 31
      3.3.3 Detecting Changes with the Sequential Probability Ratio Test 32
      3.3.4 The MCMC-Bayesian Approach is Excessively Time Consuming 35
   3.4 Experiments .............................................. 38
### 3.4.1 Experiment 1 - Effect of Magnitude of Change in the Underlying Distribution

3.4.2 Experiment 2 - Effect of Delay in Onset of Change in the Underlying Distribution

3.4.3 Experiment 3 - Performance on Real-World Data

3.4.4 Experiment 4 - Effect on Performance in Realistic Scenario

3.4.5 Performance Metrics

### 3.5 Results

3.5.1 Experiment 1 Results - Effect of Different in Magnitude on Detecting Changes

3.5.2 Experiment 2 Results - Effect of Delay of Change Onset

3.5.3 Experiment 3 Results - Real MVP Data

3.5.4 Experiment 4 - Effect of SPRT on 2D Operations

### 3.6 Extensions

### 3.7 Contributions

### 4 Global Planning for Hypothesis Falsification

4.1 Objective

4.1.1 Problem Statement

4.1.2 Hypothesis

4.2 Preliminaries

4.2.1 Model Selection

4.3 Experiment

4.3.1 Condition A: Noisy Sensor

4.3.2 Condition B: Sensor with Fault

4.3.3 Map Generation

4.3.4 Algorithms

4.4 Results

4.5 Extensions

4.6 Contribution

4.7 Introduction

4.8 Background

4.9 Method

4.9.1 Belief in Hypotheses

4.9.2 Site Selection Algorithm

4.9.3 Control Algorithm - Spatial Sampling

4.9.4 Proposed Algorithm - Hypothesis Falsification

4.9.5 Map Generation

4.9.6 Experiments

4.9.7 Analysis of Results

4.10 Results

4.11 Conclusions

4.12 Deployment in Chile
5 Simultaneous Hypothesis Generation and Falsification 85
5.1 Problem Statement 85
  5.1.1 Minimum Description Length and Kolmogorov Complexity 87
  5.1.2 Interpretability of a Hypothesis 88
5.2 Method 88
  5.2.1 Generating New Models 88
  5.2.2 When to Produce a New Model 89
5.3 Experiment 90
  5.3.1 Control Algorithm 90
  5.3.2 Testing Data 90

Appendices 91
A List of Terms 93
B Where’s Thomas Kuhn? 95
Bibliography 97
List of Figures

1.1 Science Autonomy: It does what it says on the tin. ........................................... 2
1.2 The cycle of scientific process. Hypotheses imply experiments that should be conducted. Experiments result in observations. Observations update or produce new hypotheses. ................................................................. 2
2.1 A lawnmower, or boustrophedon, trajectory ......................................................... 8
2.2 Foraging vs Greedy and Uniform Algorithm ......................................................... 17
2.3 Foraging vs Greedy and Uniform Algorithms for Experiment 2 ............................. 19
2.4 The Foraging Algorithm is Statistically Significantly Better than the control algorithms. ................................................................. 20
2.5 Detecting distribution change yields statistically significant and profoundly better performance. ................................................................. 21
3.1 KRex2 in the Mojave desert. On the rear of the vehicle is visible the stainless steel and cadmium cylinders of the Neutron Spectrometer. ................................. 26
3.2 Here we can see the average execution time for the different algorithms as a function of the number of observations collected. Error bars represent one 95% confidence interval. The SPRT algorithm is on the order of 100x faster than the Bayesian algorithm. ................................................................. 36
3.3 Considering the performance of the two algorithms over 10 trials. The performance is measured as the error between the estimated time of change in the underlying distribution and the true change time. We consider their performance before the change in the underlying distribution (t=25), shortly after the change in the underlying distribution (t=52 and t=60) and at the end of the transect (t=100). 37
3.4 Here we see a simulated map with the planned lawnmower path superimposed on it. The rate of the Poisson distribution that samples were drawn from was determined by the location of the vehicle in the map. ................................................. 40
3.5 False Negative Rates for SPRT algorithms vs Threshold ........................................ 41
3.6 False Positive Rates for SPRT algorithms are independent of background rate. 43
3.7 Mean Error Rate for SPRT algorithms vs Threshold ................................................. 45
3.8 False Positive Rate for SPRT algorithms vs Threshold ................................................. 47
3.9 Mean Error Rate for SPRT algorithms vs Threshold ................................................. 48
3.10 Mean Error for SPRT algorithms vs Threshold ......................................................... 49
3.11 Algorithm Performance on MVP Data from 2014-10-17 ......................................... 51
3.12 Algorithm Performance on MVP Data from 2014-10-18 ......................................... 51
3.13 Algorithm Performance on MVP Data from 2014-10-19 ........................................ 52
3.14 Algorithm Performance on MVP Data from 2014-10-20 ...................................... 52
3.15 Algorithm Performance on MVP Data from 2014-10-21 ...................................... 53
3.16 Algorithm Performance on MVP Data from 2014-10-22 ...................................... 53
3.17 Algorithm Performance on MVP Data from 2014-10-23 ...................................... 54
3.18 Algorithm Performance on MVP Data from 2014-10-24 ...................................... 54
3.19 Algorithm Performance on MVP Data from 2014-10-25 ...................................... 55

4.1 A geological map of Jupiter’s Moon, Io. The remote data produced from observing Io has been interpreted as a map of terrain classifications. However, observations from the ground would be required to confirm the predictions of the map creators. Image courtesy wikipedia.org .................................................. 71
4.2 Examples of a map and the hypotheses generated that anticipate the map ............. 79
4.3 Experiment 1 results. Both algorithms correctly identify $H^*$, but the proposed algorithm was more confident than the control algorithm. The control algorithm distributed belief in the hypotheses more evenly. Neither algorithm was confident that there was no correct hypothesis under consideration. Error bars one standard error, $N = 10$. ................................................................. 81
4.4 Experiment 2 results. Our proposed algorithm again outperforms the control algorithm with statistical significance. Error bars are one standard error, $N = 10$. 82
4.5 Experiment 3 results. Our algorithm correctly selects $H^*$, but the control validly selects $H_0$, no good hypotheses. Error bars are one standard error, $N = 10$. . . 83
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Parameters for Underlying Distributions in Experiment 1</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Parameters for Underlying Distributions in Experiment 2</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>Aggregate Effect Size of Foraging Algorithm</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Effect Size for Foraging vs 1st Control Algorithm</td>
<td>17</td>
</tr>
<tr>
<td>2.5</td>
<td>Effect Size for Foraging vs 2nd Control Algorithm</td>
<td>17</td>
</tr>
<tr>
<td>2.6</td>
<td>Aggregate Effect Size for Experiment 2</td>
<td>18</td>
</tr>
<tr>
<td>2.7</td>
<td>Effect Size of Foraging vs 1st Control Algorithm</td>
<td>18</td>
</tr>
<tr>
<td>2.8</td>
<td>Effect Size for Foraging vs 2nd control algorithm</td>
<td>18</td>
</tr>
<tr>
<td>3.1</td>
<td>In all cases there is a statistically significant reduction in the error in detecting the change point. Before the time change has occurred the SPRT algorithm is better at detecting there has been no change in the underlying distribution than the MCMC approach. However for all the cases after the change has occurred there is an error of 1 time step, which is a negligible error. We can say that the SPRT and MCMC algorithms have indistinguishable performance after a change has occurred.</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2.</td>
<td>42</td>
</tr>
<tr>
<td>3.3</td>
<td>The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4.</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8.</td>
<td>43</td>
</tr>
<tr>
<td>3.5</td>
<td>The size of the effect on the false positive rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2.</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4.</td>
<td>44</td>
</tr>
<tr>
<td>3.7</td>
<td>The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8.</td>
<td>45</td>
</tr>
<tr>
<td>3.8</td>
<td>The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2.</td>
<td>46</td>
</tr>
<tr>
<td>3.9</td>
<td>The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4.</td>
<td>46</td>
</tr>
<tr>
<td>3.10</td>
<td>The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8.</td>
<td>47</td>
</tr>
</tbody>
</table>
3.11 The average reduction in false positive rate due to using the SPRT algorithm vs the Memory Threshold algorithm. ................................................................. 48
3.12 The reduction in FN rate from the Memory Threshold to the SPRT algorithm. . 49
3.13 In this table we report only the results of the Memory Threshold Algorithm and the SPRT algorithm. While SPRT reports more changes than actually occurred, they are often co-located. We also see that SPRT notices changes that are not observed by the Memory Threshold algorithm. ............................................. 55
3.14 The average performance of the three different algorithms on each map. The column $E[\mu_1 - \mu_2]$ gives the average difference between the first and second place algorithm. The 95% high density interval is based around the difference. If that interval does not contain 0 then we can say with 95% probability that the means are different. If they are different we report the effect size. Effect sizes over 0.8 are considered large. Effect size is measured with Cohen’s d. ............ 57
3.15 Here we count the number of times each algorithm scored each rank on the 30 maps for the number of high-value observations collected. The SPRT algorithm is the clear winner in this case. ................................................................. 58
3.16 The average number of peaks captured by the three different algorithms on each map. The column $E[\mu_1 - \mu_2]$ gives the average difference between the first and second place algorithm. The 95% high density interval is based around the difference. If that interval does not contain 0 then we can say with 95% probability that the means are different. If they are different we report the effect size. Effect sizes over 0.8 are considered large. Effect size is measured with Cohen’s d. ....... 59
3.17 Here we count the number of times each algorithm scored each rank on the 30 maps for the number of local maxima observed. Again, the SPRT algorithm is the clear winner in this case. ................................................................. 60
4.1 The probability of the different algorithms selecting the best hypothesis, averaged over all resource budgets ................................................................. 70
4.2 This table gives the similarity between the true maps $M_i$ and the hypotheses proposed by the simulated scientists $H_j$. ............................................. 80
4.3 The hypotheses tested in the experiments. Column $H^*$ gives the best hypothesis for each experiment. Subscripts correspond to Table 4.2. ......................... 80
A.1 Interpretation of different effect sizes ...................................................... 94
## List of Algorithms

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Foraging Sampling Strategy</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>2.2 Foraging Change Detection</td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>3.1 Threshold Change Detection</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>3.2 Memory Threshold Change Detection</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>3.3 SPRT Change Detection</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>4.1 Select random sample points</td>
<td>Spatial Mutual Information sampling. This algorithm uses a measure of mutual information like the one used in Thompson (2008). It depends only on previously observed locations to find the most informative point.</td>
<td>67</td>
</tr>
<tr>
<td>4.2 Select random sample points</td>
<td></td>
<td>68</td>
</tr>
<tr>
<td>4.3 Select random sample points</td>
<td></td>
<td>68</td>
</tr>
<tr>
<td>4.4 Select random sample points</td>
<td></td>
<td>69</td>
</tr>
<tr>
<td>4.5 Site Selection Algorithm</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>4.6 Spatial Mutual Information sampling. Hypothesis Falsification sampling. We use Eq. 4.4 to increase the value of spatially diverse points.</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>4.7 Hypothesis Falsification sampling.</td>
<td></td>
<td>79</td>
</tr>
<tr>
<td>4.8 Map Generation Algorithm</td>
<td></td>
<td>79</td>
</tr>
<tr>
<td>5.1 Hypothesis Generation and Search Algorithm</td>
<td></td>
<td>90</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 What is This Thesis About?

The work presented in this thesis looks at designing algorithms for science autonomy, with particular consideration for operations in the field. They are all underpinned by the notion that the performance of robots will improve when robots consider the hypothesis linking the input observations $X$ to the observations, $Z$, that result from experiments.

We will consider this hypothesis in three settings: Opportunistic science, prospecting, and informative trajectory planning. The experiments in this thesis mirror settings from analog and actual science missions. We extend the state of the art in decision making algorithms by considering the missions constraints from field work. We extend the state of the art in science autonomy by explicitly considering the hypotheses under investigation.

Before we go into more detail on the work itself, we motivate the research into science autonomy.

1.2 Motivation

1.2.1 What is Science Autonomy?

What is science autonomy? It is the automation of the process of scientific inquiry. This includes generating hypothesis from collected data, designing experiments to falsify those models, and selecting the experiments that are most productive.

When conducting scientific inquiry there are three artifacts that are of interest: Hypotheses, experiments, and observations.

Hypotheses are models of some phenomena. Hypotheses must be falsifiably and make some prediction about the world, either implicitly or explicitly. For example, a model of the spatial distribution of geologic material implies that at specific locations there will be a specific mixture of material. In this document the words “hypothesis” and “model” are used interchangeably.

Experiments are actions that the scientist can take that will result in observations. It is important to ensure that these experiments are informative with respect to the hypotheses being
tested – that is to say they falsify one or more hypotheses. It is also important to ensure that experiments do not reproduce each other unnecessarily. In this document the word “experiment” and “action” are used interchangeably.

**Observations** are the results of experiments. They are the data that are used to construct or update hypotheses. “Observations” and “data” and “samples” are all phrases that will be used interchangeably in this document.
People disagree about which is the primary artefacts in the scientific process. Karl Popper would have us believe that hypotheses are the currency of scientists. N.R. Hanson would have us believe it is data that is the starting point of science. Ian Hacker suggests that experiments are the coin of the realm. I would suggest that it is a constant iterative loop that one can join in at any point and that all three are perfectly valid starting points.

To produce these artifacts scientists need to engage in different activities. The motion from hypotheses to experiments is called **Experiment Design**. The motion from experiment to observations is called **Experiment Selection**. The distinction between experiment design and experiment selection is this: Experiment design asks ”of all the possible actions, which ones are most informative?”. Experiment Selection asks of ”Of all the most informative actions, and given some budget, which are the next best K actions to take?”.

Finally the motion from observations to hypotheses is called **Hypothesis Generation**. Simple structured machine learning – i.e. linear regression – is a form of hypothesis generation. It chooses the hypothesis (weight settings) most likely given the constraints of the the data to be explained and the family of hypothesis (linear functions) specified by the experimenter. However choosing the structure of the hypothesis is part of the hypothesis generation and so model selection must also be considered part of the hypothesis generation. By iteratively engaging in these activities we produce and test hypotheses and thereby better understand the world.

Science autonomy, then, is research into algorithms and systems that do one or more of experiment design, experiment selection, and hypothesis generation. More importantly they are systems that may be embedded in complex, unstructured environments.

### 1.2.2 Why Automate Science?

There are three reasons to automate the scientific process. First, because future missions will demand it. Second, because human agents demand it. Third, because it poses a truly unique computational problem.

**Future Missions Require Science Autonomy**

NASA, amongst other spacefaring science agencies, will require autonomous science simply because the strain on productivity that communications latency places on missions. The further afield a robot explorer travels the lower bandwidth it will have available to communicate back to Earther, and the greater the latency will become. Including humans in the decision making loop will require designing robots that can survive the decision making cycle, or that scientist will have to make less nuanced decisions, potentially leading to less effective mission. Further, with profound delays in communications the likelihood of serendipitous discovery will drop precipitously.

An alternative consideration is that science autonomy frees science missions from the concerns of communications bandwidth. If the robot can be treated as a trusted proxy for human scientists then we can send them to locations that would previously be considered unfeasible or wasteful.

The algorithms developed for space missions of course have applications here on Earth. Prospecting for resources is an important task whether it is in support of colonies on the Moon.
Humans Require Science Autonomy

The scientific method, if it exists, is a systematic process of generating quantifiable knowledge. However, humans, who are riddled with cognitive biases are a problematic component of this process. Confirmation bias, affirming the consequent, availability bias, and clustering illusions to name a few, interfere with human ability to conduct scientific inquiry.

The literature on the design of experiments, starting at least as early as Kristin Smith Smith (1918), is an attempt toward a mathematically rigorous approach to asking questions that ameliorates human biases. Science autonomy is another step along this path.

Science Autonomy is an Interesting Problem

Science autonomy is the mathematisation of the philosophy of science. Scientific inquiry presents an interesting set of behaviours that are designed to be rational and logical, but that need to be produced in a reasonable period of time given finite computing resources.

Science autonomy necessarily fuses reason and deduction with symbols grounded in the real world and in observable quantities. However, unlike other branches of robotics, it is strictly necessary that science autonomy return information that is human interpretable.

If we contrast science autonomy with other machine learning approaches, there is not much room to return, for example, a hyperparametric function approximation. In turn, this leads to interesting questions about what it means to understand the universe and what are the limits of human understanding. While this thesis does not address these questions, they do motivate the field in general.

1.3 Document Structure

The first component of the work focuses on making decision about whether or not to sample discrete objects. It extends the notion of multi-armed bandits, even infinitely-armed bandits, to include the notion that one may not always have random access to the objects that a learning agent wishes to sample. Here the agent has no knowledge about what objects it will be able to sample, should it give up on the objects it currently has access to.

The second component of the work considers sampling agent who are using an instrument that is always on and reporting values. Here the challenge is to determine when the underlying distribution has changed, and thus warranting the execution of Area of Interest Maneuovres - moving the robot to collect more data to refine the map. This component actively considers the relative likelihood of the hypothesis that there has been a change in the underlying distribution vs the hypothesis that there has not been a change in the underlying distribution.

The third component of the work generates global trajectories that are designed to disambiguate between a series of competing hypotheses. Here the objective is to not just collect data that spans an input space, but to actively falsify the currently most likely hypothesis. This component represents the first work in science autonomy that considers multiple families of hypotheses, and directs the robot to determine which one is most accurate.
These three components address different parts necessary for an autonomous science mission. The novel contributions herein will improve the capability of autonomous field scientists when it comes to planning and choosing between different sampling opportunities.
Chapter 2

Opportunistic Sampling

The single biggest flaw in applying experiment designs to fielded exploration problems is the belief that one gets to choose what to do. While an agent does have some agency in this matter, one does not get to choose what one actually encounters in the field. What is important is having a principled way for moving on.

Multi-armed bandits are a technique developed for selecting among discrete experiments (although they have been extended to real-valued space by methods like the GP-UCP algorithm). However these algorithms make the assumption that any point in the query space is accessible at any given time – that they have random access to the input space.

They are trying to learn the value function of the different arms:

\[ V[a] \forall a \in A \text{ where } |A| \in \mathbb{N} \]  

1. Known sampling budget.
2. Unknown number of classes.
3. Role of productivity.
4. backtracking

Robots are our scientific surrogates as we peer into the depths of the ocean or into our solar neighbourhood. High-latency and low-bandwidth communication to these regions limits situational awareness and reaction times of the scientists controlling such robots. Therefore it is vital to increase the ability of robotic explorers to independently make in-mission decisions. A common exploration activity is remote sensing, in which a robot is tasked with collecting sensor data by sampling the environment at various locations. Many specialized sensors employed for activities such as biological collection and spectral mapping require long, energy-intensive sampling durations or the activation of single-use collection canisters. Constraints on mission length and payload capacity, coupled with limited remote operator awareness, necessitate some autonomy in sampling location selection for mission productivity and success.

Currently fielded robots either depend highly on operators for objectives or plan with considerable global knowledge. Operating in such conditions constrains them to rely on either remote human decision-making (requiring often impractical levels of situational awareness) or significant amounts of prior scouting, obviating the need to send a robotic agent. These limitations
are mirrored in existing literature, which fails to provide principled reasoning about what to investigate *in situ* without such reliances.

![Figure 2.1](image.png)

**Figure 2.1:** A cartoon of a path explored by a rover. The images represent different classes of desert pavements that may encountered by a rover as it follows a pre-determined path.

This paper proposes an algorithm that addresses one example of such missions, in which objects or areas in the environment lie within some easily sensed class, and each class possesses some underlying data distribution (e.g. microbial colonization) that can only be sensed with expensive specialized sensors. The overall goal is to estimate the underlying distribution of each class with maximal accuracy.
For general applicability no global information, such as prior maps of sampling opportunities, is available. Sensing opportunities are assumed to arise nondeterministically (e.g. from classes present along a pre-determined trajectory or as currents draw objects past the robot), and the robot cannot return to objects it did not sample. Thus, the problem can be thought of as a stream of sensing opportunities providing varying reward (information about underlying class distributions), each requiring a decision to sample or move on.

The proposed algorithm draws on techniques from optimal foraging theory and sequential experiment selection. Its use is motivated by observations of human and animal behavior, exemplified by geologists making decisions about investigating local phenomena without prior access to detailed maps, who are able to effectively choose between sampling materials in front of them or moving on to potentially more profitable sampling locations. These decisions may not be globally optimal, but they demonstrate an ability lacking in current exploration robots: to make decisions to stop and engage with the environment or to continue traveling in the hope of finding more informative sampling locations.

The algorithm is then extended to address another common situation in scientific exploration, namely environmental variability. As the robot traverses the environment, it may easily pass between larger regions in which the underlying distribution of classes differs. In the Atacama desert 100% of photosynthesis-promoting translucent rocks are colonized by microbes in semi-arid regions, but less than 50% of such rocks in semi-arid regions, and less than 1% in the hyperarid core ([Warren-Rhodes et al., 2007][2006]). Detecting and reacting to such changes is relevant both for scientific interest and so sampling decisions will not be based on historically observed but now inaccurate class information. An extension is proposed incorporating an additional statistical test to detect a change in class distribution to notify operators, separate data segments, and reset empirical history that might otherwise misinform upcoming sampling decisions.

The remainder of this document begins with a brief survey of the relevant literature. Next, a detailed comparison of the proposed foraging algorithm and one based upon existing principles from the design of experiments literature. Finally, discussion of experimental results from a simulated exploration scenario indicates that under limitations on sample collection and overall mission time, the foraging algorithm presents a significant improvement for a realistic range of sampling costs.

### 2.1 Problem Statement

The agent does not know how many opportunities to sample it will have. Has to choose between giving up here and now in the hopes of something better.

### 2.2 Preliminaries

This is different from other bandit approaches because it assumes that you can choose to sample anything. This is a problem across all forms of bandit algorithms. Here I take what was learned from patchy foraging and we apply it to exploration, where one does not always know what one will be encountering, and one does not always get to choose what option one will be given next.
2.2.1 Background

Automating experiment design and selection is not without precedent. Kristine Smith started the field of optimal experiment design in 1918. (Smith, 1918) Recently robots have been employed to conduct scientific exploration autonomously. (Wagner et al., 2001; King et al., 2004) Current robot scientists’ reliance on global information prevents them from operating in truly unknown environments. Additionally, previous approaches in sequential decision making from statistics do not necessarily reflect the settings that autonomous robots encounter in the real world.

2.2.2 The Secretary Problem

The secretary problem asks a decision maker to select the best candidate from sequentially presented candidates where it is not possible to return to rejected candidates. In the original setting, there is only one position for the candidate to fill, (Ferguson, 1989) and the optimal strategy is to reject the first \( N \) candidates and then accept the first candidate who is ranked better than any of the previously seen candidates. Further, the decision maker was able to objectively score the candidates without cost. In our setting, we do not know the value until sampling, and sampling a class incurs a sampling cost.

There have been many variations on this problem including selecting multiple candidates (Vanderbei, 1980), or when the total number of candidates is random (Presman and Sonin, 1972), for more variations (Ferguson, 1989) is an excellent source. What distinguishes the secretary problem from the science autonomy problem we propose is that we do not know the value of a candidate, or class, when we encounter it. Additionally repeatedly sampling the same class decreases the value to the decision maker.

2.2.3 Multi-armed bandits

Sequential experiment selection, a type of active learning, is addressed in the multi-armed bandit (MAB) literature. This was introduced by Robbins (1952) as a means of sequentially selecting which experiments to conduct with a limited budget. In Robbins’ work, selecting experiments is modelled on determining the payouts of one-armed bandit machines – each machine representing a different experiment. The player has a fixed sampling budget and has to sequentially choose which machine to play, trading off exploiting expected rewards from well-studied arms against exploring different arms, learning more accurately the payouts of those arms.

Lai and Robbins (1985) use a value function in which uncertainty in arm rewards makes an arm more interesting. Recently decision rules like Thompson sampling (Thompson, 1933) and Bayesian Optimal Control (Ortega and Braun, 2010) have gained popularity. Other techniques
addressing the exploration/exploitation problem use uncertainty as a reward metric. (Burnetas and Katehakis 1997; Auer 2003; Balcan et al. 2006) In our setting, because the agent only needs to learn the distribution and not use it for anything, uncertainty is the only necessary reward.

Balcan (Balcan et al. 2006) presents a method for learning classifiers by requesting samples from the input space with the greatest classification error. Classification error and uncertainty in function value are fungible quantities in this case. An analogy can be drawn between the classifiers used in (Balcan et al., 2006) and the bandit arms used by Auer and Ortner (Auer and Ortner, 2010).

Several factors distinguish the MAB setting from the problem explored in this paper. In MAB, the agent has access to any arm (analogous to a class in our setting) it chooses at any given time. The agent in our setting does not get to choose which of the classes it can investigate. Any previously seen classes are no longer available, and new classes arrive per a random model. Additionally, the standard MAB setting does not have switching costs, although there are some formulations which do include such costs. (Jun 2004) In our setting, there is a cost incurred with every choice to continue exploring.

2.2.4 Optimal Foraging

Foraging is the problem encountered by animals seeking to maximize energy intake when operating in unknown environments. The central question of the problem is whether it is more valuable to continue extracting resources from the current location or to seek out resources in new locations. Charnov (1976) introduced a technique for dealing with “patchy” environments, in which there are distinct regions that contain different classes of resources. The forager can extract value from these patches, with diminishing returns (modeling resources consumed), or choose to continue to wander randomly in the hopes of encountering more valuable locations.

The optimal time to leave a patch, according to Charnov’s Marginal Value Theorem, is when the expected return from continuing to sample a patch is less than the expected return from searching the environment. In this formulation, the expected return from both the current patch and the environment are offset by the cost of extracting resources in this patch and the energy spent seeking a new patch.

Pirolli and Card (1999) studied researchers attempting to acquire information. They modelled the rate of information gain and had their agent decide to leave a patch when the rate of information gain was lower than that of the environment. What differentiates their setting from ours is that their decision maker can choose which patch to sample, yet our exploring agent cannot.

Kolling et al. (2012) studied humans engaged in a gambling task in which players have to consider the option they have before them and the opportunities the environment provides. Subjects were repeatedly presented with a choice of playing a gambling game or being randomly presented with a different game. Each game was a Bernoulli trial with some unknown probability of success. Kolling et al. identify possible neural substrates for foraging decisions in humans. The behaviour was near optimal, with some skewing of probabilities near 0 or 1.
2.2.5 Science Autonomy

Thompson and Wettergreen (2008) maximize diversity of collected samples with mutual information sampling. This approach ensures diversity in the collected sample set, an act that reduces uncertainty in the input space of a function. However, this approach does not consider nondeterministic results of sampling.

Bender et al. (Bender et al., 2013) make a modification to that work, instead using Gaussian processes to identified hypothesized distributions of life across the sea floor to direct exploratory actions. The prior maps were generated by vessels passing over the sea floor prior to the robot’s exploration mission, not unlike Thompson and Wettergreen’s use of satellite imagery. The advance of Bender et al. is the use of in situ measurements to update the Gaussian process being learned. Their rover can thus be said to be generating and testing hypotheses. However, they are severely limited by a budgeting size of six “gulpers” – devices for collecting seawater samples.

Ferri et al. (2010) address prospecting where an autonomous underwater vehicle (AUV) follows a predefined track and needs to decide when to deviate to sample anomalies. The AUV examines anomalies by searching in a spiral pattern, collecting data and characterizing the environment in that location. The AUV’s sampling capacity is limited only by time. The decision to sample is based on a fixed threshold. While this may accurately encode subject matter experts’ beliefs on what is interesting, it is fragile in the face of variable and unknown environments the AUV encounters. This exploration problem is an ideal application of the algorithm proposed in this paper.

Likewise, Girdhar et al. (2013b) present an approach to autonomous exploration wherein a robot investigates a scene when it encounters unexpected phenomena. Specifically, they use topic models to describe scenes and sample when scenes do not fit into the topic models they have constructed. In these works, the vehicle has no limit on its sampling capacity and is always collecting data. By slowing the vehicle down, more samples are collected in anomalous scenes. In this fashion this is very similar to work by Thompson et al. (2013)

Girdhar et al. (2014) build upon their anomaly detection techniques to develop a path planning method to maximize information gain of paths. In that respect, it belongs with the family of curiosity-driven algorithms pioneered by Sun et al. (2011) Fundamental to these approaches is that explorers should spend time investigating regions of the world (or hypothesis space) where learned models are the least certain.

Previous work by the primary author with optimal foraging for science autonomy has considered robots with sampling budgets limited by a number of containers (removed for anonymity) and assumed knowledge of the number of sampling opportunities that would occur. While the limited sampling budget is realistic, foreknowledge of the transect is not. This paper improves upon the prior work by using productivity to reason about sampling choices and gives a constraint of time instead of an unknowable number of sampling opportunities.

As explained, real robots may not be able to predict the rewards they will earn from their actions and have to deal with unreliable arrival rates for sampling opportunities. These are concerns that are not modelled in typical sequential experiment selection algorithms. This motivates the problem setting used in this paper, described in detail in the following section.
2.3 Experiments

2.3.1 Method

A simulated scenario is considered where a rover explores a path set for it by remote scientists. The exploration budget is 100 units of arbitrary time. While following this path the agent is repeatedly presented with one of \( K \) possible materials. The agent does not know how many different types of materials it may encounter during its travels. At every presentation the agent has a choice of sampling that material, represented by taking action \( \xi_k \in \Xi \) and making an observation \( Z \), or continuing along the path in the hopes of finding a more interesting sampling opportunity. The role of the agent is to determine \( P(Z|\xi_k) = \theta_k (1 - \theta_k)^{1-z} \) \( \forall \xi_k \in \Xi \).

The experimental setup is a variation on Charnov’s patchy foraging (see Section 2.2.4). In this case we assume a patch is exhausted by taking one sample. If the agent choses to continue searching it will be presented with a new material, drawn with probability \( P(\xi_k) \). When patches are classes of random variables and the reward is information gained about the underlying distribution, the reward is less at every encounter, unless environmental conditions change the underlying distribution.

In this paper, we choose to model the different classes of materials as Bernoulli random variables, representing the common scientific exploration scenario of detecting the presence of a phenomenon of interest such as whether or not a material is colonized by microbes. We place a Beta prior on the parameter \( \theta_k \sim Beta(\alpha_k, \beta_k) \) that determines the probability that a class of material is colonized, yielding a belief post-observation that \( \mathbb{E}[\theta_k] = \frac{\alpha_k}{\alpha_k + \beta_k} \) where \( \alpha_k \) is the number of times material \( k \) was observed being colonized (“success”), and \( \beta_k \) is the number of times material \( k \) was observed as not being colonized (“failure”).

We anticipate that the agent will encounter a number \( K = |\Xi| \leq \infty \) classes of random variables while exploring. In these experiments we set \( K = 3 \). However the agent is never informed of how many classes of materials exist in the environment.

2.3.2 Algorithms

Four algorithms for sampling decision-making are evaluated in these experiments. Three of these algorithms estimate the reward of action \( \xi_k \) by using Lindley’s (1956) value of an experiment, given in Equation [2.2]. This reward represents the expected information gain over all possible observations that may result from choosing to take sampling action \( \xi_k \).

\[
R(\xi_k) = H(\theta_k|z_{k,1:t-1}, \xi_k) - \mathbb{E}_Z [H(\theta_k|z_{k,1:t}, \xi_k)],
\]

(2.2)

where \( z_{k,1:t} \) refers to the \( t \) observations that were collected for random variable \( \xi_k \).

The first algorithm, control 1, will only choose to sample \( \xi_k \) if it is has the highest reward compared to any other \( \xi_j, j \neq k \). This algorithm does not take into account the cost of moving to finding the next \( \xi_k \), nor the rate at which they arrive. This algorithm corresponds to the simple greedy strategy of maximizing immediate reward.

The second algorithm, control 2, will choose to sample \( \xi_k \) if any other random variable, \( \xi_j, j \neq k \), has been sampled more than \( \xi_k \). Like control 1, this algorithm does not take into account the cost of traverse nor the cost of taking a sampling action. This algorithm attempts to
distribute samples uniformly across all classes and provides valuable comparison as it has been previously shown to be a robustly successful strategy. (Furlong and Wettergreen, 2014)

The third algorithm, foraging (Algorithm 2.1), chooses to sample if the expected rate of reward of $\xi_k$ is greater than or equal to the expected reward from continuing to explore the environment, that is whether greater productivity is to be had from loitering or from continuing on. This captures traversal and sampling costs ($J$ in Algorithm 2.1). We place a Dirichlet prior on the occurrence of these random variables, estimating the probability of encountering class $\xi_k$ as $\hat{P}(\xi_k) = n_k / \sum_{j=0}^{\mid K \mid} n_j$, where $n_k$ is the number of times $\xi_k$ has been encountered. The distribution $\hat{P}(\xi_k)$ is used to compute the prod\_continue in Algorithm 2.1.

The fourth algorithm uses the same decision rule as Algorithm 2.1 but after it makes an observation it checks to see if the underlying distribution has changed, as in “DETECT\_CHANGE” in Algorithm 2.2. It detects the change with a likelihood ratio test. It maintains two windows of observations for each $\xi_k$, one which is initially populated with window\_size many observations, the other populated with the window\_size most recent observations. The two windows represent hypotheses about the parameter $\theta_k$. A third window of the sample\_size most recent observations is used as the test population. We employ Wald’s sequential probability ratio test (Wald, 1945) to determine if the observations in the second window represents a different distribution from the first. We select the threshold for detecting a change in the distribution, change\_threshold, as specified in (Wald, 1945). window\_size is arbitrarily set to be 30, and sample\_size to 5. If a distribution change is detected the current world model is cached, and the rover resets its sampling algorithm to an initial state.

Algorithm 2.1 Foraging Sampling Strategy

function INIT\_FORAGE\_SAMPLING

$\Xi \leftarrow \emptyset$

$R(\cdot) \leftarrow \emptyset$

$N \leftarrow \emptyset$

end function

function FORAGE\_SAMPLE($\xi_k$)

if $\xi_k \notin \Xi$ then

$\Xi \leftarrow \Xi \cup \xi_k$

$N_k \leftarrow 0$

return sample

end if

$N_k \leftarrow N_k + 1$

$\text{prod}_\text{sample} \leftarrow R(\xi_k) / J(\text{sample})$

$\text{prod}_\text{continue} \leftarrow \mathbb{E}_\Xi [R(\xi)] / (J(\text{sample}) + J(\text{search}))$

if $\text{prod}_\text{sample} \geq \text{prod}_\text{continue}$ then

return sample

else

return continue

end if

end function
2.3.3 Experiments

We conducted three experiments to demonstrate the effectiveness of our algorithm, varying the underlying distribution of each class, class arrival probability, and introducing a class distribution change during the experiment. The costs of sampling and searching were varied over \{0.1, 0.2, 0.5, 0.75, 1.0, 1.5, 2, 3, 4, 5, 6, 7, 8, 9, 10\} for experiments 1 and 2. In each experiment, for each setting of experiment parameters and costs we ran 50 trials for each algorithm.

Experiment 1 - Underlying Distribution

In the first experiment the arrival probability is fixed with a constant uniform distribution. That is to say the probability that the next random variable to be presented to the agent is \( P(\xi_k) = 1/3 \). In this experiment we vary the underlying distribution of the random variables, \( P(Z|\xi_k) \), which is the probability that the material represented by \( \xi_k \) is colonized.

Table 2.1: Experiment 1 Parameter Settings

| Experiment | \( P(Z|\xi_1) \) | \( P(Z|\xi_2) \) | \( P(Z|\xi_3) \) |
|------------|------------------|------------------|------------------|
| 1.1        | 0.001            | 0.500            | 0.999            |
| 1.2        | 0.001            | 0.300            | 0.001            |
| 1.3        | 0.001            | 0.500            | 0.001            |
| 1.4        | 0.001            | 0.750            | 0.001            |
| 1.5        | 0.001            | 0.999            | 0.001            |

Experiment 2 - Arrival Probability

In the second experiment the probability of the different materials being colonized was held constant while the arrival probability is varied. The probabilities of being colonized are \( P(Z|\xi_k) = \{0.001, 0.500, 0.999\} \). These values were because the expected rewards of the three random variables symmetrically span the range \( \theta_k \in (0, 1) \). While the entropy of \( P(Z|\xi_1) \) and \( P(Z|\xi_2) \) are the same, \( P(Z|\xi_3) \) has the maximum entropy possible for a Bernoulli distribution. This was done to determine if one random variable attracted the attention over the others because of either the expected value or the entropy of the underlying distribution.

Table 2.2: Experiment 2 Parameter Settings

<table>
<thead>
<tr>
<th>Experiment</th>
<th>( P(\xi_1) )</th>
<th>( P(\xi_2) )</th>
<th>( P(\xi_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>0.6</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>2.2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.6</td>
</tr>
<tr>
<td>2.3</td>
<td>0.1</td>
<td>0.3</td>
<td>0.6</td>
</tr>
<tr>
<td>2.4</td>
<td>0.1</td>
<td>0.6</td>
<td>0.3</td>
</tr>
<tr>
<td>2.5</td>
<td>0.3</td>
<td>0.6</td>
<td>0.1</td>
</tr>
<tr>
<td>2.6</td>
<td>0.6</td>
<td>0.1</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Experiment 3 - Distribution Change

In this experiment we compare only the Foraging algorithm against the Foraging Algorithm with change detection. In the final experiment we fix the arrival probabilities \( P(\xi_k) = \{0.4, 0.3, 0.3\} \), and fix the sampling and searching costs at 0.01 and 0.1, respectively. Halfway through this experiment the underlying distribution is changed from \( P(Z|\xi_k) = \{0.001, 0.500, 0.001\} \) to \( P(Z|\xi_k) = \{0.001, 0.001, 0.300\} \). The change occurs when the agent gets halfway along its path. Because the foraging algorithm has a prior belief defined for all sampling outcomes we can use Kullback-Leibler (KL) divergence between true and estimated \( P(Z|\xi_k) \) to measure algorithm performance.

2.4 Results

To determine the success of an algorithm we use Hoeffding’s inequality defined by

\[
P\left(|\theta_k - \hat{\theta}_k| > \gamma\right) \leq 2 \exp\left(-2\gamma^2 n_k\right)
\]

(2.3)

to determine the error in estimating the parameter \( \theta_i \). Hoeffding’s inequality was chosen over the more standard KL divergence because there were some trials where agents observed either no successes or no failures for a given \( \xi_k \), even though for all \( \xi_k, \theta_k \in (0, 1) \). In this case the KL divergence is undefined, instead we computed the error by determining the setting of \( \gamma \) for the number of observations of \( \xi_i \) and a fixed probability of \( P\left(|\theta_k - \hat{\theta}_k| > \gamma\right) \leq 0.05 \).

For experiments 1 and 2 we present a 3D plot showing how the error in estimating the parameters \( \theta_k \) is reduced by using the foraging algorithm over the control algorithm. In addition, we present 2D plots showing where either the foraging algorithm performs better than the control algorithm, or the control algorithm performs better than the foraging algorithm, or when their performance is indistinguishable.

We also report the effect size - the ratio of the mean to the standard deviation of the difference between the 50 paired trials. This is a variation of Cohen’s \( d \) value (Cohen, 2013). Values greater than 1.3 are considered to be very large, above 0.8 to be significant, and below 0.5 to be insignificant. Tables 2.4 and 2.7 give the reduction in estimate error averaged over experiments 1.1-1.5 and 2.1-2.6, respectively.

2.4.1 Experiment 1 - Underlying Distribution

Table 2.4 shows the foraging algorithm reduces parameter estimate error relative to the two control algorithms. The effect size well exceeds the threshold for significance. Figure 2.2 demonstrates that the performance of the foraging algorithm generally performs at least as well as the control algorithms, and often better. However we notice that when searching costs are low the control algorithms can outperform the foraging algorithm.

Experiment 2 - Arrival Probability

Figures 2.3 and 2.4 demonstrate that, with the exception of when searching costs are small, the foraging algorithm performs at least as good and often statistically significantly better the control
Table 2.3: Experiment 1 Foraging vs Control Algorithms

<table>
<thead>
<tr>
<th>Metric</th>
<th>Control 1</th>
<th>Control 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave. Reduced Error</td>
<td>$0.178 \pm 0.014$</td>
<td>$0.559 \pm 0.001$</td>
</tr>
<tr>
<td>Ave. Effect Size</td>
<td>$1.297 \pm 0.030$</td>
<td>$1.818 \pm 0.007$</td>
</tr>
</tbody>
</table>

Table 2.4: Experiment 1 Foraging vs Control 1

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Average Error Reduction</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>0.202</td>
<td>1.285</td>
</tr>
<tr>
<td>1.2</td>
<td>0.201</td>
<td>1.287</td>
</tr>
<tr>
<td>1.3</td>
<td>0.190</td>
<td>1.250</td>
</tr>
<tr>
<td>1.4</td>
<td>0.186</td>
<td>1.262</td>
</tr>
<tr>
<td>1.5</td>
<td>0.116</td>
<td>1.445</td>
</tr>
</tbody>
</table>

Table 2.5: Experiment 1 Foraging vs Control 2

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Average Error Reduction</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>0.558</td>
<td>1.802</td>
</tr>
<tr>
<td>1.2</td>
<td>0.558</td>
<td>1.802</td>
</tr>
<tr>
<td>1.3</td>
<td>0.559</td>
<td>1.818</td>
</tr>
<tr>
<td>1.4</td>
<td>0.560</td>
<td>1.826</td>
</tr>
<tr>
<td>1.5</td>
<td>0.561</td>
<td>1.840</td>
</tr>
</tbody>
</table>

Figure 2.2: The top row shows how the performance of the foraging algorithm compares to the first (greedy) control algorithm and the bottom row compared to the second (uniform) control algorithm. In the majority of settings for sampling and searching costs the foraging algorithm performs at least as good as the control algorithms and often (white regions) statistically significantly better.
algorithms. As we can see in Table 2.7, the effect size of the error reduction is very large by Cohen’s d.

Table 2.6: Experiment 2 Foraging vs Control Algorithms

<table>
<thead>
<tr>
<th>Metric</th>
<th>Control 1</th>
<th>Control 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ave. Reduced Error</td>
<td>0.202 ± 0.082</td>
<td>0.622 ± 0.015</td>
</tr>
<tr>
<td>Ave. Effect Size</td>
<td><strong>1.375 ± 0.122</strong></td>
<td><strong>2.209 ± 0.054</strong></td>
</tr>
</tbody>
</table>

Table 2.7: Experiment 2 Foraging vs Control 1

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Average Error Reduction</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>0.222</td>
<td>1.576</td>
</tr>
<tr>
<td>2.2</td>
<td>0.140</td>
<td>0.730</td>
</tr>
<tr>
<td>2.3</td>
<td>0.213</td>
<td>1.484</td>
</tr>
<tr>
<td>2.4</td>
<td>0.226</td>
<td>1.608</td>
</tr>
<tr>
<td>2.5</td>
<td>0.208</td>
<td>1.475</td>
</tr>
<tr>
<td>2.6</td>
<td>0.204</td>
<td>1.375</td>
</tr>
</tbody>
</table>

Table 2.8: Experiment 2 Foraging vs Control 2

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Average Error Reduction</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>0.580</td>
<td>2.381</td>
</tr>
<tr>
<td>2.2</td>
<td>0.659</td>
<td>2.271</td>
</tr>
<tr>
<td>2.3</td>
<td>0.659</td>
<td>2.271</td>
</tr>
<tr>
<td>2.4</td>
<td>0.646</td>
<td>1.995</td>
</tr>
<tr>
<td>2.5</td>
<td>0.567</td>
<td>2.070</td>
</tr>
<tr>
<td>2.6</td>
<td>0.623</td>
<td>2.267</td>
</tr>
</tbody>
</table>

**Experiment 3 - Distribution Change**

Figure 2.5 shows that the foraging algorithm with change detection performs substantially better than not using it. The leftmost bars show the performance of the foraging algorithm with and without change detection immediately before the change in the underlying distribution. By employing change detection ("After" in Figure 2.5) we see the error in estimating the underlying distribution is profoundly reduced. However it should be noted that if the number of opportunities to sample is not very large, then the agent will not be able to detect a change in an underlying distribution with any confidence. This must be considered when planning exploration missions.
Figure 2.3: Compared to the first (greedy) control algorithm, the foraging algorithm generally does at least as well or better at estimating the underlying distributions, with statistical significance. However, for low search costs the control algorithm 1 does perform better than foraging.
Figure 2.4: In the second experiment the foraging algorithm does profoundly better than the second (uniform) control algorithm. Statistical significance is achieved in the majority of sample and search cost pairings, and across all experimental parameter settings.

2.5 Conclusion

We presented an algorithm that automatically samples while exploring while identifying changes in distributions underlying sampling targets. The foraging algorithm improves learning of unknown distributions in unknown environments. The change detection component identifies environmental changes that may be relevant to remote users, but it definitely improves the performance of the learning algorithm, making it more robust to unknown and variable environments.

We can draw three conclusions from these experiments. First, accounting for the costs of searching and sampling improves the performance of learning agents. Incorporating costs motivates helping discount possible future opportunities that may not arrive. Second, accounting for arrival probabilities of the random variables improves learning the underlying distributions. Again, it helps motivate not giving up on available opportunities. The reduced error achieved in the time budget speaks to the foraging algorithm's improved productivity over the control algorithms. Both these experiments show that low searching costs obviates accounting for environmental statistics. However, the cost of searching and sampling increases a foraging approach is favourable. When the arrival probabilities of classes deviate from uniform the reduction in error from the foraging algorithm is even more pronounced.

Third, change detection is a valuable component for learning in changing environments. We demonstrated a substantial reduction in final error after a change in the underlying distributions has occurred. By tracking the observations collected by exploring robots we can increase the performance of their learning mechanisms while identifying events of interest to remote scien-
Figure 2.5: Both algorithms perform identically just before the underlying distribution change. At the end of the path the algorithm that detects changes performs substantially better than the one that doesn’t. Error bars represent $1.96 \times$ the standard error over the 50 trials.
This work can be extended in several ways. First, employ the same change detection of the sample values to the arrival probabilities. This way the exploring agent can detect when the composition of the environment changes, which may be interesting to remote scientists. Second, model more complex underlying distributions. Third, integrate site selection with a path planner in order to determine costs of different sampling actions. Finally, account for possible misclassification of the identified random variables in a scene. These additions will make progress towards robust autonomous planetary exploration.
Algorithm 2.2 Change Detection for Foraging

function INIT\_FORAGE\_SAMPLING
    $\Xi \leftarrow \emptyset$
    $R(\cdot) \leftarrow \emptyset$
    $N. \leftarrow \emptyset$
    window$_{a,k} \leftarrow \text{queue}(\emptyset) \quad \text{unless} \quad \text{empty}(\text{window}_{a,k})$
    window$_{b,k} \leftarrow \text{queue}(\emptyset)$
    sample$_{k} \leftarrow \text{queue}(\emptyset) \quad \text{unless} \quad \text{empty}(\text{sample}_{k})$
    sample\_size \leftarrow 5$
    window\_size \leftarrow 30$
end function

function DETECT\_CHANGE($k, z_{k,t}$)
    if $\text{size}(\text{window}_{a,k}) < \text{window\_size}$ then
        push(\text{window}_{a,k}, z_{k,t})
    end if
    push(\text{window}_{b,k}, z_{k,t})
    if $\text{size}(\text{window}_{b,k}) > \text{window\_size}$ then
        pop(\text{window}_{b,k})
    end if
    push(\text{sample}_{k}, z_{k,t})
    if $\text{size}(\text{sample}_{k}) > \text{sample\_size}$ then
        pop(\text{sample}_{k})
        $\theta_{a,k} \leftarrow \text{sum}(\text{window}_{a,k})/\text{size}(\text{window}_{a,k})$
        $\theta_{b,k} \leftarrow \text{sum}(\text{window}_{b,k})/\text{size}(\text{window}_{b,k})$
        $\Lambda \leftarrow \sum_{j=0}^{\text{sample\_size}} \log \left( \frac{P(\text{sample}_{k}(j)|\theta_{a,k})}{P(\text{sample}_{k}(j)|\theta_{b,k})} \right)$
        if $\Lambda > \text{change\_threshold}$ then
            cache(\text{window}_{a,k}) \quad \forall \xi_k \in \Xi$
            window$_{a,k} \leftarrow$ window$_{b,k}$ \quad \forall \xi_k \in \Xi
            init\_forage\_sampling()
        end if
    end if
end function
Chapter 3

Prospecting

Prospecting is the process of traversing a region with a cheap, possibly passive, sensor that is looking to identify places or regions to engage sampling activities that are both discrete and more expensive than the use of the prospecting sensor. Examples of such engagement are an Area of Interest Mapping manoeuvre (AIM), or collecting a soil sample with, e.g., a drill.

This particular problem assumes that a trajectory has been predetermined by some mechanism external to the agent currently under consideration. We can write that trajectory as $\tau(t)$ where $t \in [0, \infty]$. The prospecting sensor collects observations $z_t$ that are drawn from some underlying distribution $G(t) = G(\tau(t))$. Which is to say that the underlying distribution is determined by where the agent is along the trajectory.

In this chapter we seek to improve upon state of the art algorithms by being aware of the hypothesis about change points in the underlying distribution. Specifically, one change point when the distribution changes from $G_1$ to $G_2$. We use an algorithm that is built upon the Sequential Probability Ratio Test to determine with confidence that a change in the underlying distribution has occurred.

Prospecting is a distinct problem from the previously discussed Opportunistic Sampling problem chiefly because it is not interacting with discrete targets. Further where the goal of the robot in the previous chapter was to determine the predictive power of a cheap sensor with respect to a second, expensive sensor, in this setting we consider that the prospecting sensor is a reliable predictor of the quantity of interest.

Prospecting is different from sample scheduling in that it is reactive to the observations collected in situ along the trajectory. Further we do not place a constraint on how many discrete actions should be deployed. And again, this problem assumes a pre-determined trajectory. While intelligently planning a trajectory could increase the performance of prospecting it does not eliminate the need for a mechanism which makes decisions to take discrete actions, so in this way the prospecting problem is different from informative trajectory planning.

The work in this chapter was modelled on the Mojave Volatiles Prospector Project conducted in the Mojave desert in 2014 in part by the NASA Ames Intelligent Robotics Group. Here the prospecting instrument was a neutron spectrometer (NSS) and the high-cost intervention was an Area of Interest Mapping Maneuuvre (AIM). In this chapter we model an AIM as driving in a circular pattern when an area of interest is detected. The data used in this chapter was either data from the MVP project or simulated based on the distribution of NSS readings collected during
this project. More about the MVP project is discussed in section 3.1.

The state of the art in prospecting is best exemplified by the work of Ferri et al. (2010). As such this algorithm is used as a baseline for comparison in the experiments conducted in this chapter. This algorithm issues an AIM if an observation crosses a predetermined threshold. This and other related work are discussed in section 3.2.

What we propose is an algorithm that attempts to determine if the source driving the readings of the prospecting sensor has changed, and if so, take any necessary actions. This algorithm is based around the sequential probability ratio test (SPRT) [Wald (1945)], which is used to determine if one of two hypotheses are more likely, given the observed data. Our competing hypotheses is if there has been either no change in the underlying distribution or if there has been one change in the underlying distribution.

The algorithms will be discussed in detail in ??.. The experiments used to test them will be described in section 3.4 and their results discussed in section 3.5.

### 3.1 The Mojave Volatiles Prospector Project

To ground this work we consider the case of the Mojave Volatiles Prospector (MVP) project from 2014, conducted at by NASA Ames in the Mojave desert [Heldmann et al. (2015)]. The robot deployed in this project, pictured in Figure 3.1, was equipped with a Neutron Spectrometer (NSS), which was used to estimate the abundance of subsurface water.

![KRex2 in the Mojave desert](image)

**Figure 3.1:** KRex2 in the Mojave desert. On the rear of the vehicle is visible the stainless steel and cadmium cylinders of the Neutron Spectrometer.

MVP served as an early test of the kind of high-tempo scientific operations that would be required to support short-duration exploration missions like the planned Resource Prospector
mission [Andrews et al. (2014)](2014). Robot-habitable conditions on the surface of the moon, (e.g., sunlight, temperatures above freezing) only last for two week periods at a time. Unless a vehicle is designed to last lunar night, thus adding to the cost of the mission, operations must be limited to the lunar day. Consequently these missions do not have the luxury of Mars missions to take 24 hour periods to make decisions about deploying scientific instruments.

The Neutron Spectrometer (NSS) is an instrument that infers the abundance of water molecules in the field of view of the instrument. The instrument can sense water content up to a meter below the surface of the ground and in approximately a meter radius laterally along the surface of the ground.

The “just so story” of the NSS is that it reports counts of how many neutrons it estimates passed through the field of view of the instrument at the rate of once per second. These counts are modelled as a Poisson process. The average rate of the Poisson process is a function of the water abundance by weight and the depth of the water deposit. Generally speaking, a higher rate of counts implies a greater volume of water.

In the process of the MVP project it was discovered that the background rate was approximately 40 counts per second. A high number of counts, highly correlated with water is approximately 120 counts per second. A threshold for having detected water was approximately 80 counts per second. As such we used 80 as the threshold for the threshold algorithm.

Because the NSS simply counts the number of neutrons that pass through it, there is no sampling budget associated with it. The instrument does have a fixed sampling rate, however, only reporting once per second. Further, there is a fixed power budget which is necessary to operate the instrument while prospecting, but we will consider this cost as non-negotiable and fixed.

Prospecting is an important part of exploration. An NSS is planned to be part of NASA’s upcoming Resource Prospector mission. The instrument will be the primary prospecting sensor to search for subsurface deposits of solid water. The vehicle will drill at where it determines the water density is most likely highest. The tight time constraints of operating in one lunar day motivate effectively finding the true water density maxima.

### 3.2 Related Work

There are three major non-adaptive methods for determining where and when to deploy AIMs: Leave it to a human; scout out the area and determine the relative maxima; or to use a threshold to determine when an AIM should be employed. However, human judgement is not always available, or if it is, it requires demands be placed on unbiased experts whenever the prospecting sensor is deployed.

Pre-scouting an area is a viable approach should the exploration budget have sufficient margin that it can traverse the length of the path at least twice. But it does make the assumption that the interesting observations that are made along a traverse have no temporal component and will still be there when the robot returns to the site. This is not necessarily a valid assumption, and as such decisions need to be made *in situ* to be most effective.

[Yoeger et al., 2000] Uses a robot to scan an area by searching a grid. This approach is principled and avoids collecting data. However, erroneous readings can be over represented in
the data, as there is no attention paid to the statistical confidence in the reported data. Likewise, anomalies that do appear are uninvestigated, as this is a non-adaptive approach. (Yoerger et al., 2007) improves on that prior approach by having the vehicle follow a search pattern, then identify anomalies – readings that pass a certain pre-determined threshold – and then return along the path to gather more observations of those anomalies.

The third approach, of using a threshold to determine when to conduct and aim is, in principle, feasible, but only if the statistics of the environment are already well understood going into the exploration site. A thresholded approach does not adapt to the statistics of an environment, and as such it may execute AIMS either too frequently, in sufficiently abundant territories, or not frequently enough, should sites with high concentrations of the reading of interest be missed, or the higher concentration regions be not as highly concentrated as the remote scientists hoped. (Ferri et al., 2010) describes a robot that is attempting to localize undersea hydro-thermal vents. They track the vents by looking at the concentration of different chemicals in the water. Should the concentration they are currently observing pass a threshold then they initiate a spiral area of interest mapping maneouvre in order to build a more informed map of chemical concentrations.

In the case of the Resource Prospector project, the purpose is to localize maxima of the neutron signal in order to drill in a location likely to be most abundant with water. In the case of the work of Ferri et al. (2010) the objective was to localize geothermal vents, places likely to contain new and exciting ecosystems.

In some respects (Ferri et al., 2010) is an improvement on (Yoerger et al., 2007). In the previous work the vehicle had to travel to the end of the trajectory before it determines where to deploy the AIMS, where as (Ferri et al., 2010) the vehicle can determine to deploy AIMS as it is travelling. However, the reliance on an a priori threshold means that the vehicle cannot adapt itself to operations in environments that have not been previously characterized. This makes the algorithm inappropriate for exploring alien worlds.

Two more adaptive algorithms are developed presented in the work of Girdhar et al. (2013b,a; Girdhar and Dudek, 2016), and in the paper (Thompson et al., 2013).

The work developed by Girdhar focuses on processing images in situ. The images that are collected by the robot are subjected to topic analysis to create meaningful clusters of images. As the robot explores it scores the perplexity of the image, or in the case of (Girdhar and Dudek, 2016) sub-regions of the image. Topic perplexity in this case is a measure of how well the image is described by any one of the topics already discovered in the image database. When the topic perplexity is high for an image, the vehicle slows down in order to collect more observations of the perplexing scene. These vehicles do not plan with respect to overall mission objectives or goals, but simply employ their resources greedily.

There is an analogy between topic perplexity and the entropy of an image. Girdhar et al. model images as Bag of Words (BoW) vectors. Their topic content can be viewed as the dot product between the topics BoW vectors and their individual image BoW descriptions. The product between the images and all the topic models produces a distribution over the topics. An image that has a high entropy distribution over the topic models represents an anomaly that is not adequately described by the code book developed from the previous observations and as such warrants further investigation. Additional observations get added into the database that are used for producing the topic models, and as such there is an adaptation to the topic model to describe
the environment the robot is operating in.

In (Thompson et al., 2013) the focus is on deploying samples when there is a limited sampling budget. Here the explorer is carrying an instrument that is sampling at a predetermined rate and the only thing the robot can do to collect more samples of a phenomenon of interest is to slow the vehicle down. Further, the vehicle is forbidden from back-tracking. The focus of the work was introducing non-stationarity into a Gaussian Process kernel function in order to have the vehicle adapt to anomalies on-line. That being said, generating two distributions to model the data, one stationary and one not, could be used as the competing hypotheses that are considered in this work, should one need to make a discrete decision.

In the case of using an instrument such as a neutron spectrometer, as in this chapter, there is no reason to believe that there will be an explicit limit on the number of samples that can be collected. However, there may be a limit on overall mission time and/or power resources that can be deployed at any given time. Additionally, unlike (Thompson et al., 2013), our algorithm needs to make a distinct decision and does not have a mode of operation that exists along a continuum.

In this chapter we discuss a principled approach to autonomously determine when to conduct area of interest mapping activities given an unlimited proxy sensor.

3.2.1 Inadequacies of the Secretary Problem Setting

The situation addressed in (Ferri et al., 2010) is a perfect application for the secretary algorithms described in ??4. The purpose of the secretary problems is to give the decision maker some data on which to build a distribution, and only seeks to make one decision instead of deploying multiple AIMs, and does not consider if the underlying distribution is changing.

The secretary problem is a problem setting from statistical literature Ferguson (1989) which was constructed to determine when was the optimal time to stop collecting samples. The principle set up was this: A person is trying to hire a secretary. They are presented with a sequence of \( N \) candidates to consider for the position. Each candidate has a knowable score, relative to all the other candidates which can be determined by interviewing the candidate. The interviewer then can either choose to hire the candidate or dismiss them and go on to the next candidate. Once a candidate has been dismissed they cannot be recalled. The problem is to determine when is the optimal time to stop interviewing candidates. It was determined that the optimal solution for this problem was to interview the first \( N/e \) candidates, where \( e \) is Euler’s constant, and reject each one in turn. The interviewing process stops either when the first candidate who scores as good as or better than the best candidate in the first \( N/e \) candidates, or take the last candidate. This ensures finding the best candidate 37% of the time.

In principle this decision rule makes sense: Determine the distribution of the scores of candidates, then find an outlier with a high score. This problem does line up nicely with automatic AIM execution, the sensor readings from our proxy sensor maps nicely to the score of the candidates. As the robot continues to move we are encountering the sensed phenomena much like the sequence of candidates.

However, there are four reasons why the secretary problem setting is not an exact match for our present situation. First, it does not consider noise in the sensor. Second, it assumes that there is no cost to continuing to the next candidate, and that candidate cannot be recalled. This is not the case for autonomous exploration, moving between locations takes time, and any
traverse comes with risk associated with the execution of tasks in an unknown environment. Third, this setting does not allow for the underlying distribution of the candidates to change, nor does it acknowledge that candidates may be coming from a mixture of distributions. Fourth, the candidates are considered to be drawn IID from the same distribution. In our setting there can be strong spatial effects on the population from which sensor readings are derived. That is to say, near a location where is a deposit of water, the readings will come from a distribution with a higher mean than those from farther away from the water deposit.

The strong spatial dependence creates a need to not only determine how many distributions there are, but also to determine when the vehicle should be confident that it is operating in either one mode or the other. Just modelling the distribution alone is not sufficient, as simply executing AIMs when the probability of being in one regime is higher than the others could result in excessive deployment of AIMs and an unnecessary use of mission resources.

To conclude, there are applications for robots that need to determine when to deploy expensive activities – slowing vehicle speed, conducting AIMs, or involved sampling activity like drilling. To date the algorithms deployed are either greedily ignoring resources, using arbitrary and non-adaptive thresholds, or simply control speed of a vehicle proportionally to perceived anomalies without considering if there is statistical confidence that an anomaly has been observed. This chapter proposes an adaptive algorithm that determines when to deploy resources in response to statistical confidence that there has been an event that warrants observation.

3.3  Algorithms

We used two algorithms in this experiment. The state of the art algorithm, as used in Ferri et al. (2010) is to issue an AIM whenever a reading crosses a threshold. We modify this algorithm slightly to make the comparison fairer to the baseline algorithm and the algorithm described in ?? is the baseline algorithm that is employed in the experiments of this chapter.

Additionally, in subsection 3.3.4 we compare the proposed algorithm to a similar MCMC-based approach. Although performance is comparable to the proposed algorithm, and the implementation of the algorithm could be considered more intuitive than the proposed algorithm we find that the time to compute solutions is significantly higher and does not warrant that adoption without additional hardware speed-ups.

3.3.1  Detecting Changes with a Threshold

This algorithm compares every observation $z_t$ to a threshold, $\gamma$ and if $z_t > \gamma$, or conversely if $z_t < \gamma$ then a change has considered to have happen. This algorithm is memoryless and only considers one observation at a time, and each in isolation.

When the process being observed is well characterized this is a useful algorithm. However, the threshold needs to be tuned to every different environment that a robot is operating in. This makes the algorithm very brittle, but it remains a state of the art algorithm.

An analogous version of this algorithm is given a probability distribution, consider a change has occurred when a reading has a probability $P(z_t|\theta) < \delta$ then consider a change has occurred. For every such setting of $\delta$ there is a corresponding $\gamma$ to threshold the sensor values. Again, this
algorithm has no memory, and once one starts trying to accumulate data in favour of a change, we get into the proposed algorithm, the Sequential Probability Ratio Test.

**Algorithm 3.1** Threshold Change Detection. This algorithm reports that a change has occurred whenever the observed value, $z_t$, is greater than the threshold, $\gamma$. Because the algorithm has no memory, it simply reports that the change has occurred at the current time step.

```python
function INIT_THRESHOLD_CHANGE_DETECTION(\gamma)
    \gamma \leftarrow \gamma
end function

function DETECT_CHANGE(z_t)
    if $z_t \geq \gamma$ then
        return $t$
    else
        return $-1$
    end if
end function
```

This algorithm is in line with the approach taken by [Ferri et al.] (2010).

### 3.3.2 Detecting Changes with a Threshold and Memory

With little additional effort it is possible to have a slightly smarter algorithm based on thresholding the observed data. We add additional memory such that the first time an observation crosses the threshold, $\gamma$, we mark the current time as the start of the change. The algorithm reports the change point as being the time of the first observation that has crossed the threshold until the observation goes below the threshold again. This approach is coded in **Algorithm 3.2**.

Memory of when a change first occurred helps stabilize the algorithm’s estimation of when a change has occurred. Without memory the algorithm will think that every time the threshold has been crossed is a new instance of a change in the underlying distribution.
Algorithm 3.2 Threshold Change Detection with Memory. This algorithm reports that a change has occurred at the first observed value, \( z_t \), is greater than the threshold, \( \gamma \). Once the observed value goes below the threshold the change period is reset. This prevents the algorithm for constantly reporting that a new change has occurred even though it has been consistently occurring for a number of time steps.

```plaintext
function INIT_MEMORY_THRESHOLD_CHANGE_DETECTION(\( \gamma \))
\( \gamma \leftarrow \gamma \)
\( change\_start \leftarrow -1 \)
\( in\_change \leftarrow true \)
end function

function DETECT_CHANGE(\( z_t \))
if \( z_t \geq \gamma \land in\_change \)
\( change\_start \leftarrow t \)
\( return \ change\_start \)
end if
if \( z_t \geq \gamma \land in\_change \)
\( return \ change\_start \)
end if
if \( z_t < \gamma \)
\( in\_change \leftarrow false \)
\( change\_start \leftarrow -1 \)
\( return \ change\_start \)
end if
end function
```

3.3.3 Detecting Changes with the Sequential Probability Ratio Test

The sequential probability ratio test considers two hypotheses, \( h_1 \) and \( h_2 \). Each hypothesis implies some probability distribution about observations that were collected. The sequential probability ratio test asks the question which distribution better explains observed data \( z_1, \ldots, z_t \). We determined the relative probability of the data given the hypotheses is:

\[
R_t = \frac{P(z_1, \ldots, z_t|h_2)}{P(z_1, \ldots, z_t|h_1)}
\] (3.1)

If at any given time \( R > 1 \) then \( h_2 \) is more likely. If \( R < 1 \) then \( h_1 \) is more likely. However, if observations are considered to be independently and identically distributed then we can write the following:

\[
R_t = \prod_{i=1}^{t} \frac{P(z_i|h_2)}{P(z_i|h_1)}
\] (3.2)

And if we take the log of this quantity we find:
Λ_t = \sum_{i=1}^{t} [\log (P(z_i|h_2)) - \log (P(z_i|h_1))]
Λ_t = \sum_{i=1}^{t} \log (P(z_i|h_2)) - \sum_{i=1}^{t} \log (P(z_i|h_1))
(3.3)

The log form of the equations has the advantage of letting us work with sums, which are easier to keep a running total of. Additionally, since likelihoods can be come very small very quickly it is more numerically stable to work with the log of the likelihood function.

Wald determined thresholds $\alpha$ and $\beta$ based on the acceptable false positive and false negative rates for the test. If $\Lambda_t > \alpha$ then $h_2$ is accepted and if $\Lambda_t < \beta$ then $h_1$ is accepted and the experiment stops. For specified levels of error probabilities the SPRT is the test with the minimum expected stopping time 7.

In this chapter $h_1$ is the hypothesis that no change has occurred and $h_2$ is the belief that a change has occurred. We consider the hypothesis that a time has occurred at every time point between 1 and $t$, the current time. We select the most likely time change. If that potential change point crosses a threshold $\alpha$ then we consider that a change in the underlying distribution has occurred.

Fitting two distribution to a data set should always fit the data better than one distribution. The implication of this is that $R_t \geq 1$ and hence $\Lambda_t \geq 0$. So we can ignore checking if we become certain that there has no change. In fact, because we don’t do anything until confidence has built that there has been a change, no further action is required of the robot except to continue driving. The aim is only necessary when we are confident that a change has occurred.

One risk is that because $\Lambda_t \geq 0 \ \forall t \geq 0$ there is the risk that with enough observations the quantity will eventually cross the confidence threshold, even if no underlying change has occurred. To mitigate this risk we set $\alpha$ to be larger than prescribed by Wald. The task of determining the confidence level at which an AIM should be deployed is left to the people who designed the mission and be a function of their tolerance of risk and the expected duration of the entire transect.

There are a number of assumptions we make here in order to implement this algorithm. First, we assume the data from a given distribution are all IID and coming from Poisson distributions. We consider that in the case of no change in the underlying distribution that it was drawn from a Poisson distribution with rate $\theta_0$. In the case were there has been a change in the distribution we call the rate before the change $\theta_1$ and the rate after the change $\theta_2$.

Since we do not know $\theta_0$, $\theta_1$, or $\theta_2$, we estimate them by the maximum likelihood estimates from corresponding samples, $\hat{\theta}_0 = MLE(\langle z_0, \ldots, z_T \rangle)$, $\hat{\theta}_1 = MLE(\langle z_0, \ldots, z_t \rangle)$, and $\hat{\theta}_2 = MLE(\langle z_{t+1}, \ldots, z_T \rangle)$, respectively, where $T$ is the current maximum time step.

After an AIM has been conducted the window of samples gets reset to the samples after the change point, $t$, and the process continues on as before. The implementation of the algorithm is given in 3.3.
Algorithm 3.3 Sequential probability ratio test-based change detection. This algorithm considers all the data that has been collected up to the present time, $t$, and determines whether or not there has been a change in the underlying distribution.

```
function INIT_SPRT_CHANGE_DETECTION(a)
    a ← a
    data ← ⟨⟩
end function

function POISSON_LIKELIHOOD(z_1, ..., z_n, λ)
    return \prod_{k=1}^{n} P_{\text{poisson}}(z_k | λ)
end function

function DETECT_CHANGE(z_t)
    data ← data + ⟨z_t⟩
    l_0 ← poisson likelihood
    likelihoods ← ⟨⟩
    for all $k \in \{1, \ldots, t\}$ do
        $\lambda_1 ← \text{mean}(z_1, \ldots, z_k)$
        $\lambda_2 ← \text{mean}(z_{k+1}, \ldots, z_t)$
        likelihoods[k] ← likelihood(z_1, ..., z_k, $\lambda_1$) + likelihood(z_{k+1}, ..., z_t, $\lambda_2$)
    end for
    change_point ← argmax_{k ∈ \{1, ..., t\}} (likelihoods[k])
    if likelihood[change_point] − l_0 > a then
        data ← data [change_point, ..., |data|]
        return change_point
    else
        return −1
    end if
end function
```

This algorithm simply detects the changes in the underlying distribution. It is up to other mission objectives to make value judgements about the meanings of those values. For instance, observations that contradict remote sensing data may warrant a different level of interaction than data that is variable, but consistent with what is observed from orbit.

We can achieve a 2× speed up in the algorithm by realizing that we do not need to recompute the log likelihood of the first $1, \ldots, t$ data points for each $t$. However here I express the naive implementation above, for ease of communication.

Further, the relative values of the underlying distributions may have meanings that are entirely depending on the mission class. If the object is simply to map and explore, then the relative values are meaningless. If the mission is prospecting, then the vehicle may want to not only map the data, but extract resources where they have been determined to likely be at a maximum.
3.3.4 The MCMC-Bayesian Approach is Excessively Time Consuming

The preferred method to approach this problem would be to employ a Bayesian method to determine if there has been a change in the underlying distribution. Modern approaches would have one construct a probability distribution over the change point in the underlying distribution, then feed the model data and use, e.g., an MCMC sampling algorithm to determine the probability that a change has occurred for each time step, for example as conducted in ?. From there one can choose the most likely change point, and act accordingly, or pass the whole distribution over the probability that a change occurred to another Bayesian decision making algorithm. The setup for such a problem is given in as follows:

\[ C_t \sim \text{Poisson}(r_t); \quad r_t = \begin{cases} \text{late} & \text{if } s < t \\ \text{early} & \text{if } s \geq t \end{cases} \]  
(3.4)

\[ s \sim \text{U}(1, t_{\text{max}}) \]  
(3.5)

\[ \text{early} \sim \text{exp}(1) \]  
(3.6)

\[ \text{late} \sim \text{exp}(1) \]  
(3.7)

Where \( s \) is the switch point between the \( \text{early} \) and \( \text{late} \) arrival rates. The arrival rates are drawn from an exponential distribution. They, in turn, drive the Poisson distribution that produces the counts observed by the NSS. This setup, along with the observed data up until the current time \( t_{\text{max}} \) is fed into an MCMC sampler.

The MCMC approach is simple, and thanks to current libraries coding this problem is fairly easy, however, the time it takes to execute the sampler is very high, and not necessarily compatible with flight-forward hardware. For this reason, we don’t consider the Bayesian approach to solving this problem as feasible for current flight missions.

To illustrate the relative performance of our proposed algorithm vs the Bayesian-MCMC approach in Figure 3.2 we report the time it takes to execute an update in response to the latest sample for transects of length 100 samples.

The time to complete an update is substantially larger for the MCMC approach than the SPRT algorithm. The average time for the MCMC algorithm to process a sample was 61.1 seconds, the average time for SPRT to process a sample was 0.0719 seconds. The reduction in processing time was approximately 61 seconds, with a 95% High Density Interval (HDI) of \([58, 62]\) seconds and effect size (Cohen’s d) of 16.8, which is a substantial effect.
Figure 3.2: Here we can see the average execution time for the different algorithms as a function of the number of observations collected. Error bars represent one 95% confidence interval. The SPRT algorithm is on the order of 100x faster than the Bayesian algorithm.

We also consider not just how long it took to execute the two different algorithms, but also how they performed. We compare the performance of the two algorithms by looking at the error in identifying when the change in the underlying distribution occurred, $E_{cp}$. We define that error as:

\[ E_{cp} = |\hat{t}_{cp} - t_{cp}| \]  

Where $\hat{t}_{cp}$ is the estimated time of the change and $t_{cp}$ is the actual time of the change. In the ten trials we performed the change all occurred at the halfway point, $t = 50$ and the underlying distribution changed from $\lambda_1 = 40$ to $\lambda_2 = 80$. We looked at the error that occurs before the change in the underlying distribution ($t = 25$), immediately after the time change ($t = 52$), a short time after the time change ($t = 60$) and at the end of the transect ($t = 100$). The change point detection performance is shown in Figure 3.3.
Change Point Prediction Error for Bayes vs SPRT Algorithm

Figure 3.3: Considering the performance of the two algorithms over 10 trials. The performance is measured as the error between the estimated time of change in the underlying distribution and the true change time. We consider their performance before the change in the underlying distribution (t=25), shortly after the change in the underlying distribution (t=52 and t=60) and at the end of the transect (t=100).

Table 3.1 shows the relative performance of the MCMC algorithm and the SPRT algorithms in detecting the change point in the underlying distribution. In all cases the SPRT and MCMC had statistically significantly different performance.

Something we observed is that after the change had occurred at $t = 50sec$ the MCMC approach reported the change happened at $t = 49sec$ while the SPRT algorithm reported the change occurred at $t = 50sec$. This difference can be viewed as slightly different interpretation of how to report when the change happened, when is the last time step before the new data is observed or when is the first time step corresponding to the new data. So while the difference between the two algorithms is statistically significantly different, and with a substantial effect size it can be viewed as demonstrating comparable behaviour.

More telling is that before the change has occurred the SPRT algorithm is superior to the MCMC algorithm to correctly reporting that no change has occurred. This is a statistically significant difference with a profoundly large effect size. This alone could be taken as motivation for not using the MCMC approach.
Table 3.1: In all cases there is a statistically significant reduction in the error in detecting the change point. Before the time change has occurred the SPRT algorithm is better at detecting there has been no change in the underlying distribution than the MCMC approach. However for all the cases after the change has occurred there is an error of 1 time step, which is a negligible error. We can say that the SPRT and MCMC algorithms have indistinguishable performance after a change has occurred.

<table>
<thead>
<tr>
<th>T (sec)</th>
<th>$\mu_{\text{MCMC}} - \mu_{\text{SPRT}}$ (sec)</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>24</td>
<td>[24, 24]</td>
<td>$1.87 \times 10^4$</td>
</tr>
<tr>
<td>52</td>
<td>1</td>
<td>[0.97, 1.03]</td>
<td>48.2</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>[1, 1]</td>
<td>$1.88 \times 10^3$</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>[1, 1]</td>
<td>$1.86 \times 10^3$</td>
</tr>
</tbody>
</table>

Because the performance of the SPRT algorithm is indistinguishable from that of the Bayesian algorithm after the change, and superior to it before the change in the distribution, and because the time performance is substantially better, we choose to use the SPRT algorithm in favour of the Bayesian/MCMC approach.

3.4 Experiments

We test the SPRT algorithm against a thresholding algorithm in four different experiments. All of the experiments revolve around detecting a change from one underlying distribution to another. All experiments consider a sensor with noisy readings that follow a Poisson distribution. The experiments compare the performance of the algorithms:

1. As a function of the magnitude of change in the underlying distribution rate.
2. As a function of the delay in the onset of the distribution change.
3. On real MVP data.
4. On a simulated 2D environment using data drawn from the observations from the MVP project.

In all the experiments a time step was one second. For the Memory Threshold algorithm a threshold of 80 was used.

3.4.1 Experiment 1 - Effect of Magnitude of Change in the Underlying Distribution

In the first experiment the agent is presented with sequential data. There is a change in the underlying distribution half way through the simulated transect. The initial rate for the Poisson distribution is $\theta_1 = 40$ counts per second. The second rate is given by $\theta_2 \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. We ran 50 trials of each setting of $\theta_2$. The duration of the transect was $T = 100$ time steps.
3.4.2 Experiment 2 - Effect of Delay in Onset of Change in the Underlying Distribution

In experiment 2 we kept the two different rates constant with $\theta_1 = 40$ and $\theta_2 = 100$. The duration of the transect $T = 400$ time steps. The time step at which the distribution changed was varied over $t \in \{0.1T, 0.2T, 0.3T, 0.4T, 0.5T, 0.6T, 0.7T, 0.8T, 0.9T\}$. We ran 50 trials for each setting of $t$.

3.4.3 Experiment 3 - Performance on Real-World Data

We tested the two different algorithms on real-world data collected as part of the MVP project. The objective of the test was to detect changes in the underlying distribution that were annotated by humans. We tested on 9 days of data collected during October of 2014.

3.4.4 Experiment 4 - Effect on Performance in Realistic Scenario

Finally, to determine what effect the algorithm would have on performance in a mission scenario we simulated exploration by a rover in an environment modelled on data from the MVP project. We consider the objective to be the same as in Ferri et al. (2010) - to localize the local maxima of the underlying distribution. We considered a maxima to have been localized if the vehicle drove over that point.

We tested the performance of the algorithms on 30 randomly generated maps and compared them against an additional algorithm which simply followed a lawnmower pattern across the landscape. A map was produced by randomly placing 1000 samples in a $50m \times 50m$ map and the constructing a Voronoi map from those samples. The map was approximated as a $100 \times 100$ cell grid, where cell width was 0.5$m$. 
In this experiment we did not consider trafficability or obstacle avoidance and simply followed a lawnmower pattern that was 50m wide with 5m spacing between switchbacks. An example of the map with the path superimposed is shown in Figure 3.4. At any point in time, $t$, when the rover was at location $(x, y)$ the rate of the Poisson distribution from which observations were sampled was given by $\theta_t = M(x, y)$, where $M(x, y)$ was the value of the map at location $(x, y)$.

### 3.4.5 Performance Metrics

In experiments 1 and 2 we used three performance metrics. The first was the false positive rate. This is the number of time steps before the change in the underlying distribution that were reported as changes. The second metric was the false negative rate. This is the number of time steps after the change in the underlying distribution where no change was reported. The third metric was the mean error in the reported time change. This is the average difference the reported time of change and the actual time of change, for all time changes reported.

In experiment 3 we simply counted the number of changes in the distribution that were reported and saw how many of them were real and how many were fake, as well as how many that were real that were not reported by the algorithm.

In the fourth experiment we considered two metrics: The first is the number of high-value observations that were recorded by the robot. High-value in this case being readings that were at or above the threshold used by the threshold algorithm. The second performance metric was the number of local maxima in the driving rate in the two-dimensional maps used in this experiment.
This metric is inspired by \cite{Ferri et al. (2010)} as the objective of the robot was to capture the local maxima of the chemical densities it was tracking.

3.5 Results

3.5.1 Experiment 1 Results - Effect of Different in Magnitude on Detecting Changes

In Figure 3.5 we see that with the exception of where there is no change between the two arrival rates the SPRT algorithm detects the change with high accuracy. When there is no change in the arrival rates the SPRT algorithm has a much higher false negative rate. This, however, is desirable, as if there is no change in the arrival rate then we don’t want the algorithm to detect a change. Notice in this experiment that it was the SPRT algorithm with a parameter set to 10 that performed the best.

![Graph showing false negative rate as a function of difference between arrival rates with various confidence thresholds, with legend indicating different thresholds (SPRTs = 2, SPRTs = 4, SPRTs = 8, and threshold (t = 80)).](image)

Figure 3.5: Showing the rate of false negatives as a function of the difference between the arrival rates as the confidence parameter of the SPRT algorithm is tuned. For comparison
Table 3.2: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[FN]<em>{MemThresh} - \mu[FN]</em>{SPRT(2)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen's d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>1.00</td>
<td>[1, 1]</td>
<td>$1.98 \times 10^4$</td>
</tr>
<tr>
<td>-20</td>
<td>1.00</td>
<td>[1, 1]</td>
<td>$1.98 \times 10^3$</td>
</tr>
<tr>
<td>-10</td>
<td>0.99</td>
<td>[0.979, 0.993]</td>
<td>68.8</td>
</tr>
<tr>
<td>0</td>
<td>0.49</td>
<td>[0.380, 0.589]</td>
<td>1.86</td>
</tr>
<tr>
<td>10</td>
<td>0.98</td>
<td>[0.971, 0.988]</td>
<td>49.6</td>
</tr>
<tr>
<td>20</td>
<td>1.00</td>
<td>[1, 1]</td>
<td>$1.98 \times 10^3$</td>
</tr>
<tr>
<td>30</td>
<td>0.88</td>
<td>[0.867, 0.897]</td>
<td>29.6</td>
</tr>
<tr>
<td>40</td>
<td>0.48</td>
<td>[0.460, 0.500]</td>
<td>9.79</td>
</tr>
<tr>
<td>50</td>
<td>0.12</td>
<td>[0.110, 0.139]</td>
<td>3.51</td>
</tr>
<tr>
<td>60</td>
<td>0.02</td>
<td>[0.016, 0.027]</td>
<td>1.51</td>
</tr>
</tbody>
</table>

Table 3.3: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[FN]<em>{MemThresh} - \mu[FN]</em>{SPRT(4)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen's d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>1.00</td>
<td>[1, 1]</td>
<td>$1.98 \times 10^4$</td>
</tr>
<tr>
<td>-20</td>
<td>1.00</td>
<td>[0.995, 1]</td>
<td>$1.96 \times 10^3$</td>
</tr>
<tr>
<td>-10</td>
<td>0.95</td>
<td>[0.932, 0.958]</td>
<td>30.20</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>0.02</td>
</tr>
<tr>
<td>10</td>
<td>0.95</td>
<td>[0.934, 0.953]</td>
<td>32.050</td>
</tr>
<tr>
<td>20</td>
<td>0.99</td>
<td>[0.986, 1.000]</td>
<td>133.00</td>
</tr>
<tr>
<td>30</td>
<td>0.88</td>
<td>[0.868, 0.892]</td>
<td>36.70</td>
</tr>
<tr>
<td>40</td>
<td>0.48</td>
<td>[0.462, 0.500]</td>
<td>9.81</td>
</tr>
<tr>
<td>50</td>
<td>0.12</td>
<td>[0.110, 0.139]</td>
<td>3.53</td>
</tr>
<tr>
<td>60</td>
<td>0.02</td>
<td>[0.015, 0.027]</td>
<td>1.54</td>
</tr>
</tbody>
</table>
Table 3.4: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[FN]<em>{MemThresh} - \mu[FN]</em>{SPRT(8)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen's d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>1.00</td>
<td>[1.000, 1.000]</td>
<td>$1.98 \times 10^4$</td>
</tr>
<tr>
<td>-20</td>
<td>0.98</td>
<td>[0.975, 0.983]</td>
<td>99.30</td>
</tr>
<tr>
<td>-10</td>
<td>0.87</td>
<td>[0.862, 0.884]</td>
<td>16.10</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>0.87</td>
<td>[0.844, 0.903]</td>
<td>13.50</td>
</tr>
<tr>
<td>20</td>
<td>0.98</td>
<td>[0.968, 0.981]</td>
<td>96.80</td>
</tr>
<tr>
<td>30</td>
<td>0.87</td>
<td>[0.858, 0.882]</td>
<td>28.60</td>
</tr>
<tr>
<td>40</td>
<td>0.48</td>
<td>[0.459, 0.504]</td>
<td>11.00</td>
</tr>
<tr>
<td>50</td>
<td>0.12</td>
<td>[0.110, 0.138]</td>
<td>3.55</td>
</tr>
<tr>
<td>60</td>
<td>0.02</td>
<td>[0.015, 0.027]</td>
<td>1.54</td>
</tr>
</tbody>
</table>

In Figure 3.6 we see that the false positive rate is independent of the arrival rate of the counts from the sensors. While the threshold of 100 has excellent false positive rate for low arrival rate values, as the arrival rate increases the false positive rate becomes exponentially higher.

A threshold algorithm could be tuned to every environment, it would still need to be tuned for each eventuality. The SPRT algorithm is agnostic of the arrival rate, meaning that the false positive rate can be controlled without tuning the algorithm to each operating condition.

Figure 3.6: Showing the rate of false negatives as a function of the difference between the arrival rates as the confidence parameter of the SPRT algorithm is tuned. The SPRT algorithm’s performance is independent of the background arrival rate. For comparison
Table 3.5: The size of the effect on the false positive rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[FP]<em>{MemThresh} - \mu[FP]</em>{SPRT(2)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen's d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>-0.25</td>
<td>[-0.316, -0.179]</td>
<td>-1.61</td>
</tr>
<tr>
<td>-20</td>
<td>-0.25</td>
<td>[-0.313, -0.176]</td>
<td>-1.65</td>
</tr>
<tr>
<td>-10</td>
<td>-0.25</td>
<td>[-0.314, -0.176]</td>
<td>-1.66</td>
</tr>
<tr>
<td>0</td>
<td>-0.25</td>
<td>[-0.316, -0.179]</td>
<td>-1.60</td>
</tr>
<tr>
<td>10</td>
<td>-0.25</td>
<td>[-0.313, -0.177]</td>
<td>-1.67</td>
</tr>
<tr>
<td>20</td>
<td>-0.25</td>
<td>[-0.317, -0.177]</td>
<td>-1.64</td>
</tr>
<tr>
<td>30</td>
<td>-0.25</td>
<td>[-0.314, -0.177]</td>
<td>-1.60</td>
</tr>
<tr>
<td>40</td>
<td>-0.25</td>
<td>[-0.317, -0.180]</td>
<td>-1.67</td>
</tr>
<tr>
<td>50</td>
<td>-0.25</td>
<td>[-0.317, -0.178]</td>
<td>-1.67</td>
</tr>
<tr>
<td>60</td>
<td>-0.25</td>
<td>[-0.316, -0.179]</td>
<td>-1.62</td>
</tr>
</tbody>
</table>

Table 3.6: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[FP]<em>{MemThresh} - \mu[FP]</em>{SPRT(4)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen's d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>0.00</td>
<td>[-0.014, 0.0001]</td>
<td>-0.004</td>
</tr>
<tr>
<td>-20</td>
<td>0.00</td>
<td>[-0.016, 0.0001]</td>
<td>-0.003</td>
</tr>
<tr>
<td>-10</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>-0.010</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>-0.025</td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>[-0.010, 0.0001]</td>
<td>-0.013</td>
</tr>
<tr>
<td>20</td>
<td>-0.01</td>
<td>[-0.018, 0.0001]</td>
<td>-0.004</td>
</tr>
<tr>
<td>30</td>
<td>-0.01</td>
<td>[-0.018, 0.00003]</td>
<td>-0.825</td>
</tr>
<tr>
<td>40</td>
<td>-0.002</td>
<td>[-0.015, 0.0002]</td>
<td>-0.006</td>
</tr>
<tr>
<td>50</td>
<td>-0.001</td>
<td>[-0.009, 0.0002]</td>
<td>-0.006</td>
</tr>
<tr>
<td>60</td>
<td>0.00</td>
<td>[0.000, 0.000]</td>
<td>-0.009</td>
</tr>
</tbody>
</table>
Table 3.7: The size of the effect on the false negative rate using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[FN]<em>{MemThresh} - \mu[FN]</em>{SPRT(8)}$</th>
<th>95% HDI</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>-20</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>-10</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>0</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>20</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>30</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>40</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>50</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
<tr>
<td>60</td>
<td>0.00</td>
<td>[0.0, 0.0]</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Figure 3.7: For every time step that the algorithm reports a change, the error between the reported time step and the true change point.
Table 3.8: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 2.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[ME]<em>{MemThresh} - \mu[ME]</em>{SPRT(2)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen's d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>54.9</td>
<td>[53.4, 56.6]</td>
<td>15.9</td>
</tr>
<tr>
<td>-20</td>
<td>54.9</td>
<td>[53.3, 56.7]</td>
<td>15.4</td>
</tr>
<tr>
<td>-10</td>
<td>55.9</td>
<td>[54.1, 57.6]</td>
<td>13.2</td>
</tr>
<tr>
<td>0</td>
<td>58.4</td>
<td>[53.1, 64.0]</td>
<td>4.44</td>
</tr>
<tr>
<td>10</td>
<td>55.7</td>
<td>[54.0, 57.4]</td>
<td>13.2</td>
</tr>
<tr>
<td>20</td>
<td>53.8</td>
<td>[42.9, 55.8]</td>
<td>32.4</td>
</tr>
<tr>
<td>30</td>
<td>29.6</td>
<td>[27.0, 32.2]</td>
<td>5.32</td>
</tr>
<tr>
<td>40</td>
<td>29.3</td>
<td>[27.7, 31.2]</td>
<td>8.06</td>
</tr>
<tr>
<td>50</td>
<td>23.9</td>
<td>[22.5, 25.4]</td>
<td>8.07</td>
</tr>
<tr>
<td>60</td>
<td>13.2</td>
<td>[10.6, 15.7]</td>
<td>2.40</td>
</tr>
</tbody>
</table>

Table 3.9: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 4.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[ME]<em>{MemThresh} - \mu[ME]</em>{SPRT(4)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>50.0</td>
<td>[50, 50]</td>
<td>$1.92 \times 10^3$</td>
</tr>
<tr>
<td>-20</td>
<td>50.2</td>
<td>[50.1, 50.4]</td>
<td>127</td>
</tr>
<tr>
<td>-10</td>
<td>50.5</td>
<td>[49.9, 51.2]</td>
<td>30.6</td>
</tr>
<tr>
<td>0</td>
<td>33.5</td>
<td>[23.7, 42.6]</td>
<td>1.45</td>
</tr>
<tr>
<td>10</td>
<td>50.5</td>
<td>[49.9, 51.1]</td>
<td>35.9</td>
</tr>
<tr>
<td>20</td>
<td>50.0</td>
<td>[50, 50]</td>
<td>$2.02 \times 10^3$</td>
</tr>
<tr>
<td>30</td>
<td>25.4</td>
<td>[23.1, 27.4]</td>
<td>7.37</td>
</tr>
<tr>
<td>40</td>
<td>24.5</td>
<td>[23.8, 25.1]</td>
<td>23</td>
</tr>
<tr>
<td>50</td>
<td>20.0</td>
<td>[19.1, 20.8]</td>
<td>14.7</td>
</tr>
<tr>
<td>60</td>
<td>9.1</td>
<td>[6.19, 11.6]</td>
<td>2.42</td>
</tr>
</tbody>
</table>
Table 3.10: The size of the effect on the mean error using the SPRT algorithm as compared to the Memory Threshold algorithm. SPRT confidence level is set at 8.

<table>
<thead>
<tr>
<th>$\lambda_2 - \lambda_1$</th>
<th>$\mu[\text{FN}]<em>{\text{MemThreshold}} - \mu[\text{FN}]</em>{\text{SPRT}(8)}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-30</td>
<td>50.0</td>
<td>[50.0, 50.0]</td>
<td>$1.98 \times 10^3$</td>
</tr>
<tr>
<td>-20</td>
<td>50.0</td>
<td>[50.0, 50.0]</td>
<td>$1.97 \times 10^3$</td>
</tr>
<tr>
<td>-10</td>
<td>50.0</td>
<td>[49.5, 50.5]</td>
<td>33.7</td>
</tr>
<tr>
<td>0</td>
<td>0.0</td>
<td>[−0.002, 0.002]</td>
<td>0.03</td>
</tr>
<tr>
<td>10</td>
<td>50.0</td>
<td>[49.5, 50.4]</td>
<td>43.4</td>
</tr>
<tr>
<td>20</td>
<td>50.0</td>
<td>[50.0, 50.0]</td>
<td>$2.09 \times 10^3$</td>
</tr>
<tr>
<td>30</td>
<td>25.1</td>
<td>[23.0, 27.2]</td>
<td>6.34</td>
</tr>
<tr>
<td>40</td>
<td>24.5</td>
<td>[23.9, 25.2]</td>
<td>18.1</td>
</tr>
<tr>
<td>50</td>
<td>19.1</td>
<td>[18.1, 20.1]</td>
<td>7.69</td>
</tr>
<tr>
<td>60</td>
<td>7.9</td>
<td>[6.18, 9.68]</td>
<td>1.84</td>
</tr>
</tbody>
</table>

3.5.2 Experiment 2 Results - Effect of Delay of Change Onset

The SPRT algorithm once again outperforms the threshold based algorithm. The rate of false positives is largely unaffected by the change point of the algorithm, whereas the Threshold algorithm has a substantially higher rate, see Figure 3.8.

Figure 3.8: For every time step that the algorithm reports a change, the error between the reported time step and the true change point. Error bars represent a 95% confidence interval, $n = 50$. 47
Table 3.11: The average reduction in false positive rate due to using the SPRT algorithm vs the Memory Threshold algorithm.

<table>
<thead>
<tr>
<th>Change Point (sec)</th>
<th>$\mu_{FP}^{MemThresh} - \mu_{FP}^{sprt}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.016</td>
<td>[0.009, 0.022]</td>
<td>0.96</td>
</tr>
<tr>
<td>80</td>
<td>0.016</td>
<td>[0.011, 0.020]</td>
<td>1.96</td>
</tr>
<tr>
<td>120</td>
<td>0.017</td>
<td>[0.014, 0.020]</td>
<td>2.84</td>
</tr>
<tr>
<td>160</td>
<td>0.016</td>
<td>[0.013, 0.018]</td>
<td>3.13</td>
</tr>
<tr>
<td>200</td>
<td>0.016</td>
<td>[0.014, 0.018]</td>
<td>5.08</td>
</tr>
<tr>
<td>240</td>
<td>0.017</td>
<td>[0.015, 0.020]</td>
<td>4.72</td>
</tr>
<tr>
<td>260</td>
<td>0.017</td>
<td>[0.015, 0.019]</td>
<td>4.61</td>
</tr>
<tr>
<td>320</td>
<td>0.017</td>
<td>[0.015, 0.019]</td>
<td>4.94</td>
</tr>
<tr>
<td>360</td>
<td>0.017</td>
<td>[0.015, 0.019]</td>
<td>5.72</td>
</tr>
</tbody>
</table>

The false negative rate of the SPRT algorithm does increase as a function of the time change of the distribution, Figure 3.9. Simply, if there is an overwhelming amount of data from the first distribution, it becomes more difficult to identify a change late in the observation scheme. The thresholding scheme has superior performance on this front, however the analysis of the mean error shows that the Threshold algorithm has a substantial weakness.

![Figure 3.9](image-url)

Figure 3.9: For every time step that the algorithm reports a change, the error between the reported time step and the true change point. Error bars represent a 95% confidence interval, $n = 50$.  

48
Table 3.12: The reduction in FN rate from the Memory Threshold to the SPRT algorithm.

<table>
<thead>
<tr>
<th>Change Point (sec)</th>
<th>$\mu[FN]<em>{Mem\text{Thresh}} - \mu[FN]</em>{sprt}$</th>
<th>95% HDI</th>
<th>Effect Size (Cohen's d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.029</td>
<td>[0.027, 0.031]</td>
<td>5.14</td>
</tr>
<tr>
<td>80</td>
<td>0.029</td>
<td>[0.026, 0.031]</td>
<td>4.78</td>
</tr>
<tr>
<td>120</td>
<td>0.028</td>
<td>[0.026, 0.031]</td>
<td>4.41</td>
</tr>
<tr>
<td>160</td>
<td>0.028</td>
<td>[0.025, 0.031]</td>
<td>4.18</td>
</tr>
<tr>
<td>200</td>
<td>0.027</td>
<td>[0.025, 0.030]</td>
<td>6.19</td>
</tr>
<tr>
<td>240</td>
<td>0.026</td>
<td>[0.022, 0.030]</td>
<td>2.65</td>
</tr>
<tr>
<td>260</td>
<td>0.025</td>
<td>[0.021, 0.030]</td>
<td>2.52</td>
</tr>
<tr>
<td>320</td>
<td>0.025</td>
<td>[0.019, 0.031]</td>
<td>1.57</td>
</tr>
<tr>
<td>360</td>
<td>0.018</td>
<td>[0.008, 0.027]</td>
<td>0.73</td>
</tr>
</tbody>
</table>

Figure 3.10 shows that the average error in predicting when the time change occurred varies wildly for the thresholding algorithm. This is a function of the algorithm being memoryless, it can’t have a stable estimation of when the change occurred. While the SPRT may be somewhat delayed in predicting when the change occurred, the time it does estimate the change happened is very close to correct.

Figure 3.10: The average error in predicting the time point of the distribution change. The SPRT algorithm has a very small error in predicting the time change, independent of the change point. The threshold algorithm, however, has a linear relationship in the error in predicting the change point. Notice that the trend line does not cross the $\text{error} = 0$ at the halfway point of roughly 200 time steps, consistent with having false positive rate of slightly greater than 50%. Error bars represent a 95% confidence interval, $n = 50$. 
### 3.5.3 Experiment 3 Results - Real MVP Data

Here we have tested the competing algorithms on data collected during the 2014 MVP operations in the Mojave desert. We are using the NSS data that was collected from the 17th to the 25th of October, 2014.

In this experiment we consider only counts from the stainless steel channel. Figure 3.11 to Figure 3.19 shows the data that were collected during the MVP project. We set the threshold and memory threshold algorithms with the same threshold that was used to identify hot spots in the field. The SPRT algorithm was run with a confidence threshold of 8.

What we see in these figures is again the main point of this chapter - threshold based algorithms that rely purely on the raw data can only identify observations that cross the threshold. They cannot identify downward trends in the data, without the addition of additional thresholds. Further, simple threshold algorithms are blind to variations happening below the level of a pre-determined threshold.

<table>
<thead>
<tr>
<th>Change Point (sec)</th>
<th>( \mu[\text{MeanError}]<em>{\text{MemThresh}} - \mu[\text{MeanError}]</em>{\text{sprt}} )</th>
<th>95% HDI</th>
<th>Effect Size (Cohen’s d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>155</td>
<td>[155, 157]</td>
<td>39.3</td>
</tr>
<tr>
<td>80</td>
<td>136</td>
<td>[134, 138]</td>
<td>44.4</td>
</tr>
<tr>
<td>120</td>
<td>115</td>
<td>[112, 117]</td>
<td>25.1</td>
</tr>
<tr>
<td>160</td>
<td>94.2</td>
<td>[91.6, 96.9]</td>
<td>19.1</td>
</tr>
<tr>
<td>200</td>
<td>73.9</td>
<td>[70.7, 77]</td>
<td>13.8</td>
</tr>
<tr>
<td>240</td>
<td>51.9</td>
<td>[49.2, 54.5]</td>
<td>10.4</td>
</tr>
<tr>
<td>260</td>
<td>42.6</td>
<td>[40.2, 45]</td>
<td>9.47</td>
</tr>
<tr>
<td>320</td>
<td>11.8</td>
<td>[9.35, 14.2]</td>
<td>2.81</td>
</tr>
<tr>
<td>360</td>
<td>16.1</td>
<td>[−18.9, −13.4]</td>
<td>-3.42</td>
</tr>
</tbody>
</table>
Figure 3.11: The blue line indicates the NSS counts recorded on 17 October 2014. The vertical dashed lines show when the different algorithms detected a change.

Figure 3.12: The blue line indicates the NSS counts recorded on 18 October 2014. The vertical dashed lines show when the different algorithms detected a change.
Figure 3.13: The blue line indicates the NSS counts recorded on 19 October 2014. The vertical dashed lines show when the different algorithms detected a change.

Figure 3.14: The blue line indicates the NSS counts recorded on 20 October 2014. The vertical dashed lines show when the different algorithms detected a change.
Figure 3.15: The blue line indicates the NSS counts recorded on 21 October 2014. The vertical dashed lines show when the different algorithms detected a change.

Figure 3.16: The blue line indicates the NSS counts recorded on 22 October 2014. The vertical dashed lines show when the different algorithms detected a change.
Figure 3.17: The blue line indicates the NSS counts recorded on 23 October 2014. The vertical
dashed lines show when the different algorithms detected a change.

Figure 3.18: The blue line indicates the NSS counts recorded on 24 October 2014. The vertical
dashed lines show when the different algorithms detected a change.
Figure 3.19: The blue line indicates the NSS counts recorded on 25 October 2014. The vertical dashed lines show when the different algorithms detected a change.

Table 3.13: In this table we report only the results of the Memory Threshold Algorithm and the SPRT algorithm. While SPRT reports more changes than actually occurred, they are often co-located. We also see that SPRT notices changes that are not observed by the Memory Threshold algorithm.

<table>
<thead>
<tr>
<th>Date</th>
<th>Changes Detected</th>
<th>Actual Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mem Threshold</td>
<td>SPRT</td>
</tr>
<tr>
<td>2014-10-17</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2014-10-18</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2014-10-19</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2014-10-20</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>2014-10-21</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>2014-10-22</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>2014-10-23</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2014-10-24</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2014-10-25</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

3.5.4 Experiment 4 - Effect of SPRT on 2D Operations

In [Ferri et al. (2010)](#) the objective was to collect high value observations and to localize maxima of the map. To determine the effectiveness of the SPRT algorithm we have simulated an experi-
ment where a rover is following a lawnmower pattern across a terrain that has pockets of water density that will drive different average rates of the Poisson process for a NSS sensor.

We took data from the NSS readings from the MVP field experiment and fit a distribution over the number of counts. We sampled 100 observations from this distribution and used them as seeds to generate a Voroni map, called $M$, representing a 50m by 50m area, with a grid size of 0.5m.

The map was then smoothed with a Gaussian blur with a scale factor of $\sigma = 0.8$ in order to mimic the effect of the field of view of the sensor. At each point $(x, y)$ in $M$ the robot is given a sample from a Poisson distribution with $\lambda = M(x, y)$. We generated 30 maps and ran ten trials on each map for each algorithm.

The robots follow a trajectory that follows a lawnmower pattern across the width of the map and with the swaths of the path having a spacing of 5m. The trajectory is illustrated in ??.

Because the objective of the experiment in Ferri et al. (2010) was to localize the source of chemicals we set the robot in the simulation the task of collecting high-value observations. High value observations were classified in two ways: Was an observation of 80 counts or higher observed and did the rover pass over a local maxima in the map it is driving over? To score the performance of the robot we counted the number of instantaneous observations that were above the threshold over the course of the transect. To score the performance on localizing maxima we count the number of maxima that fell within the robot’s field of view while driving. In both cases higher scores are better. A threshold of 80 counts was established in conversation with member of the MVP team.

Table 3.14 gives the performance of the different algorithms on each of the 30 maps. Table 3.15 summarizes how each algorithm performed on the maps, giving the number of times each algorithm placed first, second, or third out of the algorithms. It is clear to see that in terms of collecting the highest number of high-value observations the SPRT algorithm ranks first more than any of the other algorithms by a considerable margin.

When the SPRT algorithm is in first place the effect size improvement in all but three cases are large ($>0.8$). Maps 08 and 12 have medium effect sizes ($>0.5$) and map 21 has a small effect size ($<0.2$). Map 04 produces a tie between all three algorithms with no statistical significance between their performance and hence reporting an effect size is not meaningful.
Table 3.14: The average performance of the three different algorithms on each map. The column $\mathbb{E}[\mu_1 - \mu_2]$ gives the average difference between the first and second place algorithm. The 95% high density interval is based around the difference. If that interval does not contain 0 then we can say with 95% probability that the means are different. If they are different we report the effect size. Effect sizes over 0.8 are considered large. Effect size is measured with Cohen’s $d$.

<table>
<thead>
<tr>
<th>Map Number</th>
<th>No AIMS</th>
<th>Memory Threshold</th>
<th>SPRT</th>
<th>$\mathbb{E}[\mu_1 - \mu_2]$</th>
<th>$\mu_1 - \mu_2$ 95% HDI</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>44.7 (1)</td>
<td>44.7 (1)</td>
<td>29.6 (2)</td>
<td>15.1</td>
<td>[13.4, 16.9]</td>
<td>3.9</td>
</tr>
<tr>
<td>01</td>
<td>0 (2)</td>
<td>0 (2)</td>
<td>6.2 (1)</td>
<td>6.2</td>
<td>[4.47, 7.9]</td>
<td>1.75</td>
</tr>
<tr>
<td>02</td>
<td>27.6 (1)</td>
<td>16.3 (2)</td>
<td>13.1 (3)</td>
<td>11.3</td>
<td>[10.7, 11.9]</td>
<td>8.14</td>
</tr>
<tr>
<td>03</td>
<td>27.9 (3)</td>
<td>86.5 (1)</td>
<td>63.8 (2)</td>
<td>22.8</td>
<td>[16.9, 28.9]</td>
<td>1.74</td>
</tr>
<tr>
<td>04</td>
<td>18 (1)</td>
<td>16.7 (1)</td>
<td>18.2 (1)</td>
<td>1.51</td>
<td>[-0.155, 3.13]</td>
<td>N/A</td>
</tr>
<tr>
<td>05</td>
<td>2.21 (3)</td>
<td>8.53 (2)</td>
<td>21 (1)</td>
<td>12.5</td>
<td>[9.69, 15.2]</td>
<td>2.06</td>
</tr>
<tr>
<td>06</td>
<td>14.9 (1)</td>
<td>11 (2)</td>
<td>16.4 (1)</td>
<td>5.38</td>
<td>[3.69, 7.12]</td>
<td>1.42</td>
</tr>
<tr>
<td>07</td>
<td>16.3 (2)</td>
<td>21.5 (1)</td>
<td>14.6 (2)</td>
<td>5.17</td>
<td>[3.78, 6.53]</td>
<td>1.73</td>
</tr>
<tr>
<td>08</td>
<td>30.3 (3)</td>
<td>50.3 (2)</td>
<td>54.8 (1)</td>
<td>4.48</td>
<td>[1.31, 7.80]</td>
<td>0.64</td>
</tr>
<tr>
<td>09</td>
<td>12.4 (2)</td>
<td>11.3 (3)</td>
<td>22.1 (1)</td>
<td>10.8</td>
<td>[8.5, 13.1]</td>
<td>2.15</td>
</tr>
<tr>
<td>10</td>
<td>37.6 (3)</td>
<td>89 (1)</td>
<td>62.9 (2)</td>
<td>26.1</td>
<td>[20.3, 31.8]</td>
<td>2.57</td>
</tr>
<tr>
<td>11</td>
<td>9.4 (2)</td>
<td>6.1 (3)</td>
<td>12.9 (1)</td>
<td>3.5</td>
<td>[2.55, 4.45]</td>
<td>1.78</td>
</tr>
<tr>
<td>12</td>
<td>6 (2)</td>
<td>7.4 (1)</td>
<td>8.3 (1)</td>
<td>2.27</td>
<td>[0.68, 3.92]</td>
<td>0.63</td>
</tr>
<tr>
<td>13</td>
<td>0 (2)</td>
<td>0 (2)</td>
<td>10.4 (1)</td>
<td>10.4</td>
<td>[8.32, 12.5]</td>
<td>2.34</td>
</tr>
<tr>
<td>14</td>
<td>0 (2)</td>
<td>0 (2)</td>
<td>11.6 (1)</td>
<td>11.6</td>
<td>[9.63, 13.3]</td>
<td>4.69</td>
</tr>
<tr>
<td>15</td>
<td>14.1 (2)</td>
<td>14 (2)</td>
<td>18.2 (1)</td>
<td>4.17</td>
<td>[2.57, 5.71]</td>
<td>1.37</td>
</tr>
<tr>
<td>16</td>
<td>52.3 (3)</td>
<td>54.2 (2)</td>
<td>56.4 (1)</td>
<td>4.11</td>
<td>[2.26, 5.95]</td>
<td>1.21</td>
</tr>
<tr>
<td>17</td>
<td>8.6 (1)</td>
<td>6.22 (2)</td>
<td>8.15 (1)</td>
<td>2.37</td>
<td>[1.59, 3.14]</td>
<td>1.41</td>
</tr>
<tr>
<td>18</td>
<td>2.48 (3)</td>
<td>22.6 (2)</td>
<td>31.6 (1)</td>
<td>9.13</td>
<td>[5.36, 12.9]</td>
<td>1.12</td>
</tr>
<tr>
<td>19</td>
<td>17.6 (3)</td>
<td>22.9 (2)</td>
<td>43 (1)</td>
<td>20</td>
<td>[16.1, 23.8]</td>
<td>2.45</td>
</tr>
<tr>
<td>20</td>
<td>34.5 (2)</td>
<td>48.3 (1)</td>
<td>50 (1)</td>
<td>15.5</td>
<td>[12.2, 18.9]</td>
<td>2.25</td>
</tr>
<tr>
<td>21</td>
<td>13.9 (3)</td>
<td>16.9 (2)</td>
<td>18.3 (1)</td>
<td>1.4</td>
<td>[0.004, 2.75]</td>
<td>0.47</td>
</tr>
<tr>
<td>22</td>
<td>12.5 (1)</td>
<td>7.5 (2)</td>
<td>13.1 (1)</td>
<td>5.6</td>
<td>[4.6, 6.62]</td>
<td>3.33</td>
</tr>
<tr>
<td>23</td>
<td>0.2 (2)</td>
<td>0.2 (2)</td>
<td>6.23 (1)</td>
<td>6.2</td>
<td>[5.84, 6.67]</td>
<td>9.02</td>
</tr>
<tr>
<td>24</td>
<td>30.8 (1)</td>
<td>21.5 (2)</td>
<td>18.2 (3)</td>
<td>9.27</td>
<td>[7.92, 10.7]</td>
<td>2.94</td>
</tr>
<tr>
<td>25</td>
<td>40 (3)</td>
<td>47.2 (2)</td>
<td>54.5 (1)</td>
<td>7.22</td>
<td>[3.53, 10.8]</td>
<td>1.01</td>
</tr>
<tr>
<td>26</td>
<td>14.7 (2)</td>
<td>13.6 (2)</td>
<td>49 (1)</td>
<td>34.3</td>
<td>[30.5, 38.4]</td>
<td>4.63</td>
</tr>
<tr>
<td>27</td>
<td>9.2 (3)</td>
<td>21.6 (2)</td>
<td>35.4 (1)</td>
<td>13.8</td>
<td>[11.1, 16.3]</td>
<td>4.44</td>
</tr>
<tr>
<td>28</td>
<td>34.1 (2)</td>
<td>37.1 (1)</td>
<td>38.9 (1)</td>
<td>4.91</td>
<td>[0.27, 9.46]</td>
<td>1.11</td>
</tr>
<tr>
<td>29</td>
<td>0.58 (3)</td>
<td>4.96 (2)</td>
<td>13.3 (1)</td>
<td>8.34</td>
<td>[5.02, 11.6]</td>
<td>1.8</td>
</tr>
</tbody>
</table>
Table 3.15: Here we count the number of times each algorithm scored each rank on the 30 maps for the number of high-value observations collected. The SPRT algorithm is the clear winner in this case.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>No AIMs</td>
<td>7</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>Memory Threshold</td>
<td>8</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>SPRT</td>
<td>24</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

When it comes to capturing local maxima the SPRT algorithm is the clear winner, as shown in Table 3.16 and Table 3.17. In fact, the SPRT algorithm always defeats the Memory Threshold algorithm, and always with huge effect sizes. It is clear here that the additional knowledge that is made available by using the SPRT algorithm clearly lets the robot make better decision that leads to superior performance on the given task.
Table 3.16: The average number of peaks captured by the three different algorithms on each map. The column $\mathbb{E}[\mu_1 - \mu_2]$ gives the average difference between the first and second place algorithm. The 95% high density interval is based around the difference. If that interval does not contain 0 then we can say with 95% probability that the means are different. If they are different we report the effect size. Effect sizes over 0.8 are considered large. Effect size is measured with Cohen’s $d$.

<table>
<thead>
<tr>
<th>Map Number</th>
<th>No AIMS</th>
<th>Memory Threshold</th>
<th>SPRT</th>
<th>$\mathbb{E}[\mu_1 - \mu_2]$</th>
<th>$\mu_1 - \mu_2$</th>
<th>95% HDI</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>10 (3)</td>
<td>16.6 (2)</td>
<td>42.9 (1)</td>
<td>26.3</td>
<td>22.2, 30.4</td>
<td>5.98</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>11 (2)</td>
<td>11 (2)</td>
<td>44.1 (1)</td>
<td>33.1</td>
<td>29.9, 36.3</td>
<td>9.78</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>20 (3)</td>
<td>21 (2)</td>
<td>53.4 (1)</td>
<td>32.4</td>
<td>25.9, 38.7</td>
<td>5.10</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>13 (3)</td>
<td>16.6 (2)</td>
<td>46.2 (1)</td>
<td>29.6</td>
<td>25.3, 33.7</td>
<td>6.88</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>15 (3)</td>
<td>16.1 (2)</td>
<td>54.1 (1)</td>
<td>38.0</td>
<td>34.5, 41.4</td>
<td>10.7</td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>12 (2)</td>
<td>13.1 (2)</td>
<td>53.1 (1)</td>
<td>40.0</td>
<td>35.9, 44.3</td>
<td>9.34</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>19 (2)</td>
<td>19 (2)</td>
<td>57.3 (1)</td>
<td>38.3</td>
<td>36.0, 40.4</td>
<td>19.9</td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>16 (3)</td>
<td>17.2 (2)</td>
<td>50.2 (1)</td>
<td>33.0</td>
<td>29.0, 36.5</td>
<td>8.14</td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>12 (3)</td>
<td>15.9 (2)</td>
<td>48.3 (1)</td>
<td>32.4</td>
<td>29.0, 35.7</td>
<td>9.90</td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>17 (3)</td>
<td>18 (2)</td>
<td>51.3 (1)</td>
<td>33.3</td>
<td>30.6, 37.7</td>
<td>11.8</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12 (2)</td>
<td>12 (2)</td>
<td>51.1 (1)</td>
<td>38.8</td>
<td>33.8, 43.8</td>
<td>7.81</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>15 (2)</td>
<td>15 (2)</td>
<td>48.8 (1)</td>
<td>33.8</td>
<td>31.4, 36.1</td>
<td>15.3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>18 (3)</td>
<td>19.3 (2)</td>
<td>53.0 (1)</td>
<td>33.7</td>
<td>30.2, 37.2</td>
<td>9.57</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>19 (2)</td>
<td>19 (2)</td>
<td>51.9 (1)</td>
<td>32.9</td>
<td>30.4, 35.3</td>
<td>13.5</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>24 (2)</td>
<td>24 (2)</td>
<td>58.0 (1)</td>
<td>34.0</td>
<td>31.2, 36.7</td>
<td>13.3</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>17 (3)</td>
<td>19 (2)</td>
<td>53.2 (1)</td>
<td>34.2</td>
<td>28.3, 40.4</td>
<td>5.77</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>19 (3)</td>
<td>24.2 (2)</td>
<td>59.9 (1)</td>
<td>35.7</td>
<td>31.4, 40.3</td>
<td>8.12</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>16 (2)</td>
<td>16 (2)</td>
<td>54.4 (1)</td>
<td>38.4</td>
<td>34.9, 41.9</td>
<td>10.6</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>18 (3)</td>
<td>21.1 (2)</td>
<td>50.2 (1)</td>
<td>29.1</td>
<td>24.9, 33.5</td>
<td>6.61</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>10 (3)</td>
<td>12.3 (2)</td>
<td>46.9 (1)</td>
<td>34.6</td>
<td>30.3, 39.0</td>
<td>7.94</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>22 (3)</td>
<td>23.3 (2)</td>
<td>56.4 (1)</td>
<td>33.0</td>
<td>29.8, 36.4</td>
<td>10.4</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>20 (3)</td>
<td>21.6 (2)</td>
<td>51.7 (1)</td>
<td>30.0</td>
<td>27.0, 33.1</td>
<td>9.91</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>19 (3)</td>
<td>19.6 (2)</td>
<td>63.4 (1)</td>
<td>43.9</td>
<td>40.6, 47.2</td>
<td>12.6</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>16 (2)</td>
<td>16.3 (2)</td>
<td>51.4 (1)</td>
<td>35.1</td>
<td>32.0, 38.2</td>
<td>10.9</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>18 (3)</td>
<td>20.5 (2)</td>
<td>56.3 (1)</td>
<td>35.8</td>
<td>30.9, 40.7</td>
<td>6.97</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>13 (3)</td>
<td>15.8 (2)</td>
<td>42.4 (1)</td>
<td>26.4</td>
<td>24.6, 28.1</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>14 (3)</td>
<td>16.2 (2)</td>
<td>49.3 (1)</td>
<td>33.1</td>
<td>28.5, 37.5</td>
<td>7.31</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>19 (3)</td>
<td>20.6 (2)</td>
<td>53.0 (1)</td>
<td>32.5</td>
<td>28.5, 36.7</td>
<td>7.86</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>22 (3)</td>
<td>24 (2)</td>
<td>55.8 (1)</td>
<td>31.8</td>
<td>27.0, 36.4</td>
<td>6.62</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>12 (2)</td>
<td>12.8 (2)</td>
<td>52.9 (1)</td>
<td>40.0</td>
<td>36.5, 43.7</td>
<td>11.1</td>
<td></td>
</tr>
</tbody>
</table>

We also see that 33% of the time the Memory Threshold and the No AIM algorithm result in statistically indistinguishable performance.
Table 3.17: Here we count the number of times each algorithm scored each rank on the 30 maps for the number of local maxima observed. Again, the SPRT algorithm is the clear winner in this case.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
</tr>
</thead>
<tbody>
<tr>
<td>No AIMs</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Memory Threshold</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>SPRT</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The SPRT algorithm achieved the most number of first place ranks, three times as many as the Threshold algorithm. The number of first place ranks that the Threshold Algorithm achieved is not substantially different from that of the algorithm that does not deploy AIMs.

3.6 Extensions

The following extensions would be useful improvements on the preceding work in order to make the algorithm more useful:

1. Build a spatial model to improve decision making. Right now the algorithm does not consider explicit spatial relationships, incorporating that model could help identify trends that might lead to more intelligent performance of the algorithm.

2. It would be possible to extend the work to simultaneously learn the utility of the prospecting sensors, provided sufficient time and risk budget exist in the mission to engage in such learning.

3. Right now the lawnmower pattern used has a fixed width. It could be beneficial to adaptively change width of the lawnmower pattern in order to more tightly observe interesting regions of a map.

4. The model for determining when an AIM should be deployed is fairly simple. Given that a more conservative approach to deploying the discrete, expensive sampling action may be required it would be useful to make more intelligent use of the information determined by the proposed algorithm. Given the algorithm determines the underlying rate of the region it is currently traversing we could feed these rates into an infinite secretary problem, and hence make better decisions on when to deploy an AIM.

5. In the proposed algorithm all assumptions about risk are implicit in the confidence level set for the SPRT algorithm. An explicit modelling of mission cost and risk could be used to make the algorithm more amenable to Missions System Engineering groups.

6. A non-parametric approach to modelling the underlying distributions would help the algorithm be applicable when form of the underlying distribution is not known. However, this could cause the spatial requirements of the algorithm to balloon considerably.

7. In this chapter we only considered there being two underlying distributions, however it is possible to have multiple distribution changes during a time period. It is possible to represent multi-hypotheses sequential probability ratio tests, as per Baum and Veeravalli (1994), making for a logical extension of the above work.
8. The SPRT algorithm as implemented searches through every time step in the data that have been collected from the last detected change. This is a linear search in the number of timesteps, $T$, which in turn requires an order $T$ computation, yielding $O(T^2)$ complexity. However, we can reframe this problem as searching for the roots of the equation $\lambda(t) - a$, where $a$ is the confidence level threshold for choosing one hypothesis over the other. With this reframing we could employ a bisection search which would have a worst-case complexity of $O(T \log T)$.

### 3.7 Contributions

In this chapter I have developed an algorithm for automatically deploying discrete actions in response to changes in an underlying distribution. In this case the discrete action was an Area of Interest Mapping Maneouvre.

The algorithm was tested for sensitivity to the magnitude of change in the underlying distribution, as well as the delay in the onset of the change. Further it was tested on real data from the MVP project where it was able to detect changes in the signal. Finally the algorithm was tested on a simulated environment modelled on real-world scenarios and was found to be statistically significantly superior to the state-of-the-art algorithm is field.

The SPRT-based algorithm lets autonomous agents make decisions with confidence when the underlying distribution has changed. A degree of confidence is necessary for autonomous agents to make decision competently when operating in alien environments.

The major contribution of this chapter is the use of a sequential probability ratio test based algorithm to determine when a change in the underlying distribution has changed. What this means is that the algorithm considers all of the data that have been collected thus far, and does not need to have its threshold tuned to each individual environment, but to the mission’s tolerance for false positives. What the change in the distribution means is of course up to the mission.
Chapter 4

Global Planning for Hypothesis Falsification

The objective of the scientific method is to distinguish between true and false hypotheses. Failing that, to find the hypotheses that best explains the empirical truth. This chapter deals with coordinating the operations of a robot on the surface of a planetary body with the beliefs that remote scientists had about it before they arrived, possibly taking from remote sensors.

For natural scientists it is common to propose candidate hypotheses, design experiments that will disambiguate between these hypotheses

4.1 Objective

4.1.1 Problem Statement

An agent is given a number of competing hypotheses. Initially believing them equally probable, the agent is tasked with finding which hypothesis best explains the observed phenomena while making efficient use of resources.

4.1.2 Hypothesis

The hypothesis of this experiment is that given the same budgets, a sampling algorithm that is aware of the hypotheses it is trying to chose between will be more productive

4.2 Preliminaries

At its core, the objective of the scientific process is the selection of the correct model for explaining data. This speaks back to Ramsey’s definition of the truth, as discussed in Chapter ???. Since, due to sensor noise and other sources of uncertainty, one does not anticipate that any sequence of data will be perfectly explained by any one model, one is left in the position of finding the model that does the best job of explaining the data one has at hand.
It is not uncommon to have data represented as some set of independent variables $X$ and dependent variables $Y$. The hypothesis one is often tasked with learning is $H(\cdot)$, where $Y = H(X)$ is “true” (hereafter used without scare quotes), for acceptable values of truth. However in many experiments one is given two or more hypotheses and one wants to determine which of these is the most probable, given a data set.

Note that this is different from the exploration-exploitation problem. In the exploration-exploitation problem one is simultaneously trying to maximize reward while determining which action has the best reward payout with some degree of certainty. In the case of model selection one is simply trying to explain the data. Like information foraging in Chapter 2, the reward is pure exploration – in this case information about which hypothesis is most likely.

There are a number of different types of models that can be used to explain data: Polynomials, regression models, gaussian processes, K-nearest neighbour models, artificial neural networks are all valid ways of representing relations between data. Some are more or less attractive depending on the prior knowledge of the problem state going into the experiment. However, generative models, in which the hypothesis $H(X) = P(Y)$ allows for a much more internally consistent way of directing exploration. More to the point it gives us data that can be used to update our belief about which model is most likely true.

**TODO: get citations** If one is given two collections of data, and one wants to choose between the hypothesis that an intervention (or change in environment condition, or some other factor) resulted in a change in the distribution, although not necessarily the family it came from, then one can look at the Bayes Factor (BF), defined more precisely in Equation (4.1). The Bayes Factor is a way of comparing the relative probability of two models. Likewise if one has one set of data and two probabilistic models explaining it, the bayes factor is a way of representing the relative probabilities of the two models, given the data.

\[
BF = \frac{P(M_1|D)}{P(M_2|D)}
\]  

(4.1)

However Bayes Factor alone does not take into account the complexity of the model. A fear of experimenters is that an overly complex model will overfit the data, leading to poor generalization to data points that have not yet been observed. Model metrics such as the Bayes (or Schwartz) Information Criterion (BIC) or the Akaike Information Criterion (AIC) have been developed to compare models based on not only their explanatory power, but also their relative complexity. The matter of model complexity and overfitting will be addressed in the following chapter.

In the case presented in this chapter we are interested in conducting experiments that maximally disambiguate between the more and less explanatory hypotheses. This is different from the standard multi-armed bandit approach, a technique designed by Robbins for selecting experiments sequentially. In the multiarmed bandits one would have to pick one arm and sample from it,

1. Thompson sampling/bayes optimal control, an empirical validation of thompson sampling.
2. Lindley’s value of experiment/mutual information.
3. Bayesian Experiment Design in a nutshell.
4. Bayes Factor and Bayesian information criterion and hangers on. They are variations on the likelihood function and some cost over the complexity of the algorithm. In this experiment we ignore the complexity of the algorithm.

5. active agnostic learning

What we propose is a greedy algorithm for selecting sampling points that promise to be the most productively informative over which of the competing hypotheses is true.

### 4.2.1 Model Selection

1. There are many different kinds of models (polynomials)
2. probabilistic models allow us to do Bayesian Experiment design
3. There is also the matter of selecting the most parsimonious model, but that is not for this section.
4. Non-parametric models can be seen as a type of model selection, but they are reducing an infinite number of models. Could also consider the kernels as the models, to be discussed in the next section.
5. At the end of the day we want something that accurately predicts the data, and performs well on data it has not seen yet.

### 4.3 Experiment

A robot is given competing hypotheses, and it needs to select which one is the correct (or most correct) model of terrain class distributions across the surface of a hypothetical planetary body. Here we assume that different scientists have different techniques for transforming remote sensing data into geologic classifications. However not all of these methods, representing hypotheses about what the remote sensing data means, are of the same quality, nor are they guaranteed to not have systematic biases (e.g. confusing two geologic classes for each other).

For this experiment we imagine an agent capable of transitioning between different areas on a map, where it can collect observations. The remote observations are corrupted with some noise. This is to be expected as information is imperfect. We present a hypothetical planetary robot exploring a hypothetical planetary body. We assume that the vehicle can have random access to the surface of the planetary body where it collects samples. Remote sensing vehicles such as satellites have this ability, likewise hopping vehicles have been proposed in the past [Kaplan and Seifert (1969)] for low-gravity bodies such as the moon, and as the capabilities of unmanned aerial vehicles (e.g. multi-copters, airships) increases, being able to sample without having to negotiate the 3d terrain that poses a threat to ground vehicles becomes increasingly plausible. Likewise, underwater vehicles such as ABE from Woods Hole Oceanographic Institute can operate at different levels, increasing elevation before travelling to the next data point TODO: fact check.

The robot on the surface is capable of classifying the terrain it sees into the same geologic classes that the remote scientists classified the orbital map into. The rover is given a budget that
is multiples of the distance to cover the map once, in uniform increments. The cost that the vehicle computes is the euclidean distance between the sample points. As the rover acquires new sample points it expends its travel budget and acquires information, helping it to conclude which hypothesis is the correct one.

Such a sample could be a texture classifier like the TextureCam developed at JPL [Thompson et al. (2012)], or it could be neural network algorithm like the kind being developed for different image classification schemes ?, or it could be a technique like the unsupervised terrain classifier developed at University of Sidney (Steinberg et al. 2010), or it could be some latent topic modelling approach like those used at McGill University [Girdhar et al. (2013a)]. The actual implementation is not important in this instance. What is important is that the sensor model classifies some sensor data into the same categories that the remote scientists reason about, and that it has some characterized error rate.

4.3.1 Condition A: Noisy Sensor

In this condition the robot sensor always has the probability of correctly identifying the sensor as 0.9. Conversely the probability of incorrectly labelling a terrain class $i$ is $P(i, j) = 0.1/(N − 1)$, where $N$ is the number of classes in the problem.

4.3.2 Condition B: Sensor with Fault

In this condition the robot sensor is given one class of terrain that it systematically misclassifies. It is important to note that the more populous a class of terrain is, the more vulnerable the sensor is to this fault. In this case while $P(i, i) = 0.9$ it is also the case that $P(j, i) = 0.9$

TODO: Try and do some manner of analysis on this. Probability of a bad hypothesis being upweighted? May be somewhat involved.

In this case, see how many samples are chosen from points that are labelled $j$.

4.3.3 Map Generation

Maps represent the major form of hypothesis model in this experiment. We first generate true maps that give the actual underlying spatial distribution of geologic classes on the planetary body the robot is investigating. The maps are generated from a Voronoi model. Specifically given a map of 100x100 pixels, $M$ seed points are randomly selected and assigned a terrain class. After these seed points are generated each point in the map is then assigned the label of the closest seed point.

It should be noted that this is an artificial method for producing these maps. However it does generate sharp boundaries like those observed in actual material distributions in geologic environments. Again it should be stressed that this approach is about accounting for the hypotheses and the resources required to access sampling points to support those hypotheses.
Scientist Model

The hypothetical scientists in our model are represented by taking the key nodes of the Voronoi map generation and incorrectly labelling terrain class \( i \) as terrain class \( j \) with the probability \( P(i, j) \). The scientists are given different levels of accuracy, that is to say that scientist \( k \) has the classification matrix \( P(i, i) = p_k \), where \( p_k \geq 0.5 \), and \( \sum_{i \neq j} P(i, j) = 1 - p_k \). The classification distributions of the different scientists are given in the supplementary materials.

4.3.4 Algorithms

The premise of this work is that accounting for the hypotheses under investigation as well as the resources used to investigate those hypotheses will result in an efficient detection of the correct hypotheses. To determine that this is in fact the case we employ algorithms that account for neither the hypotheses or the resources (Random Site Selection), one that accounts for resources but not the hypotheses under investigation (Uniform Sampling), one that accounts for the hypotheses but not the resources (Random where Different), and finally an algorithm that accounts for both the resources as well as the hypotheses under investigation (Maximal Disambiguation Sampling).

The algorithms follow the same general pattern: Select point(s) in the map to investigate. Acquire an observation at that point, and after every observation update the belief in the different hypotheses. The belief update step is a very simple Bayesian belief update, simply \( P(H_k) = \) 

Random Sampling

The Random Sampling algorithm randomly selects points on the map with a uniform distribution over the \( x, y \)-coordinates of the map without consideration for the distances between points, or the distance from the current point to the next. The purpose of this algorithm is to provide a control algorithm that does not account for the hypotheses under investigation or the costs of travelling between sampling points. Consequently this algorithm may backtrack, or may move to a point that has a very similar set of labellings, given the collection of hypotheses.

Algorithm 4.1 Select random sample points

\textit{selectpoints}

Does not account for hypothesis or for resources. Pick some random points Find the minimum path between them

Uniform Sampling

The purpose of the Uniform Sampling algorithm

Pick uniformly spaced points and sample them.
Algorithm 4.2 Select random sample points

\[ selectpoints \]

**Select Where Different**

This algorithm takes all of the \( N \) hypotheses proposed and determines any and all points where they are not the same. The intuition here is that it is only the points where they hypotheses disagree that informative observations can be made, assuming that it is not the robots’ sensor that is under investigation. However, this algorithm does not account for the resources necessary to acquire these samples. Consequently it samples from the interesting points randomly until it has exhausted its resources.

This algorithm needs to be the one below but that only uses informativeness and not productivity.

Algorithm 4.3 Select random sample points

\[ selectpoints \]

**Maximal Disambiguation Sampling**

Looks at points where the samples disagree, and samples from them iteratively in a way that maximizes the change in entropy.

Assumes one hypothesis is true. Looks at where other hypotheses are different, and samples randomly from them. Determines which ones would be the most informative.

The maximal disambiguation sampling algorithm asks the question: Given a sensor noise model, and assuming a particular hypothesis were true, which point in the hypothesis input space (\( x, y \)-coordinates) would most concentrate the entropy of the belief distribution over the competing hypotheses. Initially we apply a uniform prior over the different hypotheses, which is sequentially refined after each observation. To save computation time we only consider the points where the competing hypotheses differ in their predictions, otherwise the samples would not be informative. However, if the problem were to build an accurate hypothesis of the geological distributions then it would not be enough to believing the candidate hypotheses where they agree – those predictions must also be tested. Falsifying the individual hypotheses is a subject for the next chapter, however.

The main computation, which observation point is most informative, must be recomputed at every step, like wise the cost of accessing it. To save computation time locations that have been previously visited will be excluded, likewise locations where the hypotheses agree. Like Thompson sampling, at each step one of the competing hypotheses is randomly selected to be the candidate true hypotheses, and the sampled points are selected from that map. Assuming that those observations are what would be identified by the robot, and given the robot’s sensor model, one can determine how much the entropy of the belief distribution would be decreased if those observations were to be made.

It is possible that should an unlikely distribution be selected, then an observation from that site, could increase the belief in that unlikely observation. This is not viewed as a downfall of
the algorithm, as, assuming the sensor is sufficiently reliable, that erroneous prediction will be falsified. In the case that a false hypotheses and the true hypotheses agree at this point, then its credibility is rightly increased, and other competing hypotheses will have their credibility reduced.

Only sample from the points that disagree with the currently selected hypotheses.

TODO: illustrate this with a 1d graphic showing these things changing over time. Three competing hypotheses.

This approach of randomly selecting a hypotheses to be true comes from the probability matching techniques of Thompson sampling. By assuming one hypothesis to be true at any given step in the calculations, the computational complexity of the algorithm is substantially reduced. At each point the algorithm chooses the single most productive point to sample. This is then updated after each sampling step.

Algorithm 4.4 Select random sample points

The belief in the correct hypothesis is modelled as a multi-nominal distribution, where the probability of any one hypothesis (the categories in this categorical distribution) is the probability of that hypothesis being the correct one. I place a Dirichlet prior on the belief in the hypotheses, which are updated based on observations from the robot on the ground. If there is no sensor model then the result of the observation is 0 if a hypothesis disagrees with the sensor or 1 if it agrees.

However, in many settings this is not a tenable situation, so we model the sensor with a simple confusion matrix over the observation. The result after an observation then is the updated belief in the prior maps based on the sensor reading, \( P(M_i|S_t) \) where \( M_i \) is the \( i \)-th hypothesis and \( S_t \) is the \( t \)-th observation.

4.4 Results

1. Condition A: Plot of entropy of belief across hypothesis as a function of resources spent.
2. Condition B: Plot of entropy of belief across hypothesis as a function of resources spent.
3. Condition A: Plot of probability of selecting correct model once the resources are spent as a function of resource amount.
4. Condition B: Plot of probability of selecting correct model once the resources are spent as a function of resource amount.

4.5 Extensions

1. Determining when to take multiple samples so one is confident that the observation is correct. This assumes having a properly calibrated remote sensor.
Table 4.1: The probability of the different algorithms selecting the best hypothesis, averaged over all resource budgets

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Hypothesis Aware</th>
<th>Resource Aware</th>
<th>Probability Correct</th>
<th>Standard Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>No</td>
<td>No</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Uniform</td>
<td>No</td>
<td>Yes</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Differ</td>
<td>Yes</td>
<td>No</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>MDS</td>
<td>Yes</td>
<td>Yes</td>
<td>TBD</td>
<td>TBD</td>
</tr>
</tbody>
</table>


3. Using something other than a categorical sensor.

4.6 Contribution

This chapter introduces a novel planning mechanism for selecting actions that maximally inform about hypotheses.

4.7 Introduction

Robots are sent to explore other planetary bodies to determine their material properties. These robots collect evidence to support hypotheses that have been developed from precursor (remote) sensing. It is unlikely that scientists have produced only one hypothesis explaining the preliminary data. Consequently robots exploring alien worlds are tools to determine which of a number of hypotheses best explain data observed on the surface. In this paper we present a novel algorithm for autonomous planning scientific observations that is explicitly aware of the hypotheses the robot is investigating.

The key to scientific exploration is the falsification of hypotheses – seeking out where hypotheses disagree and experimenting at those points. Where hypotheses’ predictions agree they are equally correct or equally wrong. Identifying and testing hypotheses’ conflicts reveal the meaning of their internal structure and how they relate to the observable universe. Fortunately we can effectively examine competing hypotheses by determining where their predictions differ and testing at those points. In this paper we represent hypotheses as two dimensional maps, but any map from an input space to an observation fits into our framework.

To illustrate the utility of our approach we simulate a planetary mission of a robot that is determining which of a number of geological classifications most accurately predict data observed on the ground. Figure 4.1 is a map of Jupiter’s moon Io, an example of the kinds of classifications that can be produced as the result of hypothesis generated from remote sensing data. Precursor data can admit multiple explanations, implying different hypotheses about the surface. Robot observations will help resolve which of these hypotheses are correct.

In our hypothetical mission our robot is capable of moving to arbitrary locations on a map. Such a vehicle would be appropriate for a mission like exploring the Aitken basin on the south
pole of Earth’s moon. The South Pole-Aitken Basin is 2500km in diameter and as such sampling the structure requires traversing long distances. A lander vehicle capable of hopping to different locations would be ideal for this mission, and it does not have the same path planning constraints that rovers have. Our algorithm does not engage in sophisticated trajectory planning, but it could use a planner to produce informative paths for other vehicle types.

Our proposed algorithm directs robots to collect observations that quickly determine which of a set of candidate hypotheses are most likely to explain data collected in situ. The algorithm uses an information theoretic cost function that trades off between increasing the certainty in one or more of the candidate hypotheses and ensuring samples are distributed appropriately across the map.

The algorithm differs from previous science autonomy algorithms in that instead of attempting to build a model of collected data this algorithm trades off between multiple hypothesis and can determine which one is the least false. Our algorithm does not attempt to produce its own hypothesis from the observations it collects but nothing stops it from doing so.

The novel contribution of this paper is the idea of hypothesis aware science autonomy. The algorithm permits robots to plan with respect to the hypotheses that are under investigation by the overall mission. Previous approaches which look at collecting representative datasets do not explicitly prioritize disagreements between competing hypotheses. The approach described in this paper does precisely that, and thereby makes more effective use of resources from the outset.
4.8 Background

Science autonomy robots have not previously needed to choose between hypotheses, they have generally been focused on collecting data without prior knowledge. However, the design of experiments literature has been largely focused on selecting experiments that inform hypotheses. A popular example of an experiment selection technique is the multi-armed bandit. Likewise, model selection is a well studied field, but will not be reviewed here.

Multi-armed bandits (MABs) Robbins (1952) are a formalization for sequentially selecting the most rewarding of a set of experiments. There are a number of different algorithms for attacking the MAB problem. The main families of which appear to Upper Confidence Bound algorithms Lai and Robbins (1985), The Gittins Index Gittins et al. (2011), and Thompson Sampling Thompson (1933). Thompson sampling has recently gained attention for being simple to implement and for having near-optimal regret properties Chapelle and Li (2011); Agrawal and Goyal (2012); Ortega and Braun (2010).

Standard MAB settings assume the interaction with the candidate arms is simple. An arm is selected for testing, the arm is pulled, and a reward is delivered. In planetary exploration one can map candidate hypotheses to bandit arms, but executing tests is complex. Robots have to move through space and interact with the world, incurring unpredictable costs. Because targets of interest, like geologic materials, have highly spatially dependent distributions, sampling a random point is not guaranteed to be an informative action.

We look to science autonomy algorithms to determine how to select sample points. Most noteworthy of these algorithms is the OASIS/AEGIS Castano et al. (2007); Estlin et al. (2012) system, as it has actually been deployed on Mars. However this system, like Paar et al. (2012); Chien et al. (2005); Thompson (2008), either learn models from no prior data or use proxy measures of importance that encode scientists’ preference, but are unaware of the hypotheses under examination. Given that access to data from precursor remote sensing is likely, starting each new scientific mission from a blank slate seems inefficient.

Girdhar et al. Girdhar et al. (2014)’s approach to exploration focusses on visiting locations that help construct a topic model for classifying satellite imagery. This approach is improves the topic model the robot is constructing but it does not inform any hypotheses that scientists are attempting to resolve.

Thompson and Wettergreen Thompson (2008) maximize diversity of collected samples with mutual information sampling. This approach ensures samples are distributed through the input space of the function it is learning. A shortcoming of this approach, and all approaches that rely on mutual information sampling in Gaussian Processes with stationary kernel functions is that they do not depend on the observations collected. For a GP \( P(f(x)|x_1, \ldots, x_t) \) the mutual information for a new data point, \( x_{t+1} \), is proportional to:

\[
I(x_{t+1}) \propto k(x_{t+1}, x_{t+1}) - k_t(x_{t+1})^T K_t^{-1} k_t(x_{t+1})
\]

where \( k \) is the kernel function. \( k_t \) is the vector of distances between the \( t \) collected data points and the test point \( x \), and \( K_t \) is the \( t \times t \) matrix of the distances between all previously collected observations.

Mutual information sampling depends only on the points in the input space have been ob-
served and the kernel function. The sequence of sampling points selected is deterministic once
the initial point and the kernel function are known. While this is a principled approach, being
aware of the hypothesis tested would require accounting for observations of $f(x)$.

The algorithm in [Thompson et al. (2015)] is aware of observations made on the ground. Its
robot attempts to explain spectra data from satellites by collecting a library of spectra from the
ground. Each pixel of satellite data is explained as a mixture of the endmember spectra collected
by the rover. The endmembers represent a basis of examples of “pure” minerals. The rover
travels to locations where library of rover-observed spectra poorly explains the satellite data and
collects more observations to explain the satellite data. This approach is implicitly constructing
a hypothesis about the terrain composition but the algorithm itself does not test points in the
satellite data that it considers well explained, nor does it consider alternate hypotheses.

[Miller et al. (2016)] uses the expected value of the Fisher Information to determine points
of interest. Like mutual information [Lindley (1956)], Fisher information is used as a score to
select the most informative experiments. Their path planner produces smooth paths maximize
the number of high information value observations. Fisher information and the mutual infor-
mation are intimately related, but not identical, quantities. However, a rigorous comparison of
the behaviour of robots maximizing mutual information and those maximizing expected Fisher
information does not exist in the literature. Without such evidence it is difficult to select one
reward function over the other and this should be studied further.

Distributing points throughout the input space is a principled approach to learning a function
de novo. However, when trying to select between competing hypotheses it is conceivable that
a sequence of observation points $x_1, \ldots, x_t$ will miss the points of conflict between competing
hypotheses, especially with limited sampling resources. However, widely sampling an input
space is important to ensure the predictions of a hypothesis are generally accurate. Most deployed
science autonomy algorithms do not operate in the realm of hypotheses, they either focus on
distributions of data or on hand-coded proxy measures, not necessarily the support or falsification
of hypotheses under consideration. For that reason we propose the method described in Section
4.9.

4.9 Method

The objective of the mission is to build belief in the hypothesis that is least false, $H^*$. In order
to do this the robots must select locations, $l \in L$, in the input space of the hypotheses that most
productively inform the investigation. The robots collect true observations, $M(l)$, from the true
map, $M$, that can be used to update the belief in the different hypotheses based on the prediction
$H_i(l)$ from the corresponding hypothesis $H_i$. The hypotheses the robots were attempting to
validate were generated by corrupting generated map data. This is a stand-in for hypotheses that
have different interpretations of the precursor data.

4.9.1 Belief in Hypotheses

The algorithms estimates their belief that hypothesis $H_i$, where $i \in 1, \ldots, K$, is the least false
with a multinomial distribution, $P(H_i)$, where $\sum_{i=0}^{K} P(H_i) = 1$. We have a special place-holder
hypothesis $H_0$, which represents the probability that none of the proposed hypotheses are correct.

We place a Dirichlet prior on the distribution of belief in the hypotheses, with corresponding condensation parameters $\alpha_0, \ldots, \alpha_K$. The condensation parameters are initialized with an uninformative prior, $\alpha_i = 1$, as initially all hypotheses are considered equally likely. However, the condensation parameters can be initialized to reflect any prior belief in the competing hypotheses.

After an observation the condensation parameters are updated to reflect the agreement with the predictions from the different hypotheses. That is to say:

$$\alpha_{i,t+1} = \alpha_{i,t} + P(H(l(t)) = M(l(t))|H_i)$$

(4.3)

For simplicity we have assumed no uncertainty in the hypothesis predictions or the in situ observations, so the update can be rewritten as $\alpha_{i,t+1} = \alpha_{i,t} + 1(H_i(l_t) = M(l_t))$, but the algorithm can incorporate uncertainty easily.

### 4.9.2 Site Selection Algorithm

Algorithm 4.5 selects sampling sites with a modified version of Thompson sampling. At each time step, $t$, the algorithm samples a belief state $\tilde{\theta}$ from the Dirichlet prior describe in Section 4.9.1 and then chooses the $H_i$ that has the largest belief $\theta_i$. The algorithm assumes that $H_i$ is “true” for the duration of this step, and uses it to evaluate which location $l_t$ is most informative.

At the location $l_t$ the robots collect the observation $M(l_t)$. Observations are used to update the belief in the hypotheses that have been assigned to them.

We assume the robot starts in the $(0, 0)$ position on the map, which corresponds to the top left-hand location of the maps shown in Figure 4.2. The robot is capable of sampling at locations $l$ in the input space $L$, which in this experiment are $(x, y)$ locations on a two-dimensional map with integer coordinates. A Robot at location $l_t$ that chooses to travel to $l_{t+1}$ incurs a cost of $\text{cost}(l_t, l_{t+1})$. We assume that $\text{cost}(l, l) = 0$ and $\text{cost}(l_1, l_2) > 0$ when $l_1 \neq l_2$.

Once a location has been selected, the robot travels to the specified location and collects an observation from the actual map, $M(l_t)$, and the belief state is updated. Should none of the hypotheses agree with the prediction of the map at $l_t$, the place-holder condensation parameter $\alpha_0$ is incremented, increasing the belief that all hypotheses are equally false.

The reward function for candidate sampling locations is how the competing algorithms control the vehicle behaviour. The reward functions for the control algorithm and our proposed algorithm are specified in Sections 4.9.3 and 4.9.4.

### 4.9.3 Control Algorithm - Spatial Sampling

For the control algorithm we build a density estimator using a Gaussian kernel function and we select the candidate point which the increase in information would be the greatest. We compute the entropy, $\mathbb{H}(\cdot)$ of the kernel density estimator as described in Beirlant et al. (1997). The algorithm for the reward function is given in Algorithm 4.6.
Algorithm 4.5 Site Selection Algorithm

function SAMPLE-SELECTION($M, \langle H_1, \ldots, H_K \rangle, budget$)
    $\alpha_i \leftarrow 1 \forall i \in 0, \ldots, K$
    $t \leftarrow 0$
    $l_0 \leftarrow \langle (0, 0) \rangle$
    repeat
        $\bar{\theta} \sim \text{Dirichlet}(\alpha_0, \alpha_1, \ldots, \alpha_K)$
        $i \leftarrow \arg\!\max\!_i \theta_i$
        $l_t \leftarrow \arg\!\max\!_l \text{reward}(\mathbb{E}[\bar{\theta}], H_i, \langle l_0, \ldots, l_{t-1} \rangle)$
        for $i \leftarrow 1, K$ do
            $\alpha_i \leftarrow \alpha_i + 1(M(l_t) = H_i(l_t))$
        end for
        if $\sum^K_i 1(M(l_t) = H_i(l_t)) = 0$ then
            $\alpha_0 \leftarrow 1$
        end if
        $budget \leftarrow budget - \text{cost}(l_t)$
        $t \leftarrow t + 1$
    until budget = 0
end function

Algorithm 4.6 Spatial Mutual Information sampling. This algorithm uses a measure of mutual information like the one used in [Thompson, 2008]. It depends only on previously observed locations to find the most informative point.

function SPATIAL-REWARD($P(\bar{\theta}), H_i, \langle l_0, \ldots, l_{t-1} \rangle, l_t$)
    $P_t(l) \leftarrow \text{KDE}(\langle l_0, \ldots, l_{t-1} \rangle)$
    return $\mathbb{H}(P_t) - \mathbb{H}(P_t|l_t)$
end function
4.9.4 Proposed Algorithm - Hypothesis Falsification

This algorithm seeks sampling locations that concentrate belief across the hypotheses given observation \( H_i(l_t) \), as seen in Equation 4.4, where \( H_i \) is considered the “true” hypothesis as per Algorithm 4.5. These locations maximize the mutual information between the belief state \( \vec{\theta} \) and the observation \( H_i(l_t) \).

\[
I(\vec{\theta}; H_i(l_t)) = \mathbb{H}[P(\vec{\theta})] - \mathbb{H}[P(\vec{\theta}|H_i(l_t))] \tag{4.4}
\]

This will automatically seek out locations where the competing \( H_i \)’s disagree. With Equation 4.4 as the reward function at any point \( l \) where the \( H_i \)’s agree will increase the credibility of all hypotheses, resulting in negative mutual information for \( l_t \).

However, one also wants to ensure that the hypotheses are accurate. Sampling at disagreement points will help build the credibility of the best hypothesis, but it does not give confidence with the overall predictions of \( H_i \). To mitigate this problem we add to Equation 4.4 the result computed in Algorithm 4.6 to ensure the sampled points spatially diverse. The reward function used for this algorithm is given in Algorithm 4.7.

We can derive this algorithm as follows:
\[
\mathbb{H}(\theta_i, L) = - \sum_{i} \sum_{l \in L} p(\theta_i, l) \log (p(\theta_i, l)) \\
= - \sum_{i} \sum_{l \in L} p(\theta_i|l)p(l) \log (p(\theta_i|l)p(l)) \\
= - \sum_{i} \sum_{l \in L} p(\theta_i|l)p(l) \log (p(\theta_i|l)) + \log (p(l))
\]

\[
= - \sum_{i} \sum_{l \in L} [p(\theta_i|l)p(l) \log p(\theta_i|l)] - \sum_{i} \sum_{l \in L} [p(\theta_i|l)p(l) \log p(l)]
\]

\[
= - \sum_{i} \sum_{l \in L} [p(\theta_i|l)p(l) \log p(\theta_i|l)] - \sum_{i} \sum_{l \in L} [p(l) \log p(l)]
\]

\[
= - \sum_{i} \sum_{l \in L} [p(\theta_i|l)p(l) \log p(\theta_i|l)] + \mathbb{H}(L)
\]

\[
= - \sum_{i} \sum_{l \in L} [p(\theta_i|l)p(l) \log p(\theta_i|l)] + \mathbb{H}(L)
\]

\[
= \mathbb{E}_L \left[ - \sum_{i} p(\theta_i|l) \log p(\theta_i|l) \right] + \mathbb{H}(L)
\]

\[
= \mathbb{E}_L \left[ \mathbb{H}(\theta_i|l) \right] + \mathbb{H}(L)
\]

Allowing us to conclude that

\[
\mathbb{H}(\theta_i, L) = \mathbb{E}_L \left[ \mathbb{H}(\theta_i|L) \right] + \mathbb{H}(L) \tag{4.5}
\]

The expectation term of \textbf{Equation 4.5} can be difficult to compute as the space of \( L \) increases. However, since mutual information is the difference before and after observation the computation can be simplified.

\[
MI = \mathbb{H}(\theta_i, L) - \mathbb{H}(\theta_i, L|z(l_t), \xi = l_t) \tag{4.6}
\]

Substituting the derived value for entropy from \textbf{Equation 4.5} into \textbf{Equation 4.6} we can write
MI = \mathbb{E}_L [\mathcal{H} (\theta_i|L)] - \mathbb{E}_L [\mathcal{H} (\theta_i|L, z(l_t), \xi = l_t)] + \mathcal{H} (L) - \mathcal{H} (L|z(l_t), \xi = l_t) \quad (4.7)

Recognizing that \mathcal{H} (L) - \mathcal{H} (L|z(l_t), \xi = l_t) is the information gained by sampling at a location \( l_t \) given the prior belief state. The term \( \mathbb{E}_L [\mathcal{H} (\theta_i|L)] - \mathbb{E}_L [\mathcal{H} (\theta_i|L, z(l_t), \xi = l_t)] \) permits some opportunities to for simplification which all hinge on how much of the hypothesis an observation effects. Those options are:

**Global Effect** In this case we consider that every observation contributes to the average belief in the hypotheses. In this case we do not propagate belief over the input space but simply accumulate evidence to give credibility to the hypotheses.

**Naive Assumption** In this case we believe that the belief in the hypotheses at the location \( l_t \) is independent at all other locations. Here we simply need to compute how the belief at this location changes. This is an unsatisfying choice as it cannot accumulate the evidence from all other previous beliefs and simply gives credit for observing locations that have not been observed before.

**Neighbourhood Effects** If there is a relationship over the input space \( L \), some manner of smoothness in the belief state over \( L \) then if one models it with a limited range effect (e.g. a Gaussian Process using a covariance kernel with finite support)

**Non-local Effects** An observation at any location \( l_t \in L \) effects the belief state at all other locations \( l' \in L \).

**Algorithm 4.7** Hypothesis Falsification sampling. We use Eq. \[4.4\] to increase the value of spatially diverse points

```plaintext
function DISAGREE-REWARD\( (P(\bar{\theta}), H_i, (l_0, \ldots, l_{t-1}), l_t) \)
  \( P(\bar{\theta}|H_i(l_t)) \leftarrow \) update\( (P(\bar{\theta}), H_i(l_t)) \)
  \( r_h \leftarrow \mathbb{H} \left[ P(\bar{\theta}) \right] - \mathbb{H} \left[ P(\bar{\theta}|H_i(l_t)) \right] \)
  \( r_s \leftarrow \) spatial-reward\( (P(\bar{\theta}), H_i, (l_0, \ldots, l_{t-1}), l_t) \)
  return \( r_h + r_s \)
end function
```

### 4.9.5 Map Generation

We generated 10 different maps, each \( 20 \times 20 \) pixels, with each pixel containing a label from \( z \in 1, \ldots, N \). We produce the ground truth maps by selecting 20 seed locations and randomly assigning them a label \( z \) with uniform probably (denoted by the function \( U(\cdot) \)), over the \( N \) labels. We then use a Voronoi map generation algorithm, given in Algorithm 4.8. Example maps are shown in Figure 4.2a.

To generate the hypotheses maps we take the same 20 seed points used to generate a true map and mis-label the seed locations with probability \( P(z = i|z = j) = \epsilon/(N - 1) \) for all \( i \neq j, i, j \in 1, \ldots, N \). Using the corrupted seeds we then generate maps using the Algorithm
Algorithm 4.8 Map Generation Algorithm

```python
function GENERATE-MAP(numSeeds, numPixels, numLabels)
    seeds ← \{\}
    map ← zeros(numPixels, numPixels)
    for i ∈ 1, ..., numSeeds do
        x_i ∼ U(numPixels)
        y_i ∼ U(numPixels)
        label ∼ U(numLabels)
        seeds ← seeds + ⟨(x_i, y_i, label)⟩.
    end for
    for x ∈ 1, ..., numPixels do
        for y ∈ 1, ..., numPixels do
            map(x, y) ← closest(seeds, x, y).label
        end for
    end for
    return map
end function
```

(a) The True map for $M_0$.
(b) $H_1$, 93.25% similarity with $M_0$, $\epsilon = 0.1$.
(c) $H_4$, 49% similarity with $M_0$, $\epsilon = 0.5$.
(d) The $H_{10}$, 7.25% similarity with $M_0$, $\epsilon = 0.9$.

Figure 4.2: Examples of a map and the hypotheses generated that anticipate the map.

4.9.6 Experiments

We compared the performance of our proposed algorithm and of the control algorithm in three experiments. For each experiment we tested ten different maps, and each map was repeated twenty-four times. The true maps and the hypothesis maps were generated as described in Section 4.9.5. The similarity between the true map and the hypotheses are given in Table 4.2. When a robot travelled to a point on the map it was informed of the true classification of the point with no error. All code and data used in these experiments are available online at [omitted for]
Table 4.2: This table gives the similarity between the true maps $M_i$ and the hypotheses proposed by the simulated scientists $H_j$.

<table>
<thead>
<tr>
<th></th>
<th>$H_1$</th>
<th>$H_2$</th>
<th>$H_3$</th>
<th>$H_4$</th>
<th>$H_5$</th>
<th>$H_6$</th>
<th>$H_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>0.93</td>
<td>0.82</td>
<td>0.66</td>
<td>0.49</td>
<td>0.26</td>
<td>0.14</td>
<td>0.07</td>
</tr>
<tr>
<td>$M_1$</td>
<td>0.91</td>
<td>0.86</td>
<td>0.70</td>
<td>0.53</td>
<td>0.24</td>
<td>0.11</td>
<td>0.02</td>
</tr>
<tr>
<td>$M_2$</td>
<td>0.90</td>
<td>0.79</td>
<td>0.73</td>
<td>0.62</td>
<td>0.23</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>$M_3$</td>
<td>0.95</td>
<td>0.86</td>
<td>0.77</td>
<td>0.36</td>
<td>0.25</td>
<td>0.23</td>
<td>0.00</td>
</tr>
<tr>
<td>$M_4$</td>
<td>0.98</td>
<td>0.81</td>
<td>0.74</td>
<td>0.67</td>
<td>0.23</td>
<td>0.20</td>
<td>0.00</td>
</tr>
<tr>
<td>$M_5$</td>
<td>1.00</td>
<td>0.83</td>
<td>0.69</td>
<td>0.45</td>
<td>0.29</td>
<td>0.18</td>
<td>0.08</td>
</tr>
<tr>
<td>$M_6$</td>
<td>0.98</td>
<td>0.91</td>
<td>0.86</td>
<td>0.42</td>
<td>0.20</td>
<td>0.17</td>
<td>0.06</td>
</tr>
<tr>
<td>$M_7$</td>
<td>1.00</td>
<td>0.80</td>
<td>0.79</td>
<td>0.55</td>
<td>0.38</td>
<td>0.16</td>
<td>0.05</td>
</tr>
<tr>
<td>$M_8$</td>
<td>0.94</td>
<td>0.77</td>
<td>0.76</td>
<td>0.62</td>
<td>0.30</td>
<td>0.10</td>
<td>0.07</td>
</tr>
<tr>
<td>$M_9$</td>
<td>0.99</td>
<td>0.92</td>
<td>0.76</td>
<td>0.42</td>
<td>0.20</td>
<td>0.14</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 4.3: The hypotheses tested in the experiments. Column $H^*$ gives the best hypothesis for each experiment. Subscripts correspond to Table 4.2.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>$H^*$</th>
<th>Distractors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$H_1$</td>
<td>$H_2$ $H_3$</td>
</tr>
<tr>
<td>2</td>
<td>$H_1$</td>
<td>$H_4$ $H_7$</td>
</tr>
<tr>
<td>3</td>
<td>$H_5$</td>
<td>$H_6$ $H_7$</td>
</tr>
</tbody>
</table>

In Experiment 1 the hypotheses are all of reasonable quality. This represents a mission with good precursor data and generated hypotheses. In Experiment 2 the hypotheses are of mixed quality. In this scenario the precursor data is ambiguous, but there is one good hypothesis. In Experiment 3 all hypotheses are of poor quality. In this scenario the hypotheses are not fitted to the environment.

### 4.9.7 Analysis of Results

At the end of the experiments we compute mean of $P(H^*)$. Across the ten different maps we computed the effect size of our intervention. To determine the effect size we use Cohen’s $d$, with pooled standard deviation, $\sigma_{pooled}$.

$$d = \frac{\mu_1 - \mu_0}{\sigma_{pooled}} \quad (4.8)$$
where \( \mu \) and \( \sigma \) are the sample mean and standard deviation of \( P(H^*) \) for the control (0) and proposed (1) algorithms, respectively. Cohen’s \( d \) has four levels for effect size: \( d < 0.2 \) is a negligible effect, \( 0.2 < d < 0.5 \) is a small effect, \( 0.5 < d < 0.8 \) is a medium effect, and \( d > 0.8 \) is a large effect. Negative values of \( d \) mean that the control algorithm has a higher belief in \( H^* \). There are criticisms of using these standards [Lipsey et al. (2012)], but the dearth of effect size reporting in robotics does not provide alternative thresholds for effect size magnitudes.

### 4.10 Results

**Experiment 1 - All Hypotheses Reasonable**

![Bar graph showing belief states for different hypotheses](image)

Figure 4.3: Experiment 1 results. Both algorithms correctly identify \( H^* \), but the proposed algorithm was more confident than the control algorithm. The control algorithm distributed belief in the hypotheses more evenly. Neither algorithm was confident that there was no correct hypothesis under consideration. Error bars one standard error, \( N = 10 \).

Figure 4.3 shows the average belief state for the two different algorithms. In Experiment 1 our proposed algorithm was able to converge on \( H^* \) with high confidence. Our algorithm achieved a statistically significant \( (p < 0.01) \) large improvement \( (d = 1.71) \) over the control algorithm. We consider this experiment to be the most representative of mission scenarios, as hypotheses generated from precursor data should have some accuracy.

In Experiment 2 our algorithm outperforms the control, shown in Figure 4.4. Both converge on \( H_1 \) being the most credible, followed by \( H_4 \) and \( H_7 \). The control algorithm gives marginally more weight to the belief that none of the hypotheses are correct, \( H_0 \). Our algorithm outperforms
Belief

\[ P(H_0) \]

\[ P(H_1) \]

\[ P(H_2) \]

\[ P(H_3) \]

**Figure 4.4: Experiment 2 results. Our proposed algorithm again outperforms the control algorithm with statistical significance. Error bars are one standard error, \( N = 10 \).**

the control algorithm in this experiment with \( p < 0.01 \) and \( d = 1.84 \). This is a larger effect size than Experiment 1, due to the greater disparity in accuracy of the hypotheses. Our algorithm amplifies this disparity by testing on where the algorithms disagree.

In the third experiment all the hypotheses predict the true terrain poorly. The belief state at the end of sampling is shown in Figure 4.5. Again, our algorithm selects \( H^* \), with significant improvement over the control, \( p < 0.01, d = 1.54 \). On the other hand, Figure 4.5 shows that the control algorithm is significantly more confident that none of the algorithms are correct. None of the hypotheses in this experiment have a prediction accuracy greater than 50\%, \( H_0 \) - no correct hypotheses - is a valid conclusion. Our algorithm had more belief in \( H_0 \) than previously, but the control algorithm had statistically significantly higher \( P(H_0) \), \( p < 0.01, d = 5.55 \). We consider this scenario to be unlikely in planetary missions, but it does expose a weakness in our algorithm that should be addressed for when a lack of precursor data could lead to poor hypotheses.

4.11 Conclusions

We achieved our goal of developing an algorithm that can identify the least false of a set of competing hypotheses. Our algorithm is an significant improvement over a standard exploration algorithm. When the candidate hypotheses were of poor quality our algorithm was still able to pick the best one. However, the control algorithm better identified that none of the hypotheses were good. As the obvious extension of this work is to simultaneously generate and falsify hypotheses, this shortcoming must be addressed. Dynamically weighting \( r_s \) and \( r_h \) in Algorithm 4.7 may fix this problem. Alternatively it may be a weakness of the experiment. The algorithms
Experiment 3 - All Hypotheses Poor

Figure 4.5: Experiment 3 results. Our algorithm correctly selects $H^*$, but the control validly selects $H_0$, no good hypotheses. Error bars are one standard error, $N = 10$.

tested binary predictions and sought conflict – one hypotheses must always be correct. This problem may vanish when hypotheses’ predictions are high dimensional, like spectrometer readings or neural network outputs.

Sensor noise and hypothesis uncertainty were not addressed in this work, but are an important part of decision making. They will be incorporated in future experiments. By liberating our algorithm from path planning we demonstrated the utility of hypothesis-aware value functions for automate exploration. This work will be integrated into a planner for wheeled vehicles in future experiments. So equipped our exploring robots will be able to better serve us as they travel farther into space.

4.12 Deployment in Chile
Chapter 5

Simultaneous Hypothesis Generation and Falsification

This work is an experiment showing the principle behind using an understanding of how explanatory a model is to determine when to seek a better model and when to stop seeking one. Gaussian Processes were taken to work with because they are easily manipulated, and also because we can rely on the open source software presented as part of the automatic statistician project. However these techniques could be applied to any method of constructively expanding models. What is important is that there is either an explicit or implicit ranking of the complexity of the different kernels.

The complexity in this case is the sense of algorithmic complexity.

5.1 Problem Statement

Want to generate the best, most parsimonious model that explains the data.

The previous chapters illustrate how being aware of hypotheses under examination can be improve the performance of robots exploring unknown world. However, a major obstacle for conducting science is the construction of hypotheses that explain the data observed and in a parsimonious way.

We can compare the different between Kepler’s and Newton’s orbits [cite ’Scientific Discovery’ by Simon et al.]. Kepler’s orbits described the observations of the orbits, creating different descriptions of major and minor axes of the elliptical orbits and produced compendiums of the different orbits. Newton, through the introduction of the concepts of mass and gravitational forces, produced compressed, simple models that predict orbits more effectively than Kepler’s approach.

Kepler adhered strictly to the observed data in an attempt to predict how planetary orbits would proceed, but he did not produce a model that generalizes between the different planetary bodies, other than that the orbits would be elliptical. Newton produce a system that predicts the elliptical orbits based on understanding inherent properties of the system in question. Further, one can use the collected observations to infer the hidden properties (mass of planets).

Arguably, since the Newtonian approach simplifies all of the gravitational interactions in the
universe, it may be less accurate than collecting all the data that Kepler did and simply reporting it, but it won’t necessarily generalize to make predictions about how a new planet will operate. We can say that the Newtonian approach is a compressed presentation of the data that Kepler collected.

What we do then when generating hypotheses is to produce models that have an acceptable amount of descriptive error while being sufficiently simple, or parsimonious, in their description. Why parsimony? Because hypotheses can be made arbitrarily complex that they perfectly describe all observations but make wildly inaccurate predictions when encountering new situations. In machine learning literature this is known as overfitting. This harkens back to the problem of induction which was described in Chapter 1.

Given a finite amount of data we can build arbitrary models that describe the data. But we can never know, until it is observed, if that model will describe the next data point. This is why science has value as a method and a process, but is not something that is ever complete. We must collect data points that falsify our hypotheses, and in the face of new and unexplainable data revise our hypotheses, or generate new ones.

This chapter is about figuring out ways to determine which hypothesis is the most correct (read: least false) by generating explanations that are in different families of models and then determining tests to select between them. In this chapter we focus on using Gaussian Process Kernels as the models, but as long as a generation system is present (and it can be random) this approach should hold.

We are really relying on two things. First, that from any model $M$ we can generate a new model $M'$ that maintains the functional form of the original model. One can imagine starting from a constant function $M(x) = C$, optimizing it, and then branching out through the family of polynomials and the family of Fourier Series and determining at which point the models better agree.

The second thing we assume is that if model $M_1$ better describes the data than $M_2$, then it is more likely than not that there will exist an $M'_1$ that better describes the data than the best $M'_2$. This assumption can be considered a variant on the branch-and-bound strategy, or at least a prioritization scheme.

If we imagine that we can produce a graph of functions that we can explore, by generating the different functions. We know to stop exploring a graph when the explanation of the data (residual error) does not reduce.

1. Polynomial with constant variance
2. Polynomial with polynomial variance
3. Fourier series with constant variance
4. Fourier series with polynomial variance.
5. Gaussian processes with different kernels.

However, a shortcoming of this approach is that we will only be producing functions of the data, and not inventing theoretical entities, like Newton discovered mass/gravity. But because we proceed through the graph in a constructive way we will produce simpler hypotheses before we produce more complex ones.

Theoretical entities are terms in models that are not necessarily directly observable, but make
the theory work. Examples of this include the mass of an object. We can observe the weight of an object (Force of gravity acting on the object) but we cannot directly measure the mass. Mass is a unifying concept in Newtonian mechanics, and it is unlikely that anyone would deny that it exists.

Atoms are another example of theoretical entities that are used to describe physical models, and again it seems unlikely that someone would deny their existence. These sorts of theoretical entities are the things that William of Ockham, as paraphrased by John Punch, warned us that “entities must not be multiplied beyond necessity”\footnote{Ockham’s original statement was \textit{Numquam ponenda est pluralitas sine necessitate} - Plurality must never be posited without necessity. Aristotle and}

Ockham’s Razor is poetic, and an excellent principle by which to operate. However if one is to automate the generation of scientific theories this principle must be mathematized, which in this case means it must be formulated in such a way that it can be executed on a von Neumann machine\footnote{At the date of this writing von Neumann machines are the most commonly used computers. Perhaps at a later date quantum computers or neural computers may yield more comprehensive means of evaluating functions}.

Ideally our ockham measure would be a function of the data that has been collected $X = \{x_0, x_1, \ldots, x_n\}$ and the complexity of the model that is being used to explain those data, $M$. We could write that as:

$$\Upsilon(M, X) = f(X, M) + K(M)$$

Where $f(\cdot, \cdot)$ is a measure of how well data fits this model and $K(\cdot)$ is a measure of the complexity of the model. There already exist at least two widely used measures for model effectiveness. They are the Akaike Information Criterion (AIC)\footnote{AIC = \textit{Akaike} (1974) and the Bayesian Information Criterion (BIC)\footnote{BIC = \textit{Schwarz et al.} (1978).}}\cite{Akaike1974} and the Bayesian Information Criterion (BIC)\cite{Schwarz1978}.

$$AIC = 2k - 2 \ln \mathbb{P}(X|\hat{\theta}, M) \quad (5.1)$$

$$BIC = -2 \ln \mathbb{P}(X|\hat{\theta}, M) + k \ln (n) \quad (5.2)$$

Where $\hat{\theta}$ are the parameters that maximize the likelihood of the data given the mode, $k$ is number of free parameters to be estimated in the model (i.e. $|\hat{\theta}|$, and $n$ is the number of data points observed in $X$. Both these criteria penalize the effectiveness of the log likelihood by adding a function of the how the model is described.

The Bayesian Information Criterion penalizes the cost of parameters by multiplying the number of free parameters in the model. However this criterion assumes that only the exponential family of models are used. We would not necessarily want to be restricted to this family of models. For that reason we look to the Minimum Description Length\cite{Rissanen1978}.

### 5.1.1 Minimum Description Length and Kolmogorov Complexity

The amount of information contained in a string of symbols given by the length of the shortest computer program that generates the string. Research in Minimum Description Length speaks of
trying to describe sets of data with some distribution function [Myung et al. (2006)]. There is no reason, however, that we can’t apply this to describing the complexity of relations that map from one data set to another.

In fact, we proposed a criteria for measuring the suitability of a model as a two-valued vector:

$$\Upsilon(X, M) = \left( \mathbb{P}(X|\hat{\theta}, M), K(M) \right)$$

Where $K$ is the function that gives the description length of the model in whatever language of description is being used. In this case $K$ is a function that maps from the space of models to the positive integers. $K : \mathcal{M} \rightarrow \mathbb{Z}^+$. While in general the Kolmogorov complexity is not computable, if our models are computable functions, we can determine their description length in bits exactly when they are compiled to a (possibly virtual) machine code of our choosing.

However, if the model includes iterations then this can make estimating the complexity of the model significantly harder. It is not necessarily possible from an initial inspection to determine how long an iteration may take if the stopping criteria rely on settling some error to within an acceptable range (e.g. simulated annealing). This can be dealt with by adding a third term to the tuple which computes the average time to compute the result of some observation $X$ to a prediction $Y$, which, again, can be computed on some virtual computer.

By using the vector-valued approach to measuring a model’s fitness one can find the pareto optimal approach.

### 5.1.2 Interpretability of a Hypothesis

### 5.2 Method

There are (at least) two questions that must be answered:

1. How do we generate a new model from an old one?
2. How do we determine when to produce a new model?

#### 5.2.1 Generating New Models

Generally the approach here can be borrowed from other approaches such as the automatic statistician ([Duvenaud et al.] 2013), or that used by Ly and Lipson’s approach as deployed in Eurqa ([Ly and Lipson, 2012]), or the approach used by ([Grosse et al.], 2012) to generate new models, or Jurgen Schmidhuber’s approach to learning low-complexity kernels ([Schmidhuber, 1995]).

Mainly what is happening here is we are using a production grammar to produce new models from old ones. This could also work with neural networks. Your choices there are to add $X$ new neurons or to add a new layer. This is assuming the transfer functions are kept homogeneous.

In this chapter we do not necessarily examine a method that can generate hypotheses with new theoretical entities. However, this approach can choose between them regardless.

There is the fact to consider that invariant things, like the affect of gravity on mass, look like arbitrary constants until we observe phenomena occurring in different gravitational fields.
There are also matters to consider of uncertainty in observations, and uncertainty in predictions. The uncertainty in observations can result from poor resolution in the instrument that measures the phenomena of interest. Things like the discretization error in an analog to digital converter can cause errors in a signal to be propagated through a system, making observations not match predictions. Noise in a sensor, caused by, e.g., thermal effects, can also cause data to be observed incorrectly from the environment. Other approaches, like errors classification systems, can cause edge cases to be misclassified (especially if there is noise in the sensors).

Further, since predictions from models are based on collections of previous data.

5.2.2 When to Produce a New Model

The question of when to produce a model is more accurately turned around to the question “when should one stop producing new models”. The obvious answer in this case is that one should stop seeking new models when one has stopped gaining advantage in pursuing that line of modeling.

The model makes predictions about the input data it is fed. For every prediction $\hat{y}_i$ there will be a corresponding error, $\epsilon_i = \hat{y}_i - y_i$. For two competing models, $M_1$ and $M_2$ one can build the distributions $P(\epsilon_i)_1$ and $P(\epsilon_i)_2$. We can then determine if the High Density Intervals of those distributions overlap. If they do, then we can conclude with some confidence $\delta$ that these two distributions are statistically the same. If that is the case, then we can see that there was no additional advantage to adding complexity to the model. If they do not overlap, and the new model, $M_2$ has a lower expected average error, then it is a more valuable model and should be pursued further.

The first thing we do is take the starting hypotheses and we determine what minimum number of samples required to fit that model. We then sample that many data points from the oracle and fit the hypotheses. We then compute the algorithmic complexity of the different models and determine their ordering by AC criterion. Next we determine the

We want to sample from this space in a fashion similar to how we did in the previous chapter. Essentially we want to maintain at any given time a distribution over the the $P(M_i|D)$, where $D$ is the dataset and $\sum_i P(M_i|D) = 1$. This is necessary to maintain the belief in the most correct model.

Once we select a model as the one likely to be the true model (for this iteration) we determine a data point that would be most informative for separating the other models. However, it is conceivable that this list could expand to be infinitely long. What would be more appropriate is to treat this as a graph search problem. In doing so we need only maintain a list of leaf nodes from which we would like to expand.
Algorithm 5.1 Hypothesis Generation and Search Algorithm

```plaintext
function HGS(oracle, generator, seedHypotheses)
    hypotheses ← priority_queue(seedHypotheses)
    minNumPoints ← max minNumPoints(hypotheses)
    dataSet = \{x_i ∼ oracle | i ∈ \{1, . . . , minNumPoints\}\}
    for h ∈ hypotheses do
        fit(h, dataSet)
        y_i ∼ U(numPixels)
        label ∼ U(numLabels)
        seeds ← seeds + \{(x_i, y_i, label)\}.
    end for
    return map
end function
```

We maintain our belief in the leaf nodes and we select sampling actions that concentrates our belief amongst these leaf nodes. After sampling that location and updating our belief, we pick the best hypothesis to expand. We expand that hypothesis and see how well its children represent the data. If any of those child hypotheses represent a statistically significant improvement we can add it to the list of hypotheses under consideration and remove its parent. Otherwise we halt progress on that branch of hypotheses.

This becomes active model selection. Simultaneous model generation and selection.

5.3 Experiment

5.3.1 Control Algorithm

5.3.2 Testing Data

Stationary Functions

Non-Stationary Functions
Appendices
Appendix A

List of Terms

**Hypothesis**  A function that predicts from an input space $X$ to a probability distribution over an observation space, $Z$. Formally: $h : X \rightarrow (Z \rightarrow [0, 1])$

**Object**  An entity in the environment the robot can reason about.

**Sampling Opportunity**  An object that the robot can choose to sample with one of its instruments, for some cost.
left intentionally blank

Table A.1: Interpretation of different effect sizes.

<table>
<thead>
<tr>
<th>Effect Size</th>
<th>d</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very small</td>
<td>0.01</td>
<td>Sawilowsky, 2009</td>
</tr>
<tr>
<td>Small</td>
<td>0.20</td>
<td>Cohen, 1988</td>
</tr>
<tr>
<td>Medium</td>
<td>0.50</td>
<td>Cohen, 1988</td>
</tr>
<tr>
<td>Large</td>
<td>0.80</td>
<td>Cohen, 1988</td>
</tr>
<tr>
<td>Very Large</td>
<td>1.20</td>
<td>Sawilowsky, 2009</td>
</tr>
<tr>
<td>Huge</td>
<td>2.0</td>
<td>Sawilowsky, 2009</td>
</tr>
</tbody>
</table>
Appendix B

Where’s Thomas Kuhn?

Thomas Kuhn wrote the book “On the Structure of Scientific Revolutions” which is a popular book with his thoughts on the process by which scientific theories come into popularity. These thoughts were developed during Kuhn’s Ph.D. work in physics and arrived relatively unadulterated in his book.

There are two main concepts behind Kuhn’s thoughts. First, is that it takes time for scientific theories to become mainstream (a “paradigm shift”). Second, that scientific worldviews developed under different paradigms are incomparable. The first point I am happy to cede, as this is a notion well established in organization behaviour research. In any political organization it takes time for new ideas to propagate through the system and achieve acceptance. In the case of science there are notions that it is self correcting and while that may be generally true in the long term, it is still a human endeavour subject to human weaknesses. Illustrative of this point is the case of Ignaz Semmelweis.

Semmelweis published results demonstrating that when physicians wash their hands between patients reduced patient mortality to below 1% in his particular studies of childbed fever. Initially the benefits of his results were rejected by the medical community, partly because of entrenched ideas and partly because he did not have a theoretical mechanism to explain his results – Pasteur confirmed the germ theory of disease well after Semmelweis’ death. The rejection of Semmelweis’ results led him to becoming obsessed with the subject, and ultimately ending up being committed to an asylum.

The second point from Kuhn’s work is more problematic. The notion that world views generated under one scientific paradigm is utterly incomparable to the world view generated under a second paradigm raises questions about the existence of history and the ability of humans to learn and change.

Kuhn proposes an arbitrary barrier between two camps that would permit them from arguing the relative merits of their respective scientific paradigms (e.g. Newtonian vs Quantum physics). Particularly Kuhn asserted that one theory could not be proven or disproven by another theory. This is making a category error, however. The purpose of a scientific theory first and foremost is to describe some observable phenomenon, a theory is not a proof system for other theories. The neutral language that compares different theories are the predictions of the theories, observations of real data, and an error metric between the two, additionally a measure of theoretical complexity (e.g. Kolmogorov complexity). With these two measures (error, complexity) scen-
tists can compare different theories for their performance and the parsimoniousness with respect
to theoretical entities.

A more unsettling theme in Kuhn’s work is the rejection of truth. Kuhn does not seem to
allow for ether objective, absolute truth or the book maker’s truth of F.P. Ramsey – the thing that
is true is the prediction that is most correct most of the time. Theories can only be incomparable
if they do not traffic in replicable observations.

If this were the case then we would not be able to make scientific progress at all. We see
this especially in cases of testing the paranormal. Paranormal experts make predictions about
their method, they are found to not have an effect. They then propose other effects that were
not recorded the first time, tests are then conducted to observe those effects and (again) they are
found to not exist. The ultimate refuge of the woo-peddler is to fall back into a position stating
that science cannot measure either the effects or their means of transmission. Personally, I find
the notion of interventions that have undetectable effects to be somewhat unsatisfying.

Since scientific endeavours are about describing observable phenomenon then by extention
of Kuhn’s notion of incomparability different theories explaining the same observations must
be mute on those observations. By this line of reasoning we must draw the conclusion that
Kuhn’s notions of scientific theories either do not make predictions, rendering them useless, or
the predictions they make on the same data cannot be compared to each other while at the same
time being comparable to the data they are attempting to predict. The former is a council of
dispare, the later a contradiction.

In his work, Kuhn conflates how humans propagate ideas through political systems and how
scientific theories can be tested and compared. While he may have a point in the first case, in the
second he is unproductive, and his confusion renders his theories less than useful. Note that his
is not the scientific anarchy of Paul Feyerbrand, who believes that there is no scientific method.
Bibliography

Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In COLT, pages 39–1, 2012. 4.8


Asher Bender, Stefan B Williams, and Oscar Pizarro. Autonomous exploration of large-scale benthic environments. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 390–396. IEEE, 2013. 2.2.5


Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In Advances in neural information processing systems, pages 2249–2257, 2011. 4.8

Eric L Charnov. Optimal foraging, the marginal value theorem. Theoretical population biology,

Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013. 2.4


Tara A Estlin, Benjamin J Bornstein, Daniel M Gaines, Robert C Anderson, David R Thompson, Michael Burl, Rebecca Castano, and Michele Judd. Aegis automated science targeting for the mer opportunity rover. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):50, 2012. 4.8

Thomas S Ferguson. Who solved the secretary problem? *Statistical science*, pages 282–289, 1989. 2.2.2, 3.2.1


P Michael Furlong and David S Wettergreen. Budgeting samples for exploration in unknown environments. *i-SAIRAS*, 2014. 2.3.2


R.B. Grosse, R. Salakhutdinov, W.T. Freeman, and J.B. Tenenbaum. Exploiting compositionality to explore a large space of model structures. In *Uncertainty in Artificial Intelligence*, 2012. 5.2.1


Peter Pirolli and Stuart Card. Information foraging. *Psychological review*, 106(4):643, 1999. 2.2.4

Ernst L'vovich Presman and Isaac Mikhailovich Sonin. The best choice problem for a random number of objects. *Teoriya Veroyatnostei i ee Primeneniya*, 17(4):695–706, 1972. 2.2.2


Jürgen Schmidhuber. Discovering solutions with low kolmogorov complexity and high general-


Kirstine Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. *Biometrika*, pages 1–85, 1918.


