7-2-2024

Experimental Design for Scientific Discovery

Quan Minh Nguyen
Washington University – McKelvey School of Engineering

WASHINGTON UNIVERSITY IN ST. LOUIS

McKelvey School of Engineering
Department of Computer Science & Engineering

Dissertation Examination Committee:
Chien-Ju Ho, Chair
Jacob Gardner
Roman Garnett
Alvitta Ottley
Yevgeniy Vorobeychik

Experimental Design for Scientific Discovery

by
Quan Nguyen

A dissertation presented to
the McKelvey School of Engineering
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2024
St. Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

It is said that completing a Ph.D. is challenging due to its (very) sparse reward, but I was fortunate enough to receive plenty of reward during my time in the form of constant support, help, and shared laughter from friends, collaborators, and colleagues, who are happily too many to include here. Below I would like to give my sincerest thanks to some of them.

To Roman Garnett, for striking the perfect balance between being supportive with everything I want to do (be it research, internships, or teaching) and making sure I prioritize the important things, for instilling in me a sense of critical thinking and a taste for good research, and for teaching me to appreciate the differences between a "-", an "–", and an "—", to prefer "ε" over "ϵ", and to use lowercase acronyms.

To Yehu Chen and Shayan Monadjemi for being effortlessly wonderful labmates, for offering patience and empathy whenever I complain about the most minute matter (as I often do), for always giving me rides to dinners, and Yehu for accompanying me at sports bars.

To Oren Bell for initiating numerous, much needed social gatherings, for always being ready to spring into action when something in the apartment stops working, and for the movie nights. To Saumik Narayanan for the many enjoyable conversations in the office. To Ashwin Kumar for caring about me and appreciating how hard I work. To Aaron Handleman for the yellow python. To Hanyang Liu for his expert advice on cats (and dogs).

To my family for teaching me to be kind, for giving me the courage to see the degree through, and for reminding me that finishing this dissertation is rather important.

Quan Nguyen

Washington University in St. Louis
August 2024

To dad, my friend, teacher, and hero.

ABSTRACT OF THE DISSERTATION

Experimental Design for Scientific Discovery

by

Quan Nguyen

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2024

Professor Chien-Ju Ho, Chair

Experimental design offers an elegant model of many problems where one navigates within a vast search space seeking data points with certain characteristics. A multitude of applications in science and engineering fall under this umbrella, with drug and materials discovery being prime examples. The experimental design approach maintains a probabilistic model of the search space, and uses Bayesian decision theory accounting for this model to guide the accumulation of observed data to maximize an experimentation objective of interest. This dissertation explores Bayesian optimization and active search, two realizations of the experimental design framework that model discovery tasks. While existing solutions are available for these two problems under conventional settings, there are important scenarios to which these solutions cannot be readily applied, namely those of high dimensions or with multiple data sources, objectives that favor diversity in the collected data, and settings where efficient policy computation is crucial such as real-time systems and large-scale databases. We address these gaps, putting forward optimization and search policies with competitive empirical performance under their respective settings. The algorithmic solutions in our works provide practitioners with the tools to tackle a broad range of experimental design tasks, and ultimately advance machine learning-aided scientific discovery efforts.

Chapter 1

Introduction

The ability to adaptively make decisions based on data underlies many modern applications ranging from product recommendation [50] and personalized healthcare [105] to algorithms for robotic control, text generation, and learning to play games such as chess and Go at superhuman level [182, 63]. Experimental design formalizes this framework of adaptive decision-making by building a predictive model on available data, and using that model to guide the design of new experiments towards a particular goal. Of particular interest are scientific discovery problems that are characterized by (i) the sheer number of experiments one could possibly perform and (ii) the high cost of experimentation. An illustrative example is in drug discovery, where a scientist searches over a vast database of chemical compounds for those with desirable properties (such as binding activity to target proteins). Given a candidate compound, extensive computational and/or physical experiments are needed to fully characterize its properties. This cost of experimentation prevents exhaustive screening of the entire database, and motivates a sample-efficient strategy that inspects only a small number of compounds that are beneficial to the search and lead the scientist closer to their goal. In experimental design, we are concerned with the following question: Given a particular objective, how do we construct such an effective sampling strategy that can identify the beneficial experiments to perform?

This dissertation explores experimental design strategies under settings relevant to scientific discovery tasks. We begin with Bayesian optimization, a particular realization of experimental design for optimizing functions, in Chapt. 3. While traditional Bayesian optimization approaches have proven to be extremely effective in low dimensions, they struggle to perform well on high-dimensional objective functions due to curse of dimensionality. Noting the effectiveness of local optimization in high dimensions, we design a local Bayesian optimization framework to circumvent the difficulty of global optimization under the curse of dimensionality. By maximizing the probability that the next design will make progress in optimization, our algorithm can effectively navigate high-dimensional search spaces and successfully optimize challenging objective functions. This local optimization strategy allows us to tackle optimization tasks that involve a large number of decision variables, and contributes to the high-dimensional Bayesian optimization literature.

Next, we find another important instance of experimental design in active search, where the goal is to identify rare and valuable data points within a large but finite database, and study two specific settings. The first is multifidelity search discussed in Chapt. 4, where one has access to multiple experiment oracles of varying degrees of cost and accuracy (e.g., a cheap computer simulation and an expensive physical experiment conducted in a laboratory). Here, we aim to design a search policy that can leverage these multiple oracles simultaneously to maximize discovery. Chapt. 5, on the other hand, presents active search settings with utility functions that exhibit diminishing returns for similar discoveries. These utility functions model preference common in scientific settings for dissimilar over similar data, and encourage search policies to uncover diverse data sets. We extend the machinery developed from traditional active search to these new settings, first proving the hardness result that, in the worst case, there are no polynomial-time policies that can achieve a constant-factor approximation to the expected utility by the optimal policy. While these hardness results establish the theoretical limit of efficient active search policies, we also design policies that perform empirically well on real-world problems, by adopting the state-of-the-art algorithm from traditional search to multifidelity and diversity-aware search.

Finally, as the application of active search to real-time systems as well as large search spaces becomes more common, the need for highly efficient search policies arises. While the state-of-the-art active search algorithm employs a combination of simplifications of the search objective and aggressive pruning, it retains a time complexity superlinear with the size of the search space. This complexity poses a challenge in deploying the policy in real-time applications where decisions need to be made quickly or in large-scale spaces that are common in modern drug discovery. Chapt. 6 addresses this gap by having a neural network learn to search. Leveraging imitation learning techniques from reinforcement learning, we train a small, relatively shallow neural network to mimic the behavior of the expensive-to-compute state-of-the-art search policy. The trained policy network, which can rapidly make its decisions via fast forward passes, can then be deployed in the aforementioned target applications. We show that the policy network produces beneficially nonmyopic designs that are similar to those of the original expert policy, and, across diverse real-world tasks, achieves competitive performance that closely matches the expert's at a fraction of the cost, while outperforming cheaper baselines.

Most of the algorithmic solutions discussed in this dissertation take a Bayesian decision-theoretic approach. We first define a utility function that expresses preference over different outcomes resulting from possible designs – in other words, this utility function quantifies the experimental design objective. We then compute the expected utility of each possible design, marginalizing over outcome uncertainties, and select the design maximizing this expected utility. By deriving this expected-case optimal decision or, when marginalizing over all possible uncertainties is computationally intractable, efficiently approximating it, we develop principled experimental design strategies with impressive empirical performance. Overall, our works offer a wide range of solutions tackling diverse experimental design settings, providing flexibility in data-driven decision-making to accelerate scientific discovery.

This flexibility is further demonstrated in a wide range of real-life use cases to which our experimental design solutions have been applied, which are enumerated in Chapt. 7 and include the discovery of photoswitches with long half-lives in chemistry, offering guidance to users of an interactive visualization for data exploration in visual analytics, and learning of convex hulls for phase diagrams in materials science. Throughout these diverse applications, our algorithms lead to more efficient learning and discovery than less adaptive approaches to experimentation, showcasing the impacts experimental design can have on real-world tasks.

Declaration of Previous Publications

All works presented in this dissertation is the result of collaboration with other researchers, most of which have been published at peer-reviewed conferences. We outline the original work works presented in this dissertation below with references to those previous publications or unpublished manuscripts, and describe each author's contributions.

Chapter 2: Background

This chapter reviews the fundamentals upon which the rest of this dissertation builds. The presentation is mainly the candidate's own; various notions on Bayesian decision theory are partially inspired by

> Roman Garnett. *Bayesian Optimization.* Cambridge University Press, 2022.

Chapter 3: Local Bayesian Optimization

The work in this chapter appears in

> Quan Nguyen, Kaiwen Wu, Jacob R. Gardner, and Roman Garnett. Local Bayesian optimization via maximizing descent probability. In *Advances in Neural Information Processing Systems*, 2022.

Gardner and Garnett initiated the main idea of learning the gradient of the objective function for local optimization. Nguyen and Wu jointly implemented prototypes and conducted preliminary experiments. Garnett motivated maximizing descent probability, and Wu derived the closed-form solution for the acquisition function. Wu implemented the acquisition function and Nguyen performed the experiments and analyses. Gardner and Garnett provided valuable feedback throughout.

Chapter 4: Multifidelity Active Search

The work in this chapter appears in

> Quan Nguyen, Arghavan Modiri, and Roman Garnett. Nonmyopic Multifidelity Acitve Search. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Modiri and Garnett independently motivated the problem. Modiri proposed an initial baseline policy and Nguyen designed the final acquisition function. Nguyen conducted the experiments and analyses. Garnett proposed showing the cumulative utility difference versus the single-fidelity policy and offered crucial guidance throughout.

Chapter 5: Diversity-Aware Active Search

The work in this chapter appears in

> Quan Nguyen and Roman Garnett. Nonmyopic Multiclass Active Search with Diminishing Returns for Diverse Discovery. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, 2023.

Nguyen proposed the initial problem setting with the logarithm utility function, and designed the search policy. Garnett suggested lazy pruning when optimizing the submodular acquisition function to expedite policy computation, and motivated the analysis with the other utility functions such as the square root. Nguyen performed the experiments and analyses with Garnett's feedback.

Chapter 6: Amortized Active Search

The work in this chapter is currently under review and appears in the following preprint:

> Quan Nguyen, Anindya Sarkar, and Roman Garnett. Amortized nonmyopic active search via deep imitation learning. *arXiv preprint*, 2024. arXiv:2405.15031 [cs.LG].

Garnett conceived the idea of using reinforcement learning to learn to perform active search and suggested using imitation learning to obtain an initial policy. Nguyen designed the state representation, performed the training, and conducted the experiments and analyses with Sarkar's and Garnett's valuable feedback.

Chapter 7: Applications of Experimental Design

The works in this chapter appear in the following publications and preprint:

> Fatemah Mukadum, Quan Nguyen, Daniel M. Adrion, Gabriel Appleby, Rui Chen, Haley Dang, Remco Chang, Roman Garnett, and Steven A. Lopez. Efficient Discovery of Visible Light-Activated Azoarene Photoswitches with Long Half-Lives Using Active Search. *Journal of Chemical Information and Modeling*, 2021.

The Lopez group motivated the problem and Garnett proposed using active search. Nguyen adopted software implemented in previous work to implement the active search algorithm. Subsequently analyses were jointly conducted by all collaborators.

> Shayan Monadjemi, Sunwoo Ha, Quan Nguyen, Henry Chai, Roman Garnett, and Alvitta Ottley. Guided Data Discovery in Interactive Visualizations via Active Search. In *IEEE Visualization and Visual Analytics (VIS)*, 2022.

Ottley and Garnett motivated using active search for interactive data discovery. Nguyen implemented the active search policy and conducted preliminary simulated experiments to demonstrate the benefits of active search. Monadjemi and Ha conducted subsequent analyses.

> Andrew Novick, Diana Cai, Quan Nguyen, Roman Garnett, Ryan P. Adams, and Eric Toberer. Probabilistic Prediction of Material Stability: Integrating Convex Hulls into Active Learning. *arXiv preprint*, 2024. arXiv:2402.15582 [cond-mat.mtrl-sci].

Toberer motivated active learning of convex hulls. Garnett and Adams designed the active learning policy. Cai implemented a preliminary version of the policy, which Novick refined and ran with Nguyen's support.

Chapter 2

Background

We begin with the formulation of experimental design and the Bayesian decision-theoretic approach to deriving decisions that are optimal in expectation. We also discuss the realization of this expected-case optimality in active search, its exponential time complexity, as well as an approximation to this optimal policy that circumvents the exponential blowup.

2.1 Experimental Design

Suppose we have evaluation access to a function $f\colon \mathcal{X} \mapsto \mathcal{Y}$ in that we may perform experiments by evaluating the function at locations $x \in \mathcal{X}$ of our choosing, and observe the experimental outcome in the form of the function value $y = f(x)$. Here, $y \in \mathcal{Y}$ encapsulates various observation models of interest, including binary observations or those corrupted by Gaussian noise. We assume that the cost of experimentation, that is, of evaluating $f(x)$ at a specific $x \in \mathcal{X}$, is high, and we can only do it for a specified number of times; in other words, our evaluation budget is subject to a constraint.[1] Our goal is to iteratively choose the locations $x_i \in \mathcal{X}$ at which evaluate the function $y_i = f(x_i)$, and assemble a data set $\mathcal{D} = \{(x_i, y_i)\}_i$ so as to maximize a utility function of interest. We will refer to $f(x)$ as the label of the data point x, and the process of evaluating $f(x)$ as labeling x.

The utility function specifies our experimental design objective by quantifying our preference over possible data sets $\mathcal{D}$ that may result from different designs. Each combination of the underlying function f and the utility function gives rise to an instance of a experimental design problem, and the works in this dissertation examine two important instances: Bayesian

[1] There are other types of budget constraints that have been explored in the literature, but this dissertation focuses on the particular constraint that we introduce here, where the function f can only be evaluated for a specified number of times.

Algorithm 1 Experimental design

```
1: inputs function $f$, domain $\mathcal{X}$, experimental budget $T$, experimental design policy $\pi$,
   initial data $\mathcal{D}_0$
2: for $t \leftarrow 1$ to $T$ do
3:     $x_t \leftarrow \arg\max_{x \in \mathcal{X}} \pi(x \mid \mathcal{D}_{t-1})$        ▷ select the next point to query
4:     $y_t \leftarrow f(x_t)$        ▷ evaluate function $f$
5:     $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$        ▷ update observed data
6: end for
```

optimization and active search. Bayesian optimization targets optimization of a real-valued function f, while in active search, we aim to identify many data points within a finite search space $\mathcal{X}$ that yield positive values when a binary-valued function f is evaluated. These two experimental design tasks flexibly model a wide range of applications in scientific discovery. For example, Bayesian optimization has been used to optimize desirable quality of molecules such as synthesizability or high power conversion efficiency [80, 66, 72], or material properties such as durability and ease of production in materials design [45, 122]. Active search, on the other hand, has similarly been successfully applied to discover molecules that bind with target proteins for drug discovery [62], photoswitches with characteristics favorable for materials chemistry [141], and bulk metal glasses that have higher toughness and resistance than crystalline alloys [88, 145].

The assumed high cost of evaluating f is especially applicable in these scientific discovery tasks, where evaluating the function f often equates running time-consuming computer simulations and/or performing laborious experiments in a laboratory to characterize a molecule or material one would like to inspect. This cost of experimentation motivates careful consideration of which data point $x \in \mathcal{X}$ to label in order to maximize the given utility function. Our goal is to construct a policy π that facilitates this process in an iterative manner, where at each iteration, π reasons about the data observed thus far and decides which data point to label next. The iterative loop of experimental design is summarized in Alg. 1. At iteration t, the policy π produces a score for each data point x that we could possibly evaluate f with next based on the observed data $\mathcal{D}_{t-1}$. This scores quantifies the value of performing the experiment corresponding to evaluating $f(x)$, and we proceed by selecting the data point x_t with the highest score. Upon computing $f(x_t)$, the new data point is added to the observed set $\mathcal{D}_t$, and the process repeats until our experimentation budget is exhausted.

Again, how to design the policy π so that it appropriately favors data points that are beneficial to our objective (i.e., those that maximize the utility function) is the key to solving a given experimental design task. In the next section, we discuss Bayesian decision theory as a framework for designing policies that make optimal decisions in expectation.

2.2 Bayesian Decision Theory

Bayesian decision theory is a general framework for decision-making under uncertainty, which we leverage for experimental design tasks here. We first require a probabilistic model of the function f, which produces the posterior distribution of the possible values of $y = f(x)$ for a given $x \in \mathcal{X}$ conditioned on the data we have seen so far $\mathcal{D}$. We also need access to a utility function u that quantifies our true valuation of a given data set $\mathcal{D}$, denoted as $u(\mathcal{D})$; in other words, we prefer data set $\mathcal{D}$ over data set $\mathcal{D}'$ if and only if $u(\mathcal{D}) > u(\mathcal{D}')$. For example, in Bayesian optimization, this utility function is often chosen, in the maximization case, to be the so-called simple reward function $u(\mathcal{D}) = \max_{y \in \mathcal{D}} y$, the largest value observed in $\mathcal{D}$, to naturally reflect the goal of maximization.

Now, suppose that we are currently at iteration t of the experimental design loop in Alg. 1, having observed the data set $\mathcal{D}_{t-1}$ and looking for the next data point x_t to label. Bayesian decision theory identifies the optimal data point x_t^* as the one that, at the end of the loop, results in the terminal data set $\mathcal{D}_T$ with the highest expected utility:

$$x_t^* = \operatorname*{arg\,max}_{x_t \in \mathcal{X}} \mathbb{E}\big[u(\mathcal{D}_T) \mid x_t, \mathcal{D}_t\big]. \tag{2.1}$$

Here, the expectation is taken with respect to not only the label of the putative data point x_t but also labels of subsequent data points labeled at future iterations $(t+1, t+2, \dots, T)$. These future data points are identified in a manner similar to Eq. (2.1), which may be derived via dynamic programming [18]. However, as each subsequent decision depends on those before it, the entire computation of Eq. (2.1) involves nested expectations and maximizations and has a time complexity exponential in the remaining experimentation budget $\ell = T - t$. This means that computing Eq. (2.1) by fully looking ahead to the end of the loop is computationally intractable for most applications, including those in this dissertation.

A common strategy to overcome this exponential blowup in practice is to limit the lookahead, effectively assuming that the horizon ℓ is sufficiently short to enable efficient policy computation. The simplest version of this strategy is to set $\ell = 1$, assuming that we are at the last iteration of Alg. 1, where Eq. (2.1) may be computed as maximizing the one-step utility $\mathbb{E}\big[u(\mathcal{D}_t)\big]$. This myopic lookahead strategy is appealing as maximizing $\mathbb{E}\big[u(\mathcal{D}_t)\big]$ is computationally straightforward in many applications. For example, in the specific case of Bayesian optimization, optimizing the one-step utility $\mathbb{E}\big[u(\mathcal{D}_t)\big]$ under $u(\mathcal{D}) = \max_{y \in \mathcal{D}} y$ gives rise to the widely-used policy Expected Improvement. However, by ignoring the remaining decision-making horizon, policies obtained with limited lookahead might underestimate the value of exploratory designs that could yield high utility at future iterations, and ultimately produce myopically suboptimal decisions. The next section discusses an approximation to the optimal policy in Eq. (2.1) in the particular context of active search by Jiang et al. [88] that accounts for the full decision-making horizon and thus yields nonmyopic designs. This nonmyopic policy achieves impressive empirical performance across many tasks, and serves as the basis for policies developed in the candidate's own works presented in this dissertation.

We also briefly note that another way to approximate Eq. (2.1) beyond myopic lookahead is to employ an auxiliary utility function v. This alternative utility function should quantify our preference for data sets in a way that optimizing for the corresponding one-step utility $\mathbb{E}\big[v(\mathcal{D}_t)\big]$ is more conducive to optimizing for the utility of the terminal data set $\mathbb{E}\big[u(\mathcal{D}_T)\big]$ than greedily maximizing $\mathbb{E}\big[u(\mathcal{D}_t)\big]$. We see an example of this in Chapt. 3 for local Bayesian optimization, where we pursue maximization of the probability of making optimization progress as an approximation to optimizing the function f itself.

2.3 Nonmyopic Active Search

We now examine the active search framework as well as the state-of-the-art policy by Jiang et al. [88], a nonmyopic approximation to the expected-case optimal policy. An active search problem is defined by a finite search space $\mathcal{X}$, among which there exists a rare, valuable subset $\mathcal{T} \subset \mathcal{X}$. We use the term "targets" to refer to the members of this valuable subset, which we wish to collect from the entire space $\mathcal{X}$. We further use membership in $\mathcal{T}$ as the labels for the points in $\mathcal{X}$: $y_i = f(x_i) \triangleq \mathbb{I}\left[x_i \in \mathcal{T}\right], \forall x_i \in \mathcal{X}$. The targets are not known *a priori*, but whether a specific data point x_i is one can be determined by labeling the data point to obtain y_i. The goal of active search is to sequentially select which data points to label so as to find as many targets throughout the T iterations of the search as possible. To express this preference for maximizing the number of "hits" across different terminal data sets collected at the end of the search $\mathcal{D}_T$, we use the utility function

$$u(\mathcal{D}_T) = \sum\nolimits_{y_i \in \mathcal{D}_T} y_i, \tag{2.2}$$

which simply counts the number of targets in $\mathcal{D}_T$.

As discussed in the previous section, under Bayesian decision theory, the optimal decision at each search iteration is the data point maximizing the expected utility of the terminal data set $\mathbb{E}\big[u(\mathcal{D}_T)\big]$, the computation of which can be obtained in theory via dynamic programming but is computationally intractable in practice. We can instead seek to optimize the one-step lookahead utility $\mathbb{E}\big[u(\mathcal{D}_t)\big]$ as a myopic approximation to the optimal policy, which in active search simplifies to finding the most likely data point:

$$\operatorname*{arg\,max}_{x_t \in \mathcal{X} \setminus \mathcal{D}_{t-1}} \mathbb{E}\Big[u(\mathcal{D}_t) \mid x_t, \mathcal{D}_{t-1}\Big] = \operatorname*{arg\,max}_{x_t \in \mathcal{X} \setminus \mathcal{D}_{t-1}} \Pr(y = 1 \mid x_t, \mathcal{D}_{t-1}). \tag{2.3}$$

Again, limiting the lookahead undervalues exploratory designs that might yield high future rewards, and could as a result lead to suboptimal decision-making. To go beyond myopic lookahead while avoiding the high cost of reasoning about the dependence among labels of future queries, Jiang et al. [88] made the simplifying assumption that, after our next query x_t, all remaining future queries $X = (x_{t+1}, x_{t+2}, \dots x_T)$ are made simultaneously in a batch, exhausting our labeling budget. Under this assumption, the future queries in the lookahead in Eq. (2.1) – to be optimally chosen to maximize expected terminal utility – simplifies to the set of $(\ell - 1)$ most likely targets thanks to linearity of expectation, and the (approximate) expected terminal utility decomposes into:

$$\begin{aligned} \mathbb{E}\big[u(\mathcal{D}_T) \mid x_t, \mathcal{D}_{t-1}\big] \approx\ & u(\mathcal{D}_{t-1}) \\ & + \Pr(y_t = 1 \mid x_t, \mathcal{D}_{t-1}) \\ & + \mathbb{E}_{y_t}\left[\max_{X \subset \mathcal{X} \setminus \mathcal{D}_t} \sum\nolimits_{x \in X} \Pr(y = 1 \mid x, \mathcal{D}_t)\right], \end{aligned} \tag{2.4}$$

where the last term is the sum of the highest posterior probabilities achieved by X, the $(\ell - 1)$ most likely targets conditioned on $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(x_t, y_t)\}$. Here, we estimate the value of each putative query x_t with the expected utility of the union of x_t and the unlabeled points

X that are adaptively selected based on each possible label y_t. Unlike myopic policies that set the lookahead horizon ℓ to be a fixed, small number, ℓ in this policy computation exactly matches the true length of the decision-making horizon: $|X| = \ell - 1$. The resulting policy thus actively accounts for the remaining labeling budget when making its queries. Jiang et al. [88] demonstrated the benefits of this budget-awareness by showing that the policy exhibits nonmyopic, exploratory behavior when the budget is large, and automatically transitions to more exploitative queries as search progresses. This strategic exploration ultimately allows their policy to outperform many myopic baselines.

For its superior search performance, this nonmyopic policy serves as the state-of-the-art in active search and the foundation of many of the works in this dissertation. For example, Chapts. 4 and 5 realize the analogous policies under novel search settings by adopting the assumption of conditional independence in future labels. Chapt. 6 on the other hand presents an imitation learning framework that trains a neural network to mimic the behavior of this state-of-the-art policy to amortize policy computation.

Chapter 3

Local Bayesian Optimization

The optimization of expensive-to-evaluate, high-dimensional black-box functions is ubiquitous in machine learning, science, engineering, and beyond; examples range from hyperparameter tuning [183] and policy search in reinforcement learning [30, 68], to configuring physics simulations [128]. High-dimensional global optimization faces an inherent difficulty stemming from the curse of dimensionality, as a thorough exploration of the search space becomes exponentially more expensive. It is more feasible to seek to *locally* optimize these high-dimensional objective functions, as we can then sidestep this inherent burden. This is true even in settings where we cannot directly observe the gradient of the objective function, as we may appeal to sophisticated techniques such as Bayesian optimization to nonetheless learn about the gradient of the objective through noisy observations, and then use this knowledge to navigate the high-dimensional search space locally.

A realization of this scheme has been proposed by Müller et al. [142], where a Gaussian process (GP) is used to model the objective function, and observations are designed to alternate between minimizing the variance – and thus uncertainty – of the GP's estimate of the gradient of the objective at a given location, then moving in the direction of the expected gradient. Although this approach seems natural, it fails to account for some nuances in the distribution of the directional derivative induced by the GP. Specifically, it turns out that beliefs about the gradient with *identical* uncertainty may nonetheless have *different* probabilities of descent along the expected gradient. Further and perhaps surprisingly, the expected gradient is not necessarily the direction maximizing the probability of descent – in fact, these directions can be nearly orthogonal. In other words, simply minimizing the gradient variance and moving in the direction of the expected gradient may lead to suboptimal (local) optimization performance.

With this insight, we propose a scheme for local Bayesian optimization that alternates between identifying the direction of most probable descent, then moving in that direction.

The result is a local optimizer that is efficient by design. To this end, we derive a closed-form solution for the direction of most probable descent at a given location in the input space under a GP belief about the objective function. We then design a corresponding closed-form acquisition function that optimizes (an upper bound of) the one-step maximum descent probability. Taken together, these components comprise an elegant and efficient optimization scheme. We demonstrate empirically that, across many synthetic and real-world functions, our method outperforms the aforementioned prior realization of this framework and is competitive against other, significantly more complicated baselines.

3.1 Preliminaries

We first introduce the problem setting and the local Bayesian optimization framework. We aim to numerically solve optimization problems of the form:

$$\text{given } \mathbf{x}_0 \in D, \text{ find } \mathbf{x}^* = \operatorname*{arg\,min}_{\mathbf{x} \in D(\mathbf{x}_0)} f(\mathbf{x}),$$

where $f\colon D \to \mathbb{R}$ is the black-box objective function we wish to optimize locally from a starting point $\mathbf{x}_0$, and $D(\mathbf{x}_0)$ is the local region around $\mathbf{x}_0$ inside the domain D. We model the objective function as a black box, and only assume that we may obtain potentially noisy function evaluations $y = f(\mathbf{x}) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, at locations of our choosing. We further assume the gradient cannot be measured directly, but only estimated from such noisy evaluations of the function. Finally, we consider the case where querying the objective is relatively expensive, limiting the number of times it may be evaluated. This constraint on our querying budget requires strategically selecting where to evaluate during optimization.

Bayesian optimization (BO) is one potential approach to this problem that offers unparalleled sample efficiency. BO constructs a probabilistic model of the objective function, typically a Gaussian process (GP) [162], and uses this model to design the next point(s) to evaluate the objective. After each observation, the GP is updated to reflect our current belief about the objective, which is then used to inform future decisions. We refer the reader to Garnett [59] for a thorough treatment of GPs and BO.

3.1.1 Local Bayesian Optimization

In many applications, the objective function f is high-dimensional. The curse of dimensionality poses a challenge for BO, as it will take exponentially more function evaluations to sufficiently cover the search space and find the global optimum. It may be more fruitful, therefore, to instead pursue *local* optimization, where we aim to descend from the current location, by probing the objective function in nearby regions to learn about its gradient.

It turns out the BO framework is particularly amenable to this idea, as a GP belief on the objective function induces a *joint* GP belief with its gradient [162], which we may use to guide local optimization. In particular, given a GP belief about the objective function f with a once-differentiable mean function μ and a twice-differentiable covariance function K, the joint distribution of noisy function evaluations observations $(\mathbf{X}, \mathbf{y})$ and the gradient of f at some point $\mathbf{x}$ is

$$p\left(\begin{bmatrix} \mathbf{y} \\ \nabla f(\mathbf{x}) \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mu(\mathbf{X}) \\ \nabla \mu(\mathbf{x}) \end{bmatrix}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I} & K(\mathbf{X}, \mathbf{x})\nabla^\top \\ \nabla K(\mathbf{x}, \mathbf{X}) & \nabla K(\mathbf{x}, \mathbf{x})\nabla^\top \end{bmatrix}\right).$$

Here, when placed in front of K, the differential operator ∇ indicates that we are taking the derivative of K with respect to its first input; when placed behind K, it indicates the derivative is with respect to its second input. Conditioned on the observations $(\mathbf{X}, \mathbf{y})$, the posterior distribution of the derivative $\nabla f(\mathbf{x})$ may be obtained as:

$$\begin{aligned} p\big(\nabla f(\mathbf{x}) \mid \mathbf{x}, \mathbf{X}, \mathbf{y}\big) &= \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \Sigma_\mathbf{x}), \\ \text{where } \boldsymbol{\mu}_\mathbf{x} &= \nabla \mu(\mathbf{x}) + \nabla K(\mathbf{x}, \mathbf{X})\big(K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}\big)^{-1}\big(\mathbf{y} - \mu(\mathbf{X})\big), \\ \Sigma_\mathbf{x} &= \nabla K(\mathbf{x}, \mathbf{x})\nabla^\top - \nabla K(\mathbf{x}, \mathbf{X})\big(K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}\big)^{-1} K(\mathbf{X}, \mathbf{x})\nabla^\top. \end{aligned} \tag{3.1}$$

Given the ability to reason about the objective function gradient given noisy function observations, we may realize a Bayesian local optimization scheme as follows. From a current location $\mathbf{x}$, we devise a policy that first designs observations seeking relevant information about the gradient $\nabla f(\mathbf{x})$, then, once satisfied, moves within the search space to a new location (that is, update $\mathbf{x}$) seeking to descend on the objective. A particular realization of this local BO scheme named GIBO was investigated by Müller et al. [142]. In that study, the authors choose to learn about $\nabla f(\mathbf{x})$ by minimizing the uncertainty (quantified by the trace of the posterior covariance matrix) about the gradient, followed by moving in the direction of the expected gradient. This algorithm may be thought of as simulating gradient descent, as it actively builds then follows a noisy estimate of the gradient. Although effective, GIBO fails to account for nuances in our belief about the objective function gradient and may behave suboptimally during optimization as a result. Our work addresses this gap by exploiting the rich structure in the belief about $\nabla f(\mathbf{x})$ to design an elegant and principled policy for local BO.

3.2 Maximizing Probability of Descent

What behavior is desirable for a local optimization routine that values sample efficiency? We argue that we should seek to quickly identify directions that will, *with high probability,* yield progress on the objective function. Pursuing this idea requires reasoning about the probability that a given direction leads "downhill" from a given location. Although one might guess that the direction most likely to lead downhill is always the (negative) expected gradient, this is not necessarily the case.

Consider the directional derivative of the objective f with respect to a unit vector $\mathbf{v}$ at point $\mathbf{x}$:

$$\nabla_{\mathbf{v}} f(\mathbf{x}) = \mathbf{v}^\top \nabla f(\mathbf{x}),$$

which quantifies the rate of change of f at $\mathbf{x}$ along the direction of $\mathbf{v}$. According to our GP belief, $\nabla f(\mathbf{x})$ follows a multivariate normal distribution, so the directional derivative $\nabla_{\mathbf{v}} f(\mathbf{x})$ is then:

$$p\big(\nabla_{\mathbf{v}} f(\mathbf{x}) \mid \mathbf{x}, \mathbf{v}\big) = \mathcal{N}\big(\mathbf{v}^\top \boldsymbol{\mu}_{\mathbf{x}}, \mathbf{v}^\top \Sigma_{\mathbf{x}} \mathbf{v}\big),$$

where $\boldsymbol{\mu}_{\mathbf{x}}$ and $\Sigma_{\mathbf{x}}$ are the mean and covariance matrix of the normal belief about $\nabla f(\mathbf{x})$, as defined in Eq. (3.1). This distribution allows us to reason about the probability that we descend on the objective function by moving along the direction of $\mathbf{v}$ from $\mathbf{x}$, which is simply the probability that the directional derivative is negative. Thus, we have the following definition.

Definition 3.2.1 (Descent probability and most probable descent direction)**.** *Given a unit vector* $\mathbf{v}$*, the descent probability of the direction* $\mathbf{v}$ *at the location* $\mathbf{x}$ *is given by*

$$\Pr\big(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\big) = \Phi\Bigg(-\frac{\mathbf{v}^\top \boldsymbol{\mu}_{\mathbf{x}}}{\sqrt{\mathbf{v}^\top \Sigma_{\mathbf{x}} \mathbf{v}}}\Bigg), \tag{3.2}$$

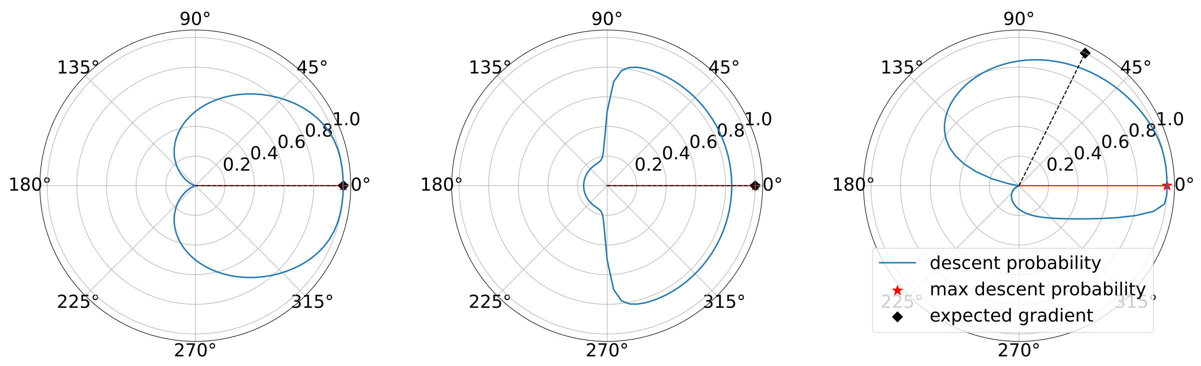
where Φ is the CDF *of the standard normal distribution. If $\mathbf{v}^*$ achieves the maximum descent probability $\mathbf{v}^* \in \arg\max_{\mathbf{v}} \Pr\left(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\right)$, then we call $\mathbf{v}^*$ a most probable descent direction.*

Note that the definition Eq. (3.2) is scaling invariant. Thus, the length of $\mathbf{v}^*$ does not matter since the descent probability only depends on its direction. Moreover, we note that descent probability depends on both the expected gradient $\boldsymbol{\mu}_{\mathbf{x}}$ and the gradient uncertainty $\Sigma_{\mathbf{x}}$. Therefore, learning about the gradient by minimizing uncertainty via the trace of the posterior covariance matrix (which does not consider the expected gradient) and moving in the direction of the negative expected gradient (which does not consider uncertainty in the gradient) in a decoupled manner may lead to suboptimal behavior. We first present a simple example to demonstrate the nuances that are not captured by this scheme and to motivate our proposed solution.

3.2.1 The (Negative) Expected Gradient Does Not Always Maximize Descent Probability

In Fig. 3.1, we show polar plots of the descent probability $\Pr\left(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\right)$ with respect to different beliefs about the gradient. The angles in the polar plots are the angles between $\mathbf{v}$ and the vector $[1, 0]^\top$. Critically for the discussion below, the uncertainty in the gradient, as measured by the trace of the covariance matrix, is identical for all three examples.

In the first example in the left panel of Fig. 3.1, the negative expected gradient happens to maximize the descent probability, and moving in this direction is almost certain to lead downhill. In the middle panel, the expected gradient is the same as in the left panel, but the covariance matrix has been permuted. Here, the negative expected gradient again maximizes the descent probability; however, the largest descent probability is now much lower. In fact, there is non-negligible probability that the descent direction is in the *opposite* direction. This is because most of the uncertainty we have about the gradient concentrates on the first element of $\boldsymbol{\mu}_{\mathbf{x}}$, which determines its direction. We note that the situation in the left panel is inarguably preferable to that in the middle panel, but distinguishing these two is impossible from uncertainty in $\nabla f(\mathbf{x})$ alone.



(a) $\mathcal{N}\left(\begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0.01 & 0\\0 & 1\end{bmatrix}\right)$ (b) $\mathcal{N}\left(\begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 0.01\end{bmatrix}\right)$ (c) $\mathcal{N}\left(\begin{bmatrix}-0.5\\-1\end{bmatrix}, \begin{bmatrix}0.01 & 0\\0 & 1\end{bmatrix}\right)$

Figure 3.1: Polar plots of descent probability (blue). The most probable descent direction $\mathbf{v}^*$ is marked in red. The direction of the (negative) expected gradient is marked in **black**. **Left:** the direction $\mathbf{v}^*$ and the negative expected gradient match exactly. **Center:** given the same level of uncertainty, the maximum descent probability has reduced from near certainty to only 84%. **Right:** the expected gradient does not maximize the descent probability. See Sect. 3.2.1 for discussion.

Finally, in the right panel, the direction of the expected gradient has rotated with respect to that in the first two panels. Now the (negative) expected gradient is entirely different from the most probable descent direction. Intuitively, the variance in the first coordinate is much smaller than in the second coordinate, and thus the mean in the first coordinate is more likely to have the same sign as the true gradient. However, using negative expected gradient as a descent direction entirely ignores the uncertainty estimate in the gradient. This example shows that, when we reason about the descent of a function, the mean vector $\boldsymbol{\mu}_{\mathbf{x}}$ and the covariance matrix $\Sigma_{\mathbf{x}}$ need to be jointly considered, as the probability of descent depends on both of these quantities (Eq. (3.2)).

3.2.2 Computing the Most Probable Descent Direction

In light of the above discussion, we propose a local BO algorithm centered entirely around the local descent probability. As a first step, we show in the following how to compute the most probable descent direction $\mathbf{v}^* = \arg\max_{\mathbf{v}} \Pr\left(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\right)$ at a given location given data.

Theorem 3.2.1. *Suppose that the belief about the gradient is* $p\big(\nabla f(\mathbf{x}) \mid \mathbf{x}, \mathbf{X}, \mathbf{y}\big) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \Sigma_{\mathbf{x}})$, *where the posterior covariance* $\Sigma_{\mathbf{x}}$ *is positive definite. Then, the unique (up to scaling) most probable descent direction is*

$$\arg\max_{\mathbf{v}} \Pr\big(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\big) = -\Sigma_{\mathbf{x}}^{-1}\boldsymbol{\mu}_{\mathbf{x}}$$

with the corresponding maximum descent probability

$$\max_{\mathbf{v}} \Pr\big(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\big) = \Phi\Big(\sqrt{\boldsymbol{\mu}_{\mathbf{x}}^\top \Sigma_{\mathbf{x}}^{-1}\boldsymbol{\mu}_{\mathbf{x}}}\Big).$$

Proof. As $\Phi(\cdot)$ is monotonic, we can reframe the problem as

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \Pr\big(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\big) = \arg\max_{\mathbf{v}} \Phi\left(-\frac{\mathbf{v}^\top \boldsymbol{\mu}_{\mathbf{x}}}{\sqrt{\mathbf{v}^\top \Sigma_{\mathbf{x}} \mathbf{v}}}\right) = \arg\max_{\mathbf{v}} -\frac{\mathbf{v}^\top \boldsymbol{\mu}_{\mathbf{x}}}{\sqrt{\mathbf{v}^\top \Sigma_{\mathbf{x}} \mathbf{v}}}.$$

Next, we square the objective, and the maximizer is still the same (up to sign). That is, if $\mathbf{v}^*$ is the maximizer of the squared objective:

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \; \frac{\mathbf{v}^\top \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^\top \mathbf{v}}{\mathbf{v}^\top \Sigma_{\mathbf{x}} \mathbf{v}}, \tag{3.3}$$

then either $\mathbf{v}^*$ or $-\mathbf{v}^*$ maximizes the descent probability. Let $\Sigma_{\mathbf{x}} = \mathbf{L}\mathbf{L}^\top$ be the Cholesky decomposition of $\Sigma_{\mathbf{x}}$, where $\mathbf{L}$ has to be nonsingular. A change of variable $\mathbf{v} = \mathbf{L}^{-\top}\mathbf{w}$ gives

$$\frac{\mathbf{v}^\top \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^\top \mathbf{v}}{\mathbf{v}^\top \Sigma_{\mathbf{x}} \mathbf{v}} = \frac{\mathbf{w}^\top \mathbf{L}^{-1}\boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^\top \mathbf{L}^{-\top}\mathbf{w}}{\mathbf{w}^\top \mathbf{w}},$$

which is exactly the Rayleigh quotient of $\mathbf{L}^{-1}\boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^\top \mathbf{L}^{-\top}$. Note that this is a rank-1 matrix with top eigenvector $\mathbf{L}^{-1}\boldsymbol{\mu}_{\mathbf{x}}$ and corresponding eigenvalue $\boldsymbol{\mu}_{\mathbf{x}}^\top \Sigma_{\mathbf{x}}^{-1}\boldsymbol{\mu}_{\mathbf{x}}$. Thus, the maximizer $\mathbf{w}^*$ is given by

$$\mathbf{w}^* = \mathbf{L}^{-1}\boldsymbol{\mu}_{\mathbf{x}}.$$

Therefore, the maximizer to Eq. (3.3) is $\mathbf{v}^* = \mathbf{L}^{-\top}\mathbf{L}^{-1}\mathbf{w}^* = \Sigma_{\mathbf{x}}^{-1}\boldsymbol{\mu}_{\mathbf{x}}$. Plug both $\Sigma_{\mathbf{x}}^{-1}\boldsymbol{\mu}_{\mathbf{x}}$ and $-\Sigma_{\mathbf{x}}^{-1}\boldsymbol{\mu}_{\mathbf{x}}$ back into Eq. (3.2). It is easy to check that the direction along $-\Sigma_{\mathbf{x}}^{-1}\boldsymbol{\mu}_{\mathbf{x}}$ is the desired maximizer. □

Theorem 3.2.1 states that the most probable descent direction can be computed by simply solving a linear system. Being able to compute this quantity allows us to always move within the search space in the direction that most likely improves the objective value, which, as we have seen, is not necessarily the negative expected gradient. This helps us to realize the "update" portion of our local BO algorithm, where we iteratively move from the current location $\mathbf{x}$ in the most probable descent direction $\mathbf{v}^*$. That is, we repeatedly update $\mathbf{x}$ with $\mathbf{x} + \delta\mathbf{v}^*$, where δ is a small constant that acts as a step size. This procedure is iterative in that we do not take one single step along a direction, but multiple small steps, always in the most probable descent direction at the current point, throughout. (Note that we do not observe the value of the objective function at any of these steps.)

It is important that we stop this iterative procedure when it becomes uncertain whether we can continue to descend. This is because we aim to move to a new location that decreases the value of the objective function, and thus should only move when descent is likely. A natural approach is to again use the maximum descent probability, which we can compute using Theorem 3.2.1. Specifically, we stop the iterative update when the maximum descent probability falls below a prespecified threshold p_*. Once we have stopped, the final updated $\mathbf{x}$ is the location we move to at the current iteration of the BO loop. In our experiments, we set the step size to $\delta = 0.001$ and the descent probability threshold to $p_* = 65\%$, which we find to work well empirically.

3.2.3 Acquisition Function via Look-Ahead Maximum Descent Probability

When the maximum descent probability falls below the threshold p_*, we begin selecting queries to learn about the gradient in the current location so as to maximize the probability of descent. Here we derive an acquisition function seeking data that will, in expectation, best improve the highest descent probability. For maximum flexibility, we consider the batch setting where we may gather multiple measurements simultaneously, although we only use the sequential case in our experiments.

In particular, the acquisition function we would like to use for a batch of potential query points $\mathbf{Z}$ is:

$$\begin{aligned}\alpha_0(\mathbf{Z}) &= \mathbb{E}_{\mathbf{y}|\mathbf{Z}}\Big[\max_{\mathbf{v}} \Pr\left(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{Z}\right)\Big] \\ &= \mathbb{E}_{\mathbf{y}|\mathbf{Z}}\left[\Phi\left(\sqrt{\boldsymbol{\mu}_{\mathbf{x}|\mathbf{Z}}^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}\boldsymbol{\mu}_{\mathbf{x}|\mathbf{Z}}}\right)\right],\end{aligned} \tag{3.4}$$

where $\boldsymbol{\mu}_{\mathbf{x}|\mathbf{Z}}$ and $\Sigma_{\mathbf{x}|\mathbf{Z}}$ are the posterior mean and covariance of the belief about $\nabla f(\mathbf{x})$, conditioned on a batch of observations at $\mathbf{Z}$ and a previously collected training set $(\mathbf{X}, \mathbf{y})$ which we have omitted for notational clarity. Note that the second equality is due to Theorem 3.2.1. The above acquisition function is exactly the look-ahead maximum descent probability. Namely, $\alpha_0(\mathbf{Z})$ is the expected maximum descent probability after querying $\mathbf{Z}$.

Unfortunately, this expectation is challenging to compute, so we opt for another acquisition function that approximates Eq. (3.4) via computing the expectation of an upper bound:

$$\alpha(\mathbf{Z}) = \mathbb{E}_{\mathbf{y}|\mathbf{Z}}\left[\boldsymbol{\mu}_{\mathbf{x}|\mathbf{Z}}^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}\boldsymbol{\mu}_{\mathbf{x}|\mathbf{Z}}\right]. \tag{3.5}$$

We discard the (monotonic and concave) transformation given by the normal CDF and square root, thus optimizing an upper bound by Jensen's inequality. The advantage to this acquisition function α is that, remarkably, it has a closed-form expression, as we show below.

Note that $\boldsymbol{\mu}_{\mathbf{x}|\mathbf{Z}} = \boldsymbol{\mu}_{\mathbf{x}} + \Sigma_{\mathbf{x}\mathbf{Z}}\Sigma_{\mathbf{Z}}^{-1}(\mathbf{y}_{\mathbf{Z}} - \boldsymbol{\mu}_{\mathbf{Z}})$, where $\mathbf{y}_{\mathbf{Z}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{Z}}, \Sigma_{\mathbf{Z}})$. Thus, the acquisition function in Eq. (3.5) is an expectation of a quadratic function over a Gaussian distribution. Let $\mathbf{L}\mathbf{L}^{\top} = \Sigma_{\mathbf{Z}}$ be the Cholesky decomposition of $\Sigma_{\mathbf{Z}}$ and denote $\mathbf{A} = \Sigma_{\mathbf{x}\mathbf{Z}}\mathbf{L}^{-\top}$. Then, the acquisition function can be written as an expectation over a standard normal $\boldsymbol{\zeta}$:

$$\alpha(\mathbf{Z}) = \mathbb{E}_{\boldsymbol{\zeta}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[(\boldsymbol{\mu}_{\mathbf{x}} + \mathbf{A}\boldsymbol{\zeta})^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}(\boldsymbol{\mu}_{\mathbf{x}} + \mathbf{A}\boldsymbol{\zeta})\right].$$

Expanding, we have:

$$(\boldsymbol{\mu}_{\mathbf{x}} + \mathbf{A}\boldsymbol{\zeta})^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}(\boldsymbol{\mu}_{\mathbf{x}} + \mathbf{A}\boldsymbol{\zeta}) = \boldsymbol{\mu}_{\mathbf{x}}^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}\boldsymbol{\mu}_{\mathbf{x}} + 2\boldsymbol{\mu}_{\mathbf{x}}^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}\mathbf{A}\boldsymbol{\zeta} + \boldsymbol{\zeta}^{\top}\mathbf{A}^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}\mathbf{A}\boldsymbol{\zeta}.$$

The expectation of each term can be computed in closed form. The first term is a constant and the second term vanishes. Finally, the third term is the expectation of a quadratic form, yielding:

$$\alpha(\mathbf{Z}) = \boldsymbol{\mu}_{\mathbf{x}}^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}\boldsymbol{\mu}_{\mathbf{x}} + \operatorname{tr}\left(\mathbf{A}^{\top}\Sigma_{\mathbf{x}|\mathbf{Z}}^{-1}\mathbf{A}\right).$$

Algorithm 2 Local BO via MPD

```
1: inputs starting location $\mathbf{x}$, number of iterations $N$, number of samples for learning the
   gradient $M$, step size $\delta$, and minimum descent probability threshold $p_*$.
2: Initialize the GP.
3: for $t = 0, \dots, N$ do
4:     Observe the objective value: $y = f(\mathbf{x}) + \varepsilon$.
5:     Update the training data $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{x}, y)\}$ and retrain the GP.
6:     for $m = 1, \dots, M$ do                                         ▷ learn the gradient
7:         Query point: $\mathbf{z}^* = \arg\max_{\mathbf{z}} \alpha(\mathbf{z})$.
8:         Observe the objective value: $y_{\mathbf{z}} = f(\mathbf{z}) + \varepsilon$.
9:         Update the training data $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{z}, y_{\mathbf{z}})\}$ and the GP.
10:    end for
11:    while $\max_{\mathbf{v}} \Pr\big(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\big) > p_*$ do          ▷ move iteratively
12:        Compute most probable descent direction $\mathbf{v}^* \leftarrow \arg\max_{\mathbf{v}} \Pr\big(\nabla_{\mathbf{v}} f(\mathbf{x}) < 0 \mid \mathbf{x}, \mathbf{v}\big)$.
13:        Move in the most probable descent direction: $\mathbf{x} \leftarrow \mathbf{x} + \delta \mathbf{v}^*$.
14:    end while
15: end for
```

This compact expression gives the closed-form solution to our acquisition function. Note that solving a linear system with respect to $\Sigma_{\mathbf{x}|\mathbf{Z}}$ can be performed efficiently using low-rank updates to the Cholesky decomposition of $\Sigma_{\mathbf{x}}$. Further, we may differentiate the acquisition function easily via automatic differentiation. This allows us to optimize the acquisition function trivially using any gradient-based optimizer such as L-BFGS with restart.

This completes our algorithm, local BO via most-probable descent, or MPD, which is summarized in Alg. 2. The algorithm alternates between learning about the gradient of the objective function using the acquisition function discussed above, and then iteratively moving in the most probable descent direction until further progress is unlikely, as described in Sect. 3.2.

3.3 Experiments

We now present results from extensive experiments that evaluate our method MPD against three baselines: (1) GIBO [142], which performs local BO by minimizing the trace of the posterior covariance matrix of the gradient and uses the expected gradient in the update step; (2) ARS [131], which estimates the gradient of the objective via finite difference with random perturbations; and (3) TuRBO [52], a trust region-based Bayesian optimization method.

Müller et al. [142] provide code implementation under the MIT license for GIBO, ARS, and various test objectives. We extend this codebase to implement MPD and conduct our own numerical experiments. For the synthetic (Sect. 3.3.1) and reinforcement learning (Sect. 3.3.2) objectives, we use the provided experimental settings. For the other objectives (Sect. 3.3.3), we set the number of samples to learn about the gradient per iteration $M = 1$. For each objective function tested, we run each algorithm ten times from the same set of starting points sampled from a Sobol sequence over the (box-bounded) domain. In each of the following plots, we show the progressive mean objective values as a function of the number of queries with error bars indicating (plus or minus) one standard error.

3.3.1 Synthetic Objectives

Our first experiments involve maximizing, over the d-dimensional unit hypercube $[0, 1]^d$, synthetic objective functions that are generated by drawing samples from a GP with an RBF kernel. We refer to §4.1 of Müller et al. [142] for more details regarding the experimental setup. While Müller et al. [142] tested for dimensions up to 36, we opt for much higher-dimensional objectives: $d \in \{25, 50, 100\}$. Each run has a budget of 500 function evaluations. We visualize the results in Fig. 3.2, which shows that MPD was able to optimize these functions at a faster rate than the other baselines. Note that the difference in performance becomes larger as the dimensionality d grows, indicating that our method scales well to high dimensions.

3.3.2 MuJoCo Objectives

The second set of experiments are reinforcement learning MuJoCo locomotion tasks [191], where each task involves learning a linear policy that maps states to actions to maximize the reward received from the learning environment. We use the same three environments in Müller et al. [142], CartPole-v1 with 4 parameters, Swimmer-v1 with 16, and Hopper-v1 with 33, to evaluate the methods and show the results in Fig. 3.3. MPD is competitive in the first two tasks but progresses slower than the other baselines on Hopper-v1.

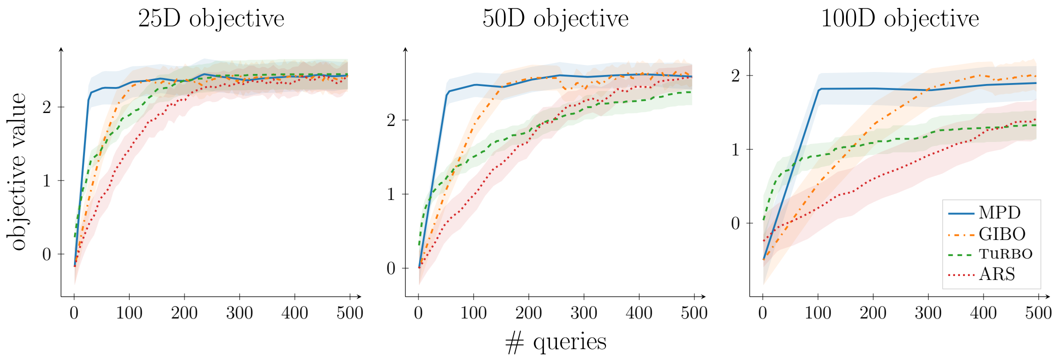


Figure 3.2: Progressive optimized objective value on high-dimensional synthetic functions. MPD consistently finds higher objective values faster than other baselines.

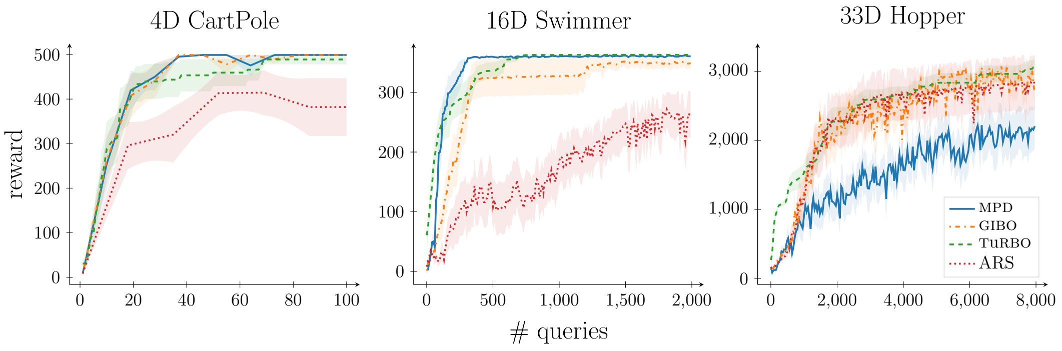


Figure 3.3: Progressive objective values observed on the MuJuCo tasks. MPD is competitive on CartPole and Swimmer.

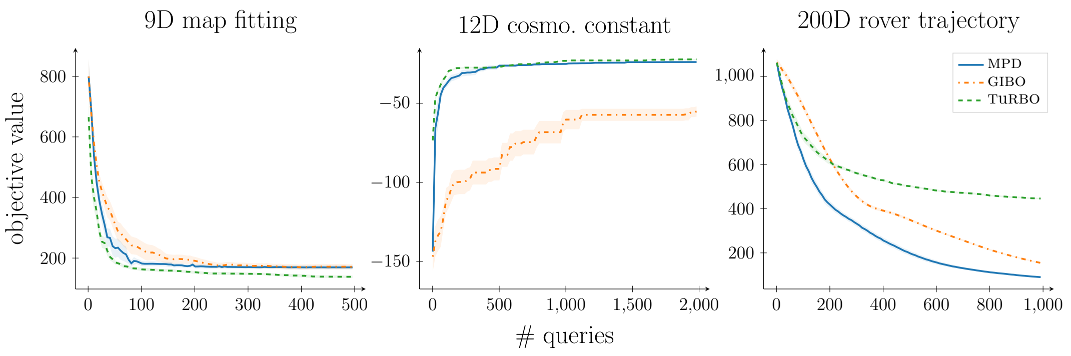


Figure 3.4: Progressive objective values observed on real-world tasks. MPD is competitive against other baselines on all tasks.

3.3.3 Other Objective Functions

We further evaluate our method on other real-world objective functions. The first two functions represent inverse problems from physics and engineering. The first is from electrical engineering, where we seek to maximize the fit of a theoretical physical model of an electronic circuit to observed data. There are nine parameters in total, and we set the budget to 500 evaluations. The second is a problem from cosmology [165], where we aim to configure a cosmological model/physical simulator to fit data observed from the Universe. In particular, our objective is to maximize the log likelihood of the physical model parameterized by various physics-related constants that are to be tuned. We follow the setting in Eriksson et al. [52], which presents a harder optimization problem with 12 parameters and much larger bounds, and set the budget at 2000 evaluations. Our third objective function uses the rover trajectory planning problem [202]. This involves tuning the locations of 100 points on a two-dimensional space that map the trajectory of a rover to minimize a cost, thus making up a 200-dimensional optimization problem. We set the budget to be 1000 function evaluations.

We visualize optimization performance on these three objective functions in Fig. 3.4. Our proposed policy MPD is consistently competitive against both GIBO and TuRBO. Most notably, in the cosmological constant learning problem, MPD was able to make significant progress immediately and ultimately outperforms its closest spiritual competitor GIBO.

3.3.4 Ablation Study

We now present results from various ablation studies to offer insight into the components of our method MPD and its hyperparameters, specifically the descent probability threshold p_* (65% as the default) and the step size δ (0.001 as the default), as described in Sect. 3.2.

First, one may reasonably ask which of the two novel components of MPD – either the learning phase that seeks to maximize expected posterior descent probability, or the update phase that moves in the most probable descent direction – is responsible for the performance improvement compared to GIBO. We address this question by comparing the performance of MPD against two variants: (1) trace + **mpd**, which learns about the gradient by minimizing the trace of the posterior covariance matrix and moves in the most probable descent direction, and (2) **mpd + expected gradient**, which uses our scheme for identifying the most

Table 3.1: Average terminal optimized objective values and standard errors of different variants of MPD. Results that are better than those of our baseline GIBO are highlighted **bold**.

	16D Swimmer (maximization)	12D cosmo. constant (maximization)	200D rover trajectory (minimization)
MPD$\big(p_* = 65\%, \delta = 10^{-3}\big)$	**360.50 (0.61)**	**−23.97 (0.34)**	**89.89 (3.88)**
GIBO	348.88 (10.11)	−55.25 (3.23)	152.77 (2.26)
trace + MPD	350.58 (9.35)	**−27.72 (1.16)**	**84.17 (2.10)**
MPD + expected gradient	340.12 (12.75)	**−21.24 (0.04)**	293.08 (8.12)
MPD$(p_* = 50\%)$	342.36 (13.10)	**−24.29 (0.10)**	**51.48 (3.44)**
MPD$(p_* = 85\%)$	294.67 (38.16)	**−31.08 (0.86)**	**142.63 (5.57)**
MPD$(p_* = 95\%)$	15.97 (5.46)	**−31.86 (0.25)**	**140.44 (6.95)**
MPD$\big(\delta = 10^{-4}\big)$	**362.06 (0.63)**	**−24.22 (0.53)**	**90.99 (3.29)**
MPD$\big(\delta = 10^{-2}\big)$	350.15 (10.92)	**−25.73 (0.39)**	**98.72 (4.42)**

probable descent direction, then moves in the direction of the (negative) expected gradient. The second section of Tab. 3.1 shows the average terminal objective values of these MPD variants on three tasks that MPD outperforms GIBO: Swimmer-v1, cosmological constant learning, and rover trajectory planning. We observe that swapping out either component of MPD does not consistently improve from GIBO as much as MPD does. This indicates that the two components of our MPD algorithm work in tandem and both are needed to successfully realize our local BO scheme.

In particular, the components of our method are coupled: because the expected gradient and the most probable descent direction are not the same in general, spending evaluation budget to learn about one and then using the other to move may not work well. GIBO's acquisition function minimizes the trace of the posterior covariance and therefore aims to make the expected gradient estimate more accurate, but it is unclear whether it will necessarily estimate the most probable descent direction accurately. On the other hand, our acquisition function focuses on the one-step maximum descent probability directly. GIBO's "moving" policy, moving in the direction of the (negative) expected gradient (which may not be the most probable descent direction), may not necessarily benefit from having a descent direction with a high descent probability (which could point in a different direction), and is therefore incompatible with our acquisition function.

We also tested MPD with three other values for the minimum descent probability threshold $p_* \in \{50\%, 85\%, 95\%\}$ (described in Sect. 3.2). The first variant with $p_* = 50\%$ is less conservative when moving to a new location than our default policy with $p_* = 65\%$, while the other two variants are more conservative. In the third section of Tab. 3.1, we observe that the more conservative variants of MPD are not as competitive. For example, MPD($p_* = 85\%$) sees a drop in performance on the Swimmer task, while MPD($p_* = 95\%$) fails to make significant progress altogether. Interestingly, while the less conservative policy with $p_* = 50\%$ also does not perform as well on the two Mujoco tasks, we do observe an increase in performance in the rover trajectory planning problem. From our experiments, we find that this rover objective function is piecewise linear within most of its domain, making finding a descent direction "easier" and allowing a lower value of p_* to perform better.

The interpretation of the threshold p_* is quite natural: it sets a threshold of the minimum probability that we would make progress by moving to a new location. Intuitively, this hyperparameter trades off robustness versus optimism, with higher thresholds spending more budget before moving, but being more confident in their moves. While $p_* = 65\%$ performs well in our experiments, a user can set their own threshold depending on their use case. As observed with the rover trajectory planning problem, if there are structures within the objective function that make it "easy" to find a descent direction, MPD may benefit from a lower threshold. We might also consider dynamically setting the value of p_* based on recent optimization progress – that is, we might increase p_* if we believe that we are approaching a local optimum and therefore that finding a promising descent direction is becoming more challenging.

Finally, the lower section of Tab. 3.1 shows the performance of the variants of MPD with two additional step sizes, 10^{-4} and 10^{-2}. We observe that MPD with $\delta = 10^{-2}$ occasionally fails to perform better than GIBO, illustrating the potentially detrimental effect of a step size that is too large. This step size parameter δ balances between faster convergence and taking steps that are too large, analogous to gradient descent, and may even be problem dependent. It would be additionally interesting to analyze whether there are good "rules of thumb" for setting δ based on the length scale of the GP, as smoother functions can likely support larger step sizes.

3.4 Conclusions

We develop a principled local Bayesian optimization framework that revolves around maximization of the probability of descending on the objective function. This novel scheme is realized with (1) an update rule that iteratively moves from the current location in the direction of maximum descent probability, and (2) a mathematically elegant, computationally convenient acquisition function that aims to maximize the probability of descent prior to our next move. Our extensive experiments show that our policy outperforms natural baselines on a wide range of applications.

(Local) Bayesian optimization has seen a wide range of applications across science, engineering, and beyond; an extensive annotated bibliography of these applications was compiled by Garnett [59] [appendix D]. However, it is possible to leverage BO for nefarious purposes as well; a concrete example is constructing adversarial attacks on machine learning models [187, 198]. Further, BO requires human expertise and ethical considerations in many important applications, and fully automated optimization systems may run the risk of perpetuating misaligned goals in machine learning. The authors judge the potential positive impacts on society resulting from better methods for local optimization to outweigh the potential negative impacts.

Chapter 4

Multifidelity Active Search

While the last chapter discusses our contribution to (local) Bayesian optimization, an experimental design framework for optimizing functions, from this chapter onwards, we will examine active search, an alternative framework for modeling search and discovery tasks. We begin with multifidelity active search in this chapter.

The goal of active search is to identify members of a rare and valuable class among a large pool of unlabeled points. This is a simple model of many real-world discovery problems, such as drug discovery and fraud detection. Active search proceeds by successively querying an oracle that returns a binary label indicating whether or not a chosen data point exhibits the desired properties. In many applications, this oracle is expensive, limiting the number of queries that could be made. The challenge is to design a policy to sequentially query the oracle in order to discover as many targets as possible, subject to a given labeling budget.

Active search has been extensively studied under various settings [61, 88, 89, 90]. Notably, Jiang et al. [88] proved a hardness of approximation result, showing that no polynomial-time policy can approximate the performance of the Bayesian optimal policy within any constant factor. Thus active search is surprisingly hard. However, the authors also proposed an efficient approximation to the optimal policy that delivers impressive empirical performance. The key feature of their policy is its ability to account for the remaining budget and dynamically trade off exploration and exploitation.

Most previous work on active search has assumed access to only a single expensive oracle providing labels. In practice, however, we may have several methods of probing the search space, including cheap, low-fidelity surrogates. For example, a computational simulation may serve as a noisy approximation to an experiment done in a laboratory, and we may reasonably seek to use a cheaper computational search to help design the expensive experiments. This motivates the problem of *multifidelity active search,* where oracles of different fidelities may be simultaneously accessed to accelerate the search process. The central question in this task is how to effectively leverage these oracles in order to maximize the rate of discovery.

We present a model for multifidelity active search and study the problem under the framework of Bayesian decision theory. We propose a novel policy for this setting inspired by the state-of-the-art single-fidelity policy mentioned above. The result is an efficient approximation to the optimal multifidelity policy that is specifically tailored to take advantage of low-fidelity oracles. We also create aggressive branch-and-bound pruning strategies to increase the efficiency of our proposed algorithm, enabling scaling to large datasets. In a series of experiments, we investigate the performance of our policy on several real-world datasets for scientific discovery. Our solution outperforms various baselines from the literature by a large margin.

4.1 Problem Definition

We briefly reintroduce the active search formulation here. Suppose we are given a finite set of points $\mathcal{X} \triangleq \{x_i\}$, which includes a rare, valuable subset $\mathcal{T} \subset \mathcal{X}$. The members of $\mathcal{T}$, which we call *targets* or *positives*, are not known *a priori*, but whether a given point $x \in \mathcal{X}$ is a target can be determined by making a query to an oracle that returns the binary label $y \triangleq \mathbb{I}\{x \in \mathcal{T}\}$. We assume that querying the oracle is expensive and that we only have a limited budget of t queries to do so. We denote a given dataset of queried points and their corresponding labels as $\mathcal{D} = \{(x_i, y_i)\}$. At times, we will use $\mathcal{D}_i$ to denote the dataset collected after i queries.

4.1.1 Multifidelity Active Search

In addition to the expensive oracle that returns the exact label of a query, we have access to other cheaper but more noisy oracles that are low-fidelity approximations to the exact oracle. We limit our setting to two levels of fidelity – one exact oracle and one noisy oracle, denoted as H and L, respectively – but note that our proposed algorithm can be extended to more than two fidelities with minor modifications, as we will discuss later.

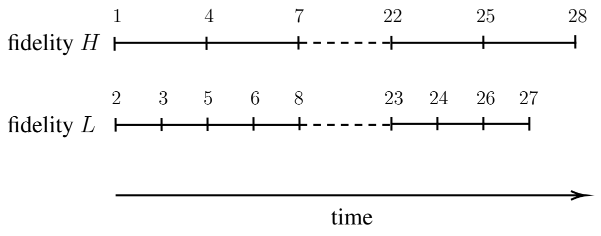


Figure 4.1: An illustration of our multifidelity active search model. Each short vertical line indicates when a query finishes and the next query is made. The numbers indicate the order in which queries are made across the two oracles. The low-fidelity oracle is $k = 2$ times faster than the high-fidelity oracle. The budget on H is $t = 10$; in total, $T = t + kt - k = 28$ queries are made.

For a given point $x \in \mathcal{X}$, we denote its exact label on H as y_H and its noisy label on L as y_L. Under this setting, a dataset $\mathcal{D}$ can be partitioned into $\mathcal{D}_L$, the observations made on L, and $\mathcal{D}_H$, those that are made on H. Recall that our objective is to query as many targets as possible; as such, to express our preference over different datasets, we use the natural utility function

$$u\left(\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_H\right) \triangleq \sum_{y_i \in \mathcal{D}_H} y_i,$$

the number of targets discovered on the H fidelity.

We assume that queries to different oracles are run *in parallel*, but a high-fidelity query takes longer to complete than a low-fidelity one. In particular, we will assume that each query on H is k times slower than a query on L.[2] That is, each time we make a query on H, we may make k sequential queries on L while waiting for the result. The process will then repeat until the budget is depleted. This model aims to emulate real-life scientific discovery procedures, the main motivation for active search, where multiple fidelities (e.g., computational and experimental campaigns) are often run in parallel but vary in response time.

If we are given a budget of t queries on H, we can make $kt - k$ queries on L; in total, $T = t + kt - k$ queries are made. (Although a total of kt queries can be made on L, after the final query on H is made at iteration T, further queries on L do not affect the final utility.

[2]This assumption is for simplicity; *asynchronous* queries could be addressed with a slight modification of our proposed algorithm.

We thus terminate after T queries.) This query schedule illustrated in Figure Fig. 4.1 for $t = 10$ and $k = 2$.

4.1.2 The Bayesian Optimal Policy

We now derive the optimal (expected-case) policy using Bayesian decision theory. First, we assume access to a probabilistic classification model that computes the posterior probability that a point $x \in \mathcal{X}$ is a target given a dataset $\mathcal{D}$, $\Pr(y_H = 1 \mid x, \mathcal{D})$, as well as the posterior probability that the same x is a positive on L, $\Pr(y_L = 1 \mid x, \mathcal{D})$.

Suppose we are currently at iteration $i + 1 \leq T$, having observed the dataset $\mathcal{D}_i$, and now need to make the next query, requesting the label of an unlabeled point x_{i+1}. The Bayesian optimal policy selects the point that maximizes the expected utility of the terminal dataset $\mathcal{D}_T$, assuming future queries will too be made optimally:

$$x^*_{i+1} = \operatorname*{arg\,max}_{x_{i+1} \in \mathcal{X} \setminus \mathcal{D}_i} \mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_{T-1} \mid x_{i+1}, \mathcal{D}_i\right)\big].$$

If the query is on H, the expectation is taken over the posterior distribution of the label of the putative query x_{i+1}; if the query is on L, it is over the *joint* distribution of the L label of x_{i+1} and the label of the pending H query (recall that we sequentially query k points on L while waiting for an H query that runs in parallel to finish).

To compute this expected utility, we follow the backward induction procedure described in Bellman [18]. In the base case, we are at iteration T and need to make the last H query and

$$\begin{aligned}
&\mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_{T-1} \mid x_T, \mathcal{D}_{T-1}\right)\big] \\
&\qquad = \sum_{y_H} u\left(\mathcal{D}_T \setminus \mathcal{D}_{T-1}\right) \Pr\left(y_H \mid x_T, \mathcal{D}_{T-1}\right) \\
&\qquad = \Pr\left(y_H = 1 \mid x_T, \mathcal{D}_{T-1}\right).
\end{aligned}$$

The optimal decision at this final step is therefore to greedily query the point most likely to be a target, maximizing $\Pr(y_H = 1 \mid x_T, \mathcal{D}_{T-1})$. For the second-to-last query, which is on L, the expected utility is

$$\mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_{T-2} \mid x_{T-1}, \mathcal{D}_{T-2}\right)\big] = \mathbb{E}\left[\max_{x_T} \Pr\left(y_H = 1 \mid x_T, \mathcal{D}_{T-1}\right)\right].$$

This is the expected future reward to be collected at the next and final step, which we have shown to be optimally the greedy query. Again, at this iteration, the second-to-last H query is still pending, so the expectation is taken over the joint distribution of the label of that query and that of the putative L query. In general, we compute this expected utility at iteration $i+1 \leq T$ for an L query recursively as

$$\mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_i \mid x_{i+1}, \mathcal{D}_i\right)\big] = \mathbb{E}\left[\max_{x_{i+2}} \mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_{i+1}\right) \mid x_{i+2}, \mathcal{D}_{i+1}\big]\right]. \tag{4.1}$$

For an H query, this expected utility may still be recursively computed in the same manner but has a different expansion:

$$\begin{aligned}\mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_i \mid x_{i+1}, \mathcal{D}_i\right)\big] &= \Pr\left(y_H = 1 \mid x_{i+1}, \mathcal{D}_i\right) \\ &\quad + \mathbb{E}\left[\max_{x_{i+2}} \mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_{i+1}\right) \mid x_{i+2}, \mathcal{D}_{i+1}\big]\right].\end{aligned} \tag{4.2}$$

The new term in (Eq. (4.2)), $\Pr\left(y_H = 1 \mid x_{i+1}, \mathcal{D}_i\right)$, accounts for the possibility that our running reward increases if $x_{i+1} \in \mathcal{T}$, as we are querying on fidelity H. This is simply the probability that x_{i+1} is indeed a target. Also different from (Eq. (4.1)), the expectation is taken over the distribution of the label y_H of the putative point x_{i+1} only, as there is no pending query at this time. An intuitive interpretation of the sum in (Eq. (4.2)) is the balance between exploitation, the immediate reward $\Pr\left(y_H = 1 \mid x_{i+1}, \mathcal{D}_i\right)$, and exploration, the expected future reward $u\left(\mathcal{D}_T \setminus \mathcal{D}_{i+1}\right)$ conditioned on this putative query. Again, the exploitation term is not present in (Eq. (4.1)) when an L query is made, as the query cannot possibly increase our utility immediately.

Perhaps unsurprisingly, computing the optimal policy is a daunting task: the time complexity of computing the expectation term in (Eq. (4.1)) and (Eq. (4.2)) is $\mathcal{O}\big((4n)^{\ell}\big)$, where $\ell = T - i$ is the number of remaining queries and n is the number of unlabeled points. This optimal policy is computationally intractable, and suboptimal approximations are needed in practice. One natural solution is to shorten the lookahead horizon, pretending there are only $\ell' < T - i$ iterations remaining. This idea constitutes *myopic* policies, the most straightforward of which is the greedy strategy that queries the point with the highest probability of being a target in the single-fidelity setting, setting $\ell' = 1$. However, the design of a greedy policy for an L query is not obvious, as no immediate reward can be obtained by making the query.

4.1.3 Hardness of Approximation

In addition to demonstrating the intractability of the analogous Bayesian optimal policy in the classical single-fidelity setting, Jiang et al. [88] proved that *no* polynomial-time policy can approximate the optimal policy (in terms of the expected terminal utility) by *any* constant factor. This was done by constructing an adversarial family of active search problems featuring "hidden treasures" that are provably difficult to uncover without exponential work. By modifying details of this construction, the performance of any polynomial-time policy can be made arbitrarily worse in expectation than that of the optimal policy.

This hardness result naturally extends to our multifidelity model, as even access to a *perfectly faithful* low-fidelity oracle cannot aid a polynomial-time active search policy in one of these adversarial examples. If we assume positives on L also count towards our utility, one such active search problem with a faithful low-fidelity oracle reduces to a single-fidelity problem with a budget on H increased by a factor of $(k + 1)$. Unless k is *exponential* in the initial budget, any polynomial-time policy remains incapable of approximating the optimal policy. Under our model, only positives on H count towards our utility, so the performance of any policy is further reduced. We therefore obtain the same hardness of approximation result. However, we can still reasonably aim to design efficient approximations to the optimal policy that perform well in practice.

4.2 Efficient Nonmyopic Approximation

Our proposed algorithm is inspired by the ENS policy, introduced by Jiang et al. [88] for the single-fidelity active search setting. ENS offers an efficient and nonmyopic approximation to the optimal policy by assuming that after the putative query, all remaining budget will be spent *simultaneously in one batch*. Under this heuristic, the optimal decision following the putative query is to greedily construct the batch with points having the highest probabilities. The expected utility of this batch is simply the sum of these highest probabilities due to linearity of expectation and therefore can be computed efficiently. An interesting interpretation by Jiang et al. [88] about ENS is that it matches the optimal policy given that after the putative query, the labels of the remaining unlabeled points are conditionally independent.

4.2.1 One-stage Approximation

As a stepping stone to our proposed policy, consider the following adoption of ENS. We reapply the heuristic by assuming that after the putative query, which can be on either L or H, all remaining H queries will be made simultaneously in one batch. Again, the optimal choice for this batch is the set of most promising points, which we will refer to as the greedy H batch. Using the summation-prime symbol $\sum'_s$ to denote the sum of the top s terms, we approximate the maximum expected utility of the remaining portion of the search in (Eq. (4.1)) and (Eq. (4.2)) as

$$\max_{x_{i+2}} \mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_{i+1}\right) \mid x_{i+2}, \mathcal{D}_{i+1}\big] \approx \sum\nolimits'_{\ell_H} \Pr\left(y_H = 1 \mid x_{i+2}, \mathcal{D}_{i+1}\right), \tag{4.3}$$

where $\ell_H = \lfloor (T-i-1)/(k+1) \rfloor$ is the number of remaining H queries. The resulting policy queries the point that maximizes the expected utility in (Eq. (4.1)) and (Eq. (4.2)), using (Eq. (4.3)) as an approximation. This policy is cognizant of the remaining budget on H. However, by assuming that the greedy H batch will be queried immediately following the putative point, the strategy fails to consider future queries that could be made to the low-fidelity oracle in its lookahead; this motivates the design of our proposed policy described below.

4.2.2 Two-stage Approximation

Our main contribution is a policy we call MF–ENS, an efficient approximation to the optimal policy that additionally accounts for future L queries. As above, we assume in our lookahead that after the putative query, our H budget will be spent simultaneously. However, prior to committing to that final batch, we assume we may make $\overline{k} \leq k$ additional L queries (also simultaneously) with the goal of improving the expected utility of the final H batch. To faithfully emulate our search model, we set $\overline{k}$ to be the number of L queries remaining before the next H query is made.[3] We denote this batch of exploratory L queries as $X_L \subset \mathcal{X} \setminus \mathcal{D}_L$ and the corresponding labels as $Y_L \in \{0,1\}^{\overline{k}}$. In the language of *approximate dynamic programming* [21], the policy described in the previous subsection is a *one-stage rollout* policy where the base policy selects the optimal batch on H following the putative query. MF–ENS on the other hand is a *two-stage rollout* policy whose base policy first queries the L batch X_L, and upon observing Y_L, adaptively queries the updated optimal H batch.

How should we construct X_L to best improve the greedy H batch at the second stage? One option would be to appeal to nonmyopic policies for *batch active search* [89], but the best-promising batch policies become prohibitively expensive in this context as we would need to construct a new batch for *each* putative query and label. In general, the conditional dependence among the labels in the set Y_L poses a computational challenge in approximating the expected utility gained from querying a given X_L batch.

Recall that in the single-fidelity setting, the ENS policy is optimal if labels become conditionally independent after the chosen point. Let us make a similar assumption to ease computation: we assume that labels become conditionally independent after the putative query, *except* for the pair of labels (on H and L) corresponding to each point. That is, revealing the label y_L of a point x is allowed to affect our belief about the corresponding label y_H, but not the belief about any other label of any other point. This structure allows for efficiently sharing information between the fidelities and enables our multistage lookahead approach.[4]

We now aim to quantify the value of querying an unlabeled point on L in improving the final H batch. Our solution is motivated by a heuristic search for alternatives to the members of the greedy H batch that is assembled under the one-stage approximation. Given a putative query, we still construct that same greedy H batch. Then, for each candidate x not in the greedy H batch, we consider the expected marginal gain in utility of querying it on L and modifying the membership of the H batch in light of its newly revealed label y_L.

Recall that observing y_L only changes the distribution of y_H of the same x under our assumption. Let p_* be the lowest success probability among the current H batch and consider two cases. If $\Pr(y_H = 1 \mid x, \mathcal{D})$ *exceeds* p_* as y_L is revealed, we swap out the corresponding

[3] When making an H query, $\overline{k} = k$; when making an L query, we subtract the number of L queries since the pending H query.

[4] This is only used in policy construction and not in inference!

least-promising member of the batch with x and thus increase the final batch's expected utility. Otherwise, we do not modify the current batch. The value of querying x on L and observing y_L, denoted as $v\left(x \mid \mathcal{D}\right)$, is then

$$v\left(x \mid \mathcal{D}\right) \triangleq \mathbb{E}_{y_L}\big[\max\left(\Pr\left(y_H = 1 \mid x, y_L, \mathcal{D}\right) - p_*, 0\right)\big].$$

This score can be rapidly computed for most models under our conditional independence assumption. With this value function in hand, we then greedily construct X_L with the candidates having the highest values. Another computational benefit of label independence is that $v\left(x \mid \mathcal{D}\right)$ automatically vanishes for points already labeled on H, as querying its L label does not affect the belief of our model and thus cannot improve the H batch. This reduces our search space in computing the exploratory batch X_L.

We now proceed to the last step of the rollout procedure in MF–ENS: marginalizing over Y_L, the labels of X_L. As previously described, for each possible value of Y_L, we approximate the optimal sequence of queries following the putative one and X_L with the updated greedy H batch given the newly revealed labels Y_L:

$$\max_{x_{i+1}} \mathbb{E}\big[u\left(\mathcal{D}_T \setminus \mathcal{D}_i\right) \mid x_{i+1}, \mathcal{D}_i\big] \approx \mathbb{E}_{Y_L}\left[\sum\nolimits'_{\ell_H} \Pr\left(y_H = 1 \mid x, X_L, Y_L, \mathcal{D}_i\right)\right]. \qquad (4.4)$$

At each iteration, MF–ENS queries the candidate maximizing the expected utility in (Eq. (4.1)) and (Eq. (4.2)), as approximated by (Eq. (4.4)). As an extension of ENS, our approach is non-myopic and aware of the remaining budget; we will demonstrate the impact of this reasoning in our experiments. The policy also factors in future L queries, actively taking advantage of the ability to query the low-fidelity oracle.

We give the pseudocode for MF–ENS in Algorithm Alg. 3 (for H queries) and Algorithm Alg. 4 (for L queries). The two versions are nearly identical; however, in Algorithm Alg. 4, we marginalize both the putative label y_L and the pending label y_H for L queries, and the $\Pr(y_H = 1 \mid x, \mathcal{D})$ term is dropped from the final line.

4.2.3 Extension to Multiple Low Fidelities

So far, we have assumed our model only consists of one exact oracle H and one noisy oracle L. To extend MF–ENS to settings where there are multiple noisy oracles $\{L_i\}$ that approximate

Algorithm 3 MF–ENS for H queries

```
Require: $x$, $\mathcal{D}$
Ensure: approximate expected utility for querying $x$ on $H$, $f(x)$   ▷ design by maximizing
1: for $y_H \in \{-,+\}$ do   ▷ probabilities: $\Pr(y_H \mid x, \mathcal{D})$
2:     determine $X_L$ by finding top-$k$ $v$ scores among unlabeled $L$ points
3:     for $Y_L \in \{-,+\}^k$ do   ▷ probabilities: $\Pr(Y_L \mid x, y_H, X_L, \mathcal{D})$
4:         $s\,(x, y_H, X_L, Y_L) \leftarrow \sum'_{\ell_H} \Pr\,(y'_H = 1 \mid x', x, y_H, X_L, Y_L)$   ▷ (4)
5:     end for
6:     $f\,(x \mid y_H) \leftarrow \mathbb{E}_{Y_L}\big[s\,(x, y_H, X_L, Y_L) \mid x, y_H, X_L\big]$   ▷ (4)
7: end for
8: return $f(x) \leftarrow \Pr\,(y_H = 1 \mid x, \mathcal{D}) + \mathbb{E}_{y_H}\big[f\,(x \mid y_H) \mid x, \mathcal{D}\big]$   ▷ (2)
```

Algorithm 4 MF–ENS for L queries

```
Require: $x$, $\mathcal{D}$
Ensure: approximate expected utility for querying $x$ on $L$, $f(x)$   ▷ design by maximizing
1: for $y_L, y_H \in \{-,+\}^2$ do   ▷ probabilities: $\Pr(y \mid x, \mathcal{D})$
2:     determine $X_L$ by finding top-$\overline{k}$ $v$ scores among unlabeled $L$ points
3:     for $Y_L \in \{-,+\}^{\overline{k}}$ do   ▷ probabilities: $\Pr(Y_L \mid x, y_L, X_L, \mathcal{D})$
4:         $s\,(x, y_L, X_L, Y_L) \leftarrow \sum'_{\ell_H} \Pr\,(y'_H = 1 \mid x', x, y_L, X_L, Y_L)$   ▷ (4)
5:     end for
6:     $f\,(x \mid y_L) \leftarrow \mathbb{E}_{Y_L}\big[s\,(x, y_L, X_L, Y_L) \mid x, y_L, X_L\big]$   ▷ (4)
7: end for
8: return $f(x) \leftarrow \mathbb{E}_{y_L}\big[f\,(x \mid y_L) \mid x, \mathcal{D}\big]$   ▷ (1)
```

H, potentially at different levels of fidelity, we still aim to design each query to maximize the expected utility on H, marginalizing future experiments on the $\{L_i\}$ oracles. Once again limiting the conditional dependence among labels to those of the same point, now between *each* lower-fidelity L_i and H, we identify a batch of appropriate size for each L_i, marginalize their labels, update the probabilities on H, and compute the approximate expected utility. When there is only one low fidelity, this strategy reduces to the base version of MF–ENS we have presented here.

4.2.4 Implementation and Pruning

Active search requires a classification model computing a given point's success probability with an oracle. We extend the k-nearest neighbor introduced by Garnett et al. [61] to our multifidelity setting by allowing information observed on L to propagate to H. Specifically, when calculating the probability that an unlabeled point $x \in \mathcal{X}$ is a positive on H, $\Pr(y_H = 1 \mid x, \mathcal{D})$, we take into account the revealed labels of its nearest neighbors on both fidelities, as well as its own L label, y_L. Effectively, we treat each given point $x \in \mathcal{X}$ as having two separate copies: one corresponding to its H label, denoted as x_H, and one corresponding to its L label, denoted as x_L. Compared to the single-fidelity k-nearest neighbor, the set of nearest neighbors of x_H is now doubled to include both copies of its original neighbors and its own copy on L, x_L. Formally, denote $\textsc{nn}_{\text{single}}(x)$ as the original nearest neighbor set of x. The nearest neighbor set of x_H under our multifidelity predictive model is

$$\begin{aligned}\textsc{nn}(x_H) \triangleq \{x_L\} &\cup \left\{x'_H : x' \in \textsc{nn}_{\text{single}}(x)\right\} \\ &\cup \left\{x'_L : x' \in \textsc{nn}_{\text{single}}(x)\right\}.\end{aligned}$$

To account for the unknown accuracy of the low-fidelity oracle, we apply a damping factor $q \in (0, 1)$ to the weights of the neighbors on L; q is dynamically set at each iteration via maximum likelihood estimation.

This model performs well in practice, is nonparametric, and can be efficiently updated in light of new data. The last feature is essential in allowing for fast lookahead, the central component of our method. The time complexity of a naive implementation of MF–ENS is

$\mathcal{O}\big(2^k\, n^2 \log n\big)$,[5] where k is the number of L queries made for each H query and n is the size of the candidate pool. The corresponding time complexity of the single-fidelity ENS is $\mathcal{O}\big(n^2 \log n\big)$ [88], to which MF–ENS adds a factor of 2^k from the exhaustive marginalization of the exploratory L labels. We may also take advantage of the implementation trick developed by Jiang et al. [88], which reduces the time complexity to $\mathcal{O}\big(2^k\, n\,(n + m \log m)\big)$, where $m \ll n$ is the maximum number of unlabeled points whose probabilities are affected by a newly revealed label. The interested reader may refer to §3.2 of Jiang et al. [88] for more detail.

We also extend existing branch-and-bound pruning strategies to further reduce the computation time of our policy at each search iteration. First, following previous work [61, 88], we establish an upper bound of the score function that is the approximate expected utility defined by (Eq. (4.4)). This allows us to eliminate candidates whose score upper bounds are lower than the current best score we have found, as their actual scores cannot possibly be the final best score. These upper bounds are computationally cheap to evaluate, so applying this pruning check only adds a trivial overhead to each search iteration. Further, we develop an extension of this pruning strategy by making use of the fact that computing the score of a point involves marginalizing over its unknown label. We thus apply similar pruning checks at every step during this marginalization, which allows us to identify and prune suboptimal candidates "on the fly," avoiding any unnecessary computation.

Concretely, denote by $f(x)$ the score of an unlabeled point $x \in \mathcal{X}$, defined by (Eq. (4.4)) for MF–ENS. Suppose x has a posterior probability of $\pi = \Pr\left(y = 1 \mid x, \mathcal{D}\right)$, where y is the label to be returned by the oracle we are currently querying. As previously described, we compute $f(x)$ by calculating its *partial values* while marginalizing over y:

$$f(x) = \pi\, f\left(x \mid y = 1\right) + (1 - \pi)\, f\left(x \mid y = 0\right),$$

where $f(x \mid y)$ is the partial score of x according to (Eq. (4.4)) when conditioned on a value of y. Suppose before computing either $f(x \mid y = 0)$ or $f(x \mid y = 1)$, we know these partial scores are upper bounded by $\overline{u}(x)$ given a new positive label and $\underline{u}(x)$ given a new negative label:

$$f(x \mid y = 1) < \overline{u}(x); \quad f(x \mid y = 0) < \underline{u}(x).$$

[5]Note that we are assuming k is small enough that 2^k is a small constant; if this is not the case, we may approximate (4.4) by sampling instead, replacing 2^k by s, the number of samples used.

Table 4.1: Experiment results with an H budget of $t = 300$, averaged across all repeated experiments for each setting. θ is the simulated false positive rate of the low-fidelity oracle; k is the number of L queries made between two H queries (i.e., the speed ratio between the two fidelities). Each entry denotes the average number of targets found across the repeated experiments and the corresponding standard error in parentheses. The best performance in each column is highlighted bold.

	ECFP4				GpiDAPH3				BMG			
	$\theta = 0.1$		$\theta = 0.3$		$\theta = 0.1$		$\theta = 0.3$		$\theta = 0.1$		$\theta = 0.3$	
	$k = 2$	$k = 5$	$k = 2$	$k = 5$	$k = 2$	$k = 5$	$k = 2$	$k = 5$	$k = 2$	$k = 5$	$k = 2$	$k = 5$
ENS	218 (4.5)	218 (4.5)	213 (4.2)	213 (4.2)	197 (4.7)	197 (4.7)	186 (5.3)	186 (5.3)	279 (1.6)	279 (1.6)	278 (1.7)	278 (1.7)
MF–UCB	227 (4.8)	238 (4.4)	200 (5.7)	206 (5.8)	200 (5.7)	212 (5.7)	187 (5.5)	200 (5.3)	284 (1.0)	289 (1.0)	274 (2.0)	277 (1.7)
UG	228 (4.9)	237 (4.4)	207 (5.8)	209 (5.8)	204 (5.6)	211 (5.7)	191 (5.6)	202 (5.6)	283 (1.2)	287 (1.1)	278 (2.0)	280 (1.6)
MF–ENS	**244 (3.5)**	**250 (3.2)**	**226 (4.5)**	**230 (4.4)**	**230 (3.8)**	**243 (3.4)**	**208 (5.0)**	**221 (4.3)**	**290 (0.7)**	**294 (0.6)**	**286 (1.1)**	**286 (1.1)**

As such, $f(x)$ need not be evaluated if

$$\pi\,\overline{u}(x) + (1 - \pi)\,\underline{u}(x) < f^*, \tag{4.5}$$

where f^* is the current best score we have found. This pruning strategy has been found to offer a significant speedup in previous work [61, 88, 89].

We extend this strategy by considering the case in which a given candidate x is not pruned because (Eq. (4.5)) is not satisfied, and we proceed with the calculation of $f(x)$. Now, suppose we have computed only $f\,(x \mid y = 1)$ and not yet $f\,(x \mid y = 0)$ and observe that

$$\pi\, f\,(x \mid y = 1) + (1 - \pi)\,\underline{u}(x) < f^*,$$

then we may also safely conclude that $f(x)$ cannot possibly exceed f^* without needing to go further and compute $f(x \mid y = 0)$. If this condition is met, we simply abort the computation of the current score $f(x)$ and move on to the next unpruned candidate. For each H query, we apply this partial pruning check once (either after conditioning on $y_H = 1$ and before on $y_H = 0$ or vice versa) for each candidate that is not eliminated by the full pruning check (Eq. (4.5)). For an L query, we may do this at most three times for each candidate, as the marginalization over the joint distribution of the putative label and the pending H label when computing $f(x)$ involves four different possible label combinations.

4.3 Related Work

This work is an extension of the larger active search paradigm, first introduced by Garnett et al. [61]. Active search is a variant of *active learning* [179] where the goal is not to learn an accurate model but to find and query positive labels. Previous work has studied active search under different settings such as finding a given number of targets as fast/cheaply as possible [205, 204, 90], making queries in batches [89], or when points have real-valued utility [196]. Our work generalizes ENS, the policy proposed by Jiang et al. [88] from the single-fidelity setting. The authors of that work demonstrated that their policy is nonmyopic and aware of its exact budget, allowing it to automatically balance between exploration and exploitation during search and outperform various baselines by a large margin. We will make the same observations about our policy.

Multifidelity active search was first examined by Klyuchnikov et al. [100], who specifically considered the search problem of a recommender system: identifying items that users of a given application are interested in. They modeled predictions made by a trained preference model as output of a low-fidelity oracle and proposed a co-kriging predictive model [6] to perform inference on the users' true preferences. Under their setting, queries to the oracles are made sequentially, one after another. Our multifidelity setting is different, modeling situations where oracles of different fidelities are available to run in parallel and vary in their response times, common in scientific experiments and testing. Regardless, our proposed policy could be naturally adopted to their sequential model; the only difference in the computation is that the marginalization over a pending H label is no longer necessary. Further, as we will show in later experiments, our algorithm outperforms the adoption of their upper confidence bound (UCB) policy for various datasets. To our knowledge, our work is the first to tackle multifidelity active search using Bayesian decision theory.

Active search is equivalent to *Bayesian optimization* (BO) [28, 183] with binary observations and cumulative reward. Multifidelity BO itself has been studied considerably, and policies corresponding to common acquisition functions have been adopted to multifidelity settings, including expected improvement [83, 158], knowledge gradient [159, 210], and UCB [94]. However, most of these policies are derived from or motivated by greedy approximations to the optimal policy under different utility functions, and to our knowledge, no *nonmyopic* multifidelity BO policies have been proposed.

Active search is related to the *multi-armed bandit* (MAB) problem [112]. In particular, querying the label of a point can be viewed as "pulling an arm" in MAB; in active search, an arm cannot be pulled twice but is correlated to its neighbors. Kandasamy et al. [93] studied a formulation of multifidelity MAB in which each arm may be pulled on different fidelities at different costs, and proposed a policy that is a variant of UCB. By assuming the accuracy of the lower fidelities is known, the authors derived strong theoretical guarantees for their proposed policy.

4.4 Experiments

We now compare the empirical performance of MF–ENS against several benchmarks. As previously described, ENS is a state-of-the-art, nonmyopic active search policy in the single-fidelity setting. In our experiments, this policy simply ignores the low-fidelity oracle, serving as a single-fidelity baseline to illustrate the benefit of having access to low-fidelity queries. We also test against the MF–UCB policy for multifidelity active search, recently proposed by Klyuchnikov et al. [100]. Under this policy, each candidate x has a UCB–style score of $\alpha\left(x, \mathcal{D}\right) = \pi + \beta\sqrt{\pi(1-\pi)}$, where π is the probability of x having a positive label on the fidelity being queried and β is the exploration/exploitation tradeoff parameter.[6] We set $\beta = 0.01$ for L queries and $\beta = 0.001$ for H queries, as suggested in the same work. For another benchmark, we consider a simple but natural heuristic for multifidelity optimization, that low-fidelity queries should serve to narrow down the most-promising search regions (exploration), so that more informed queries could be made on the higher-fidelity oracle (exploitation). Inspired by this heuristic, we design a policy we call UG (for *uncertainty* and *greedy* sampling) that always queries the most uncertain points on L and the points most likely to be positive on H. Uncertainty sampling is chosen for the role of exploration due to its popularity as an active learning technique [115]. UG may also be viewed as the limit of UCB when β approaches infinity on L for maximum exploration and 0 on H for maximum exploitation.

Datasets for multifidelity active search are not readily available due to the relative novelty of the problem setting, at least not in our motivating area of scientific discovery. However,

[6]The authors also considered an odd scenario in which the correlation between the two oracles is *negative*. We assume that this correlation is always positive, and it is constructed to be so.

numerous high-quality datasets are available in the single-fidelity setting, which we will adopt and use to simulate multifidelity search. Namely, for each dataset, we simulate the noisy labels returned by the low-fidelity oracle using the following procedure. We first create a duplicate of the true labels. Given $\theta \in (0, 1)$, we randomly select a fraction θ of the positives from this duplicate set and "flip" their labels to negative. We also randomly select the same number of negatives and flip their values to positive. This perturbed set of labels is then used as the L labels. This construction yields simulated L labels with a false positive rate of θ and a false negative rate of $\theta\, r/(1 - r)$, where $r \in (0, 1)$ is the prevalence rate of the positive set $\mathcal{T}$ in $\mathcal{X}$. In a typical active search problem, $\mathcal{T}$ is rare and $r \ll 1$, making the false negative rate much lower than the false positive rate, a common characteristic of many real-world scientific discovery and testing procedures.

We set $\theta \in \{0.1, 0.3\}$. We set k, the number of L queries that are made for each H query, to be either 2 or 5, and set the budget on H to be 300. In each experiment, a policy starts with an initial training dataset of one randomly selected target whose L label is also positive.

4.4.1 Datasets

We conducted experiments on three real-world scientific discovery datasets used in previous studies [88, 89, 90]. The first two come from drug discovery, where the goal is to discover chemical compounds exhibiting binding activity with a given protein. Each protein defines the target for an active search problem. Here we used the first 50 proteins from the BindingDB database [121] described by Jiang et al. [88]. A set of 100 000 compounds sampled from the ZINC database [184] served as a shared negative set. Features for the compounds are binary vectors encoding chemical characteristics, also known as a chemoinformatic fingerprint. We considered two fingerprints delivering good performance in previous studies: ECFP4 and GpiDAPH3. The size of the positive set for these 50 targets ranged from 205 to 1488 (mean 538), having an average prevalence rate of $r \approx 0.5\%$. For each of these two datasets and 50 targets, we repeat each experiment five times, for a total of 500 search simulations.

The other dataset is related to a materials science application. The targets in this case are alloys that can form bulk metallic glasses (BMGs), which have higher toughness and better wear resistance than crystalline alloys. This dataset comprises 106 810 alloys from

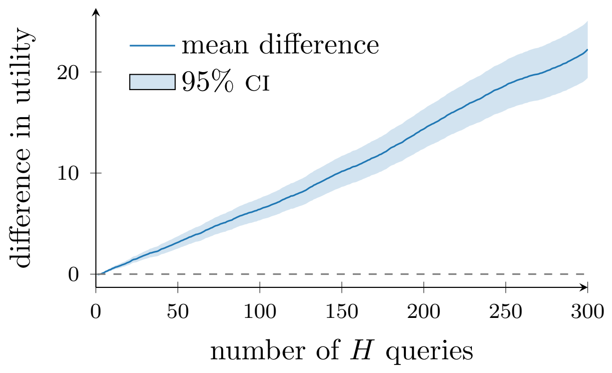


Figure 4.2: The difference in cumulative targets found between MF–ENS and ENS, averaged across all experiments and datasets.

the materials literature [95, 203], 4275 of which exhibit glass-forming ability ($r \approx 4\%$). We repeated each experiment 50 times for this dataset.

We report the average number of targets found by each policy across the experiments with standard errors in parentheses in Table Tab. 4.1; each column corresponds to a specific setting of θ and k under a dataset. We observe that our policy MF–ENS outperforms all baselines by a large margin. In each column, a two-sided paired t–test rejects the hypothesis that the average difference in the number of targets found between MF–ENS and any baseline is zero with overwhelming confidence, returning a p–value of at most 4×10^{-5}. Finally, looking across the columns, we notice the expected trends: performance of all algorithms except for ENS, which does not utilize fidelity L, improves with higher k (when L is cheaper) or with lower θ (when L is more accurate).

4.4.2 Performance Gain from Multifidelity Search

To further examine the benefit of having access to more than just the exact oracle, we visualize the difference in the cumulative number of targets found between MF–ENS and the single-fidelity policy ENS, averaged across all experiments, in Figure Fig. 4.2. We observe that MF–ENS completely dominates ENS, finding roughly linearly more targets throughout the search. This large difference in empirical performance illustrates the usefulness of the simulated low-fidelity oracle in our experiments, and suggests that our approach is likely to benefit search with any budget.

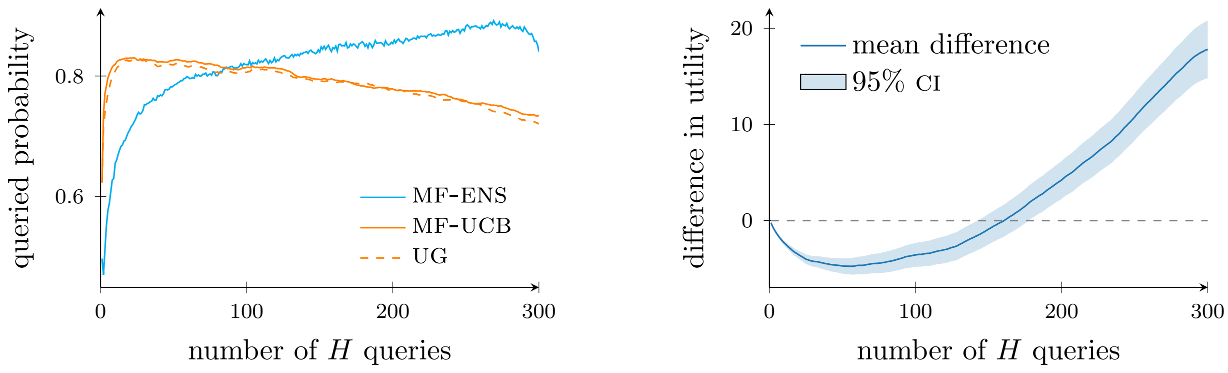


Figure 4.3: An illustration of budget-awareness exhibited by MF–ENS. Left: The average progressive probabilities of points queried on H by different active search policies. Right: The difference in cumulative targets found between MF–ENS and MF–UCB. The results are averaged across all experiments and datasets.

4.4.3 Nonmyopic Behavior

We have claimed that MF–ENS is nonmyopic and aware of its remaining budget at any given time during a search. We demonstrate this nonmyopia by first comparing the progressive probabilities of the points queried on H (at the time of the queries being made) by the policy against the myopic baselines MF–UCB and UG, averaged across all experiments, in the left panel of Figure Fig. 4.3. Initially, MF–ENS chooses points with lower probabilities, exploring the space. As the search progresses, MF–ENS queries more promising points, smoothly transitioning to exploitation. The opposite trend can be observed for UCB and UG, whose probabilities decrease over time due to greedy behavior. This difference translates to distinct patterns in the cumulative reward achieved by these policies. The right panel of Figure Fig. 4.3 shows the difference in the cumulative number of targets found between MF–ENS and MF–UCB, also averaged across all experiments. During the first half of the search, MF–ENS appears to perform *worse* than MF–UCB, but quickly recovers and outperforms the latter in the end. The corresponding plot comparing MF–ENS and UG shows a similar trend.

Overall, this phenomenon perfectly highlights the automatic tradeoff between exploration and exploitation exhibited by MF–ENS: the policy makes its initial queries to explore the search space, often requesting labels that are not the most likely to be positive and failing to collect substantial immediate reward; however, as the budget decreases, its queries grow more exploitative and are ultimately more successful than those from myopic policies by leveraging what it has learned.

Table 4.2: Average pruning rate and the contribution of each of the two pruning methods.

policy	full coverage rate	given full coverage	
		total prune %	partial prune %
MF–ENS	27.6%	89.4%	9.2%

4.4.4 Effects of Pruning

We examine the effectiveness of our pruning strategies in helping MF–ENS cover the entire search space. Each of the two policies has an upper limit on how many candidates are to have their scores fully computed. If this limit is reached, we only consider a random subset of the remaining candidate pool. We classify each time a policy returns while there are unpruned candidates remaining as an instance of failure to cover the entire space.

First, we compute the fraction of iterations, across all experiments, in which this limit is not exceeded, or in other words, how often each policy can exactly consider all candidates with the help of pruning. When a point is pruned, it may be before any actual computation (total pruning) if its score upper bound is lower than the current highest score f^*. Otherwise, it may be pruned during the calculation of its scores (partial pruning) if its partial score, combined with partial upper bounds, is lower than f^*. We keep track of the fraction of points pruned by each of the two methods; these results are averaged across the iterations in which pruning helps cover the entire candidate pool. We report these statistics in Table Tab. 4.2.

We observe that while we do not cover the entire candidate pool in many iterations, the effect of pruning is dramatic when it is successful. The existing pruning method (total pruning) helps eliminate most of the candidates, and our extension (partial pruning) raises the combined pruning rate to roughly 99% on average. In our experiments, successful pruning could reduce the time a policy takes to produce a decision from under an hour to mere seconds.

At an iteration where the current best score f^* does not exceed the majority of the score upper bounds, many candidates may be left unpruned. In order to help our policy avoid having to calculate the scores of a large number of candidates and consequently spending too much time on a single iteration, we place an upper limit on how many candidates are to be considered before we terminate the search. Our approach is to first follow the lazy-evaluation strategy introduced by Jiang et al. [89] and sort the candidates by their score upper bounds. With this sorting, candidates with higher upper bounds will be considered first, and a candidate will never be considered if it will be pruned later on. Now, at each iteration, if after having considered u candidates and noticing that there are still unpruned points remaining, we simply consider a randomly selected subset of size at most s of the unpruned set, before terminating the search and returning the current best candidate. In our experiments, we set $u = s = 500$ for MF–ENS.

We note that this strategy is only used when pruning fails to reduce our search space to be below $u + s$, and to allow us to collect results over a long horizon over many repeated experiments. When applied in a real-life planning setting, MF–ENS can still cover the entirety of a large pool of candidates if desired, even with a significant portion of the pool unpruned. In our experiments, MF–ENS takes approximately 30 seconds to reach its quota of 1000 candidates when pruning is unsuccessful; the time for it to fully cover a pool of 100 000 points (about the size of the real-world datasets used in our experiments) is thus well under one hour. In short, our policy remains tractable in real-life settings, even without successful pruning.

4.5 Conclusion

We have proposed a multifidelity active search model in which an exact oracle and a cheaper surrogate are queried in parallel. We presented a novel nonmyopic policy (based on two-stage rollout) for this setting that reasons about the remaining queries on both fidelities seeking to maximize the total number of discoveries. Our policy is aware of the remaining budget and dynamically balances exploration and exploitation. Experiments on real-world data demonstrate that the policy significantly outperforms myopic benchmarks.

Chapter 5

Diversity-Aware Active Search

Prior work on active search (AS) has operated in a binary setting where every data point is either valuable or not. The total number of discoveries made in a given budget is then used as a utility function during experimental design, encoding equal marginal utility for every discovery made. However, this may not adequately capture preferences over experimental outcomes in many practical scenarios, where there may be *diminishing returns* in finding additional members of a frequently observed class. This is often the case in, e.g., scientific discovery, where a discovery in a novel region of the design space may offer more marginal insight than the 100th discovery in an already densely labeled region. In this work, we will consider a *multiclass* variant of the AS problem, wherein discoveries in rare classes are awarded more marginal utility than those in an already well-covered classes. As we will see, this approach naturally encourages *diversity* among the points discovered.

After defining this problem, we study it through the lens of Bayesian decision theory. We begin by outlining how we can capture the notion of diminishing returns in marginal discovery through an (arbitrary) positive, increasing, and concave utility function. We then extend the hardness of approximation result by Jiang et al. [88] from the linear utility to this much larger family, demonstrating that search is fundamentally difficult for a broad range of natural utility functions. We then propose an approximation to the optimal policy for problems in this this class that is both computationally efficient and nonmyopic. We show that the resulting algorithms effectively encourage diversity among discoveries, and, similar to nonmyopic policies from previous work, leverage budget-awareness to dynamically balance exploration and exploitation. We demonstrate the superior empirical performance of our approach through an exhaustive series of experiments, including in a challenging drug discovery setting. Across the board, our proposed policy recovers both better balanced and richer data sets than a suite of strong baselines.

5.1 Problem Definition

We first introduce the multiclass active search problem with diminishing returns and present the Bayesian optimal policy. This policy will be hard to compute (or even approximate), but will inspire an approximation developed in the next section. Suppose we are given a large but finite set of points $\mathcal{X} \triangleq \{x_i\}$, each of which belongs to exactly one of C classes, denoted by $[C] \triangleq \{1, 2, \ldots, C\}$, where $C > 2$. We assume class-1 instances are abundant and uninteresting, while other classes are rare and valuable; we call the members of these classes *targets.* The class membership of a given point $x \in \mathcal{X}$ is not known *a priori* but can be determined by making a query to an oracle that returning its label $y = c$. We assume this labeling procedure is expensive and can only be accessed a limited number of times $T \ll n \triangleq |\mathcal{X}|$ (the querying budget). Denote a given data set of queried points and labels as $\mathcal{D} = \{(x_i, y_i)\}$, and $\mathcal{D}_t$ as the data set collected after t queries to the oracle in a given search.

Our high-level goal is to design a policy that decides which elements of $\mathcal{X}$ should be queried in order to uncover as many targets as possible. Preferences over different data sets (experimental outcomes) are expressed via a *utility function;* previous work [61, 88] has used a linear utility in the binary setting $C = 2$:

$$u\,(\mathcal{D}) = \sum\nolimits_{y \in \mathcal{D}} \mathbb{I}\{y > 1\}, \tag{5.1}$$

which effectively groups all targets in a common positive class and assigns equal reward to each discovery. In many practical scenarios, however, once a target class has been thoroughly investigated, the marginal utility of finding yet more examples decreases and we would prefer to either expand a rarely sampled class or discover a new one. For example, in drug discovery – one of the main motivating applications for AS – screening procedures optimized for hit rate tend to propose very structurally similar compounds and lead to an overall decline in usefulness of these discoveries downstream [58]. This has lead to efforts to artificially encourage diversity when generating new screening experiments, as a way to induce the desired search behavior [19, 156].

The preferences above reflect the notion of *diminishing returns.* We propose to capture diminishing returns for marginal discoveries in a known class c (and thereby encourage diversity in discoveries) with a reward function f_c:

$$u(\mathcal{D}) \triangleq \sum_{c>1} f_c\Big(\sum_{y\in\mathcal{D}} \mathbb{I}\{y=c\}\Big) = \sum_{c>1} f_c(m_c), \tag{5.2}$$

where f_c is a positive, increasing, and concave function quantifying our reward given the number of found targets from a given class c. The term m_c denotes the number of targets of class c in data set $\mathcal{D}$; we also use $m_{c,t}$ to denote the corresponding number in $\mathcal{D}_t$ at time t. Any utility encoding decreasing marginal gains (that is, concave) is an appropriate choice for this setting, and we will see later in Sect. 5.2 that the key element of our algorithm is valid with any concave utility. In our experiments, we use the logarithmic function $f_c(x) = \log(x+1)$ (which has seen a wide range of applications, namely modeling utility of wealth [20] or production output [155] in economics) as a reasonable default with which to generate our main results. We also present results showing that our methods generalize well to other possibilities such as the square root utility $f_c(x) = \sqrt{x}$.

Under this model, the marginal gain of an additional discovery decreases with the size of the corresponding class in $\mathcal{D}$. Consider a toy problem illustrated in Fig. 5.1, where we wish to search for points close to the center and corners of the unit square, each representing a target class. The state-of-the-art ENS policy [88], which uses a linear utility, recovers many targets but over-exploits the center region. This is undesirable behavior when we would prefer to have balanced discoveries across all classes; ideally, we would like to achieve a uniform target distribution (an equal number of hits across all classes for maximum diversity), where all five regions in the plot are transparent. We will develop policies that can achieves such balance in the next section.

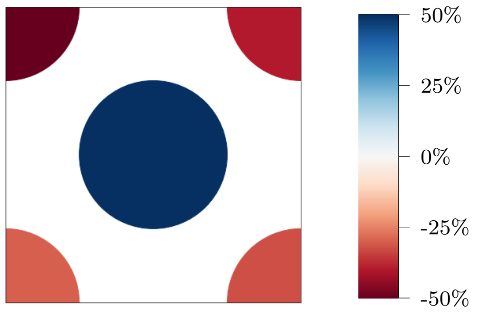


Figure 5.1: Discoveries by region relative to a uniform target distribution by the state-of-the-art policy ENS, in a toy search problem where points within the visible regions in the unit square are considered search targets. ENS over-samples the center region, the most common and easiest-to-identify target class, and collects a highly unbalanced data set.

5.1.1 The Bayesian Optimal Policy

We now derive the optimal policy in the expected case using Bayesian decision theory. We first assume access to a model computing the posterior probability that a point $x \in \mathcal{X}$ belongs to class $c \in [C]$ given an observed data set $\mathcal{D}$, denoted by $p_c(x \mid \mathcal{D}) \triangleq \Pr(y = c \mid x, \mathcal{D})$. (We sometimes omit the dependence on $\mathcal{D}$ in the notation when the context is clear.) This model may be arbitrary. Now, suppose we are currently at iteration $t + 1 \leq T$, having collected data set $\mathcal{D}_t$, and now need to identify the next point $x_{t+1} \in \mathcal{X} \setminus \mathcal{D}_t$ to query the oracle with. The optimal policy selects the point that maximizes the expected utility of the *terminal* data set $\mathcal{D}_T$, conditioned on the current query, recursively assuming that future queries will also be made optimally:

$$x^*_{t+1} = \operatorname*{arg\,max}_{x_{t+1} \in \mathcal{X} \setminus \mathcal{D}_t} \mathbb{E}\big[u(\mathcal{D}_T) \mid x_{t+1}, \mathcal{D}_t\big]. \tag{5.3}$$

This expected optimal utility may be computed using backward induction [18]. In the base case where $t = T - 1$ and we are faced with the very last query x_T,

$$\mathbb{E}\big[u(\mathcal{D}_T) \mid x_T, \mathcal{D}_{T-1}\big] = \sum_{c \in [C]} u(\mathcal{D}_T)\, p_c(x_T \mid \mathcal{D}_{T-1}). \tag{5.4}$$

Maximizing this expectation is equivalent to maximizing the expected *marginal utility gain*

$$\Delta(x_T \mid \mathcal{D}_{T-1}) \triangleq \sum_{c > 1} p_c(x_T \mid \mathcal{D}_{T-1}) \Big[f_c(m_{c,T-1} + 1) - f_c(m_{c,T-1})\Big]. \tag{5.5}$$

For each class c, this quantity not only increases as a function of the positive probability p_c, but also decreases as a function of the number of targets already found in that class. Therefore, even at this very last step, the optimal decision balances between hit probability and discovery/extension of a rare class. When more than one query remains in our budget, the expected optimal utility in Eq. (5.3) expands into

$$\begin{aligned}\mathbb{E}\big[u(\mathcal{D}_T) \mid x_{t+1}, \mathcal{D}_t\big] = {} & u(\mathcal{D}_t) + \Delta(x_{t+1} \mid \mathcal{D}_t) \\ & + \mathbb{E}_{y_{t+1}}\Big[\max_{x_{t+2}} \mathbb{E}\big[u(\mathcal{D}_T) \mid x_{t+2}, \mathcal{D}_{t+1}\big] - u(\mathcal{D}_{t+1})\Big],\end{aligned} \tag{5.6}$$

where $\mathbb{E}\big[u(\mathcal{D}_T) \mid x_{t+2}, \mathcal{D}_{t+1}\big]$ is the expected utility that is a step further into the future and may be recursively computed using the same expansion. Here, we note while the first term in the sum on the right-hand side (the utility at the current step $u(\mathcal{D}_t)$) is a constant, the other two terms may be interpreted as balancing between exploitation from the immediate reward (the marginal gain $\Delta\left(x_{t+1} \mid \mathcal{D}_t\right)$), and exploration from the future rewards to be optimized by subsequent queries (the expected future utility). Overall, computing this expectation involves $(\ell-1)$ further nested expectations and maximizations, where $\ell = T-t$ is the search horizon. This has a time complexity of $O\left((C\,n)^{\ell}\right)$, making finding the optimal decision intractable for any large data set.

A potential solution to this problem is to limit the lookahead horizon by pretending that ℓ is small, thus myopically approximating the optimal policy. The simplest form of this is to set $\ell = 1$ and greedily optimize for the one-step expected marginal utility gain in Eq. (5.5). We refer to the resulting policy as the *one-step* policy. Since our utility function has elements with diminishing returns, a question naturally arises as to whether the results from the submodularity [104] and adaptive submodularity [65] literature apply here, and whether the greedy strategy of the one-step policy could approximate the optimal policy well. In the next subsection, we present the perhaps surprising result that *no* polynomial-time policy can approximate the optimal policy by any constant factor.

5.1.2 Hardness of Approximation

Assuming access to a unit-cost conditional probability $p_c\left(x \mid \mathcal{D}\right)$ for any point $x \in \mathcal{X}$ and data set $\mathcal{D}$, we obtain the same hardness result of Jiang et al. [88] for the broad range of utility functions considered here:

Theorem 5.1.1. *There is no polynomial-time policy providing any constant factor approximation to the optimal expected utility in the worst case.*

Our proof strategy follows that of Jiang et al. [88]. We construct a family of hard problem instances, where in each instance a secret set of points encodes the location of a larger "treasure" of targets. The probability of discovering this treasure is extremely small without observing the secret set first, which in itself is vanishingly unlikely to happen in polynomial time. Further, the average hit rate outside of the treasure set is vanishingly low, making it infeasible to compete with the optimal policy. Remarkably, we can construct such hard problem instances for any utility that is positive, increasing, unbounded, and concave in the number of discoveries in each class. The complete proof is included in Appx. A.

Despite this negative result, we hope to design *empirically* effective policies. Previous work has demonstrated that nonmyopic policies offer both theoretical and empirical benefits when working with the linear utility function, and that budget-awareness is in particular can be especially beneficial for a policy to effectively balance exploration and exploitation. In the next section, we propose an efficient, nonmyopic approximation to the optimal policy for the class of utility functions we consider here, which we will show later also improves practical performance.

5.2 Efficient Nonmyopic Approximation

We propose a batch rollout approximation to the optimal policy similar to the ENS algorithm for binary AS [88] and the GLASSES algorithm for Bayesian optimization [67]. The key idea is to assume that after a proposed query in the current iteration, all remaining budget will be spent *simultaneously* on a single batch of queries exhausting the budget. This assumption simplifies the decision tree we must analyze, reducing its depth to 2 while expanding the branching factor of the last layer. Under the linear utility model, the expected marginal utility of a final batch of queries conveniently decomposes into a sum of positive probabilities of individual batch members. The optimal final batch therefore consists of the points with the highest probabilities, which may be computed efficiently. By matching the size of the following batch to the number of queries remaining, we can effectively account for our remaining budget when computing the expected utility of a given putative query. Unfortunately, the linear decomposition enabling rapid computation in ENS does not hold in our setting due to our nonlinear utility (5.2), and designing an effective batch policy requires more care.

5.2.1 Making a Batch of Queries

We first temporarily consider the subproblem of designing a batch of b queries X given a data set $\mathcal{D}$ to maximize the expected utility of the combined observation set, i.e., $\mathbb{E}_Y\big[u\left(\mathcal{D} \cup X, Y\right)\big]$, where the expectation is taken over Y,[7] the labels of X. As labels may be conditionally dependent and the utility function u is not linear, exact computation of this expectation

[7]In this section, expectations over Y are universally conditioned on knowledge of X and $\mathcal{D}$; we drop this conditioning from the expressions to clarify the main ideas.

requires iterating over all C^b realizations of the label set Y. This is infeasible unless b is very small, and represents the primary challenge we must overcome.

A crude but effective mechanism to address the conditional dependence of labels is to simply ignore the dependence (a "mean field approximation"). This relieves us from having to update the posterior for unseen points given fictitious observations arising in the computation. However, in our setting, even if we assume conditional independence, we still face challenges in computing the expected utility:

$$\mathbb{E}_Y\big[u\,(\mathcal{D} \cup X, Y)\,\big] = \sum_{c>1} \mathbb{E}_Y\Big[f_c\,(m'_c)\,\Big], \tag{5.7}$$

where m'_c is the total number of targets belonging to class c in the union set of $\mathcal{D}$ and a particular realization of Y. In the interest of effective computation, we use Jensen's inequality to obtain an upper bound on the expected utility:

$$\sum_{c>1} \mathbb{E}_Y\Big[f_c\,(m'_c)\,\Big] \le \sum_{c>1} f_c\Big(\mathbb{E}_Y\,[m'_c]\,\Big). \tag{5.8}$$

Now, for a given class c, the inner expectation may be rewritten as the sum of probabilities (and some constants):

$$\mathbb{E}_Y\,[m'_c] = m_c + \sum_{x \in X} p_c\,(x \mid \mathcal{D})\,. \tag{5.9}$$

We then upper-bound the overall expected utility:

$$\mathbb{E}_Y\big[u\,(\mathcal{D} \cup X, Y)\,\big] \le \overline{u}(X) \triangleq \sum_{c>1} f_c\Big(m_c + \sum_{x \in X} p_c(x \mid \mathcal{D})\Big). \tag{5.10}$$

We propose to use this upper bound, $\overline{u}$, to approximate the expected utility of a batch for the purposes of policy computation. (We note that we may derive this bound for *any* concave utility.) We later present simulation results comparing the fidelity of this approximation to that of Monte Carlo sampling. Overall, our method offers competitive accuracy even against sampling with a large number of samples of Y ($>$1000), while being significantly more computationally lightweight. Here, speed is paramount since in batch rollout, computing the utility of a batch is a subroutine that needs to run many times (C times for each putative query candidate, once for each putative label). Another attractive feature of our approach is that $\overline{u}$ is a monotone submodular set function, which will facilitate efficient (approximate) maximization.

Algorithm 5 Diversity-aware active search (DAS)

```
1: inputs observations $\mathcal{D}_t$, remaining budget $T - t$
2: returns $x^*_{t+1}$ maximizing the score in Eq. (5.11)
3: for $x_{t+1} \in \mathcal{X} \setminus \mathcal{D}_t$ do                              ▷ iterate over candidates
4:     for $y_{t+1} \leftarrow 1$ to $C$ do
5:         $\alpha\left(x_{t+1} \mid y_{t+1}\right) \leftarrow \overline{u}\left(X \mid \mathcal{D}_t \cup \{(x_{t+1}, y_{t+1})\}\right)$   ▷ approximate future rewards
6:     end for
7:     $\alpha(x_{t+1}) \leftarrow \sum_c p_c(x_{t+1})\, \alpha\left(x_{t+1} \mid c\right)$
8: end for
9: $x^*_{t+1} \leftarrow \arg\max_{x_{t+1}} \alpha(x_{t+1})$                              ▷ design by maximizing
```

Our goal now is to find the batch X that maximizes $\overline{u}$, as an approximation to the batch maximizing the expected one-step utility. Naïvely maximizing $\overline{u}$ requires iterating over $\binom{n}{b}$ candidate batches to compute the corresponding score. However, we note that this score is a sum of concave, increasing functions, which are monotone submodular, and therefore $\overline{u}$ is a monotone submodular function itself. We thus opt to *greedily* optimize $\overline{u}$ by sequentially maximizing the pointwise marginal gain; the resulting batch provides a $(1 - 1/e)$-approximation for the optimal batch [144, 104]. We briefly remark on the nature of the batches resulting from this greedy procedure, which naturally encourages batch members to be diverse in their likely labels: once a point having a high p_c is added to X, others with high $p_{c'}$ for another target will be prioritized during the next selection. This is a desideratum of a batch policy when seeking to encourage diversity, indicating the output of the algorithm is a reasonable approximation of the optimal batch. Further, when $b = 1$ (that is, at the second-to-last iteration), this procedure makes the true expected-case optimal decision – a reassuring feature.

5.2.2 Completing the Algorithm

With a method of constructing approximate one-step optimal batches in hand, we now complete our proposed policy, *diversity-aware active search*, or DAS. For a candidate observation x_{t+1}, we condition on each possible label $y_{t+1} \in [C]$, approximate the optimal batch observation following (x_{t+1}, y_{t+1}), and average the resulting approximate terminal utility $\overline{u}$ over the labels y_{t+1}:

$$\alpha(x_{t+1}) = \mathbb{E}_{y_{t+1}}\left[\overline{u}\left(X\right) \mid x_{t+1}, \mathcal{D}_t\right], \tag{5.11}$$

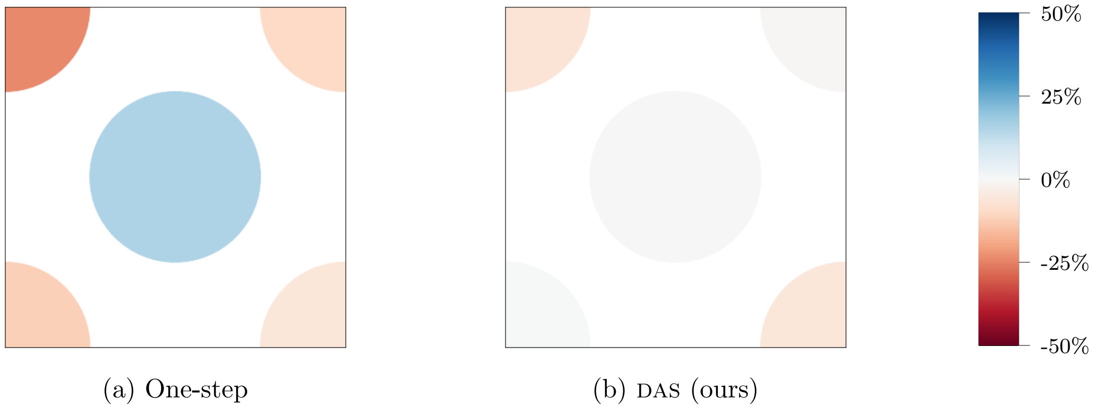


Figure 5.2: Discoveries by region relative to a uniform target distribution by the one-step policy and our proposal DAS, in the problem visualized in Fig. 5.1. One-step distributes queries more equally than the previously seen ENS; however, center points are still over-represented. DAS constructs more diverse data sets and finds more rare corner targets.

where $\overline{u}$ depends on the putative data $\mathcal{D} \cup (x_{t+1}, y_{t+1})$. DAS proceeds by selecting the candidate x^*_{t+1} that maximizes the score α. This procedure is summarized in Alg. 5.

As mentioned, with the lookahead batch construction simulating future queries, DAS accounts for not only the immediate reward but also the impact of the chosen query on future rewards. Additionally, the latter quantity naturally decreases as a function of the remaining budget b, allowing our policy to be budget-aware and dynamically balance exploitation and exploration without any tradeoff parameter. We briefly demonstrate the benefits of our approach by continuing with the example previously seen in Fig. 5.1, where ENS, in seeking to maximize only recovery, collected highly unbalanced data sets. Fig. 5.2 shows the results of the one-step policy and DAS under the same setting. Compared to ENS, one-step distributes its queries more equally, but it is our proposed policy DAS that constructs the most diverse data set.

5.2.3 Implementation

A naïve implementation of the batch subroutine in DAS has a complexity of $O(b\,n)$, and the entire DAS procedure has a complexity of $O\big(C\,n\,(b\,n)\big) = O\left(C\,b\,n^2\right)$, where again $n = |\mathcal{X}|$ is the size of our search space and b is the remaining budget.

We now describe the k-NN model used in our experiments, introduced by Garnett et al. [61] in the binary setting. The idea is to use the proportion of class-c members among the observed nearest neighbors of a given point x as the posterior marginal probability $p_c(x \mid \mathcal{D})$. Formally, denote the set of nearest neighbors of x as $\textsc{nn}(x)$ and the (potentially empty) subset of labeled neighbors as $\textsc{lnn}(x) \subseteq \textsc{nn}(x)$. Then, the posterior probability of x belonging to class c is

$$p_c(x \mid \mathcal{D}) = \frac{\gamma_c + \sum_{x' \in \textsc{lnn}(x)} \mathbb{I}\{y' = c\}}{1 + |\textsc{lnn}(x)|}, \tag{5.12}$$

where each $\gamma_c \in (0, 1)$ is a hyperparameter acting as a "pseudocount", or our prior belief about the prevalence of class c (since $p_c(x \mid \mathcal{D}) = \gamma_c$ if $\textsc{lnn}(x) = \varnothing$).

The k-NN achieves reasonable generalization error in practice (in the sparsely labeled setting we are considering here), and can be rapidly updated given a new observation, which is a valuable feature with respect to our method. Further, the k-NN only uses the similarity matrix for $\mathcal{X}$, whose calculation only needs to be done once and can be accelerated by modern similarity search libraries such as Faiss [91]. Another benefit of this model is that it is possible to cheaply compute a *posterior probability upper bound* p^*, given any data set $\mathcal{D}$ and an additional observation with label y:

$$\max_{x' \in \mathcal{X} \setminus \mathcal{D}} p_c\big(x' \mid \mathcal{D} \cup \{(x, y)\}\big) \leq p^*_c(y, \mathcal{D}).$$

This upper bound is useful in that we may then bound the approximate expected terminal utility $\overline{u}$ conditioned on label c when computing the score in Eq. (5.11), i.e., $\alpha(x \mid y = c)$, and therefore the overall score $\alpha(x)$. With these score upper bounds in hand, we employ branch-and-bound pruning strategies used in previous work [89, 147]. Specifically, before evaluating the score α of a given candidate x, we compare the upper bound of $\alpha(x)$ against the current best score α^* we have found. If α^* exceeds this bound, computing $\alpha(x)$ is unnecessary and we proceed to the next candidate. Otherwise, since we have access to the *conditional* score upper bounds for $\alpha(x \mid y = c)$, as we marginalize over each label y, we further check whether the conditional scores computed thus far, when combined with the complementary conditional upper bounds, are less than α^*. If this is the case, we terminate the current α computation "on the fly." The upper bounds p^*_c are cheap to evaluate, and these checks add a trivial overhead to the entire procedure in the pessimistic situation of no pruning – in practice, this cost is well worth it. Finally, a lazy-evaluation strategy is used, where candidates are evaluated in descending order of their score upper bounds, so that a given point that may be pruned will not be evaluated.

We also employ a pruning strategy for the inner batch-building procedure. We first note that, in this procedure, points having the same marginal probabilities p_c (conditioned on a putative query) are interchangeable as we have assumed conditional independence. We point out one particular set of such points with equal p_c: those whose marginal probabilities have not been updated from the prior due to having no labeled nearest neighbors. In a typical AS iteration, there may be many such points, especially in early stages of the search. Pruning duplicates appropriately allows the follow-on batch to be built more efficiently, and we empirically observed a drastic improvement in our experiments. A welcomed property of this method is that it has the most impact in the early iterations of a search, which would usually be the longest-running iterations otherwise. Overall, the combination of these strategies allows our algorithm to scale to large data sets (>100 000 points).

In the event where pruning is unsuccessful, we can sub-sample the space (e.g., subject to a user-specified cardinality constraint) and conduct the current search within the sampled pool. This technique is extensively explored in Mirzasoleiman et al. [136] and used by Nguyen et al. [147].

5.3 Related Work

Active search [61] is a variant of active learning (AL) [179] where we aim not to learn an accurate model, but to collect members of rare and valuable classes. Previous work has explored AS under a wide range of settings, such as when the goal is to hit a targeted number of positives as quickly/cheaply as possible [205, 204, 90], when queries are made in batches [89], or with multifidelity oracles [147]. These studies all assumed there is only one target class, and collecting a target constitutes a constant reward. Ours is the first to our knowledge to tackle multiclass AS.

Diversity as an objective has enjoyed great interest from the broader AL community. A common approach is to modify a typical AL acquisition function to encourage diversity in the resulting queries. For example, Gu et al. [73] and Yang et al. [215] encouraged diversity by incorporating dissimilarity terms (computed via an RBF kernel) into uncertainty sampling schemes. Brinker [27] used the angles between the hyperplanes induced by adding new points to the training set of an SVM, and Zhdanov [219] considered the minimum distance between any pair of labeled points. Others have employed coreset-based strategies [178, 3] to identify a set of diverse representative points. Another popular strategy is to partition a given data set into different groups (e.g., using a clustering algorithm) and inspect the groups in a round-robin manner [127, 116, 124, 38]. We will apply a round-robin heuristic to a number of benchmarks in our experiments.

Vanchinathan et al. [196] was motivated by a similar problem of uncovering a diverse, valuable subset. Theirs is a regression setting in which diversity is measured in the feature space – via the logdet of the Gram matrix of the collected data. The proposed policy is myopic, maximizing the expected one-step marginal gain in a weighted sum of reward and diversity. He and Carbonell [78] studied a related problem where the objective is to detect at least one instance of each rare target class as quickly as possible. By assuming the target classes are highly concentrated, they design a policy that optimizes the difference in local density between a given point and its nearest neighbors, which is effective at identifying targets on the boundary. Malkomes et al. [130] considered the *constraint active search* problem, in which they seek to find a diverse set of points satisfying a set of constraints. The authors propose maximizing the expected improvement in a coverage measure given a new observation. We include these policies as AL baselines in our experiments.

Closest to the motivation of our work, a line of research [102, 103] has explored a unified AL framework for querying rare, diverse subsets of a large pool using submodular information measures. Specifically, the neural network-based SIMILAR algorithm [102] consists of AL policies that can tackle problematic yet realistic learning scenarios such as imbalance in the training data, rare classes, out-of-distribution test data, and redundancy. While it can be used for active search, SIMILAR is not designed to specifically target discovery tasks.Further, our diversity-aware active search framework allows the utility function to be adjusted by the user to dynamically balance between discovery and diversity, a valuable feature in many discovery tasks.

Diversity has also been explored in the related task of Bayesian optimization [59]. A common approach is to incorporate a determinantal point process [107] to induce diversity in the feature space [201, 143]. In the multiobjective setting, many policies leverage the diversity of their collected data in the Pareto space to design queries [75, 181, 123].

5.4 Experiments

We performed a series of experiments to evaluate the empirical performance of our policy DAS. As baselines, we considered related active learning/search algorithms [78, 196, 130] (see Sect. 5.3), as well as the one-step lookahead policy, which greedily maximizes the marginal utility at each iteration. Another baseline was ENS [88], the state-of-the-art for binary AS, where we lumped all targets into a single positive class.

We also considered a family of policies that design queries in a *round-robin* (RR) manner. In each iteration t, we choose a target class c_t and seek to make a discovery for this target class. A round-robin policy then continually rotates the target class among the positive classes throughout the search, devoting an equal amount of resources to each class. The first of these policies is RR-greedy, which for a given class index c_t queries the point that maximizes the probability p_{c_t}. Another round-robin baseline is RR-UCB, which maximizes an upper confidence bound score [11] corresponding to c_t: $p_{c_t} + \beta\sqrt{p_{c_t}(1 - p_{c_t})}$. Here β is a hyperparameter trading off exploitation (class membership probability) and uncertainty (as measured by the standard deviation of the binary indicator $[y = c_t]$). We evaluate this policy for $\beta \in \{0.1, 0.3, 1, 3, 10\}$ and report the result of the best performing value of β, denoted β^* in our results (Tab. 5.1). Finally, we consider RR-ENS, which applies the ENS heuristic to the subproblem of finding positives in the target class c_t. The remaining budget is equally allocated among the positive classes, and we adjust the remaining budget when constructing lookahead batches accordingly.

When relevant in the following experiments, we used a utility function that was logarithmic in marginal discoveries for each class: $u(\mathcal{D}) = \sum_{c>1} \log(1 + m_c)$; however, we also conducted a robustness study comparing performance for this utility function with an alternative that had significantly faster (square root) growth in each class. We set our budget $T = 500$ unless specified otherwise and run each policy 20 times, each time with an initial training set containing a randomly selected target. We tested our policies on a wide range of data sets representing a diverse set of applications, which we briefly describe below.

Product recommendation. Our first task uses the Fashion-MNIST data set [211] of fashion item images to simulate a product recommendation setting. Here, we assume a user is looking for specific classes of fashion articles while shopping online. To simulate two different users (corresponding to two search problems), we select specific classes to be the products each user is looking for. We sub-sample these classes uniformly at random, making the positives harder to uncover; the hit rate of a random policy is roughly 2%. While the k-NN model has been found to perform well on the remaining data sets in previous studies [61, 88, 141], with this data, we can select targets that are either easy or difficult to identify with the k-NN, to simulate different levels of search difficulty and study the effect of the quality of the k-NN on the performance of our methods. To this end, we measure the predictive performance of the k-NN and split the two search problems into an "easy" one, where the model scores highly on precision-at-k metrics (particularly important in AS), and a "hard" one, where the k-NN scores lower.

Photoswitch discovery. We also consider the task introduced by Mukadum et al. [141] of searching for photoswitches (molecules that change their properties upon irradiation) in chemical databases that exhibit both desirable light absorbance and long half-lives. Roughly 36% of the molecules in the search space are targets. In their study, the authors partitioned the points into 29 groups by their respective substructures, thus defining a multiclass AS problem with $C = 30$ (a negative class and 29 positive classes). As this data set is smaller in size, we set the budget $T = 100$.

The CiteSeer$^{\text{x}}$ data set. We use the CiteSeer$^{\text{x}}$ citation network data [61], which contains papers published at popular computer science conferences and journals. The label of each paper is its publication venue, and our targets are machine learning and artificial intelligence proceedings. We conduct two sets of experiments with different numbers of classes C. For $C = 5$, we select papers from NeurIPS, ICML, UAI, and JMLR as our four target classes (roughly a 14% hit rate). For $C = 10$, we further include IJCAI, AAAI, JAIR, *Artificial Intelligence,* and *Machine Learning* (ML) as targets, yielding a 31% hit rate.

Drug discovery. Finally, we experiment with a drug discovery task using a massive chemoinformatic data set. The goal is to identify chemical compounds that exhibit selective binding activity to a given protein. The data set consists of 120 such activity classes from BindingDB [121]. For each class, there are a small number of compounds with significant binding activity – these are our search targets. In each experiment for a given $C \in \{5, 10, 15\}$, we select $(C - 1)$ of the 120 classes uniformly at random without replacement to form the targets. They are then combined with 100 000 "drug-like" entries in the ZINC database [184], which serve as the negatives, to make up our search space. We note that each of these active classes has a unique structure and behavior, and combining multiple classes in one AS

Table 5.1: Logarithmic utility and standard errors across 20 repeated experiments for each setting. C is the total number of unique classes in the search space. The best performance in each column is highlighted in **bold**; policies not significantly worse than the best (according to a two-sided paired t-test with a significance level of $\alpha = 0.05$) are in *blue italics*.

	Fashion-MNIST		Photoswitch	CiteSeer^x		Drug discovery		
	easy	hard		$C = 5$	$C = 10$	$C = 5$	$C = 10$	$C = 15$
He and Carbonell [78]	3.99 (0.16)	3.95 (0.16)	12.65 (0.22)	11.30 (0.11)	24.61 (0.14)	3.49 (0.23)	7.19 (0.27)	10.35 (0.35)
Vanchinathan et al. [196]	9.47 (0.50)	6.99 (0.51)	7.16 (0.25)	16.28 (0.06)	31.95 (0.49)	10.19 (0.70)	18.67 (0.88)	25.51 (1.40)
Malkomes et al. [130]	11.31 (0.34)	10.93 (0.47)	6.39 (0.03)	11.28 (0.01)	22.16 (0.01)	7.41 (0.31)	13.77 (0.56)	19.60 (0.66)
ENS	10.48 (0.29)	10.91 (0.14)	5.04 (0.21)	16.57 (0.08)	32.54 (0.50)	10.29 (0.68)	13.79 (0.85)	17.00 (1.15)
RR-greedy	11.41 (0.36)	10.41 (0.44)	11.70 (0.30)	16.66 (0.14)	33.08 (0.13)	10.87 (0.82)	18.66 (1.13)	24.87 (0.95)
RR-UCB	11.51 (0.37) ($\beta^* = 1$)	11.28 (0.47) ($\beta^* = 1$)	11.70 (0.30) ($\beta^* = 3$)	16.68 (0.12) ($\beta^* = 1$)	33.22 (0.13) ($\beta^* = 3$)	11.60 (0.90) ($\beta^* = 3$)	19.27 (1.13) ($\beta^* = 3$)	26.45 (0.96) ($\beta^* = 10$)
RR-ENS	*13.97 (0.09)*	**12.78 (0.31)**	12.54 (0.11)	**17.47 (0.11)**	33.78 (0.16)	10.56 (0.72)	17.83 (0.66)	23.58 (0.88)
One-step	11.83 (0.39)	11.57 (0.47)	8.67 (0.13)	16.77 (0.13)	34.01 (0.13)	11.68 (0.92)	19.06 (0.98)	27.40 (1.11)
DAS (ours)	**14.02 (0.06)**	*12.38 (0.30)*	**13.85 (0.27)**	*17.37 (0.09)*	**34.45 (0.11)**	**13.34 (0.79)**	**23.39 (0.83)**	**31.48 (1.25)**

Table 5.2: Average number of rarest papers found by different policies in the citation network experiment. Ours finds more rare instances and achieves a better balance.

	UAI	JMLR	ML
ENS	51.80	25.25	34.95
One-step	45.75	30.40	29.50
DAS (ours)	44.45	33.85	37.40

problem makes ours a challenging task. Here, the average prevalence of a target is 0.2%; the hit rates are 1%, 2%, and 3% for $C = 5$, 10, and 15, respectively.

Discussion. We report the performance of the policies, quantified by the logarithmic utility function, in Tab. 5.1. Overall, DAS either is the winner or does not perform significantly worse than the best policy. This consistent performance highlights the benefits of our non-myopic, budget-aware approach. Under Fashion-MNIST, most methods work better on the easy problem than on the hard problem, showing the importance of the predictive model in AS. That said, our method remains competitive even under the harder setting. Inspecting the optimal values for RR-UCB's tradeoff parameter β^* in the CiteSeer^x and drug discovery experiments, we notice a natural trend: as C increases, so does the need for exploration, and larger values of β are thus selected.

To illustrate DAS is effective at constructing *diverse* observations, we show in Tab. 5.2 the average numbers of discoveries by the best three policies under the CiteSeer^x $C = 10$ experiments for the three rarest classes: UAI, JMLR, and ML. Here, DAS successfully finds more

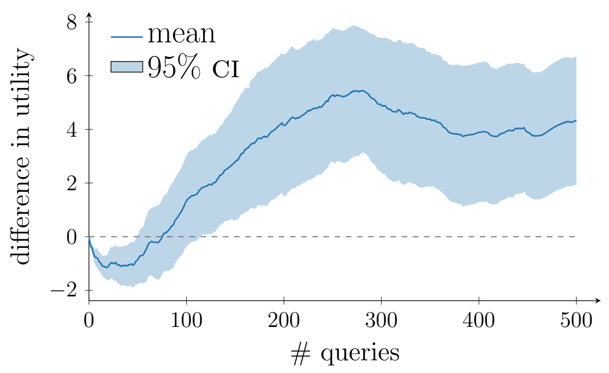


Figure 5.3: Difference in cumulative reward between DAS and one-step across the drug discovery $C = 10$ experiments. DAS dynamically balances exploration and exploitation.

targets from the rarer classes of JMLR and ML, with a better balance among these classes as well.

By design, DAS is always aware of its remaining budget during search and therefore dynamically balances exploration and exploitation. We demonstrate this with the difference in the cumulative reward of DAS vs. one-step under the drug discovery $C = 10$ setting in Fig. 5.3. DAS collects fewer rewards in the beginning while exploring the space. As the search progresses, the policy transitions to more exploitation and ultimately outperforms the myopic one-step.

Quality of approximation. We run simulations to compare the performance of the Jensen's upper bound $\overline{u}$ against Monte Carlo (MC) sampling, under the logarithmic utility function $u(\mathcal{D}) = \sum_{c>1} \log(1 + m_c)$. Using the CiteSeer$^{\text{x}}$ data set, we first construct a training data set $\mathcal{D}$ by randomly selecting 50 points for each class ($|\mathcal{D}| = 50\,C$), and compute the posterior probabilities p_c with the k-NN to simulate a typical AS iteration. A random target batch X of size b is then chosen, and the considered approximation methods are applied on this batch. This entire procedure is repeated 10 times for each setting of (C, b), and the average root mean squared error (RMSE) and time taken for each method to return are reported in Tabs. 5.3 and 5.4. (MC(s) denotes MC sampling using s samples.) Note that in settings with large C or b, exact computation of Eq. (5.7) is prohibitively expensive, in which case MC(2^{15}) is used as the ground truth. Overall, our Jensen's approximation offers a good tradeoff between accuracy and speed.

Table 5.3: Quality and time of our approximation method against MC sampling, averaged across 10 random repeats of CiteSeer^x $C = 5$ experiments. Under each setting, the approximation with the lowest error (with respect to the chosen ground truth) is highlighted **bold**, and so is the fastest method.

	RMSE					Time in seconds				
Ground truth	Exact		MC(2^{15})			Exact		MC(2^{15})		
b	3	10	30	100	300	3	10	30	100	300
Exact	-	-	NA	NA	NA	0.0031	53.7149	NA	NA	NA
MC(2^5)	0.0021	0.0027	0.0060	0.0090	0.0167	0.0026	0.0015	0.0021	0.0056	0.0195
MC(2^{10})	0.0004	0.0008	0.0014	**0.0021**	**0.0025**	0.0190	0.0304	0.0576	0.1574	0.4581
MC(2^{15})	**0.0001**	**0.0001**	-	-	-	0.4908	0.7933	1.7341	4.7440	14.4539
Ours	0.0001	0.0003	**0.0010**	0.0028	0.0059	**0.0001**	**0.0003**	**0.0003**	**0.0003**	**0.0004**

Table 5.4: Quality and time of our approximation method against MC sampling, averaged across 10 random repeats of CiteSeer^x $C = 10$ experiments. Under each setting, the approximation with the lowest error (with respect to the chosen ground truth) is highlighted **bold**, and so is the fastest method.

	RMSE					Time in seconds				
Ground truth	Exact	MC(2^{15})				Exact	MC(2^{15})			
b	3	10	30	100	300	3	10	30	100	300
Exact	-	NA	NA	NA	NA	0.0056	NA	NA	NA	NA
MC(2^5)	0.0023	0.0056	0.0091	0.0141	0.0275	0.0008	0.0016	0.0022	0.0060	0.0178
MC(2^{10})	0.0005	0.0013	**0.0010**	**0.0031**	**0.0032**	0.0167	0.0281	0.0536	0.1522	0.4390
MC(2^{15})	**0.0001**	-	-	-	-	0.5122	0.8178	1.6678	4.7923	13.9236
Ours	0.0002	**0.0007**	0.0017	0.0055	0.0120	**0.0003**	**0.0005**	**0.0004**	**0.0005**	**0.0007**

Other utility functions and misspecification. One may reasonably ask whether DAS still works under other possible utility functions, and how robust the method is against utility misspecification. To tackle these questions, we reran the CiteSeer^x $C = 10$ and drug discovery $C \in \{10, 15\}$ experiments with the square root utility $u(\mathcal{D}) = \sum_{c>1} \sqrt{m_c}$, which rewards additional discoveries of a known class more than the logarithm. This presents an alternative utility with a different asymptotic behavior, but our method can be applied without any algorithmic modification. DAS again consistently performs the best across these settings shown in Tab. 5.5, showing that the policy generalizes well to this utility. As another note on the flexibility of our framework, a user may select different reward functions f_c (e.g., by weighting the functions differently) to prioritize certain classes, and DAS can still run as-is.

Table 5.5: Square root utility and standard errors across 20 repeated experiments for each setting. C is the total number of unique classes in the search space. The best performance in each column is highlighted in **bold**; policies not significantly worse than the best (according to a two-sided paired t-test with a significance level of $\alpha = 0.05$) are in *blue italics*.

	CiteSeer^x		Drug discovery			
	$C = 10$		$C = 10$		$C = 15$	
He and Carbonell [78]	35.51	(0.22)	9.40	(0.33)	13.60	(0.45)
Vanchinathan et al. [196]	48.61	(1.22)	32.28	(1.26)	41.10	(1.99)
Malkomes et al. [130]	31.32	(0.01)	18.26	(0.79)	25.82	(0.87)
ENS	59.31	(0.81)	32.12	(1.17)	36.71	(1.71)
RR-GREEDY	57.37	(0.33)	30.66	(1.82)	37.51	(1.38)
RR-UCB	57.87 ($\beta^* = 3$)	(0.32)	30.77 ($\beta^* = 1$)	(1.47)	38.02 ($\beta^* = 3$)	(1.49)
RR-ENS	60.20	(0.30)	*37.09*	*(1.20)*	45.05	(1.27)
One-step	60.59	(0.46)	*38.86*	*(1.59)*	46.60	(1.70)
DAS (ours)	**62.41**	**(0.24)**	**38.89**	**(1.76)**	**55.50**	**(1.41)**

As for robustness against utility misspecification, we look for any performance drop when DAS is evaluated under a utility different from what the policy uses during search. First, we classify the variants of DAS by the utility they use: (1) the version optimizing the logarithmic utility shown in Tab. 5.1 is denoted by DAS_{log}, (2) the version using the square root utility is DAS_{sqrt}, and (3) the version with the linear utility, $\text{DAS}_{\text{linear}}$, reduces to ENS. We then evaluate these policies using all three utility functions. We also include the one-step policy optimizing the correct utility in the results in Tab. 5.6. Overall, each DAS variant is competitive under the correct utility, always outperforming the corresponding one-step counterpart. Crucially, both concave variants, DAS_{log} and DAS_{sqrt}, perform well "cross-utility" in each other's setting, even outperforming *the one-step policy with the correctly specified objective.* Thus there is merit in adopting DAS even when there may be uncertainty regarding the nuances of the user's "true" utility function. We hypothesize this is because the concave utilities are similar in behavior – both encourage diversity in the collected labels and are optimized by balanced data sets – and a policy effective under one utility is likely to perform well under the other. The linear utility, on the other hand, does not exhibit diminishing returns and behaves differently from the others. Utility misspecification here is thus more costly: $\text{DAS}_{\text{linear}}$ is not competitive under concave utilities, and neither are the concave variants under linear utility.

Table 5.6: Average search utility and standard errors under various utility functions. C is the total number of unique classes in the search space. The best performance in each column is highlighted in **bold**; policies that are not significantly worse than the best (according to a two-sided paired t-test with a significance level of $\alpha = 0.05$) are in *blue italics*.

		utility function	One-step		DAS$_{\text{linear}}$ (ENS)		DAS$_{\text{sqrt}}$		DAS$_{\text{log}}$	
CiteSeer$^{\text{x}}$	$C = 10$	linear	445.90	(5.74)	**459.25**	**(3.23)**	448.35	(2.48)	433.55	(4.42)
		sqrt	60.59	(0.46)	59.31	(0.81)	**62.41**	**(0.24)**	61.41	(0.33)
		log	34.01	(0.13)	32.54	(0.50)	**34.72**	**(0.09)**	34.45	(0.11)
Drug discovery	$C = 10$	linear	370.00	(18.37)	**415.25**	**(12.05)**	304.35	(22.73)	269.25	(18.47)
		sqrt	36.33	(1.48)	32.12	(1.17)	*38.89*	*(1.76)*	**40.44**	**(1.66)**
		log	19.06	(0.98)	13.79	(0.85)	*21.07*	*(1.11)*	**23.39**	**(0.83)**
	$C = 15$	linear	384.65	(11.08)	**427.95**	**(12.88)**	327.50	(14.82)	269.25	(18.47)
		sqrt	46.60	(1.70)	36.71	(1.71)	**55.50**	**(1.41)**	49.20	(1.98)
		log	27.40	(1.11)	17.00	(1.15)	**34.03**	**(1.06)**	*31.48*	*(1.25)*

5.5 Conclusion

We propose a novel active search framework that rewards diverse discoveries and study the problem from the Bayesian perspective. We first prove a hardness result, showing the optimal policy cannot be approximated by a constant in polynomial time. We then design a policy that simulates approximate optimal future queries in an efficient manner. This nonmyopic planning allows our method to be aware of its remaining budget at any point during search and trade off exploitation and exploration dynamically. Our experiments illustrate the empirical success of the proposed policy on real-world problems and its ability to build diverse data sets.

While many real-world applications are modeled by our multiclass AS framework, our model assumes we know *a priori* how many classes are present, which may be violated in many use cases. In other applications, one might also consider the *multilabel* setting where a data point can belong to more than one target class. Investigating the problem under these settings is an interesting future work. Another direction is to extend our approach to the batch setting where multiple queries run simultaneously.

Chapter 6

Amortized Active Search

Active search (AS) has been thoroughly studied in previous work and sophisticated search policies have been developed [61, 88, 89, 90]. Of particular interest is the work of Jiang et al. [88], who derived the Bayesian optimal policy under the simplifying assumption that future experiments are chosen simultaneously in a batch. Here, the number of future experiments is set to be the remaining labeling budget so that this remaining budget is actively accounted for during policy computation. The authors called the resulting policy *efficient nonmyopic search* (ENS), which can be viewed as a budget-aware approximation to the true optimal policy. They showed this budget-awareness induces nonmyopic decisions that automatically balance between strategic exploration of the space and timely exploitation of regions that likely yield many targets, and ultimately achieve state-of-the-art (SOTA) search performance across many tasks. Although the aforementioned simplifying assumption, combined with aggressive pruning, allows ENS to be feasibly applied to problems of considerable size (100 000+ in Jiang et al. [88, 89, 90], for example), the policy retains a *superlinear* computational complexity. This complexity poses a challenge in (1) deploying in real-time applications where decisions need to be made quickly and (2) scaling to large spaces. For instance, a guided data discovery task [139] in visual analytics combines a search algorithm with an interactive visualization to assist a user with their analytic goals in real time, and the time available for the search algorithm to run is thus severely constrained. Modern recommender systems such as YouTube and Amazon must quickly search over millions of items to make recommendations for a large number of users [50]; similarly, there exist databases with billions of synthesizable molecules acting as search spaces for drug discovery [190].

We aim to alleviate the computational cost of budget-aware search by training a small, relatively shallow feedforward neural network to mimic the behavior of the SOTA, expert policy ENS; policy computation is thus amortized as we deploy the trained network as the search policy. We train this policy using the imitation learning technique DAGGER [169], which aids the goal of behavior cloning by iteratively querying the expert's actions on states encountered by the network being trained. This procedure is done with small, synthetic search problems where ENS is cheap to query. We find that the trained policy network successfully learns a beneficial nonmyopic search behavior and, despite the synthetic training data, incurs only a minor decrease in search performance at real-world tasks, in exchange for much faster decision-making. We showcase the usefulness of this computationally lightweight policy with a wide range of search problems spanning diverse applications, including drug discovery tasks of an unprecedented multi-million scale.

6.1 Nonmyopic Search via Budget-Awareness

We briefly reintroduce the SOTA, nonmyopic active search policy that serves as the basis for our solution. To avoid the high cost of reasoning about the dependence among labels of future queries, Jiang et al. [88] made the simplifying assumption that, after our next query, all remaining future queries are made at the same time in a batch. Under this assumption, the future queries in the lookahead – to be optimally chosen to maximize expected terminal utility – can be quickly identified as the set of $(\ell - 1)$ most likely targets [88]. Their policy ENS thus estimates the value of each putative query x_i with the expected utility of the union of x_i and the top $(\ell - 1)$ unlabeled points that are adaptively selected based on each possible label y_i. Again, as the number of future queries in this policy computation is set to exactly match the true length of the decision-making horizon, ENS actively accounts for the remaining labeling budget when making its queries. The authors demonstrated the benefits of this budget-awareness by showing that ENS exhibits nonmyopic, exploratory behavior when the budget is large, and automatically transitions to more exploitative queries as search progresses. This strategic exploration ultimately allows ENS to outperform many search baselines including the one-step policy.

While the aforementioned batch assumption avoids an exponential blowup in computational complexity, ENS still incurs a considerable cost, especially under large values of n, the size of the search space. A naïve implementation with a generic classifier has a complexity of $O\big(n^2 \log n\big)$. The official implementation by Jiang et al. [88], on the other hand, uses a lightweight k-nearest neighbor (NN) classifier that (reasonably) assumes a certain level of locality when probabilities $\Pr(y \mid x, \mathcal{D})$ are updated in light of new data. This structure allows for a faster computation of the batch of future queries in ENS's lookahead, and brings the complexity down to $O\big(n\,(\log n + m \log m + T)\big)$, where m is the largest degree of any node within the nearest neighbor graph corresponding to the k-NN [88]. Unfortunately, this reduced complexity still poses a substantial challenge in two scenarios commonly encountered in AS: large search spaces (e.g., drug discovery) and settings where queries must be rapidly computed (recommender and other real-time systems). We address this problem by training an estimator, specifically a neural network, to learn the mapping from possible candidate queries to the output of ENS, thus amortizing policy computation; the next section details our approach.

6.2 Amortizing Budget-Aware Active Search

Our goal is to amortize search with a neural network, replacing the time-consuming policy computation of ENS with fast forward passes through the network. Crucially, this network should learn a beneficial strategy that outperforms the greedy one-step policy, so that the cost of training and deploying the network outweighs one-step's speed and ease of use. We now discuss our approach using reinforcement learning, specifically imitation learning, to effectively train one such network.

6.2.1 Learning To Search With Imitation Learning

We start with the goal of training a neural network to learn to search using reinforcement learning (RL), as the utility function in AS can be naturally treated as a reward function, and each search run of T iterations as belonging to a budget-constrained episodic Markov decision process. Given a search space defined by $\mathcal{X}$, the current state at iteration t is given by $\mathcal{D}_t$, the data that we have collected thus far, while the unlabeled data points $\mathcal{X} \setminus \mathcal{D}_t$ make up the possible actions that can be taken. Unlike many RL settings, though, the size of the action space in a typical AS problem makes it challenging for common RL training algorithms.[8] This is because many of these algorithms rely on thoroughly exploring the action space to learn about the value of each specific action in a given state, and as $n = |\mathcal{X}|$ grows larger, this task becomes increasingly more daunting.

[8]Not to mention one of our main goals, scaling AS to large search spaces.

Noting that we have access to ENS, an expert policy with demonstrated superior performance throughout previous works, we forgo learning to search from scratch and seek to instead rely on ENS for guidance. This proves to be more feasible, as we can leverage imitation learning techniques in which we collect a data set of state and expert's action pairs $S = \{s, \text{ENS}(s)\}$ and train a neural network to learn this mapping. Here, we wish our neural network to output the same decision generated by ENS (i.e., which unlabeled point to query) given the current state (the observed data) of a search problem. This is done by treating the goal of imitating the expert policy as a classification problem, where a data point is characterized by a given state s of a search, and the corresponding label is the expert's decision $\text{ENS}(s)$. A neural network classifier is then trained to correctly classify $\text{ENS}(s)$ as the desirable label among all possible actions, by minimizing the corresponding cross-entropy loss.

Algorithm 6 DAGGER for imitation learning

```
1: inputs number of training iterations $N$,
   expert policy $\pi_*$, problem generator $G$
2: initialize $S \leftarrow \emptyset$
3: initialize $\hat{\pi}_0$ randomly
4: for $i = 1$ to $N$ do
5:     sample AS problems $\mathcal{X} \sim G$
6:     roll out $\hat{\pi}_{i-1}$ on $\mathcal{X}$ to obtain states $\{s\}$
7:     assemble $S_i = \{(s, \pi_*(s))\}$
8:     aggregate $S \leftarrow S \cup S_i$
9:     train $\hat{\pi}_i$ on $S$ until convergence
10: end for
11: returns best $\hat{\pi}_i$ on validation
```

The training data S for imitation learning can be assembled in several ways. For example, we could run ENS on training problems and record the states encountered and the decisions computed. However, this leaves the possibility that as the trained network is deployed, it will arrive at a state very different from those seen during training, and thus output unreliable decisions. We use DAGGER [169], a well-established imitation learning technique, to address this problem. DAGGER is a meta-learning algorithm that iteratively rolls out the policy currently being trained (i.e., it uses the current policy to make decisions), collects the expert's actions on the encountered states, and appends this newly collected guidance to the training set to improve the policy being trained. This iterative procedure allows the expert policy to be queried more strategically, targeting states the current policy network is likely to be in. Alg. 6 summarizes this procedure.

6.2.2 Constructing Search Problems for Training

To realize DAgger, we require access to a search problem "generator" that provides AS problems in which we are free to roll out the network being trained and observe its performance. One may consider directly using one's own real-world use case to train the policy network; however, running DAgger on real-life AS problems might prove infeasible. This is because DAgger is an iterative training loop that requires many training episodes to be played so that the collected training data S could cover a wide range of behaviors of the expert to be imitated. We cannot afford to dedicate many real search campaigns to this task, especially under our assumption of expensive labels. Instead, we turn to synthetic problems generated in a way that is sufficiently diverse to present a wide range of scenarios under which we may observe ENS's behavior. In addition to constructing these problems and running a policy currently being trained on them at little computational cost, we can limit the size of the problems so that ENS can be queried efficiently.

When called, our data-generating process constructs a randomly generated set $\mathcal{X}$. We sample from a Gaussian process (GP) [162] at the locations in $\mathcal{X}$ to obtain a real-valued label for each $x \in \mathcal{X}$, which is then converted to a binary label by thresholding at a chosen quantile. The generated search space and labels are returned as a training problem. Although this procedure is quite simple and, in using a GP, assumes a certain level of smoothness in the labels, we observe that the generated problems offer reasonable variety of structures with "clumps" of targets of variable number and size. This variety successfully facilitates imitation learning, as later demonstrated by the empirical performance of our trained policy on real-world problems. We include more details in Appx. B.

6.2.3 Feature Engineering & Implementation

The effectiveness of any training procedure in RL crucially depends on the quality of the representation of a given state during a search. To characterize a state in a way that aids learning, we use the following features to represent each unlabeled data point $x \in \mathcal{X} \setminus \mathcal{D}$ remaining in a search:

- the posterior probability that the data point has a positive label $\Pr(y = 1 \mid x, \mathcal{D})$,
- the remaining budget $\ell = T - t$,

- the sum of posterior probabilities of the $(\ell - 1)$ unlabeled nearest neighbors of x:

$$\sum_{x' \in \textsc{nn}(x,\, \ell-1)} \Pr(y' = 1 \mid x', \mathcal{D}), \tag{6.1}$$

  where $\textsc{nn}(x, k)$ denotes the set of k unlabeled nearest neighbors of a given $x \in \mathcal{X}$, and
- the sum of similarities between x and its $(\ell - 1)$ unlabeled nearest neighbors:

$$\sum_{x' \in \textsc{nn}(x,\, \ell-1)} s(x, x'), \tag{6.2}$$

  where $s(x, x') \in [0, 1]$ denotes the similarity between two given points $x, x' \in \mathcal{X}$.

Which similarity function s to use to compute the nearest neighbors of each point and the corresponding similarity values depends on the application. We use the radial basis function kernel $s(x, x') = \exp\big(-\frac{\|x-x'\|^2}{2\,\lambda^2}\big)$ during training and upon deployment for appropriate tasks in Sect. 6.4, but this can be replaced with another function more applicable to a given domain.

We specifically design the third and fourth features to relate the value of an unlabeled point we may query to the characteristics of its nearest neighbors. Intuitively, a point whose neighbors are likely targets is a promising candidate, as it indicates a region that could yield many hits. On the other hand, points that are close to its neighbors (e.g., cluster centers) could also prove beneficial to query, as they help the policy explore the space effectively. Further, the number of nearest neighbors to include in these computations is set to match the length of the horizon, allowing these features to dynamically adjust to our remaining budget. Overall, the four features make up the feature vector of each candidate point, and concatenating all feature vectors gives the state representation of a given search iteration. We also note that our state representation is task-agnostic and applicable across AS problems of varying structures and sizes, which is crucial for training our policy on the different problems we generate, as well as for when we deploy our trained policy on unseen problems.

Here, finding the nearest neighbors of each point may prove challenging under large spaces. We leverage the state-of-the-art similarity search library FAISS [91] to perform efficient approximate nearest neighbor search when exact search is prohibitive.[9] FAISS allows us to

[9] We set the number of clusters into which the search space is split when performing approximate neighbor search at $\lfloor 4\sqrt{n} \rfloor$, following `https://github.com/facebookresearch/faiss/issues/112`.

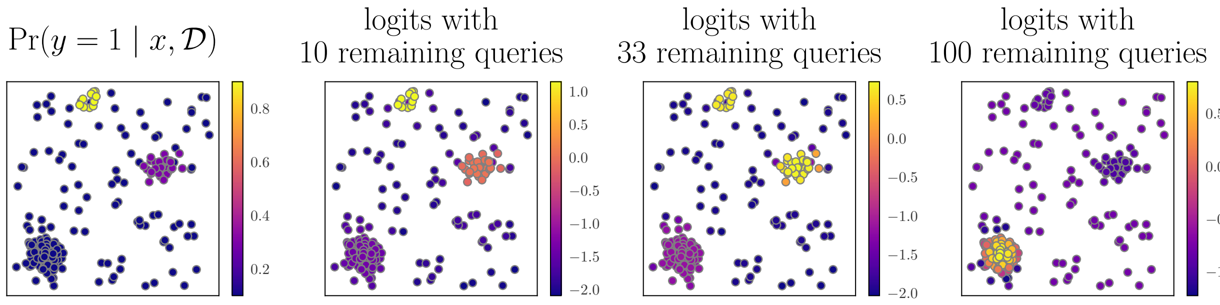


Figure 6.1: Demonstration of our trained policy's budget-awareness with a toy example. **Left panel**: the probability that a point is a target. **Remaining panels**: computed logits and the point selected to be the next query under different labeling budgets. Our policy appropriately balances between exploitation under a small labeling budget and strategic exploration if the budget is large.

significantly accelerate this step, completing, for example, the neighbor search for our largest problem in Sect. 6.4 of 6.7 million points in roughly one hour. Once this neighbor search is done before the actual search campaign, the time complexity of constructing the features above at each search iteration is $O(n)$.

We run Alg. 6 for $N = 50$ iterations, each consisting of 3 training problems. At the end of each iteration, we train a small policy network with 5 fully connected hidden layers (with 8, 16, 32, 16, and 8 neurons, respectively) and ReLU activation functions using minibatch gradient descent with Adam optimizer [97]. The trained policy is then evaluated on a fixed set of 3 unseen validation problems. We set the labeling budget $T = 100$ across all generated problems.

6.2.4 Demonstration of Learned Search Strategy

Before discussing our experiment results, we briefly demonstrate the learned behavior of our trained policy network using an illustrative toy example visualized in Fig. 6.1. The search space $\mathcal{X}$ consists of uniformly sampled points as well as 3 distinct clusters of different sizes. The left panel shows $\Pr(y = 1 \mid x, \mathcal{D})$, the probability that each point is a target, specifically set so that:

- for each of the 100 uniformly sampled points, $\Pr(y = 1 \mid x, \mathcal{D}) = 0.1$,
- for each point in the small cluster of size 10 at the top, $\Pr(y = 1 \mid x, \mathcal{D}) = 0.9$,

- for each point in the medium cluster of size 30 on the right, $\Pr(y = 1 \mid x, \mathcal{D}) = 0.3$, and
- for each point in the large cluster of size 100 on the bottom left, $\Pr(y = 1 \mid x, \mathcal{D}) = 0.1$.

This problem structure presents an interesting choice between exploiting the small cluster of very likely targets and exploring larger clusters that contain less likely targets. A good policy should select exploitation if the remaining budget is small and choose to further explore otherwise. The remaining panels in Fig. 6.1, which visualize the logits computed by our trained policy under different remaining budgets $\ell \in \{10, 33, 100\}$, show that this is exactly the case: the policy targets the cluster of likely targets when the budget is small, and moves to larger clusters as the budget increases. Further, when exploring, the policy appropriately favors cluster centers, which offer more information about the space. This balance between exploitation and strategic exploration our trained policy exhibits indicates that the policy has learned a meaningful search behavior from ENS, which translates into good empirical performance, as later shown in Sect. 6.4.

6.3 Related Work

Active search. We continue the line of research on active search (AS) [61, 39], which previous works have also referred to as active learning and adaptive sampling for discovery [205, 204, 212] or active covering [87]. Garnett et al. [61] studied the Bayesian optimal policy, and Jiang et al. [88] proposed ENS as a budget-aware approximation demonstrating impressive empirical success. ENS has since then been adopted under various settings, including batch [89], cost-aware [205, 204, 90], multifidelity [147], and diversity-aware AS [146]. We propose to amortize policy computation of ENS using imitation learning, scaling nonmyopic search to large data sets.

Amortization via neural networks. Using neural networks to amortize expensive computations has seen increasing interests from the machine learning community. Of note is the work of Foster et al. [56], who tackled amortizing maximizing expected information gain [125, 126] for Bayesian experimental design (BED) [117] with a design network trained on a specialized loss function, and was a major inspiration for our work. Subsequent works [24, 180] have studied BED under other settings such as those with discrete action spaces.

Liu et al. [120] tackled amortizing Gaussian process (GP) inference by training a transformer-based network as a regression model to predict optimal hyperparameters of GPs with stationary kernels; Bitzer et al. [23] later extended the approach to more general kernel structures. Andrychowicz et al. [9], on the other hand, learned an optimization policy with a recurrent neural network that predicts the next update to the parameters to be optimized based on query history; the policy network was shown to outperform many generic gradient-based optimizers. Also related is the work of Konyushkova et al. [101], where a regressor was trained to predict the value of querying a given unlabeled data point for active learning.

Reinforcement learning. Reinforcement learning (RL) has proven a useful tool for learning effective strategies for planning tasks similar to AS. Examples include active learning policies for named entity recognition [54, 118], neural machine translation [119], and active learning on graphs [82]. Igoe et al. [84] were interested in path planning for drones, framed as a specialized AS setting with linear models and many agents. Sarkar et al. [174] and Sarkar et al. [175] studied the problem of visual AS, a realization of AS on images for geospatial exploration. Overall, the methodologies in these works rely on being able to generate many training episodes to learn an effective RL policy from scratch, which cannot be realized in our setting. Having access to ENS, we instead leverage imitation learning to learn to search from this expert on synthetically generated search problems. When deployed, our trained policy can be applied to a diverse set of use cases, as demonstrated in the next section.

6.4 Experiments

We tested our policy network, which we call *amortized nonmyopic search*, or ANS, and a number of baselines on search problems spanning a wide range of applications. For each problem included, we run each policy 10 times from the same set of initial data $\mathcal{D}_0$ that contains one target and one non-target, both randomly sampled. Each of these runs has a labeling budget of $T = 100$.

Baselines. We compare our method against the expert policy ENS which ours was trained to mimic, as well as the one-step policy that greedily queries the most likely target. We also implement a number of baseline policies from the literature. The first is a family of upper confidence bound (UCB) policies [10, 32] that rank candidates by the following score: $p + \beta\sqrt{p(1-p)}$, where $p = \Pr(y = 1 \mid x, \mathcal{D})$ is the posterior probability that a given candidate is a target, and β is the tradeoff parameter balancing exploitation (favoring large p) and exploration (favoring large uncertainty in the label, as measured by $\sqrt{p(1-p)}$). Here, we have β take on values from $\{0.1, 0.3, 1\}$. Another baseline is from Jiang and Rostamizadeh [87], who proposed a simple explore-then-commit (ETC) scheme that uniformly samples the space for m iterations and then switches to the greedy sampling strategy of one-step for the remaining of the search. We run this ETC policy with $m \in \{10, 20, 30\}$. Finally, we include the policy by Xu et al. [212], which uses the information-directed sampling (IDS) heuristic [170, 171] that scores each candidate query x by the ratio between (1) information about the labels of the current ℓ most likely targets ($\ell = T - t$ is the length of the remaining horizon), gained by querying x and (2) the expected instant regret from querying x. We note that Xu et al. [212] proposed this policy under specialized AS settings that allow information gain to be efficiently computed; they also only considered problems with fewer than 1000 points. For our experiments, we can only apply IDS to relatively small spaces where the policy is computationally feasible.

Search problems. We now discuss the search problems making up our experiments. The first was posed by Andrade-Pacheco et al. [8], who sought to identify disease hotspots within a region of interest. The provided data sets correspond to 4 distinct AS problems of finding locations with a high prevalence of schistosomiasis in Côte d'Ivoire and Malawi and of lymphatic filariasis in Haiti and the Philippines. Each problem consists of 1500 points, with targets accounting for 10%–34% of the space. For our second task, following Nguyen and Garnett [146], we simulate product recommendation problems using the Fashion-MNIST data [211], which contains 70 000 images classified into 10 classes of clothing articles. We first randomly select 3 out of the 10 classes as products a user is interested in (i.e., our search targets). We further sub-sample these 3 classes uniformly at random to increase the difficulty of the problem; the resulting prevalence rate of the targets is roughly 6%. We repeat this process 10 times to generate 10 search problems with this data.

Borrowing from previous works [88, 89], we use a data set from the materials science literature [95, 203] containing 106 810 alloys, of which 4275 can form bulk metallic glasses with high toughness and wear resistance and are our search targets. Another application comes from drug discovery, where we aim to identify "active compounds", chemical compounds that bind with a targeted protein. Garnett et al. [62] assembled a suite of such drug discovery problems, each of which consists of the active compounds for a specific protein from the BindingDB database [121], and 100 000 molecules sampled from the ZINC database [184] that

Table 6.1: Average number of targets found and standard errors by each search policy across 10 repeats of all instances of a given task. Settings that are computationally prohibitive for a given policy are left blank. The best policy in each setting is highlighted **bold**; policies not significantly worse than the best (according to a two-sided paired t-test with a significance level of $\alpha = 0.05$) are in *blue italics*. The state-of-the-art ENS achieves the best performance in all feasible settings and is closely followed by our policy network ANS, which in turn yields the best result in the large-scale problems.

	Disease hotspots	Fashion-MNIST	Bulk metal glass	Drug discovery (small)	GuacaMol	Drug discovery (large)
ETC($m = 10$)	50.50 (4.44)	47.79 (3.89)	81.70 (3.66)	75.46 (2.64)	10.37 (1.80)	*33.97 (4.14)*
ETC($m = 20$)	52.25 (3.47)	42.97 (3.47)	73.40 (3.67)	68.46 (2.37)	9.42 (1.61)	30.42 (3.70)
ETC($m = 30$)	48.75 (3.12)	38.16 (3.03)	65.70 (3.17)	60.76 (2.11)	8.56 (1.45)	26.59 (3.24)
UCB($\beta = 0.1$)	48.20 (4.21)	51.66 (4.24)	*88.80 (4.00)*	80.74 (2.86)	11.21 (1.93)	*36.94 (4.49)*
UCB($\beta = 0.3$)	48.15 (4.20)	51.66 (4.24)	*88.80 (4.00)*	80.74 (2.86)	11.21 (1.93)	*36.94 (4.49)*
UCB($\beta = 1$)	45.48 (3.52)	51.10 (4.18)	81.80 (3.15)	75.14 (2.59)	11.21 (1.93)	*36.94 (4.49)*
one-step	48.20 (4.21)	51.66 (4.24)	*88.80 (4.00)*	80.74 (2.86)	11.21 (1.93)	*36.94 (4.49)*
IDS	48.83 (4.08)	51.66 (4.24)	—	—	—	—
ENS	**57.67 (3.22)**	**88.15 (1.52)**	**91.10 (3.48)**	**86.82 (2.41)**	—	—
ANS (ours)	*57.25 (3.58)*	85.72 (1.69)	*89.90 (4.20)*	*85.29 (2.47)*	**15.51 (2.36)**	**39.83 (3.76)**

act as the negative pool. Our experiments include the first 10 problems where on average the active compounds make up 0.5% of the search space.

Finally, to demonstrate the ability to perform search on large spaces achieved by our method ANS, we consider two large-scale, challenging drug discovery tasks. The first employs the GuacaMol database of over 1.5 million drug-like molecules that were specifically curated for drug discovery benchmarking tasks involving machine learning [29]. In addition to these molecules, the database offers a family of objective functions to measure the molecules' quality using a variety of criteria. We use each objective function provided to define a search problem as follows. We first randomly sample a set of 1000 molecules which we fully label using the objective functions. We then define the search targets as those of the remaining unlabeled molecules whose scores exceed the 99-th percentile of the labeled set; in other words, the goal of our search is the top 1% molecules. In total, we assemble 9 such AS problems with GuacaMol. For our second task, we follow the procedure by Garnett et al. [62] described above with the BindingDB and ZINC databases, this time expanding the

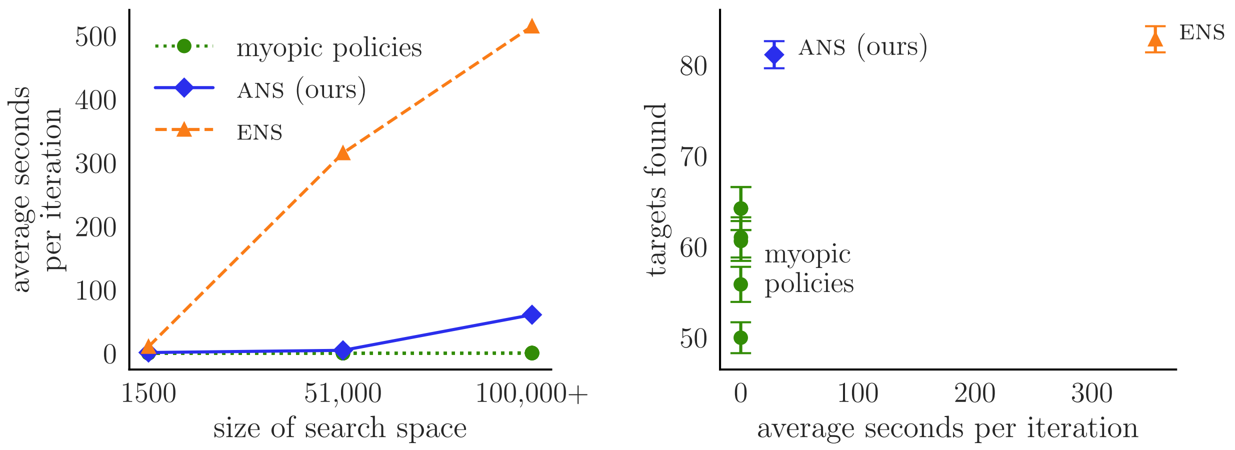


Figure 6.3: The time taken per iteration by different policies in the small- and medium-scale experiments. **Left**: average number of seconds per iteration with respect to search space size. **Right**: average number of targets found and standard errors vs. time per iteration.

negative pool to all drug-like molecules in ZINC [190]. This results in a search space of 6.7 million candidates, of which 0.03% are the active compounds we aim to search for.

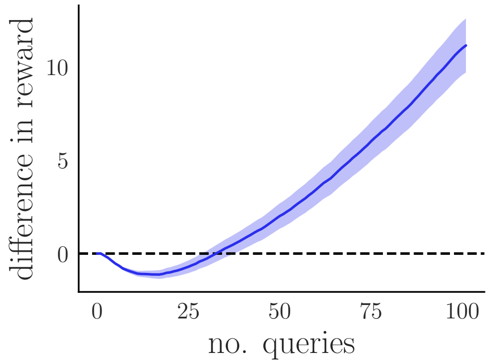


Figure 6.2: The average difference in cumulative reward and standard errors between our policy and one-step. Our policy spends its initial budget exploring the space and finds fewer targets in the beginning but smoothly switches to more exploitative queries and outperforms one-step at the end.

Discussions. Tab. 6.1 reports the performance of the search policies – measured in the number of targets discovered – in each of these tasks, where settings that are computationally prohibitive for a given policy are left blank. From these results, we observe a clear trend: the state-of-the-art policy ENS consistently achieves the best performance under all settings that it could feasibly run, while our trained policy network ANS closely follows ENS, sometimes outperforming the other baselines by a large margin. Our method also yields the best result in large-scale problems, demonstrating its usefulness in large search spaces. Among the baselines, we note the difficulty in setting the number of exploration rounds m for ETC, since no value of m performs the best across all settings. Results from UCB policies, on the other hand, indicate that prioritizing exploitation (setting the parameter β to a small value) is beneficial, a trend also observed in previous work [88].

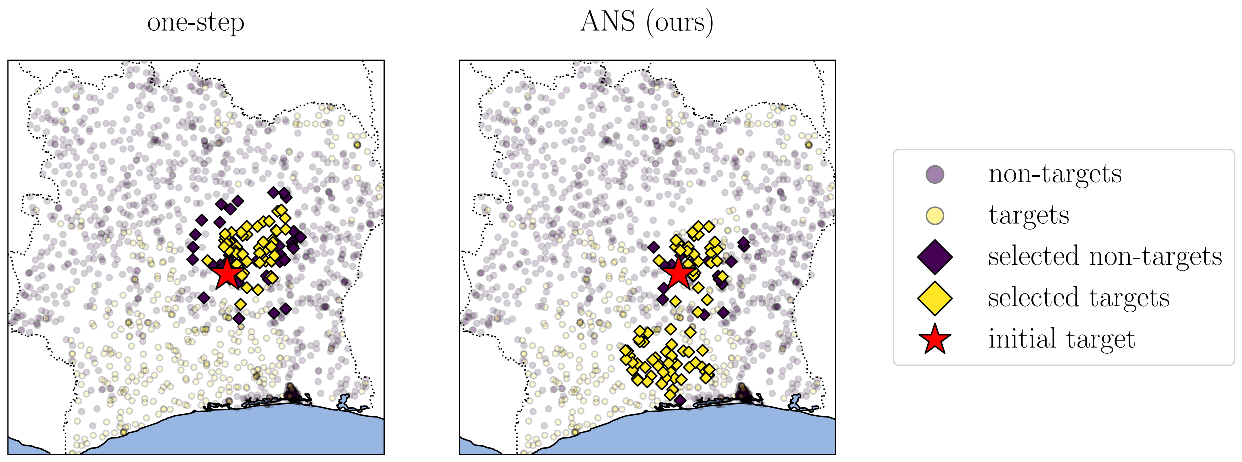


Figure 6.4: Locations in Côte d'Ivoire selected by the one-step policy and by ours in an illustrative run with the disease hotspot data, where our policy discovers a larger cluster.

To illustrate the tradeoff between performance and speed achieved by ANS, the left panel of Fig. 6.3 shows the average time taken by ANS, ENS, and myopic baselines per iteration as a function of the size of the data, while the right panel shows the number of discoveries by each policy vs. the same average time per iteration. These plots do not include the results from the large-scale problems so that the comparison with ENS is fair, or IDS which is slower than ENS but does not perform as well. We see that ANS finds almost as many targets as ENS but is much more computationally lightweight. We thus establish a new point on the Pareto frontier of the performance vs. speed tradeoff with our search policy. In the 6.7 million-point drug discovery problems, ANS on average takes 36.94 ± 0.15 minutes per iteration, which we deem entirely acceptable given the boost in performance compared to faster but myopic baselines, the fact that the time cost of labeling is typically much higher, and ENS, in comparison, is estimated to take roughly 10 hours per iteration on the same scale.

As a demonstration of our policy's strategic explorative behavior learned from ENS, Fig. 6.4 shows the result of an illustrative run by the one-step policy vs. ANS from the problem of finding schistosomiasis hotspots in Côte d'Ivoire [8]. We note that the queries made by one-step are localized within the center region containing the target in the initial data $\mathcal{D}_0$, while ANS is able to discover a larger cluster of targets to the south. Further, Fig. 6.2 visualizes the cumulative difference in utility between ANS and one-step across all experimental settings. Here, ANS initially finds fewer targets than one-step, as the former tends to dedicate its queries to exploration of the space when the remaining budget is large; however, as the search progresses, ANS smoothly transitions to more exploitative queries and ultimately

Table 6.2: Average number of targets found and standard errors by each ablated policy across 100 product recommendation tasks with FashionMNIST. The best policy is highlighted **bold**.

without target probability	without remaining budget	without neighbor probability sum	without neighbor similarity sum	imitation learning without DAgger	REINFORCE without imitation learning	ANS (ours)
66.93 (1.51)	80.11 (1.80)	73.35 (1.61)	63.29 (3.35)	59.66 (3.80)	75.15 (1.52)	**85.72(1.69)**

outperforms the greedy policy. The same pattern of behavior has been observed from ENS in previous works [88, 147, 146].

Ablation study. We use the 10 product recommendation tasks from the FashionMNIST data to quantify the value of various components of our framework. First, we trained four additional policy networks, each learning from ENS without one of the four features discussed in Sect. 6.2.3. We also trained another network using imitation learning but without DAgger's iterative procedure: we ran the expert policy ENS on $3 \times 50 = 150$ generated search problems (the same number of problems generated to train the policy examined in the main text), kept track of the encountered states and selected actions, and used these data to train the new network until convergence only once. Finally, we trained a policy network without imitation learning using the REINFORCE policy gradient algorithm [186]. The performance of these policies, along with that of ANS as a reference, is shown in Tab. 6.2. We see by removing any component of our imitation learning procedure, we incur a considerable decrease in performance, which demonstrates the importance of each of these components.

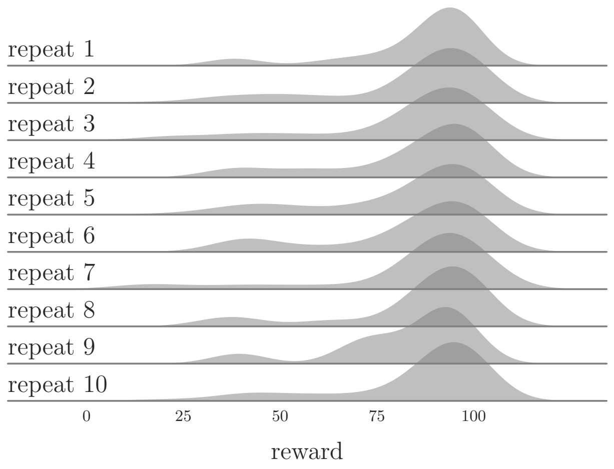


Figure 6.5: Distributions of the number of targets found across 100 product recommendation tasks with FashionMNIST by 10 policy networks trained with different initial random seeds. The distributions are comparable, indicating that the trained policy networks behave similarly.

Training stability. We rerun our training procedure with DAgger for 10 times using different random seeds and evaluate the trained policy networks using the experiments with the FashionMNIST data. Each row of Fig. 6.5 shows the distribution of the number of targets found by each of these 10 policy networks across the 100 search problems. We observe that the variation across these 10 distributions is quite small, especially compared to the variation across different search runs by the same policy network. This shows that our training procedure is stable, resulting in policy networks that behave similarly under different random seeds.

Refinement under repeated search. In many settings that AS targets, multiple search campaigns may be conducted within the same search space. For example, as in our drug discovery experiments in Sect. 6.4, a scientist may explore a molecular database to identify candidates with different desirable properties. As the search for a given property concludes, the next search stays within the same database but now targets a different property. In these situations, we may reasonably seek to refine our search strategy throughout these episodes using the results we observe, so that our search policy could improve using its past experiences. We identify two approaches to such refinement:

- If a neural network is used as the search policy, it can be updated by a policy gradient algorithm such as REINFORCE [186] after each episode.
- If a deep autoencoder (DAE) is used to produce a representation of the search candidates (on which the nearest neighbor search described in Sect. 6.2.3 is conducted), the autoencoder can be updated with a semisupervised loss [99] that accounts for the labels it iteratively uncovers throughout the search.

To investigate the effects of each of these approaches on the search performance of our policy trained with imitation learning and examined in the main text, we engineer another version of the FashionMNIST data set [211] that simulates a setting of repeated search. We first randomly choose 5 out of 10 classes in the data set to act as possible target sets throughout the repeated searches. These selected classes are then sub-sampled uniformly at random so that there are only 1000 data points per class; this yields a data set of 40 000 points in total. We then use a variational autoencoder [98] to learn a two-dimensional representation of these 40 000 candidates.

We allow 100 search episodes within this database, where in each episode, 1 of the chosen 5 classes is randomly selected as the target class. To implement the second approach to search refinement, we train a variational Gaussian process classifier on the observed data $\mathcal{D}$ and use the corresponding evidence lower bound (ELBO) to make up the supervised component of the joint loss of the semisupervised model. While we update the search policy using

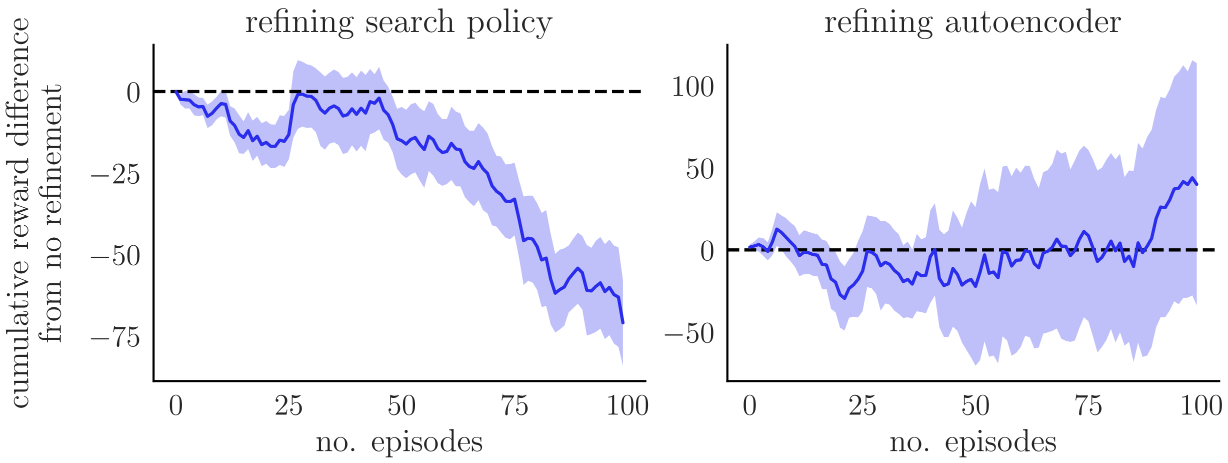


Figure 6.6: Average cumulative difference in the number of targets found and standard errors between (**left**) updating the policy network using REINFORCE or (**right**) updating the autoencoder producing the representation of the candidates vs. performing no updates.

the REINFORCE loss at the end of each episode, an update to the semisupervised VAE is performed for every 20 iterations within one episode.

Fig. 6.6 shows the value of each of the two update schemes as the cumulative difference in the number of targets found between each scheme compared to performing no updates (both the search policy and the representation of the data points are kept fixed) throughout 100 search episodes across 10 repeats. Surprisingly, attempting to further refine the search policy using REINFORCE actually hurts performance, resulting in an increasing gap in reward between the initial policy and the one continually updated. On the other hand, we see that updating the initial unsupervised VAE to account for the observed labels yields an improvement in performance on average, but this improvement is not consistent across the 10 repeats. Overall, we show the difficulty in further updating our trained search policy using real experiences under repeated searches, and hypothesize that more sophisticated reinforcement learning procedures such as the double Q-learning algorithm [137] are needed to improve learning, which we leave as future work.

6.5 Conclusion

We propose an imitation learning-based method to scale nonmyopic active search to large search spaces, enabling real-time decision-making and efficient exploration of massive databases common in product recommendation and drug discovery beyond myopic/greedy strategies. Extensive experiments showcase the usefulness of our policy, which mimics the state-of-the-art policy ENS while being significantly cheaper to run. Future directions include deriving a more effective reinforcement learning strategy to train our policy network, potentially outperforming ENS, as well as extending to other active search settings such as batch [88] and diversity-aware search [146, 145].

Chapter 7

Applications of Experimental Design

Throughout this dissertation, we have shown that experimental design is a flexible framework that can be applied to a wide range of use cases to accelerate exploration and discovery. In this chapter, we demonstrate this flexibility one more time by enumerating instances of successful application of our algorithms to real-world experimental design campaigns.

7.1 Efficient Discovery of Visible Light-Activated Azoarene Photoswitches with Long Half-Lives Using Active Search

Photoswitches are molecules that undergo a reversible, structural isomerization after exposure to different wavelengths of light. The dynamic control offered by molecular photoswitches is favorable for materials chemistry, photopharmacology, and catalysis applications. Ideal photoswitches absorb visible light and have long-lived metastable isomers. We used high throughput virtual screening to predict the absorption maxima ($\lambda_{\max}$) of the E-isomer and half-lives ($t_{1/2}$) of the Z-isomer. However, computing the photophysical and kinetic properties of each entry of a virtual molecular library containing 10^3–10^6 entries with density functional theory is prohibitively time-consuming. We applied active search to intelligently search a chemical search space of 255 991 photoswitches based on 29 known azoarenes and their derivatives. Scheme 7.1 shows an illustration of the iterative processes used to identify ideal photoswitches.

Phase 1: An initial screen of 50–100 molecules is processed through an automated computational workflow developed by Abreha et al. [2]. RDKit[113] is used to generate 3-D coordinates from a simplified molecular-input line-entry system (SMILES)[206] string, followed

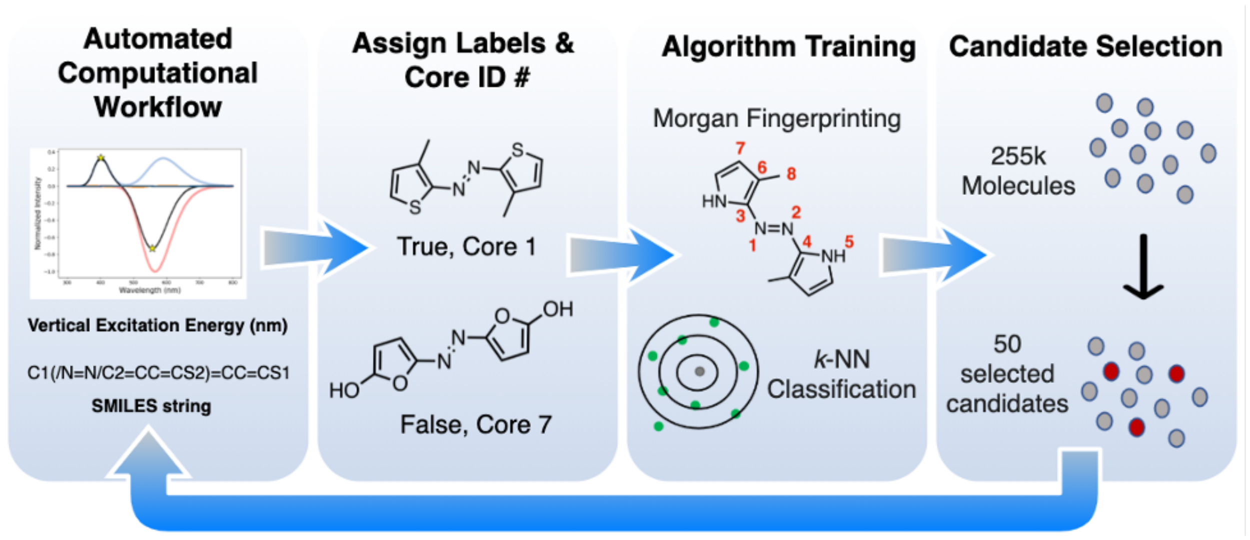


Figure 7.1: The multipronged iterative procedure used to update the active search algorithm with DFT results.

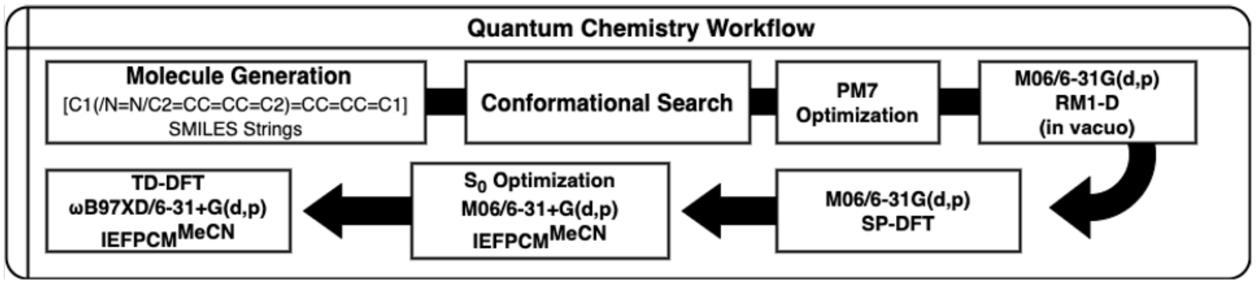


Figure 7.2: Quantum chemical workflow for computing the $\lambda_{\max}$ for all molecules considered in this study.

by a low-mode conformational search where each conformer (4 total) is minimized with the Universal Force Field [161]. The lowest energy conformer is determined through semi-empirical optimizations and a single-point energy calculation. The lowest energy structure is optimized with M06 [218]/6-31+G(d,p) [57, 48] and IEFPCM$^{\text{MeCN}}$, [192] and a vibrational analysis confirms the stationary point as the true minimum if it has only positive frequencies. The $\lambda_{\max}$ is calculated with a single point energy calculation using ωB97XD[33]/6-31+G(d,p)//M06[218]/6-31+G(d,p). Figure 7.2 shows the automated workflow of quantum chemical calculations used to compute the excitation energies and corresponding $\lambda_{\max}$ for selected molecules from our virtual library.

Phase 2: An in-house Python script assigns a “core ID” (1–29) to each computed structure. Cores are determined using a substructure analysis included in RDKit. `True` or `False` labels are assigned to each smiles string based on the pre-determined threshold, λ_{max} greater than 450nm.

Phase 3: A machine learning model is trained on the set of labeled molecules to guide the search algorithm. First, we generate the Morgan fingerprint [167] of each molecule and compute the Tanimoto similarity coefficient [208] between each pair of molecules. We then build a k-nearest neighbors (k-NN) predictive model that computes the probability of a given unlabeled molecule having a positive label, given the data we have observed thus far. This k-NN model is then utilized by the search algorithm. Note that the Morgan fingerprints and Tanimoto similarity coefficients only need to be computed once, while the k-NN is updated with newly labeled data at each iteration

Phase 4: The active search algorithm builds the set of 50 recommendations, selecting among all unlabeled molecules. These recommendations are then sent to Phase 1 to be computed and labeled. This procedure repeats for a total of 40 iterations, sampling 1 962 molecules from the space.

7.1.1 Methods

We adapted the active search method, which has shown impressive performance in molecular discovery in previous studies [62, 88, 89]. The method was first introduced by Garnett et al. [61] and extended to the batch setting by Jiang et al. [89]. Formally, suppose we have a large set of elements $\mathcal{X} = \{x_i\}$, among which there is a small subset $\mathcal{R} \subset \mathcal{X}$ of valuable elements that we wish to search for (i.e., molecules exhibiting a desired property). We do not know which members of $\mathcal{X}$ belong to $\mathcal{R}$ *a priori*, but whether a specific element x belongs to $\mathcal{R}$ can be determined by querying an oracle, requesting for the binary label $y = \mathbb{1}\{x \in \mathcal{R}\}$, where $\mathbb{I}\{\cdot\}$ is the indicator function. In this work, the binary label denotes whether a molecule exceeds the λ_{max} threshold of 450nm. Further, we assume that at each iteration of the search, b elements are inspected simultaneously, requiring that queries to the oracle be made in batches of size b. This models experimental settings in which multiple experiments may be run in parallel to maximize throughput, contrasting with the fully sequential setting where queries are made one after another; here, $b = 50$. The goal is to design a sequence of queries limited by a predetermined budget, such that the number of target elements uncovered by querying the oracle is maximized. As such, we naturally define the utility of a given set of observations $\mathcal{D} = \{(x_i, y_i)\}$ to be the total number of targets found:

$$u(\mathcal{D}) = \sum_{y_i \in \mathcal{D}} y_i.$$

We aim to determine the sequence of queries that maximizes our definition of utility in the expected case using Bayesian decision theory. This framework first requires a classification model that computes the posterior probability that an unlabeled point x belongs to $\mathcal{R}$, given the elements we have inspected thus far in $\mathcal{D}$, $\Pr(y = 1 \mid x, \mathcal{D})$. The active search method is model-agnostic and does not make any further assumptions about this predictive model. In the next section, we describe the k-nearest neighbors model we use for this classification task.

We denote $T = t\,b$ to be the total number of queries allowed to be made given our budget, where t is the number of search iterations). We further denote by $\mathcal{D}_i$ the observations collected at the end of iteration i. At iteration $i + 1 \leq t$, the best batch of queries (of size b) we can make, denoted as X_{i+1}, maximizes the expected value of the utility of the dataset at termination $\mathcal{D}_t$:

$$X_{i+1} = \arg\max_{X} \mathbb{E}\Big[u(\mathcal{D}_t) \mid X, \mathcal{D}_i\Big].$$

Jiang et al. [89] analyzed and provided further interpretation for this expected utility. Specifically, it decomposes into the sum of the expected number of positives in the current batch and the expected number of positives found in the future, assuming optimal behavior [89]. The second term in this sum is large relative to the first when the remaining budget is large; as such, the objective naturally balances between exploration and exploitation. This quantity only coincides with the expected number of molecules in the current batch at the very last iteration where the greedy batch is optimal (the second term is zero). We refer the reviewer to Equation (3) in Jiang et al. [89] for more discussion on this objective function.

Although this objective can be derived using the standard procedure of backward induction [21], it involves $t - i$ nested steps of sampling over unknown labels of candidate queries and maximizing the future expected utility. This computation is prohibitively expensive for horizons $t - i \geq 3$, rendering the optimal query infeasible to calculate in practice.

We adopt the *sequential simulation* strategy proposed by Jiang et al. [89] as an efficient approximation to the optimal batch of queries. First, the strategy builds on the efficient nonmyopic search algorithm ENS [88] in the sequential setting where only one query is made at each iteration. ENS itself approximates the optimal sequential strategy by assuming that all future queries after the current iteration are made at the same time. Jiang et al. [88] demonstrated that ENS actively explores the search space when the remaining budget is large, recommends increasingly promising molecules as the search progresses, and achieves significant improvements in performance over greedy strategies. Our sequential simulation active search algorithm under the batch setting builds its recommendations by iteratively adding elements to an initially empty set using the ENS algorithm until the desired size ($b = 50$) is reached. As a new element is added, we assume that this element will return a negative label (i.e., the element is assumed to lack the desired property). Jiang et al. [89] demonstrated that by taking on this pessimistic view, the algorithm encourages the elements within the same batch to be diverse, which helps explore the search space more effectively. During batch construction, as each point is assumed to be negative, the probabilities of the neighbor points are lowered, effectively causing future points in the same batch to be "pushed away" from the current one. Please see Section 5.2 in Jiang et al. [89] for more discussion and theoretical motivation for this interpretation (this policy also greedily maximizes the probability that *at least one* batch member is positive). The authors further showed that the algorithm significantly outperformed popular baselines in the machine learning literature such as the greedy and the upper confidence bound (UCB) policy.

Finally, we aim to distribute our queries equally across the 29 cores. Our sequential simulation strategy may be naturally modified in service of this goal as follows. As a new element is added to the running batch in the iterative procedure described above, we temporarily remove other candidates having the same core ID as the newest batch member from the search space. When no candidate remains, we add all removed molecules back to our search space. This simple procedure effectively forces each batch of queries to be constructed to span the available cores equally.

7.1.2 Results and Discussion

We had a dataset of 1 436 azoarenes that we had previously computed ($\lambda_{\max}$) using the method described in Figure 2. These azoarenes were generated with a different substitution

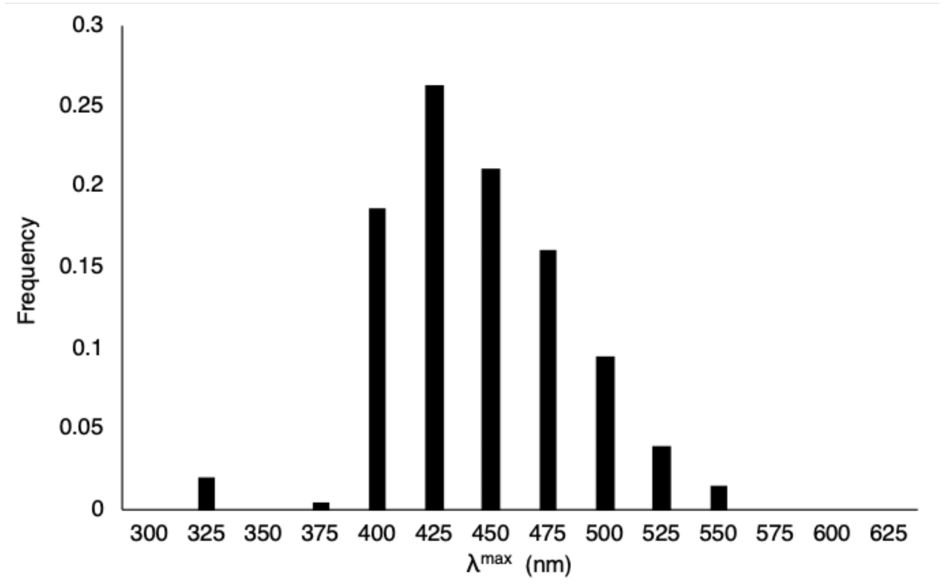


Figure 7.3: Distribution of the λ_{max} values of the photoswitch training set.

policy than previously described and a detailed description can be found in the supporting information. 981 out of 1 436 had vertical excitation energies that corresponded to λ_{max} greater than 450nm. We initially ran two iterations of active search with 100 molecules each. These two AS iterations selected molecules from the chemical space of 255 991 molecules. However, we realized that the algorithm would exploit a single core structure; and the time required to perform 100 calculations for each iteration was expensive. We then decided to create a core restriction policy where the algorithm would equally sample all cores. We re-trained the algorithm with the 198 molecules previously selected. Of the 198, 20 molecules absorbed greater than 450 nm. We also decreased the batch size to 50 molecules per batch to decrease the turnaround time for the quantum chemical calculations. A histogram of the λ_{max} of these 198 azoarenes is shown in Figure 7.3.

Figure 7.3 shows that the λ_{max} ranges from 301 to 541 nm for the selected 198 azoarenes. To train the AS algorithm, we assigned each candidate a label of `True` or `False`, depending on whether the following expression is satisfied, λ_{max} greater than 450 nm. 62 of the 198 azoarenes were assigned `True` and 136 were assigned `False`.

We then iteratively applied the algorithm 40 times on our new molecular dataset. Each molecular batch featured 50 AS-suggested candidates that would enter our computational workflow. The first 20 iterations used an "equidistributed" policy, which equally sampled molecules belonging to each core family of the 29. Since the AS selected 50 molecules for each iteration, we sampled the 29 cores by constraining the algorithm to select at least one molecule per core. The remaining 21 slots for each batch were selected in a similar fashion where no more than two molecules were selected for each core. The remaining iterations (21–40) used a "targeted" policy that only selected molecules from a subset of 15 cores that had derivatives where the $\lambda_{\max}$ greater than 450 nm. Cores that did not show derivatives that fit the criteria were excluded from the subset. After each iteration, we added a binary label to each molecule based on whether $\lambda_{\max}$ greater than 450 nm. Figure 3 summarizes this iterative procedure. We compared the AS strategy to the performance of a random search strategy by sampling three molecules selected at random from each of the 29 cores. Figure 7.4 shows the distribution of the $\lambda_{\max}$ values from AS and the random search.

Figure 7.4 describes the effect of applying AS. The random search showed that 11 out of the 87 molecules (13%) had $\lambda_{\max}$ greater than 450 nm. The active search increases the number of molecules that are selected with $\lambda_{\max}$ greater than 450 nm. The overall increase in average can be seen for bins (501, 551] and (551, 602] where the random search was unable to select any molecules in the latter bin, respectively. Figure 7.5 shows how the proportion of hits changes with respect to the first 20 iterations using the equidistributed policy. We define the hit rate as the percentage of molecules with a $\lambda_{\max}$ greater than 450 nm from the current batch.

The dotted orange line indicates a random search hit rate of 13%. The black data points indicate the hit rate as the active search is iteratively applied. The equidistributed search shows a range of hit rates from [12% to 35% (batch 3 and 18, respectively)]. The slope is +0.82; the hit rate is improved relative to the random search in nearly all iterations. We then turned our attention to the targeted AS policy to maximize the number of hits corresponding to the subset of cores with molecules that had a $\lambda_{\max}$ greater than 450 nm, shown in Figure 7.6.

For iterations 21–40, the AS algorithm selected three derivatives corresponding to each of the 15 cores for a total of 45 selected molecules. To keep the batch size consistent to 50, AS chooses five more from the top-ranked derivatives of the 15 core subset. Figure 7.7 shows the hit rate for iterations 21–40 with the targeted policy.

In the targeted policy, the hit rate varied from 44% to 56%; the average hit rate was 49%. Unlike the equidistributed policy, Figure 7.7 does not show an increase in hit rate as a

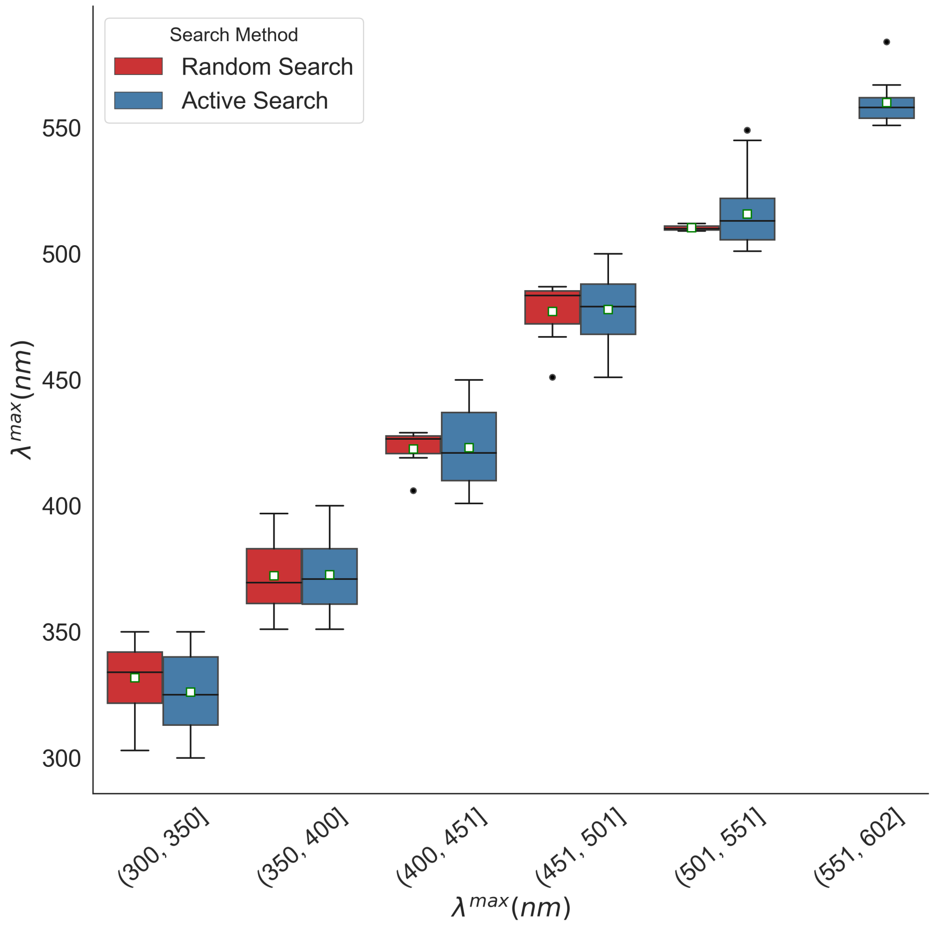


Figure 7.4: Box plot of the random search compared to active search. For the random search, molecules are sampled for each core, resulting in a total of 87 molecules. Active search calculations entail 1 962 computed azoarenes. The bin size is 50 nm. The median is denoted by the horizontal line and the average for each bin is denoted by the white squares. Outliers are shown as black circles.

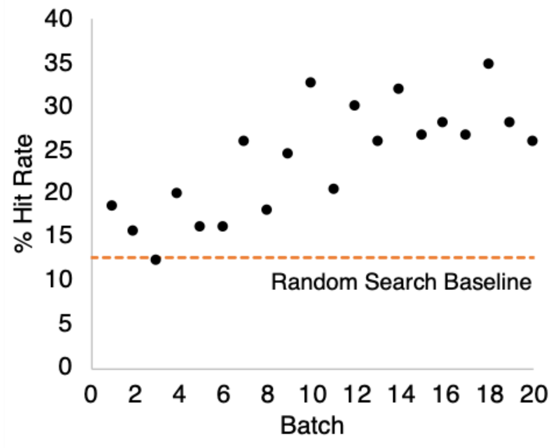


Figure 7.5: The hit rate of the first 20 iterations of the search with the reset policy. The orange dotted line indicates the hit rate for the random search of 87 molecules which was 13%. A linear regression gave the following equation describing the correlation between the hit rate and batch number, [%HR=0.82(batch) + 15.26] with an R^2 of 0.57.

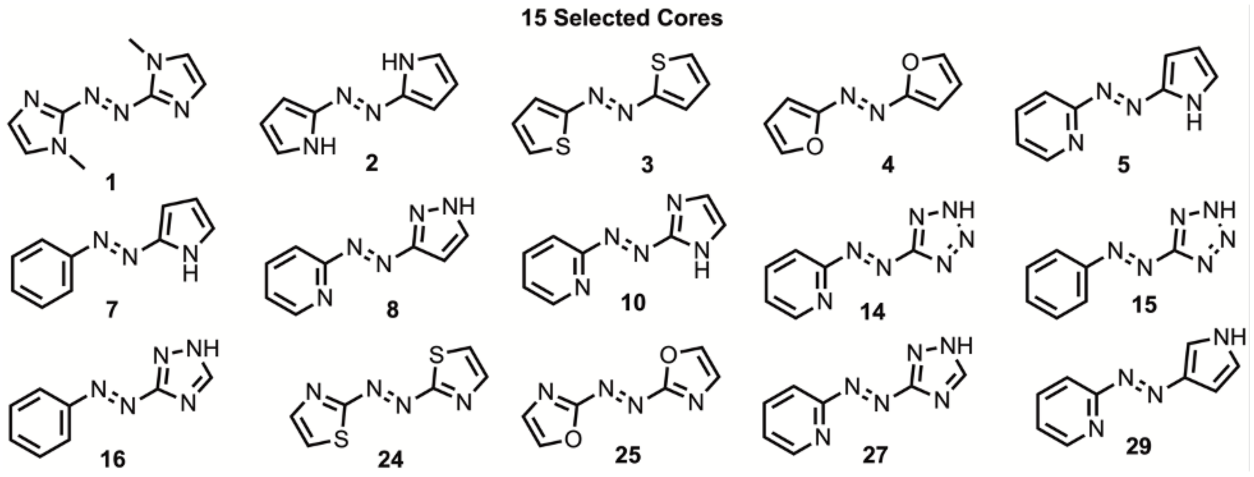


Figure 7.6: A subset of cores searched for the second half of iterations from 21–40. Cores represented yielded at least one substituted molecule that had a λ_{max} exceeding 450 nm.

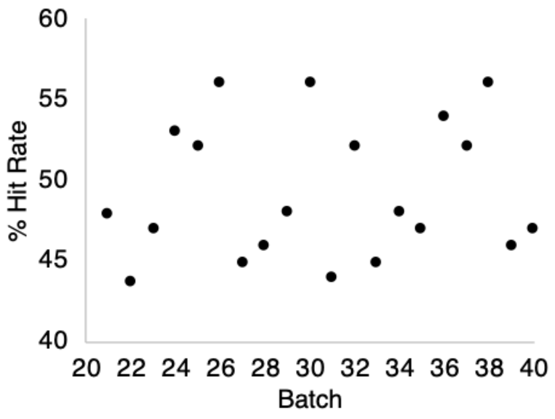


Figure 7.7: The hit rate of the second 20 iterations of the search policy with 15 cores.

function of the batch number. The relatively high hit rate led to the rapid discovery of 485 candidates with λ_{max} greater than 450 nm in batches 21–40.

Overall, we identified a total of 717 photoswitches with λ_{max} greater than 450 nm after the 40 batches (1 962 molecules) of AS-assisted virtual screening. The resulting hit rate is 37%, corresponding to a tripling of the 13% hit rate from the random search. A two-sample z–test rejects the null hypothesis that the two strategies result in equal hit rates with overwhelming confidence, yielding a p–value of 5×10^{-6}.

7.1.3 Conclusion

We created a molecular dataset of 255 991 azoarenes to find photoswitches with high λ_{max} values and high activation energies for therapeutic applications. We leveraged quantum mechanical calculations to sample just 1% of the search space and computing 2 117 DFT calculations in total over 40 iterations. The iterative process of applying AS to photoswitch screening was highly effective and tripled the discovery rate of novel photoswitches compared to a random search. The AS algorithm identified 717 photoswitches with high λ_{max} values ranging from 451 nm to 602 nm.

7.2 Guided Data Discovery in Interactive Visualizations via Active Search

Recent advances in visual analytics have enabled us to learn from user interactions and uncover analytic goals. These innovations set the foundation for actively guiding users during data exploration. Providing such guidance will become more critical as datasets grow in size and complexity, precluding exhaustive investigation. Meanwhile, the machine learning community also struggles with datasets growing in size and complexity, precluding exhaustive labeling. Active learning is a broad family of algorithms developed for actively guiding models during training, with active search is a particular paradigm aimed at discovery. We consider the intersection of these analogous research thrusts. First, we discuss the nuances, and present results of a user study for the particular task of data discovery guided by an active search algorithm.

To investigate the feasibility and impact of this human–computer partnership on visual data foraging, we created a prototype system and designed a simple data foraging task. We chose a data set published in the Visual Analytics Science and Technology (VAST) community to represent a scenario in which an epidemic breaks out in the fictional city of Vastapolis and authorities are searching through social media posts to identify impacted individuals and parts of the city. Next, we performed a series of simulations to confirm our model assumptions are appropriate for the VAST data set and the active search algorithm is indeed capable of generating plausible queries.

Finally, using our prototype system, we conducted two crowd-sourced user studies to investigate the impact of the active search algorithm in assisting users during visual data exploration and discovery. We randomly assigned participants to one of two groups: (1) an *active search* group that performed an information foraging task with visual cues powered by the active search algorithm and (2) a *control* group who performed the same task without assistance. Our quantitative analysis of the user study indicates that users assisted by the active search algorithm make more relevant discoveries while interacting with fewer irrelevant data points. However, we found that a non-trivial percentage of the *active search* group ignored the recommendations, and an analysis of the subjective responses revealed these same participants reported a low level of perceived "trust" in the system.

7.2.1 Assisted Visual Data Discovery via Active Search

Human–computer collaboration refers to the process of two or more agents working towards a shared goal where at least one agent is a human and at least one agent is a computer [188]. Using this conceptual framework, we formulate assisted data foraging as a human–computer collaboration in which the shared goal is to discover relevant data points by interacting with a visualization. Specifically, we consider an interactive visual metaphor of a data set, where each data point is represented by an element on the visual metaphor. The objective of the user is to search through this data set via the visualization to discover data points deemed valuable for a given task. With the goal of accelerating visual exploration and discovery, we augment interactive visualizations with active search. Starting with a data set, we create an interactive visualization with which the analyst interacts in order to inspect individual data points. As the user sequentially inspects and discovers relevant data points, we create a cycle where user interactions with the interface train a classifier on the relevance of unobserved data points and the active search algorithm picks a set of promising points to present to the analyst for further investigation. Details of this workflow are shown in Figure 7.8.

In the remainder of this section, we first formalize data foraging as a sequential decision-making process (Section 7.2.1). Then, we present two major interactive components of this technique: observing user interactions with the visualization to learn relevance of points (Section 7.2.1), and presenting active search queries to the user through the visualization (Section 7.2.1).

Problem Formulation

We assume there is a dot-based visual metaphor for a given data set, $\mathcal{X} = \{x_1, x_2, ..., x_n\}$, where each data point in $\mathcal{X}$ has a representative on the visualization. We further assume that each data point is classified as either *relevant* or *irrelevant,* and the objective is to recover as many relevant points as possible without getting distracted by irrelevant points. As users begin providing labels by interacting with the visualization, we maintain a set of observations, $\mathcal{D} = \{(x_1, y_1), ..., (x_m, y_m)\}$, where $y_i \in \{0, 1\}$ denotes the binary classification for a point x_i. A label of $y_i = 1$ indicates the point x_i is *relevant* to the task at hand, whereas $y_i = 0$ indicates the point x_i is *irrelevant.* Note that in an active search setting, a very small portion of the data set is typically labeled (i.e., $m \ll n$). The objective is to recover as many relevant points as possible, defined by the utility function u where:

$$u(\mathcal{D}) \triangleq \sum\nolimits_{y_i \in \mathcal{D}} y_i,$$

which simply is the number of relevant points discovered.

In an active search procedure, the algorithm relies on a model to predict the relevance of unobserved data points in light of observations. This model is used by a querying policy that, given the current user interactions, suggests unlabeled points to the user for further investigation with the goal of maximizing the total number of discoveries at the end of the search process (denoted by u above). As suggested by Garnett et al. [61], we pick a simple k-NN model that provides the posterior probability of an unlabeled point x being relevant given the observed data: $\Pr(y = 1 \mid x, \mathcal{D})$. This choice of model is non-parametric, fast to update in light of new observations, and is simple in that it only relies on a distance metric between data points. Some examples of distance functions for various structured and unstructured data types include the Euclidean distance for numerical values, Word Mover's Distance for text documents [109], and ImageNet for images [46].

In scenarios where datasets contain multiple (say d) attributes, practitioners may build a k-NN model on each attribute, $\{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_d\}$, and merge the predictions via the following weighted sum where the values of $q_i \in [0, 1]$ are tuned to maximize the likelihood of observed interactions:

$$\Pr(y = 1 \mid x, \mathcal{D}) = \sum_{i=1}^{d} q_i \Pr(y = 1 \mid x, \mathcal{D}, \mathcal{M}_i).$$

Once the model over data relevance and an active search algorithm are in place, the next primary consideration is the communication between humans and the active search procedure. In this workflow, we consider a bidirectional communication channel in which the active search algorithm needs a proxy to receive feedback from user interactions and humans need to be presented with active search queries through user friendly means.

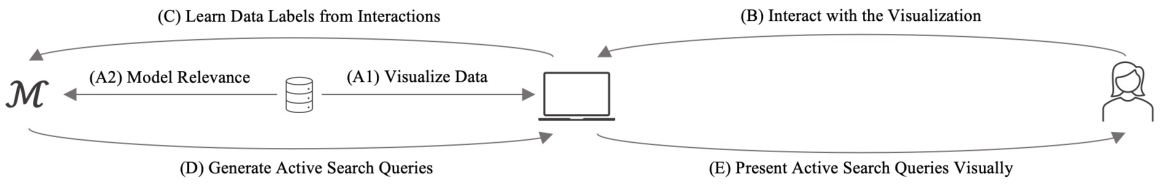

(C) Learn Data Labels from Interactions
(B) Interact with the Visualization
$\mathcal{M}$
(A2) Model Relevance
(A1) Visualize Data
(D) Generate Active Search Queries
(E) Present Active Search Queries Visually

Figure 7.8: The workflow of assisted data foraging. The first step is to visualize a data set and create a model over the relevance of data points to one another (A1, A2). The workflow cycle begins with the user interacting with the visualization (B), the interactions translating into data labels and updating the relevance model (C), an active search algorithm using the model to generate queries for inspection (D), and the queries appearing on the visualization for user inspection (E).

Learning Data Labels Interactively

In this section, we consider how human interactions with an interface can train the underlying models about the relevance of data points. Visual analytics researchers have analyzed low-level interactions to uncover information about users and the task at hand. In particular, they have discovered that analyzing low-level interactions can result in better performance at inferring user expertise [26], inferring exploration patterns [55, 138], and modeling the cognitive sense-making process [157]. These successful attempts at analyzing interactions naturally bring us to the following question: can user interactions with a system provide an active search algorithm with a seamless, yet robust, labeling mechanism? In the simple case which we examine in Section 7.2.2, certain low-level interactions can directly map into certain training labels for active search. For example, clicking a button to bookmark a data point (or disregard one) can signal a positive (or negative) label. In more complex settings, however, a more ambiguous set of interactions may be used in order to provide labels seamlessly. For example, examining the frequency of hovers on a certain data point and the length of the hover may be a more robust approach to uncovering labels from interactions.

Querying the User Seamlessly

Similar to how well-designed mechanism are needed to translate user interactions into robust labels for active search, we need a mechanism to communicate active search queries to the user effectively. In the most intrusive case, the system would explicitly query the user to provide labels for a given set of points. However, this may cause frustration for the user and undermine the role of humans in leading data exploration. Alternatively, we envision active search queries as presented to the user in the form of visual cues such as color, opacity, and size. In this work, we assume the user leads the analysis and take a non-intrusive approach where active search queries are presented in a distinct color on the visualization. Depending on the visual channels available for a specific application, the risk associated with missing a relevant data point, and the intended degree of human involvement in analysis, practitioners may design other methods of interactive queries.

7.2.2 Proof of Concept Prototype

In this section, we present our user study prototype of an visual analytic system for data foraging. Using the VAST Challenge 2011 epidemic data set, we create an interactive map visualization of microblogs based on their posting location and ask participants to discover posts by sick individuals. We discuss the details of the data set in Section 7.2.2, the details of the k-NN model over the data set in Section 7.2.2, and the details of our visualization interface in Section 7.2.2.

Data Set

The Visual Analytics Science and Technology (VAST) community published a fictitious epidemic data set for their annual challenge in 2011. The story involves a terrorist attack in the fictitious city of Vastopolis, where a truck accident over a major river contaminates the water and air with harmful chemicals. The water flow and wind transport these chemicals to two distinct parts of the city, causing citizens to exhibit symptoms of an illness. Those who live downstream of the river show waterborne digestive symptoms, whereas those who live downwind of the accident show respiratory symptoms. The data set contains 1,023,077 microblogs posted on social media from various parts of town during a 21-day period (04/30/2011-05/20/2011). The fictitious attack occurred on 05/17/2011 and the outbreaks appeared during 05/18/2011-05/20/2011.

Labeling Heuristic

For evaluation purposes, we need ground-truth labels for this data set. Since the relevance of each data point to the spread of epidemic is not included in the data set, we rely on a heuristic for labeling. Specifically, microblogs containing a set of pertinent keywords such as "sore throat," "diarrhea," and "pneumonia" are labeled *relevant,* and the remaining microblogs are labeled *irrelevant*. Refer to the supplemental material for the full list of keywords. The daily incidence rate (proportion of *relevant* points per day) according to our heuristic is ∼2% for 04/30/2011-05/17/2011 and it increases to 26-38% for 05/18/2011-05/20/2011 after the terrorist attack. We acknowledge that relying on this heuristic for labeling introduces false positives, but inspection of the data set before the epidemic (i.e. the ∼%2 incidence rate on 04/30/2011-05/17/2011) suggests the false-positive rate during the epidemic is less than 2%.

Data Selection

To demonstrate active search on datasets with varying rates of relevant points, we choose three subsets of the data set: (1) the first two days of the epidemic (05/18/2011 and 05/19/2011) with ∼33% of points being related to illness (*high-incidence*), (2) a random sample of the data set with ∼9% of points being related to illness (*medium-incidence*), and (3) the first two available days in the data set (04/30/2011-05/01/2011) with ∼3% of points being labeled positive by the heuristic (*low-incidence*). In the upcoming sections, we use the low- and high-incidence datasets in our simulations to validate our models and demonstrate active search's exceptional ability to identify rare points of interest. Furthermore, we use the high-incidence data set as a relatively easy task in our crowd-sourced user study to demonstrate active search's ability to identify relevant data points in an interactive environment (Section 7.2.3).

Probabilistic Classifier over Data Set

As mentioned in Section 7.2.1, performing active search relies on a probabilistic model that computes the probability of an unlabeled point being relevant given the observed data. The k-NN classification model is non-parametric and only relies on a distance definition among data points. This flexibility in the choice of distance function offers practitioners from various fields the option to tailor this model to their domain-specific datasets.

For this data set specifically, we build two k-NN models over the data. The first one ($\mathcal{M}_L$) is based on the posting *location* of microblogs, where the distance between two data points is the Euclidean distance between locations from which they were posted. The second one ($\mathcal{M}_T$) is based on the microblog *texts*, where the distance between two data points is the cosine distance between the vector representation of their texts. We define the vector representation of a microblog to be the normalized average over *word2vec* representation of its individual tokens (after removing numerical values, punctuation, and stop words) trained on a large set of news articles [164]. Given some observed data, each of these two models, $\mathcal{M}_i$, calculates the probability that an unlabeled data point x is relevant: $\Pr(y = 1 \mid x, \mathcal{D}, \mathcal{M}_i)$. To combine these two predictions, we use a parameter $q \in [0, 1]$ as the weight of the text-based prediction (the location-based prediction thus has a weight of $1 - q$):

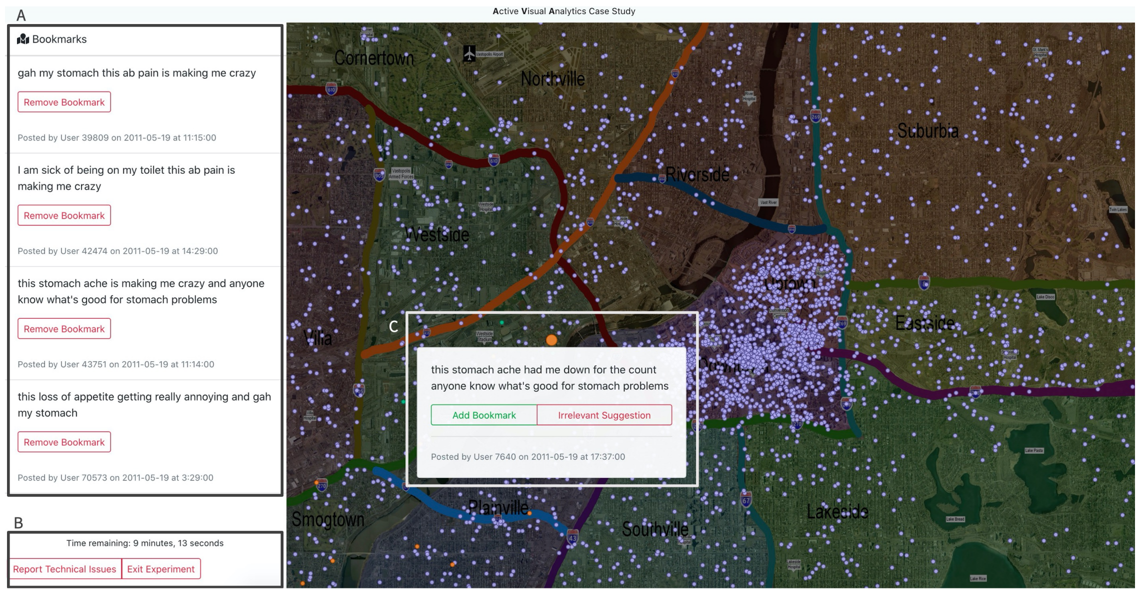
$$\begin{aligned}\Pr(y = 1 \mid x, \mathcal{D}, \mathcal{M}_T, \mathcal{M}_L) = &\quad q \ \Pr(y = 1 \mid x, \mathcal{D}, \mathcal{M}_T) \\ &+ (1-q) \Pr(y = 1 \mid x, \mathcal{D}, \mathcal{M}_L),\end{aligned}$$

where q is chosen using the maximum-likelihood estimation method to maximize the likelihood of the observed data $\mathcal{D}$.

Interactive Interface

The interface of our prototype is shown in Figure 7.9. We aimed for a simple interface and clear means of interaction for greater usability. There are two primary components on our interface: an interactive map visualization of microblogs based on their posting location (details to follow), and a side bar containing a list of current bookmarks, time remaining for task completion, and control buttons to report technical issues or leave the experiment. Although a text-based visualization (e.g. t-SNE may seem more appropriate for the given task, we assume the primary attribute determining relevance (i.e. microblog text in our case) is not known a priori. We provide more details on the visualized data set as well as the means of displaying active search recommendations in the following two sections.

Users hovered on data points to see a tooltip containing the microblog (Figure 7.9, C). The tooltip allowed user feedback in one of three ways: (1) if the hovered data point was



gah my stomach this ab pain is making me crazy

Posted by User 39809 on 2011-05-19 at 11:15:00

I am sick of being on my toilet this ab pain is making me crazy

Posted by User 42474 on 2011-05-19 at 14:29:00

this stomach ache is making me crazy and anyone know what's good for stomach problems

Posted by User 43751 on 2011-05-19 at 11:14:00

this loss of appetite getting really annoying and gah my stomach

Posted by User 70573 on 2011-05-19 at 3:29:00

Time remaining: 9 minutes, 13 seconds

this stomach ache had me down for the count anyone know what's good for stomach problems

Posted by User 7640 on 2011-05-19 at 17:37:00

Figure 7.9: A view of the prototype system on the epidemic data set. **Green** dots on the map indicate *relevant* data points bookmarked by the user (also shown in the left panel, A). **Orange** dots indicate active search recommendations of *potentially relevant* data points (C). **Violet** dots indicate the remaining data points. Hovering on data points triggers a tooltip containing the microblog and feedback options (C). A countdown of the remaining time was shown, and users had the option to exit the experiment at any time or report technical issues (B).

suggested by the active search algorithm, the user could either *add bookmark* or report an *irrelevant suggestion*; (2) if the hovered data point was already bookmarked, the user could *remove bookmark*; (3) if the hovered data point was not already bookmarked nor suggested by active search, the user could only *add bookmark*. We utilized three distinct colorblind-safe colors to distinguish between suggested dots, discovered dots, and the remaining dots. To make potential feedback modifications easier, we displayed a list of bookmarks on the sidebar along with an option to *remove bookmark* (Figure 7.9, A).

7.2.3 Data Discovery Throughput Experiment

We designed a crowd-sourced user study[10] to investigate the impact of active search on visual exploration and data foraging. We adopted the 2011 VAST challenge, which describes an epidemic in the fictional city of Vastopolis. The VAST challenge has a long history of

[10]This experiment was pre-registered on Open Science Foundation.

providing complex, realistic analytic tasks and datasets for visual analytics research [40, 176]. We used the mini-challenge which included twitter-like messages and their location data to provide information about the spread of various disease symptoms.

Task

Participants were told that health professionals had reported a spike in reported illnesses with flu-like symptoms, including fever, chills, sweats, nausea and vomiting, diarrhea, and death. We informed participants that the authorities are interested in identifying the impacted parts of the city by analyzing social media activity, and that we have access to social media posts and their posting locations. Their task was to assist the authorities by searching through a data set of microblogs via an interactive map and bookmarking as many posts containing illness-related information as possible. The data set was presented with reference to a satellite image of the city, highlighting major landmarks, regions, and water bodies, as well as tweet-like messages spanning one month of activity.

Participants

We recruited 130 participants via Amazon's Mechanical Turk platform. Participants were 18 to 65 years old, from the United States, and fluent in English. Each participant had a HIT approval rating of greater than 98% with more than 100 approved HITs. After data cleaning steps outlined in 7.2.3, there were 46 women, 76 men, and 1 participant with undisclosed sex in our subject pool with ages ranging from 18 to 62 years ($\mu = 36$, $\sigma = 9$). About 72% of our participants self-reported to have at least an associate degree. The average completion time (including reading the tutorial, performing the task, and completing the survey) was 12 minutes. The instructions specified that participants will be compensated \$1.00 base pay and an additional \$0.10 bonus for every relevant microblog they identify (with a maximum of \$4.00). Although the advertised payment structure was designed to incentivize participants to complete the task, we ultimately decided to pay everyone the maximum bonus of \$4.00 for fairness.

Procedure

The experiment complied with an approved protocol per [*redacted for anonymity*]'s IRB. Workers who accepted the HIT followed a URL to the study platform. Our system randomly assigned each participant to one of the following groups: the *active search group,* which received a batch of 10 active search queries in the form of visual clues that were updated after every bookmark, and the *control group,* which did not receive any assistance during exploration. Upon giving consent to participate in our study, participants were given a tutorial on their task and their corresponding system. Both groups initiated their task without any initial "clues," and in particular the active search group did not receive assistance for their selecting their first bookmark. Participants were given at most 10 minutes to identify as many microblogs related to the epidemic as they could using an interactive map visualizing microblogs as dots placed their posting locations. Hovering on visualized dots triggered a tooltip containing the post, and users could click on a button to bookmark the post if they judged it to contain illness-related content. Once the users were either satisfied with their search for illness-related documents or the 10 minutes were up, they were directed to a post-experiment survey to collect demographic information and general feedback on the system. In case our participants experienced technical difficulties with the system, we provided them with the option to report issues, gracefully exit the session, and receive compensation.

Data Collection

We analyze our user study data by focusing on two interactions: inspection of microblogs (*hovers)* and discovery of relevant posts (*bookmarks*). These two types of interaction inform us about the speed and accuracy of visual data foraging through the metrics listed in Table 7.1. The bookmark and hover purity metrics are the proportion of bookmarks and hovers that involved relevant data points, respectively. The bookmarks- and hovers-per-minute metrics inform us about the speed at which users interacted with data points. The relevant hovers and relevant bookmarks-per-minute metrics are the rate at which users interacted with relevant data points, quantifying both speed and accuracy of interactions. Finally, we measure the number of relevant bookmarks discovered by the end of the session and number of unique illness-related keywords contained in the discovered microblogs.

Data Cleaning and Exclusions

In a pre-processing step, we filtered the collected data to exclude participants who did not attempt the task or were unable to finish the experiment. Specifically, we eliminated participants based on the following four criteria:

1. those who failed the attention checks in the survey (eliminating 1 subject),
2. those who reported technical issues with the interface using the button provided (eliminating 1 subject),
3. those who hovered on less than 10 data points (eliminating 4 subjects) – we consider a valid hover to be one that lasts at least 500 milliseconds (300 milliseconds for triggering the tooltip, and 200 milliseconds for skimming the text), and
4. those who did not meet the age qualification (eliminating 1 subject).

A total of 123 subjects remained after filtering (74 in the *control* group and 49 in the *active search* group).

Suggestion Quality

We begin our analysis by examining the quality of suggestions provided by the active search algorithm when seeded with real users' interaction data. We use the labeling heuristic detailed in Section 7.2.2 to assign a label of *relevant* or *irrelevant* to each microblog in the data set. These labels were hidden from the active search algorithm, which generated suggestions solely based on observed interactions. Thus, the labeling heuristic serves only as a proxy for ground truth in this analysis, allowing us to evaluate suggestion quality. We define suggestion purity to be the proportion of unique microblogs recommended to the user throughout a given session that were relevant. On average, active search group participants had a suggestion purity of 79%. We observe a moderate positive correlation between bookmark purity and suggestion purity ($R^2_{adj} = 0.594$, $p < 0.0001$), suggesting that the active search algorithm provides useful recommendations for participants who interacted with known symptoms.

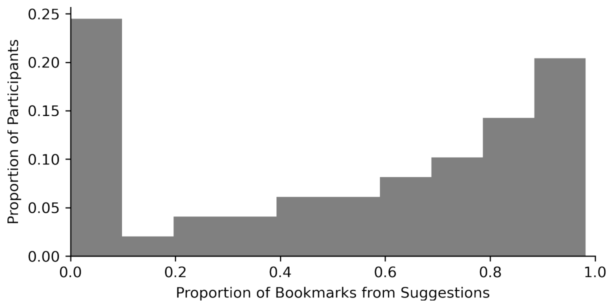


Figure 7.10: Distribution of proportion of bookmarks resulting from the active search suggestions

Suggestion Usage

Upon inspecting the sessions, we observed an unexpected pattern in the active search group. As shown in Figure 7.10, for approximately 24% of participants in the active search group, suggested microblogs accounted for less than 10% of their bookmarks. 9 out of 49 participants did not bookmark any of the suggestions presented to them at all. Further inspection reveals that the 9 active search participants who ignored the suggestions had on average $82 \pm 9\%$ suggestion purity and $76 \pm 11\%$ bookmark purity. This compares to the 40 active search participants who did interact with the suggestions, who had on average $79 \pm 5\%$ suggestion purity and $82 \pm 5\%$ bookmark purity. Finally, we observed a difference between how subjects reported their trust towards system suggestions on a 1–5 Likert scale in the post-experiment survey (3.3 ± 0.46 for those who ignored suggestions vs. 4.2 ± 0.24 for those who interacted with the suggestions).

The following analyses focus on the impact of suggestions on data exploration and information foraging. Thus, we exclude the 9 participants in the *active search* group who did not interact with the system's suggestions, leaving us with 74 participants in *control* group and 40 participants in the *active search* group.

The Effect of Suggestions on Data Foraging

We performed a series of two-sample t-tests to investigate differences in behavior in our two study conditions: *control* and *active search*. Table 7.1 summarizes our findings. We found that participants in the *active search* group bookmarked ($t(112) = 3.98, p = 0.0001; d = 0.79$) and hovered over ($t(112) = 4, p = 0.0001; d = 0.79$) significantly more relevant microblogs per

Table 7.1: The results of two-sample t-tests on the metrics discussed in 7.2.3

	95% CI				
Metric	Control $_{N=74}$	Active Search $_{N=40}$	p-value	t-statistic	Cohen's d
Hovers per Minute	16.7 ± 1.19	14.3 ± 1.23	**0.0112**	-2.58	-0.51
Relevant Hovers per Minute	6.7 ± 0.68	9.2 ± 1.12	**0.0001**	4.00	0.79
Hover Purity	0.39 ± 0.02	0.63 ± 0.05	$<$ **0.0001**	9.70	1.92
Bookmarks per Minute	6.9 ± 0.77	9.5 ± 1.41	**0.0006**	3.52	0.70
Relevant Bookmarks per Minute	5.4 ± 0.68	8.1 ± 1.26	**0.0001**	3.98	0.79
Bookmark Purity	0.77 ± 0.04	0.82 ± 0.05	0.2249	1.22	0.24
Relevant Microblogs Bookmarked	53.9 ± 6.80	73.4 ± 11.50	**0.0026**	3.09	0.61
Unique Keywords Identified	16.1 ± 0.83	15.6 ± 1.32	0.4980	-0.68	-0.13

minute than the *control* group. Furthermore, our findings show that the *active search* group performed fewer exploratory hovers per minute ($t(112) = -2.58, p = 0.0112; d = -0.51$) than the *control* group, implying that the suggestions resulted in a more efficient exploratory analysis.

For a more fine-grained analysis, we examine bookmark discoveries as a function of time. Figure 7.11 shows the average number of bookmarks over time for the *active search* and *control* groups. We can observe that the *active search* group consistently outperformed the control group by bookmarking more relevant microblogs throughout the ten-minute information foraging session. However, it is noteworthy that although the suggestions improved the quantity of the bookmarks, we found no measurable difference in the quality or content of the bookmarked discoveries. Both the *active search* and *control* groups collectively examined similar geographical regions and symptom sets (see supplementary material for analysis details).

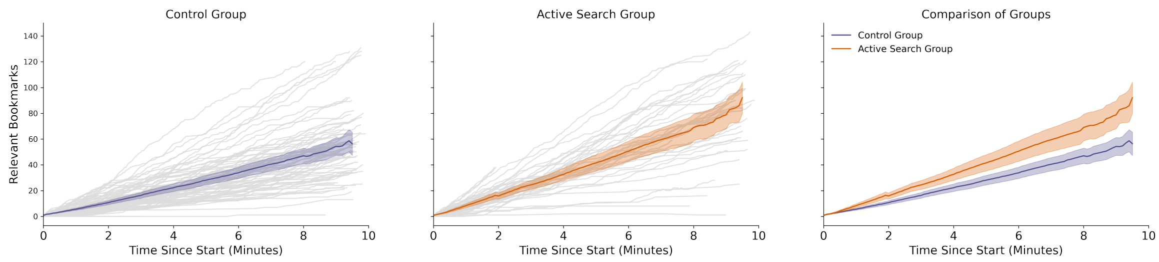


Figure 7.11: The number of relevant microblogs discovered over time for individual participants (gray) and the 95% confidence interval for each group.

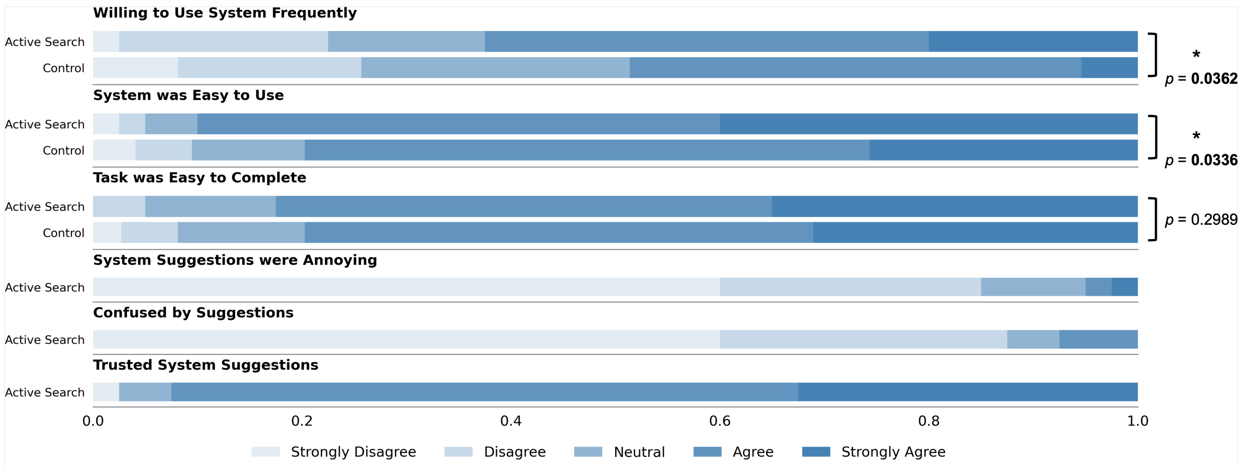


Figure 7.12: Post-experiment survey responses. The results show a slight tendency in the Active Search group to find the system more usable and the task easier. The last three questions were only applicable to the Active Search group, showing that participants did not find recommendations intrusive, confusing, and untrustworthy. (*) denotes statistically significant difference between groups according to a Mann-Whitney U test at $\alpha = 0.05$.

Impact of Active Search Suggestions on Usability

In a post-experiment survey, we asked subjects in both groups three questions on *willingness to use*, *ease of use*, and *ease of task completion*. We performed a Mann–Whitney U statistical test at $\alpha = 0.05$ to determine if there was a significant difference between the control and active search groups. The analysis showed some evidence that the active search group found the system easier to use ($U = 1199.50$, $p = 0.0336$, $r = 0.16$) and were more willing to use the system frequently ($U = 1192$, $p = 0.0362$, $r = 0.16$). However, the effect sizes were small for both. Furthermore, we did not find a significant difference between the control and active group's response to ease of task completion ($U = 1397.50$, $p = 0.2989$, $r = 0.07$). Figure 7.12 summarizes the post-experiment survey results.

Discussion

Our results indicate that a human–machine partnership could significantly improve information foraging and data discovery. For example, participants in the active search group hovered on fewer points per minute while hovering on more relevant data points per minute than the control group. These findings show that they successfully disregarded irrelevant data and were more mindful of the relevant data points. As a result, users with assistance from active search inspected a more relevant subset of data points than the control group and made more discoveries (per minute).

Overall, we demonstrate that adding active search to the information foraging workflow improves an analyst's throughput and the quality of interactions significantly. However, let's consider the potential downstream hypothesis generation and decision-making that might result from this initial data foraging and collection. Our findings also show that the data from the *active search* and *control* groups would produce similar conclusions. Both groups yielded a similar set of keywords and geographical coverage. This convergence in data exploration indicates two things. First, the baseline task was reasonably manageable without the machine's assistance. The control group hovered an average of 17 points per minute and bookmarking 7 points per minute, on average. Additionally, participants in the control group generally believed that the task was easy to complete. Second, although we selected a greedy policy for the active search policy algorithm, the data exploration convergence indicates that showing suggestions did not limit the diversity of discoveries, which is a general concern for greedy algorithms [88, 61].

From the data collected in our post-experiment survey, we observe an encouraging and consistent tendency among the active search group to find the system and task easier and being more willing to use the interface. However, it is noteworthy that a non-trivial percentage of the active search group ignored the suggestions entirely, and those participants reported lower levels of perceived trust in the system. Trust is a complex and multifaceted construct, but it is essential for human–computer partnership [173]. In the context of visual analytics, trust can be defined as "the truster (user)'s belief that the trustee (VA system) will help them correctly identify and visually distill the most valuable and relevant information content" [76]. Our findings highlight a vital factor for nurturing the human–computer partnership so that analysts can accelerate the process of information foraging. The analysts need to be able to trust the suggested microblogs provided by the active search algorithm. Although our study provides only a coarse Likert scale for assessing trust, the combination of low trust ratings and low interaction provides suggestive evidence that the system's method of providing suggestions (visual cues with no explanations) was sub-optimal for building trust with the human, leading to distrust, and ultimately, low engagement. There is existing work on how to elicit trust through design, e.g., showing explanations [51] and being transparent about uncertainty [43, 173]. This raises the question of how explainable suggestions from active search might impact the interaction behavior of participants.

7.3 Probabilistic Prediction of Material Stability: Integrating Convex Hulls into Active Learning

Understanding thermodynamic stability is foundational to chemical and materials design. Phase relations provide mechanistic insight and accelerate discovery in disparate areas such as drug solubility [209, 12], polymer blend stability [166, 154, 220], and phase transitions in metallic alloys [150, 36]. To accelerate stability predictions, computational research often focuses on producing high-fidelity surrogate models [17, 49, 193, 92, 16, 15]. However, phase stability prediction remains a persistent challenge for complex systems without effective surrogate models; examples include high-entropy materials [31, 64, 152, 77, 1], liquids and glasses [221, 129, 189], materials at high temperatures [71], and highly correlated materials [42, 96, 217, 7, 135]. In this work, we address the frontiers of phase stability prediction by constructing an active learning approach that directly learns about the convex hull.

Phase transitions often occur across length- and time-scales too large to be directly observed using simulations. Instead, thermodynamic potentials need to be evaluated across a vast space of competing compositions and phases. The outcome of this competition is encapsulated in the convex hull: a single mathematical object that wraps the energy surface and defines the set of stable phase-composition pairs. Convex hulls are often associated with predicting the stability of compounds without external fields at 0 K [41, 151, 172, 86, 14, 134, 79, 153, 111], but they have also been used to calculate phase transitions induced by temperature [71], pressure [216, 114, 85], anisotropic stresses in thin films [213, 214], magnetic fields [110, 70], and applied voltages in battery materials [194, 195]. Indeed, the convex hull formalism can be used to predict stability under any set of thermodynamic conjugate variables [4, 185]. Beyond phase diagrams, convex hulls have been recently leveraged in understanding chemical reaction networks and synthesis pathways [207, 132, 168, 35].

The global nature of convex hulls implies that it is not obvious which composition-phase pairs will reside on the hull. For instance, it is possible for the exact value of the energy to be certain, while still being uncertain that the composition is on the hull. A brute force

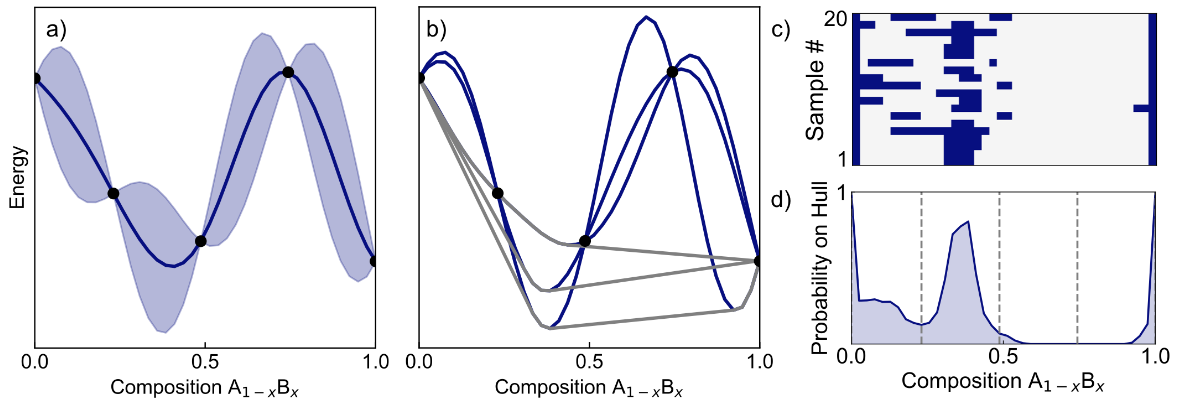


Figure 7.13: (a) For a single phase, the search procedure begins by modeling the energy surface with a Gaussian process. The black points denote observed compositions, the blue curve represents the mean of the Gaussian process posterior, and the blue shaded region corresponds to two standard deviations from the mean. (b) Sampling from the Gaussian process posterior allows an ensemble of energy surfaces to be hypothesized. The convex hull (grey) is constructed for each energy surface; single-phase regions are where the energy surface touches the hull. (c) Each convex hull can be reduced to a composition vector with a binary classification of phase stability. Here, each row of the matrix corresponds to a separate sampled hull; blue denotes single phase compositions. (d) Interrogating this ensemble of hulls yields the probability of being on the hull. We note that observing the energy of compositions (dashed lines) does not necessarily give absolute information about their stability.

approach to predicting the convex hull would require calculating the energy for all competing phases and compositions. However, when the cost of individual energy evaluations is large, or the space of possible competing compositions is high-dimensional, exhaustively evaluating the energies is prohibitively expensive. Thus, there are two complimentary modes of acceleration: efficiently producing surrogate models that lower the cost of energy calculations and minimizing the number of energy evaluations necessary to define the convex hull. Both approaches can leverage active learning [179], since it is a natural method for selecting expensive data points that are expected to maximally increase the information about a function.

To optimize the information gain about a surrogate energy function, active learning has been used to iteratively select first-principles calculations that minimize uncertainty in the surrogate model. Surrogate models like cluster expansion [34] and interatomic potentials

[74] have been trained with active learning; they were then leveraged to conduct numerous energy evaluations for predicting the underlying convex hull. Active learning has also been biased to identify phase-composition pairs that are expected to be on or near the convex hull [177, 108, 197]. While these approaches have been shown to be more efficient than random and grid-based search procedures, the active learning was only biased using proxies that incorporate a local view of the hull rather than directly reasoning about the entire convex hull as a singular, global object.

We develop convex hull-aware active learning (CAL) to accelerate stability predictions. CAL distinguishes itself from more conventional Bayesian approaches by reasoning directly about the entire convex hull. CAL uses separate Gaussian process regressions to model the energy surfaces of phases across the composition space. From the Gaussian processes, a posterior belief is produced over possible convex hulls. This induced posterior enables the algorithm to identify composition-phase pairs that are expected to minimize the uncertainty in the convex hull itself, not the constituent energy surfaces. By focusing exclusively on the convex hull, it is possible to make more effective decisions on what compositions to consider.

7.3.1 Approach

The overall goal is to establish a methodology that approximates the convex hull with minimal observed data. We begin by establishing a probabilistic view of the hull (Fig. 7.13) and then present the policy for determining the next observation (Fig. 7.14). We provide additional details on both the model and policy in the Methods section.

Probabilistic view of the hull In this and all subsequent examples, the energy surfaces are assumed to be continuous and differentiable across alloy compositions. We also assume that there is a finite set of candidate compositions that represent a dense subset of the space. In our first example, we begin with a single phase for which we have observed the energies of the parent compounds and three alloy compositions. These observations are denoted as $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$, with x_n taking values in composition space and y_n being energies.

We model the energy surface with a Gaussian process (GP), which provides a prior on energy surfaces specified by a mean and covariance function [163, 44]. Conditioning on the observations $\mathcal{D}$ results in a posterior distribution over energy surfaces that is itself a Gaussian process. Let $F_{\mathcal{D}}$ be the random function associated with the posterior on energy surfaces; then $H_{\mathcal{D}} = \mathcal{C}[F_{\mathcal{D}}]$ is the induced random (lower) convex hull, where $\mathcal{C}$ is the convex hull operator. The random function $H_{\mathcal{D}}$ is the object of primary interest in this work.

As we are only considering a finite set of candidate compositions, it is possible to generate samples from this induced posterior by 1) drawing a sample from the multivariate Gaussian distribution resulting from the GP posterior, and 2) using a standard algorithm such as QuickHull [13] for computing the lower convex hull of a set of points. Fig. 7.13a shows a posterior distribution over the energy surface, $F_{\mathcal{D}}$, and Fig. 7.13b depicts three posterior samples and their associated convex hulls.

Our epistemic uncertainty about the true convex hull is captured by the random function $H_{\mathcal{D}}$; the Shannon entropy $\mathbb{S}[H_{\mathcal{D}}]$ then quantifies our (lack of) knowledge about the convex hull. By framing our problem as one of minimizing $\mathbb{S}[H_{\mathcal{D}}]$, we can more rapidly gain information about the structure in which we are most interested.

In addition to the hull itself, various properties of interest can be derived from $H_{\mathcal{D}}$, so we can reason about their posterior distributions as well. For example, the (random) set

$$\mathcal{S}_{\mathcal{D}} := \{x : F_{\mathcal{D}}(x) = H_{\mathcal{D}}(x)\}$$

contains the stable compositions as these are the compositions for which the minimum-energy phase is tight against the convex hull.

Fig. 7.13c shows 20 samples of stable sets after the 3 iterations in 7.13b. These binary classifications can be averaged to estimate the marginal probability that any given composition is on the hull, i.e., is stable (Figure 7.13d). Note that these marginal probabilities reveal an important way in which this problem is different from conventional Bayesian optimization and active learning tasks: the global nature of the convex hull means there is uncertainty about stability even for compositions in which the energy has been noiselessly observed. In this example, the observed compositions are marked with dashed vertical lines in Fig. 7.13d and there is uncertainty about the stability in two of the three cases.

Refining the convex hull With a probabilistic view of convex hulls in place, our goal in each iteration of the search is to identify the candidate observation x^*, which is expected to

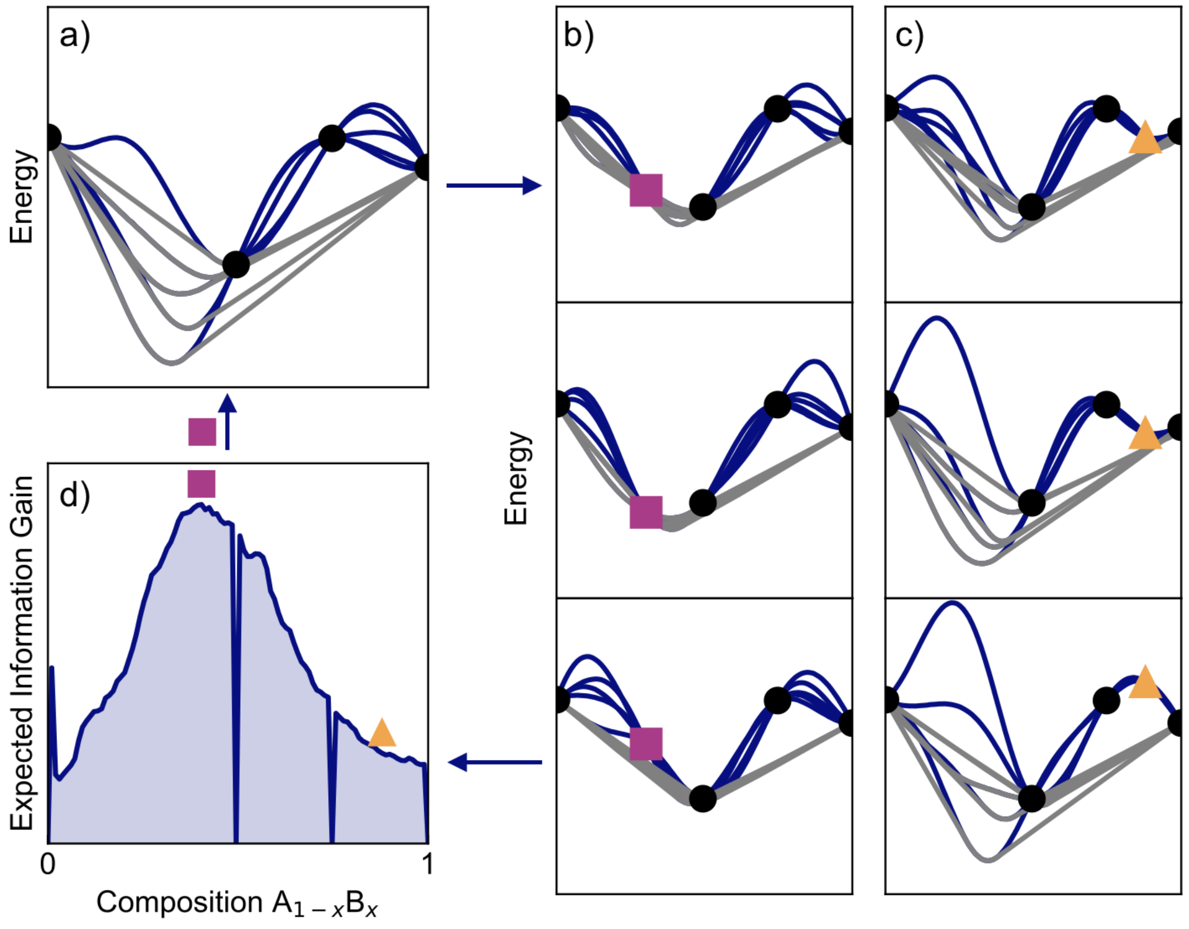


Figure 7.14: (a) Given some set of existing observations, energy surfaces are sampled from the trained GP and the corresponding hulls are calculated. (b) To determine the expected information gain for a potential observation at composition x', hypothetical energies that could result from such observations are predicted. These hypothetical energies are generated using the conditional distribution of the GP at $x = x'$. (c) For contrast, a set of potential observations for a different x composition are also highlighted. (d) This procedure is repeated to calculate the expected information gain across all compositions. The optimal composition $x \rightarrow x^*$ for subsequent observation is found by identifying the composition with the highest expected information gain. After conducting an observation at x^*, the process repeats until the uncertainty in the convex hull is sufficiently small.

minimize the Shannon entropy $\mathbb{S}[H_{\mathcal{D}}]$. This objective can be viewed as a Bayesian experimental design procedure in which the policy is to greedily maximize the information gain (Fig. 7.14).

Like many Bayesian optimization and search algorithms, the selection of x^* requires approximating the expected information gain (EIG) across the space of possible designs, which in our case is the set of compositions [126, 125]. The EIG is simply the difference between the Shannon entropy of the current state (reflected in the observed data, $\mathcal{D}$) and the expected Shannon entropy after making an observation at an unobserved composition x. Of course, the energy value y is unknown at this point and so the new set of observations $\mathcal{D} \cup (x, y)$ is considered in expectation:

$$\mathrm{EIG}(x\,;\,\mathcal{D}) := \mathbb{S}[H_{\mathcal{D}}] - \mathbb{E}_y[\mathbb{S}[H_{\mathcal{D}\cup(x,y)}]]\,. \tag{7.1}$$

Finally, the expected information gain is used within each iteration to select x^*, the candidate composition to be evaluated:

$$x^* = \arg\max_x \mathrm{EIG}(x\,;\,\mathcal{D})\,.$$

Fig. 7.14 illustrates how the EIG is evaluated in practice. In Fig. 7.14a, we start with a GP conditioned on some data, $\mathcal{D}$. Energy surfaces are sampled from the resulting posterior distribution, convex hulls are calculated, and the Shannon entropy of state $\mathcal{D}$ is calculated, giving us the first term in equation 7.1.

For a given candidate composition x, we sample from the conditional Gaussian process posterior at x to obtain a set of K possible energy values, denoted y_k. In other words, these y_k values correspond to different energies for composition x given the current uncertainty within our energy model. For each of these K samples, the entropy $\mathbb{S}[H_{\mathcal{D}\cup(x,y_k)}]$ is estimated in three steps. 1) The Gaussian process is conditioned on this "fantasized" pair of observations (x, y_k), and energy surfaces for all considered compositions are sampled from the resulting distribution. 2) For each of these sampled energy surfaces, a convex hull is computed. 3) The convex hull samples are used to estimate the Shannon entropy. The expectation value of the Shannon entropy is then calculated by averaging the K entropy estimates, thereby completing our evaluation of the EIG.

We continue to illustrate this algorithm in panels Fig. 7.14b where three hypothetical energy values for composition x lead to three different hull distributions. For contrast, a different composition is selected for Fig. 7.14c, resulting in visibly greater variation in the hulls and thus a higher expected Shannon entropy. In Fig. 7.14d, the process is repeated across composition space to determine the composition with the maximum EIG (i.e., x^*). (For panel b, the optimal value x^* was intentionally selected to visually emphasize the impact that sampling at x^* would have.) Finally, an observation is made at x^* to update $\mathcal{D}$ and the algorithm repeats to refine the convex hull. We reiterate that this approach seeks to minimize the Shannon entropy in the convex hull, not simply observe points that are on the hull. Here, observing the composition x^* is advantageous because regardless of its energy, the resulting distribution in possible convex hulls narrows significantly.

Application of the Convex Hull Having sufficiently iterated to build an accurate hull, relevant thermodynamic intensive variables can be directly calculated. For example, the elemental chemical potentials can be determined by combining the tangent and energy value of the hull. Figure 7.15a highlights that the elemental chemical potentials can be directly read off the y-intercepts of the composition boundaries (i.e., $x = 0$ and $x = 1$). Here, the energy surface is a single sample from a GP with an associated convex hull. Sweeping over the derivative of the convex hull changes the elemental chemical potentials, as shown in Figure 7.15b. All compositions within the two-phase region (shaded in blue) are in thermodynamic equilibrium, and as such, the chemical potentials stay constant. Figure 7.15c shows the mean chemical potential and affiliated uncertainty ($\pm 2\sigma$) associated with a distribution of convex hulls.

Elemental chemical potentials are critical in predicting defect concentrations, as defect creation involves exchanges with element and charge reservoirs. For example, in LiZnSb, the limited chemical potential window of Li renders the compound significantly Li-deficient even in the presence of secondary phases with excess Li (e.g. Li_3Sb) [69]. Chemical potentials of charged species can also be leveraged to produce intercalation voltage curves in battery materials [195], as was done for Li_xCoO_2 [194]. Lastly, pressure is an intensive variable that can be determined from the convex hull of an energy surface that is a function volume [216, 114, 85]. For example, the impact of volumetric confinement on the freezing point of water can be readily determined from the hull [160].

Multiple phases CAL can be naturally expanded to search across multiple competing phases. In such cases, the n phases are modeled with n independent GPs. By adopting separate GPs, we make no assumptions concerning correlations between the energy surfaces

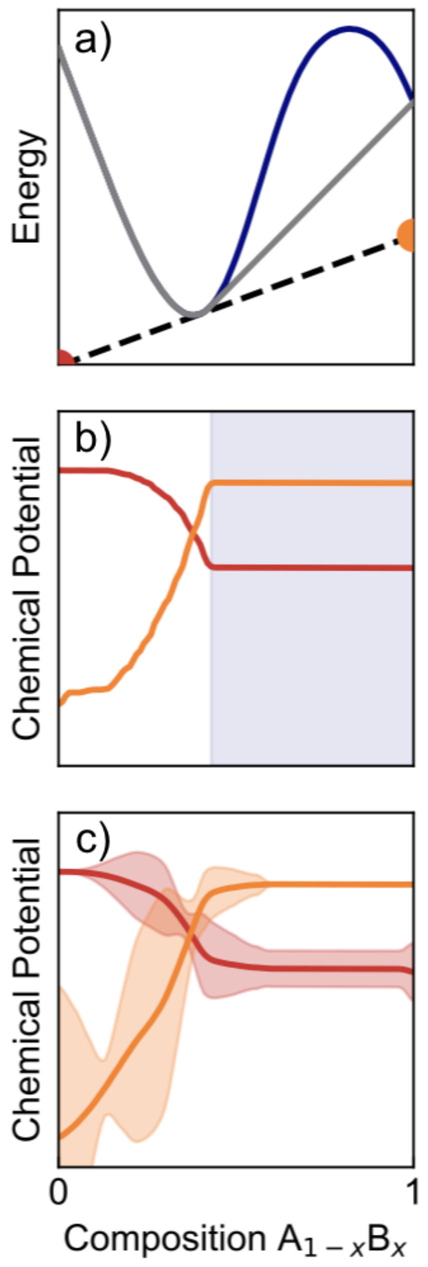


Figure 7.15: (a) Given a sampled energy surface from the GP (blue), intensive properties can be obtained from the associated hull (grey); when considering $E(x)$, the tangent (black) to the hull yields the elemental chemical potentials upon intersection with $x = 0$ and $x = 1$, denoted by the red and orange points. (b) For the single sampled hull, the chemical potentials are derived across the composition space. Within the two-phase region (shaded), the chemical potentials are constant. (c) From an ensemble of convex hull samples, the corresponding distribution in elemental chemical potentials are also represented as a distribution. The uncertainty in these potentials can be used to inform stopping criteria.

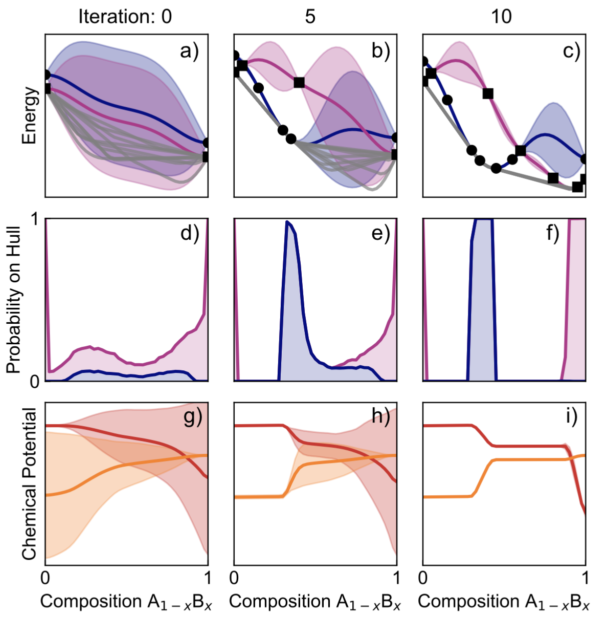


Figure 7.16: Chemical systems with multiple competing phases are represented with independent GPs; here, two phases (blue and purple) are considered in a binary space. (a) Having observed nothing but the endpoints, there is significant uncertainty across the composition space. Ten example convex hull samples are shown in grey, and they also vary widely. With (b) 5 and (c) 10 iterations, the distribution of hulls converges. (d-f) The probability that a given phase is on the hull likewise converges with observation iterations. These are stacked plots such that the total probability for being on the hull is broken up into the individual phase contributions. (g-i) The elemental chemical potentials also converge after 10 iterations (μ_A: red; μ_B: orange).

of different phases. For further efficiency, the set of n phases could be described with a joint GP, as mentioned in the Discussion. To construct the corresponding convex hull distribution, each GP is sampled s times, resulting in s^n permutations of n energy surfaces. For a given permutation, the n energy surfaces, corresponding to the n phases, can once again be wrapped with a single convex hull. From the convex hull we can predict the probability that a given phase-composition pair is on the hull, as will be shown in Fig. 7.16. The search process extends gracefully to multiple phases; the expected information gain is evaluated for each phase-composition pair.

Case Example I: 1D, 2 Phases To see this methodology applied to an iterative loop, we consider the case of a 1-dimensional binary composition space with two competing phases.

Figure 7.16a shows how the initial energy surfaces are ambiguous and this uncertainty propagates to the convex hull. The probability of any composition being on the hull is then derived from the convex hull distribution. In Fig. 7.16b and c, increasing observations leads to a tightening of the energy and convex hull distributions. However, CAL leaves significant ambiguity in the energy surfaces when they are well above the hull. The probability of a given phase being on the hull is shown across Fig. 7.16d-f; these curves quantify the evolving uncertainty in the stability predictions. A similar evolution is seen in the elemental chemical potentials (Fig. 7.16g-i).

As previously mentioned, CAL acquires observations that minimize the uncertainty in the convex hull distribution. The behavior of the algorithm can be characterized by two steps. In the first few iterations when there is large uncertainty, Figure 7.16b shows that the algorithm tends to explore the energy surface, producing a coarse estimate for the convex hull. As the estimate of the convex hull develops, the algorithm focuses its next iterations increasingly on regions that are purportedly on the hull or close to it. These subtle refinements to the convex hull distribution are reflected in Figure 7.16c, where the convex hull samples converge.

Quantitative performance assessment Hulls are intriguing objects as they involve both classification and quantitative prediction. In part, we seek to classify if a given composition is on the hull. Knowing about the energies and slopes of the hull are also important for deriving intensive variables and quantifying the energy above the hull for an unstable composition. For this reason, we use three metrics in order to assess these dual aims: mean absolute error (MAE) for the hull energy, true positive rate (TPR), and false positive (FPR). Here, TPR refers to the percentage of stable compositions that are correctly identified as being on the hull, while FPR is the percentage of unstable compositions that are incorrectly identified as being on the hull. Mathematical definitions for these metrics can be found in the Methods.

In low dimensions, producing an accurate hull can be achieved via brute force. However, the necessity for efficient hull construction emerges in spaces that involve multiple competing phases and large composition spaces. To test the efficiency of CAL in such a space, we pit it against a challenging opponent: a baseline algorithm (BASE) that still models the energy surfaces using a Gaussian process. However, BASE seeks to minimize the uncertainty in the energy surfaces and has no knowledge of convex hulls. See the Methods for further information about the BASE policy.

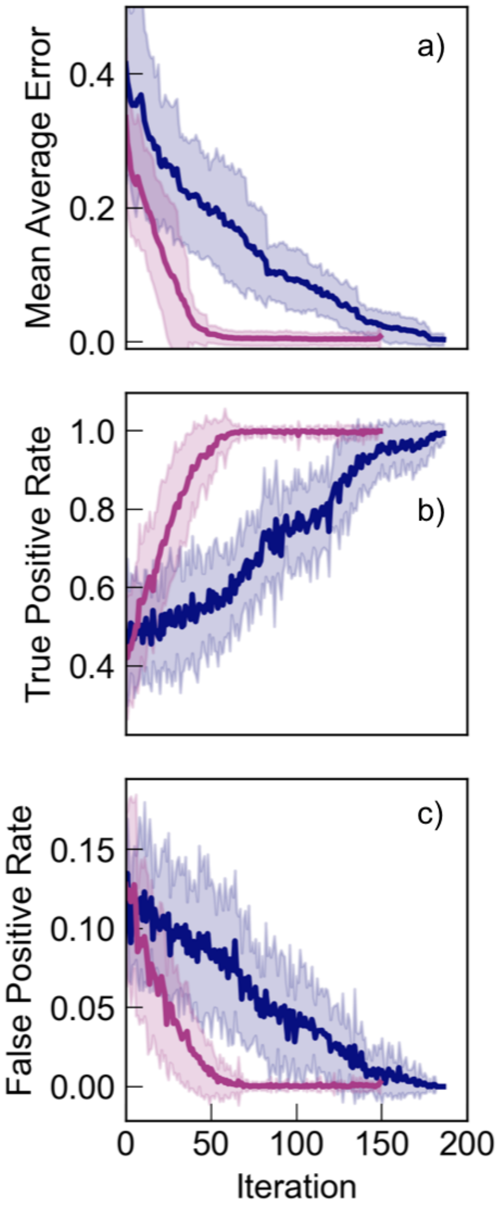


Figure 7.17: To compare the performance of CAL (pink) and BASE (blue), we consider a more complex search problem: ternary composition spaces with three competing phases. (a) Concerning the regression problem for the convex hull, we calculate the average error in the convex hull energy across the composition space. (b,c) The classification accuracy is also evaluated using the true and false positive rates. Across all metrics, CAL outperforms BASE. Here, we show the performance averaged across 40 sets of energy surfaces. The bands represent one standard deviation from the mean.

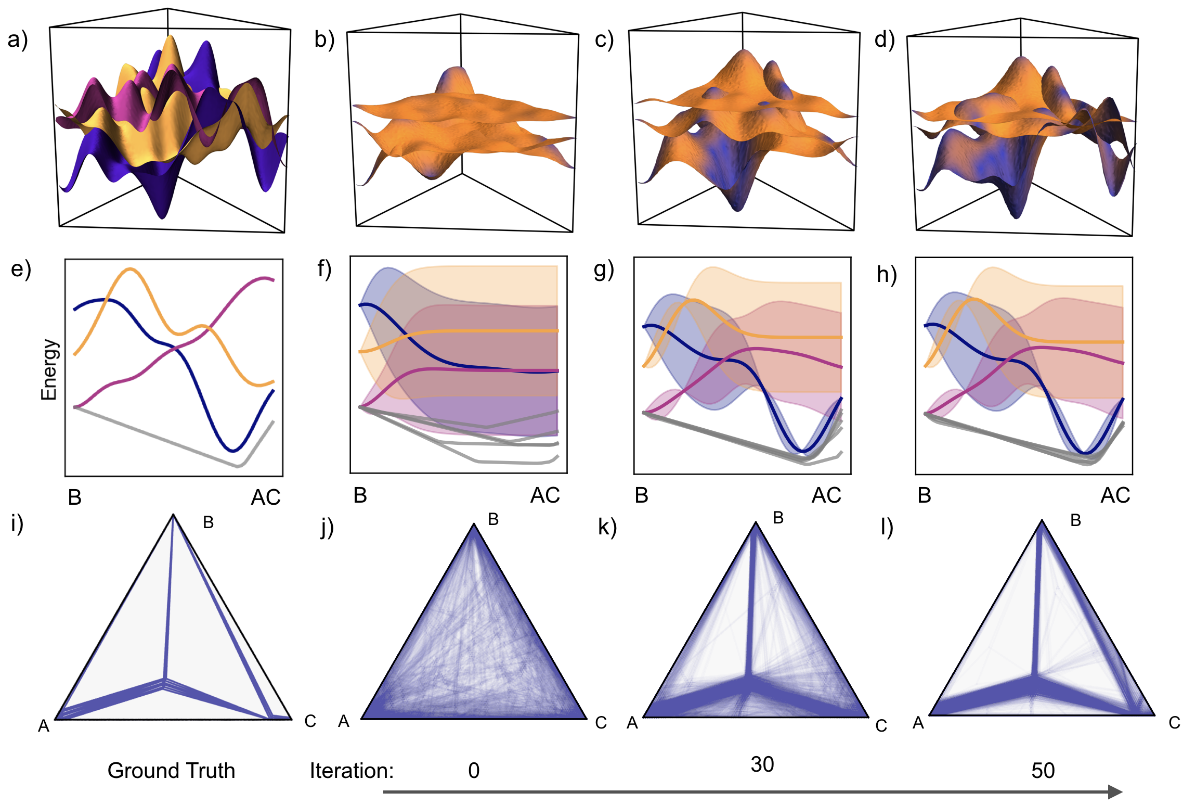


Figure 7.18: The evolution of the CAL performance is shown quantitatively in Fig 7.17; further insight can be gained by visualizing the evolution of the GP and the associated hull for a single set of energy surfaces. To investigate how CAL performs with three phases spanning a ternary composition space (continuing Fig. 7.17), a single example is considered with increasing observations. (a) Each phase has an energy surface that spans the composition space. (e) A slice of the ternary space from B to AC shows the energies of these competing phases and a corresponding slice of the convex hull. (i) The full convex hull is represented as a ternary phase diagram. (b) After 10 iterations of CAL, the three Gaussian processes are illustrated by plotting their means and coloring the surfaces with their associated uncertainties. (c,d) With increasing iteration, CAL prioritizes learning about phase-composition pairs that are relevant to the convex hull, resulting in regions transitioning from high (orange) to low (purple) uncertainty. (f-h) A similar progression can be seen in the slice from B to AC. Ultimately, we are interested in predictions of the hull and the associated phase diagram. j) After 10 iterations, the uncertainty in the convex hull distribution is represented by overlaying 100 convex hull samples on a ternary phase diagram. (k,l) With increasing iteration, the distribution tightens and converges around the true convex hull.

Case Example II: Ternary Composition Space with Three Phases Here we highlight a ternary composition space of the form $A_{1-x-y}B_xC_y$ with three different competing phases. This example is chosen to show how CAL navigates multiple dimensions and prioritizes phases that are more relevant to the convex hull. With composition steps of 0.1, the search space consists of 66 discrete compositions and 198 phase-composition pairs. We repeat the search process for 40 different sets of energy surfaces to reveal the typical differences between the two policies.

Across all three metrics shown in Fig. 7.17, CAL significantly outperforms BASE. For CAL, the mean absolute error (MAE) is nearly zero by 50 iterations. Similar convergence is found for the true positive and false positive rates. Together, these metrics indicate that by 50 iterations (i.e., 25% of the search space), CAL is able to predict the energy of the convex hull as well as classify which compositions are on and off the hull. BASE, however, takes significantly longer to come to these conclusions. Considering that there only 198 phase-composition pairs in this space, BASE requires observing nearly all phase-composition pairs to understand the convex hull. Not only does BASE finish far slower, but its rate of learning is consistently lower through the search process, as shown by its smaller slopes in Fig. 7.17a-c. Finally, from the width of the shaded regions, we conclude that BASE is much more variable than CAL.

Fig. 7.18 shows a representative example from Fig. 7.17 to understand the root of how CAL so efficiently and consistently reveals the hull. The true energetic landscape is shown in panel (a) with energy surfaces corresponding to the three distinct phases. A slice through these energy surfaces is shown in (e); here, we show from B to intermediate composition AC. Additionally, a slice of the true convex hull is included below in grey. In panel (i), the complete convex hull is projected onto two dimensions as a ternary phase diagram. The three energy surfaces are similar in energy, resulting in a fairly complex phase diagram. As such, this is a challenging task for hull determination.

We model the three energy surfaces using separate Gaussian processes and conduct a total of 50 observations within this system. In panels (b-d), we show the mean of each GP and color the three surfaces by their standard deviation. In (b), before any observations, all energy surfaces have significant uncertainty and are thus orange. With increasing iteration, both the mean energies evolve and the uncertainties decrease for select composition regions; it will be made clear that these regions are targeted by CAL for their relevance to the convex hull. The evolution of energetic uncertainties can be clearly seen in the $B - AC$ slice. Composition-phase pairs near the hull show evidence of significant observation and an associated reduction in uncertainty. It is important to note that only observing the lowest energy phase would not have been an optimal solution–different phases affect the hull in different regions.

In panels (j-l), 100 hulls are projected and overlaid onto the ternary phase diagram. As expected, no coherent expectation for the hull is present initially. By 30 iterations, most of the single-phase regions have been identified, but there is still significant uncertainty. As such, some unstable compositions are classified as having a non-zero probability of being on the hull, resulting in a smearing out of the ternary phase diagram. Finally, after 50 iterations, much of the lingering uncertainty has dissipated and the convex hull is well understood.

7.3.2 Discussion

The above case examples demonstrate CAL as a fundamentally distinct approach to resolving phase diagrams. There are a variety of ways in which the general method presented can be adjusted to specific search problems. Herein, we consider joint Gaussian processes as tools for capturing correlations between separate phases. As a natural extension of joint Gaussian processes, we discuss conducting CAL simultaneously over a variety of temperatures. We then list ways in which the computational cost of CAL can be reduced for truly vast composition spaces.

It is also explained how our method may play a role in a broader uncertainty-based thermodynamic workflow. First, the importance of uncertainty quantification is discussed, then we consider how CAL may interact with sources of uncertainty that precede it in a workflow. Finally, we talk through how the uncertainty in CAL predictions is propagated forward to other thermodynamic predictions.

Correlated Energy Surfaces For simplicity, we used separate GPs for modeling the energy surface of each competing phase. If there are compositional correlations between energy surfaces, the set of GPs are not learning from them. For systems where strong compositional correlations are expected, it would be advantageous to use observations of one phase-composition pair to help inform the beliefs about a separate phase for similar compositions.

Joint Gaussian processes are well-suited for incorporating compositional correlations into the energy model [6, 60]. In a joint Gaussian process, the energy surface of each phase would be modeled simultaneously; the inputs for such a model would be observations across all phases, and the outputs would be the energy surfaces for each phase. Incorporating joint GPs into CAL would leave the acquisition function unchanged.

Temperature Often, it is favorable to produce phase diagrams over a range of temperatures; example applications include tuning synthesis conditions or identifying phase transitions that limit the operating conditions for a material. To incorporate temperature into the CAL workflow, the free energy surface could be modeled as a function of both composition and temperature. Such an approach would allow for the GP to explicitly learn the relationship between free energy surfaces at differing temperatures. As a terminology note, here we use the term “free energy” to explicitly denote the temperature dependence of the thermodynamic potential.

The policy for determining the next optimal observation would need to be extended in order to account for temperature as an added dimension in the design space. The added complexity derives from the free energy convex hull only being defined over composition space at a single temperature. As such, the total expected information gain for a single phase-composition-temperature triplet would need to be assessed as a sum over the expected information gains across temperatures of interest. In practice, the temperature range would need to be discretized to make evaluating the total information gain feasible.

A special case of temperature-dependent search involves thermodynamic methods where calculating the enthalpy of formation is the computationally limiting factor and the entropy can be approximated analytically [148, 222, 37]. As such, with these methods the free energy can be predicted at multiple temperatures with no additional cost . The ramifications of this set of observations would need to be incorporated into the acquisition function.

Computational Scaling and Approaches for Cutting Cost The computational cost of CAL will often be dwarfed by that of first-principles calculations. However, there is some cost to CAL, especially when moving to multi-dimensional composition spaces with many possible phase-composition pairs. If the cost of CAL is unacceptably large compared to the energy evaluations, there are multiple shortcuts for speeding up the algorithm.

Evaluating the expected information gain (EIG) across phase-composition pairs is the main source of cost for CAL. Indeed, one could use Bayesian optimization to efficiently find the optimal phase-composition pair that maximizes the EIG. One could also imagine using a coarse grid of compositions to begin with and then iteratively increasing the granularity of the composition grid as the convex hull distribution continues to tighten.

In truly large spaces, one may want to prioritize composition sub-regions. The acquisition function can be readily altered to exclusively focus on such regions. Here, the expected information gain would only reflect minimizing the uncertainty for the convex hull in those prioritized regions. The resulting efficiency gain will be dependent on how many different multi-phase regions enclose the specified compositions.

Other approaches center around decreasing the cost of the EIG. For instance, the EIG could be calculated with fewer convex hull samples. Another approach would employ BASE in the beginning of the search and CAL only after some number of iterations. Since CAL is more expensive, it would be reserved for later in the search when there is sufficient information about the hull such that the CAL policy results in significantly different decisions from BASE. Finally, one could approximate the joint entropy as a sum of the entropies across individual compositions. This is a strong approximation for the entropy and should be taken with caution since it assumes convex hulls have no correlations between compositions. All these shortcuts add parameters requiring tuning to negotiate between speed and quality.

Opportunities for Uncertainty-based Workflows Understanding how uncertainty propagates throughout a workflow allows for the rational prioritization of certain segments of the workflow. Thermodynamic stability prediction is one such workflow–it often involves a series of convoluted steps, and at each step there is opportunity to estimate and propagate uncertainty. Such uncertainties could be produced from first-principles calculations [199, 25], fitting surrogate models [149, 34], or numerical approaches to approximating free energies [148]. The GP within CAL could incorporate uncertainties from previous steps as noise in its observations. Such noise would be reflected in the convex hull distribution and resulting predictions.

In an uncertainty-based thermodynamic workflow, CAL could be useful in iteratively training surrogate models with energetic uncertainties like the Bayesian approach to cluster expansion [140, 106, 5, 149, 34]. Here, completing the necessary first-principles calculations to train such models is the limiting factor. Such training would be focused on minimizing the uncertainty in the convex hull rather than predicting energies.

Specifically, instead of the GP used in our work, the surrogate model would be leveraged to produce uncertainty in the convex hull distribution before and after a potential observation. The simplicity of such an inexpensive surrogate makes it computationally feasible to retrain numerous times, which is necessary for choosing the optimal observation. Once an optimal composition is identified by CAL, its energy would be calculated using first-principles, and the result would be included in the training set for the surrogate model.

Ultra-fine Convex Hulls Bayesian modeling also allows for propagating uncertainty to subsequent steps in the thermodynamic workflow. We have shown such propagation for both stability predictions and chemical potentials, and herein we highlight one more example–the production of ultra-fine convex hulls from coarse-grid composition spaces. Producing fine-grained convex hulls is advantageous due to their ability to resolve single-phase regions, but conducting CAL on ultra-fine composition grids heavily increases its computational cost. As such, we use CAL to conduct search on coarse grids and use post-processing to produce the fine-grained convex hulls shown in Fig. 7.18j-l. Specifically, a new GP is trained on the existing energy observations from the coarse grid and produces energetic predictions over a fine composition space. The resulting convex hull distribution is subsequently derived. The associated uncertainty with interpolating to fine grids is naturally included in the convex hull predictions.

7.3.3 Conclusion

Efficient, scalable calculations coupled with end-to-end uncertainty predictions are critical for the next generation of computational materials design. Here, CAL provides a crucial component of this workflow with the ability to efficiently and accurately predict thermodynamic stability. This enhancement comes from developing an acquisition function for active learning that is focused on minimizing the uncertainty of the convex hull. Rather than attempt to characterize the entire space, CAL prioritizes observing compositions that are on or near the hull. As a result, we see a factor of four gain in search efficiency for complex ternary spaces. While we focus on ternary spaces, our approach generalizes across dimensions; thus, it can be applied to pernicious problems such as generating phase diagrams for high-entropy alloys. Uncertainty quantification of both phase stability and associated intensive variables emerges naturally from this hull-aware Bayesian method. Such intensive variables (e.g., pressure, chemical potential, voltage) are critical for linking CAL's results into a predictive workflow for informing experimental campaigns.

Chapter 8

Conclusion and Future Directions

This dissertation studies multiple experimental design settings that are relevant to scientific discovery tasks. By taking a decision-theoretic approach, our policies achieve superior empirical performance than previously proposed baselines. Overall, we provide practitioners with a wide range of algorithmic solutions to tackle diverse experimental design problems.

We first studied Bayesian optimization in high dimensions and developed a local optimization framework that maximizes the probability of descending on the objective function (i.e., making optimization progress). This maximization of descent probability is achieved by exploiting the rich structure of the Gaussian process belief about the objective function, which allows us to (i) compute the direction of maximally likely descent in closed form and (ii) gather data to closely approximate maximizing the posterior maximum descent probability. The resulting policy leads to more efficient optimization than baselines across many tasks. A natural next step in this direction is to treat our local optimization algorithm as a component within a larger framework for global optimization, such as those in McLeod et al. [133], Wang et al. [200], Diouane et al. [47]. The outer optimization scheme could be responsible for managing multiple local optimization runs, choosing the initial starting point for each run, as well as terminating runs that either have converged or are unlikely to yield good results. Here, the closed-form maximum descent probability from our method could play a role in determining whether a run should be terminated if, for example, after the local region has been thoroughly been explored, the maximum descent probability does not exceed a certain threshold. In addition to maximizing descent probability, future directions could explore other heuristics that are conducive to local optimization. For example, Fan et al. [53] made a connection between moving in a descent direction and minimizing the upper confidence bound of Gaussian process belief about the objective function, and derived local optimization policies that leverage this upper confidence bound.

This dissertation also forwards the research on active search methodologies by extending the machinery in traditional active search to multifidelity and multiclass scenarios that model many relevant scientific discovery tasks. We first prove the same hardness result in the original to these two new settings, showing that the performance of the optimal policy cannot be approximated by that of any polynomial-time policy by any constant ratio. We further adopt the nonmyopic search policy to these settings by efficiently approximating optimal future designs under their respective utility functions. The resulting policies are shown to produce nonmyopic decisions that carefully reason about the remaining search budget, and overall outperform various baselines. A future direction could extend these policies to batch scenarios, where multiple experiments are conducted at the same time to maximize experimentation throughput.

In the multiclass case, we propose a utility function with diminishing returns in the number of targets found in each class to encourage diverse discoveries across the classes (i.e., diversity in the label space), but alternative search settings that reward diversity in the feature space are also possible. An example of this is the work of Nguyen and Dieng [145], who introduced a utility function that acts as an "effective count" for the binary setting. As computing this utility function involves decomposing the covariance matrix of the input data and thus requires cubic effort, adopting a nonmyopic lookahead similar to that of Jiang et al. [88] was not straightforward, and the authors proposed greedily optimizing the one-step lookahead. Future works could consider developing efficient rollout strategies that well-approximate future designs under such active search settings.

Another contribution of this dissertation lies in an imitation learning framework for training a neural network to mimic the behavior of the state-of-the-art policy by Jiang et al. [88], as a way to amortize policy computation. Our imitation learning procedure was successful in training the network to inherit the beneficial search strategy from the expert policy. The trained policy network ultimately achieved strong empirical performance that closely approximates that of the expert, while being significantly less computationally demanding. This increased computational efficiency opens doors to the application of active search to real-time systems where decisions must been made quickly, as well as large-scale search spaces such as drug discovery databases. With that said, behavior cloning of the expert policy from Jiang et al. [88] is not necessarily the end goal, and one could seek to further improve from our imitator policy network by learning directly from experiences using modern reinforcement learning techniques. Hester et al. [81], for instance, proposed a procedure in which the policy network initially imitates an expert policy to achieve a reasonable behavior, and subsequently keeps learning from interacting with the target environment to outperform that expert and establish the new state of the art in game-playing tasks; adopting similar strategies for active search poses an interesting future direction.

Another attractive feature of directly learning from past experiences with reinforcement learning is that, under active search settings where the utility function is not known, we only require evaluation access to the utility function to determine the reward at the end of each search campaign. While previous works have assumed the true utility function that accurately reflects one's objective is known *a priori* to be able to simulate future designs,[11] in many real-world applications, we might start out with an unknown utility function (an example is the problem setting considered in Bhatia et al. [22]). Assuming a particular structure of the unknown utility function could lead to utility misspecification and consequently misguided designs. Instead, a potential direction may consider eliciting valuations from a human evaluator by having them "score" the performance of the current policy in each search episode, without needing to learn the underlying utility function. An important consideration in this user-oriented direction is that the system frugally queries the human evaluator for their scoring, so as to avoid straining the user and resulting in noisy evaluations.

Finally, this dissertation includes a number of applications of experimental design to real-world discovery spanning chemistry, materials science, and human–computer interaction for data discovery. By leveraging the tools experimental design, these campaigns lead to more efficient learning and discovery than traditional, non-adaptive techniques, and demonstrate the benefits of our methods.

[11] Although an exception is found in Chapt. 5, the setting explored in that work does assume that all possible utility functions share a common characteristic, specifically diminishing returns from similar discoveries.

References

[1] Solveig S Aamlid, Mohamed Oudah, Jorg Rottler, and Alannah M Hallas. Understanding the role of entropy in high entropy oxides. *Journal of the American Chemical Society*, 145(11):5991–6006, 2023. (Cited on pg. 110.)

[2] Biruk G Abreha, Snigdha Agarwal, Ian Foster, Ben Blaiszik, and Steven A Lopez. Virtual Excited State Reference for the Discovery of Electronic Materials Database: An Open-Access Resource for Ground and Excited State Properties of Organic Molecules. *J. Phys. Chem. Lett.*, 10(21):6835–6841, 2019. (Cited on pg. 85.)

[3] Sharat Agarwal, Himanshu Arora, Saket Anand, and Chetan Arora. Contextual Diversity for Active Learning. In *European Conference on Computer Vision*, pages 137–153. Springer, 2020. (Cited on pg. 60.)

[4] Robert A Alberty. Use of legendre transforms in chemical thermodynamics (iupac technical report). *Pure and Applied Chemistry*, 73(8):1349–1380, 2001. (Cited on pg. 110.)

[5] Manuel Aldegunde, Nicholas Zabaras, and Jesper Kristensen. Quantifying uncertainties in first-principles alloy thermodynamics using cluster expansions. *Journal of Computational Physics*, 323:17–44, 2016. (Cited on pg. 125.)

[6] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for Vector-Valued Functions: A Review . *Foundations and Trends in Machine Learning*, 4(3): 195–266, 2012. (Cited on pgs. 42 and 124.)

[7] J. M. An, S. V. Barabash, V. Ozolins, M. van Schilfgaarde, and K. D. Belashchenko. First-principles study of phase stability of gd-doped euo and eus. *Phys. Rev. B*, 83: 064105, Feb 2011. doi: 10.1103/PhysRevB.83.064105. URL `https://link.aps.org/doi/10.1103/PhysRevB.83.064105`. (Cited on pg. 110.)

[8] Ricardo Andrade-Pacheco, Francois Rerolle, Jean Lemoine, Leda Hernandez, Aboulaye Meïté, Lazarus Juziwelo, Aurélien F Bibaut, Mark J van der Laan, Benjamin F Arnold, and Hugh JW Sturrock. Finding hotspots: development of an adaptive spatial sampling approach. *Scientific Reports*, 10, 2020. (Cited on pgs. 77 and 80.)

[9] Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29*, 2016. (Cited on pg. 76.)

[10] Peter Auer. Using Upper Confidence Bounds for Online Learning . In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 270–279. IEEE, 2000. (Cited on pg. 76.)

[11] Peter Auer. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002. (Cited on pg. 61.)

[12] Shrawan Baghel, Helen Cathcart, and Niall J O'Reilly. Polymeric amorphous solid dispersions: a review of amorphization, crystallization, stabilization, solid-state characterization, and aqueous solubilization of biopharmaceutical classification system class ii drugs. *Journal of pharmaceutical sciences*, 105(9):2527–2544, 2016. (Cited on pg. 110.)

[13] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996. (Cited on pg. 113.)

[14] Christopher J Bartel. Review of computational approaches to predict the thermodynamic stability of inorganic solids. *Journal of Materials Science*, 57(23):10475–10498, 2022. (Cited on pg. 110.)

[15] Albert P Bartók and Gábor Csányi. Gaussian approximation potentials: A brief tutorial introduction. *International Journal of Quantum Chemistry*, 115(16):1051–1057, 2015. (Cited on pg. 110.)

[16] Albert P Bartók, Mike C Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13):136403, 2010. (Cited on pg. 110.)

[17] Simon Batzner, Albert Musaelian, and Boris Kozinsky. Advancing molecular simulation with equivariant interatomic potentials. *Nature Reviews Physics*, 5(8):437–438, 2023. (Cited on pg. 110.)

[18] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957. (Cited on pgs. 9, 32, and 52.)

[19] Mostapha Benhenda. ChemGAN challenge for drug discovery: can AI reproduce natural chemical diversity? *arXiv preprint*, 2017. arXiv:1708.08227 [stat.ML]. (Cited on pg. 50.)

[20] Daniel Bernoulli. *Exposition of a New Theory on the Measurement of Risk*. Econometrica, 1954. (Cited on pg. 51.)

[21] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific Belmont, MA, 1995. (Cited on pgs. 36 and 88.)

[22] Kush Bhatia, Peter L Bartlett, Anca D Dragan, and Jacob Steinhardt. Agnostic learning with unknown utilities. *arXiv preprint*, 2021. arXiv:2104.08482. (Cited on pg. 130.)

[23] Matthias Bitzer, Mona Meister, and Christoph Zimmer. Amortized Inference for Gaussian Process Hyperparameters of Structured Kernels. In *Uncertainty in Artificial Intelligence*, pages 184–194, 2023. (Cited on pg. 76.)

[24] Tom Blau, Edwin V Bonilla, Iadine Chades, and Amir Dezfouli. Optimizing Sequential Experimental Design with Deep Reinforcement Learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 2107–2128, 2022. (Cited on pg. 75.)

[25] Emanuele Bosoni, Louis Beal, Marnik Bercx, Peter Blaha, Stefan Blügel, Jens Bröder, Martin Callsen, Stefaan Cottenier, Augustin Degomme, Vladimir Dikan, et al. How to verify the precision of density-functional-theory implementations via reproducible and universal workflows. *Nature Reviews Physics*, 6(1):45–58, 2024. (Cited on pg. 125.)

[26] Nadia Boukhelifa, Anastasia Bezerianos, Ioan Cristian Trelea, Nathalie Méjean Perrot, and Evelyne Lutton. An Exploratory Study on Visual Exploration of Model Simulations by Multiple Types of Experts. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2019. (Cited on pg. 98.)

[27] Klaus Brinker. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of the 20th International Conference on Machine Learning*, pages 59–66, 2003. (Cited on pg. 60.)

[28] Eric Brochu, Vlad M Cora, and Nando De Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *arXiv preprint*, 2010. arXiv:1012.2599 [cs.LG]. (Cited on pg. 42.)

[29] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. GuacaMol: Benchmarking Models for de Novo Molecular Design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019. (Cited on pg. 78.)

[30] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. In *Annals of Mathematics and Artificial Intelligence*, volume 76, pages 5–23, 2016. (Cited on pg. 13.)

[31] Brain Cantor, ITH Chang, Peter Knight, and AJB Vincent. Microstructural development in equiatomic multicomponent alloys. *Materials Science and Engineering: A*, 375:213–218, 2004. (Cited on pg. 110.)

[32] Alexandra Carpentier, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, and Peter Auer. Upper-Confidence-Bound Algorithms for Active Learning in Multi-armed Bandits. In *International Conference on Algorithmic Learning Theory*, pages 189–203, 2011. (Cited on pg. 76.)

[33] Jeng-Da Chai and Martin Head-Gordon. Long-Range Corrected Hybrid Density Functionals with Damped Atom–Atom Dispersion Corrections. *Phys. Chem. Chem. Phys.*, 10(44):6615–6620, 2008. (Cited on pg. 86.)

[34] Hantong Chen, Sayan Samanta, Siya Zhu, Hagen Eckert, Jan Schroers, Stefano Curtarolo, and Axel van de Walle. Bayesian active machine learning for cluster expansion construction. *Computational Materials Science*, 231:112571, 2024. (Cited on pgs. 111 and 125.)

[35] Jiadong Chen, Samuel R Cross, Lincoln J Miara, Jeong-Ju Cho, Yan Wang, and Wenhao Sun. Navigating phase diagram complexity to guide robotic inorganic materials synthesis. *arXiv preprint arXiv:2304.00743*, 2023. (Cited on pg. 110.)

[36] Jian Chen, Xueyang Zhou, Weili Wang, Bing Liu, Yukun Lv, Wei Yang, Dapeng Xu, and Yong Liu. A review on fundamental of high entropy alloys with promising high–temperature properties. *Journal of Alloys and Compounds*, 760:15–30, 2018. (Cited on pg. 110.)

[37] Wei Chen, Antoine Hilhorst, Georgios Bokas, Stéphane Gorsse, Pascal J Jacques, and Geoffroy Hautier. A map of single-phase high-entropy alloys. *Nature Communications*, 14(1):2856, 2023. (Cited on pg. 124.)

[38] Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch Active Learning at Scale. *Advances in Neural Information Processing Systems 34*, 2021. (Cited on pg. 60.)

[39] Cody Coleman, Edward Chou, Julian Katz-Samuels, Sean Culatana, Peter Bailis, Alexander C Berg, Robert Nowak, Roshan Sumbaly, Matei Zaharia, and I Zeki Yalniz. Similarity Search for Efficient Active Learning and Search of Rare Concepts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 2022. (Cited on pg. 75.)

[40] Kristin Cook, Georges Grinstein, and Mark Whiting. The vast challenge: History, scope, and outcomes: An introduction to the special issue, 2014. (Cited on pg. 103.)

[41] Stefano Curtarolo, Wahyu Setyawan, Gus LW Hart, Michal Jahnatek, Roman V Chepulskii, Richard H Taylor, Shidong Wang, Junkai Xue, Kesong Yang, Ohad Levy, et al. Aflow: An automatic framework for high-throughput materials discovery. *Computational Materials Science*, 58:218–226, 2012. (Cited on pg. 110.)

[42] Elbio Dagotto. Complexity in strongly correlated electronic systems. *Science*, 309 (5732):257–262, 2005. (Cited on pg. 110.)

[43] Aritra Dasgupta, Joon-Yong Lee, Ryan Wilson, Robert A. Lafrance, Nick Cramer, Kristin Cook, and Samuel Payne. Familiarity vs trust: A comparative study of domain scientists' trust in visual analytics and conventional analysis methods. *IEEE*

Transactions on Visualization and Computer Graphics, 23(1):271–280, 2017. (Cited on pg. 110.)

[44] Volker L Deringer, Albert P Bartók, Noam Bernstein, David M Wilkins, Michele Ceriotti, and Gábor Csányi. Gaussian process regression for materials and molecules. *Chemical Reviews*, 121(16):10073–10141, 2021. (Cited on pg. 112.)

[45] Volker L Deringer, Albert P Bartók, Noam Bernstein, David M Wilkins, Michele Ceriotti, and Gábor Csányi. Gaussian Process Regression for Materials and Molecules. *Chemical Reviews*, pages 10073–10141, 2021. (Cited on pg. 8.)

[46] Thomas Deselaers and Vittorio Ferrari. Visual and Semantic Similarity in ImageNet. In *CVPR 2011*, pages 1777–1784. IEEE, 2011. (Cited on pg. 97.)

[47] Youssef Diouane, Victor Picheny, Rodolphe Le Riche, and Alexandre Scotto Di Perrotolo. TREGO: a Trust-Region Framework for Efficient Global Optimization. *arXiv preprint*, 2021. arXiv:2101.06808 [stat.ML]. (Cited on pg. 128.)

[48] R Ditchfield, WJ Hehre, and JA Pople. Self-Consistent Molecular Orbital Methods. VI. Energy Optimized Gaussian Atomic Orbitals. *J. Chem. Phys.*, 52(10):5001–5007, 1970. (Cited on pg. 86.)

[49] Ralf Drautz. Atomic cluster expansion for accurate and transferable interatomic potentials. *Physical Review B*, 99(1):014104, 2019. (Cited on pg. 110.)

[50] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep Reinforcement Learning in Large Discrete Action Spaces. *arXiv preprint*, 2015. arXiv:1512.07679 [cs.AI]. (Cited on pgs. 1 and 68.)

[51] Mary T Dzindolet, Scott A Peterson, Regina A Pomranky, Linda G Pierce, and Hall P Beck. The role of trust in automation reliance. *Computer Studies*, page 22, 2003. (Cited on pg. 109.)

[52] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. (Cited on pgs. 22 and 25.)

[53] Zheyi Fan, Wenyu Wang, Szu Hui Ng, and Qingpei Hu. Minimizing UCB: a Better Local Search Strategy in Local Bayesian Optimization. *arXiv preprint*, 2024. arXiv:2405.15285. (Cited on pg. 128.)

[54] Meng Fang, Yuan Li, and Trevor Cohn. Learning how to Active Learn: A Deep Reinforcement Learning Approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, 2017. (Cited on pg. 76.)

[55] Mi Feng, Evan Peck, and Lane Harrison. Patterns and Pace: Quantifying Diverse Exploration Behavior with Visualizations on the Web. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 501–511. IEEE Computer Society, 2018. (Cited on pg. 98.)

[56] Adam Foster, Desi R Ivanova, Ilyas Malik, and Tom Rainforth. Deep Adaptive Design: Amortizing Sequential Bayesian Experimental Design. In *Proceedings of the 38th International Conference on Machine Learning*, pages 3384–3395, 2021. (Cited on pg. 75.)

[57] Michelle M Francl, William J Pietro, Warren J Hehre, J Stephen Binkley, Mark S Gordon, Douglas J DeFrees, and John A Pople. Self-Consistent Molecular Orbital Methods. XXIII. A Polarization-Type Basis Set for Second-Row Elements. *J. Chem. Phys.*, 77(7):3654–3665, 1982. (Cited on pg. 86.)

[58] Warren RJD Galloway, Albert Isidro-Llobet, and David R Spring. Diversity-oriented synthesis as a tool for the discovery of novel biologically active small molecules. *Nature communications*, 1(1):1–13, 2010. (Cited on pg. 50.)

[59] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2022. (Cited on pgs. 14, 28, and 60.)

[60] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023. (Cited on pg. 124.)

[61] Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff Schneider, and Richard Mann. Bayesian Optimal Active Search and Surveying. In *Proceedings of the 29th International Conference on Machine Learning*, 2012. (Cited on pgs. 29, 39, 40, 41, 42, 50, 58, 59, 62, 68, 75, 87, 97, and 109.)

[62] Roman Garnett, Thomas Gärtner, Martin Vogt, and Jürgen Bajorath. Introducing the 'active search' method for iterative virtual screening. *Journal of Computer-Aided Molecular Design*, 29:305–314, 2015. (Cited on pgs. 8, 77, 78, and 87.)

[63] Andrew Gelman and Aki Vehtari. What are the most important statistical ideas of the past 50 years? *Journal of the American Statistical Association*, 116(536):2087–2097, 2021. (Cited on pg. 1.)

[64] Easo P George, Dierk Raabe, and Robert O Ritchie. High-entropy alloys. *Nature reviews materials*, 4(8):515–534, 2019. (Cited on pg. 110.)

[65] Daniel Golovin and Andreas Krause. Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011. (Cited on pg. 53.)

[66] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS central science*, 2018. (Cited on pg. 8.)

[67] Javier González, Michael Osborne, and Neil Lawrence. GLASSES: Relieving The Myopia Of Bayesian Optimisation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 790–799, 2016. (Cited on pg. 54.)

[68] Aditya Gopalan and Shie Mannor. Thompson Sampling for Learning Parameterized Markov Decision Processes. In *Proceedings of The 28th Conference on Learning Theory*, volume 40, pages 861–898, 2015. (Cited on pg. 13.)

[69] Prashun Gorai, Anuj Goyal, Eric S Toberer, and Vladan Stevanović. A simple chemical guide for finding novel n-type dopable zintl pnictide thermoelectric materials. *Journal of Materials Chemistry A*, 7(33):19385–19395, 2019. (Cited on pg. 116.)

[70] DI Gorbunov, MD Kuz'min, K Uhlířová, M Žáček, M Richter, Y Skourski, and AV Andreev. Magnetic properties of a gdmn6sn6 single crystal. *Journal of alloys and compounds*, 519:47–54, 2012. (Cited on pg. 110.)

[71] Sean D Griesemer, Yi Xia, and Chris Wolverton. Accelerating the prediction of stable materials with machine learning. *Nature Computational Science*, 3(11):934–945, 2023. (Cited on pg. 110.)

[72] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 2020. (Cited on pg. 8.)

[73] Yingjie Gu, Zhong Jin, and Steve C Chiu. Active learning combining uncertainty and diversity for multi-class image classification. *IET Computer Vision*, 9(3):400–407, 2015. (Cited on pg. 59.)

[74] Konstantin Gubaev, Evgeny V Podryabinkin, Gus LW Hart, and Alexander V Shapeev. Accelerating high-throughput searches for new alloys with active learning of interatomic potentials. *Computational Materials Science*, 156:148–156, 2019. (Cited on pg. 112.)

[75] Sunil Gupta, Alistair Shilton, Santu Rana, and Svetha Venkatesh. Exploiting Strategy-Space Diversity for Batch Bayesian Optimisation. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, pages 538–547, 2018. (Cited on pg. 60.)

[76] Wenkai Han and Hans-Jorg Schulz. Beyond trust building — calibrating trust in visual analytics. In *2020 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX)*, pages 9–15. IEEE, 2020. (Cited on pg. 109.)

[77] Gus LW Hart, Tim Mueller, Cormac Toher, and Stefano Curtarolo. Machine learning for alloys. *Nature Reviews Materials*, 6(8):730–755, 2021. (Cited on pg. 110.)

[78] Jingrui He and Jaime Carbonell. Nearest-Neighbor-Based Active Learning for Rare Category Detection. In *Advances in Neural Information Processing Systems 20*, pages 633–640, 2007. (Cited on pgs. 60, 61, 63, and 66.)

[79] Vinay I Hegde, Muratahan Aykol, Scott Kirklin, and Chris Wolverton. The phase stability network of all inorganic materials. *Science advances*, 6(9):eaay5606, 2020. (Cited on pg. 110.)

[80] José Miguel Hernández-Lobato, James Requeima, Edward O Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. (Cited on pg. 8.)

[81] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep Q-learning from Demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. (Cited on pg. 129.)

[82] Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. Graph Policy Network for Transferable Active Learning on Graphs. In *Advances in Neural Information Processing Systems 33*, pages 10174–10185, 2020. (Cited on pg. 76.)

[83] Deng Huang, Theodore T Allen, William I Notz, and R Allen Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382, 2006. (Cited on pg. 42.)

[84] Conor Igoe, Ramina Ghods, and Jeff Schneider. Multi-Agent Active Search: A Reinforcement Learning Approach. *IEEE Robotics and Automation Letters*, 7(2):754–761, 2021. (Cited on pg. 76.)

[85] John E Jaffe, James A Snyder, Zijing Lin, and Anthony C Hess. Lda and gga calculations for high-pressure phase transitions in zno and mgo. *Physical Review B*, 62(3): 1660, 2000. (Cited on pgs. 110 and 116.)

[86] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1), 2013. (Cited on pg. 110.)

[87] Heinrich Jiang and Afshin Rostamizadeh. Active Covering. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5013–5022, 2021. (Cited on pgs. 75 and 77.)

[88] Shali Jiang, Gustavo Malkomes, Geoff Converse, Alyssa Shofner, Benjamin Moseley, and Roman Garnett. Efficient Nonmyopic Active Search. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1714–1723, 2017. (Cited on pgs. 8, 10, 11, 12, 29, 34, 35, 40, 41, 42, 44, 49, 50, 51, 53, 54, 61, 62, 68, 69, 70, 75, 77, 79, 81, 84, 87, 89, 109, 129, 152, and 153.)

[89] Shali Jiang, Gustavo Malkomes, Matthew Abbott, Benjamin Moseley, and Roman Garnett. Efficient nonmyopic batch active search. In *Advances in Neural Information Processing Systems*, volume 31, 2018. (Cited on pgs. 29, 36, 41, 42, 44, 48, 58, 59, 68, 75, 77, 87, 88, and 89.)

[90] Shali Jiang, Roman Garnett, and Benjamin Moseley. Cost effective active search. In *Advances in Neural Information Processing Systems 32*, pages 4880–4889, 2019. (Cited on pgs. 29, 42, 44, 59, 68, and 75.)

[91] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. (Cited on pgs. 58 and 73.)

[92] Sara Kadkhodaei and Jorge A Muñoz. Cluster expansion of alloy theory: a review of historical development and modern innovations. *JOM*, 73(11):3326–3346, 2021. (Cited on pg. 110.)

[93] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. The Multi-fidelity Multi-armed Bandit. In *Advances in Neural Information Processing Systems 29*, pages 1785–1793, 2016. (Cited on pg. 43.)

[94] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian Optimisation with Continuous Approximations. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2861–2878, 2017. (Cited on pg. 42.)

[95] Y Kawazoe, J-Z Yu, A-P Tsai, and T Masumoto, editors. *Nonequilibrium Phase Diagrams of Ternary Amorphous Alloys*, volume 37A of *Condensed Matters*. Springer-Verlag, 1997. (Cited on pgs. 45 and 77.)

[96] B Keimer and JE Moore. The physics of quantum materials. *Nature Physics*, 13(11): 1045–1055, 2017. (Cited on pg. 110.)

[97] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceddings of the 3rd International Conference for Learning Representations*, 2015. (Cited on pg. 74.)

[98] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proceddings of the 2nd International Conference for Learning Representations*, 2014. (Cited on pg. 82.)

[99] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised Learning with Deep Generative Models. *Advances in Neural Information Processing Systems 27*, 2014. (Cited on pg. 82.)

[100] Nikita Klyuchnikov, Davide Mottin, Georgia Koutrika, Emmanuel Müller, and Panagiotis Karras. Figuring out the User in a Few Steps: Bayesian Multifidelity Active Search with Cokriging. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 686–695, 2019. (Cited on pgs. 42 and 43.)

[101] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Learning Active Learning from Data. In *Advances in Neural Information Processing Systems 30*, 2017. (Cited on pg. 76.)

[102] Suraj Kothawade, Nathan Beck, Krishnateja Killamsetty, and Rishabh Iyer. SIMILAR: Submodular Information Measures Based Active Learning In Realistic Scenarios. *Advances in Neural Information Processing Systems 34*, pages 18685–18697, 2021. (Cited on pg. 60.)

[103] Suraj Kothawade, Vishal Kaushal, Ganesh Ramakrishnan, Jeff Bilmes, and Rishabh Iyer. PRISM: A Rich Class of Parameterized Submodular Information Measures for Guided Data Subset Selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10238–10246, 2022. (Cited on pg. 60.)

[104] Andreas Krause and Carlos Guestrin. Near-optimal Observation Selection using Submodular Functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 21, 2007. (Cited on pgs. 53 and 56.)

[105] Dinesh Krishnamoorthy and Francis J Doyle. Safe Bayesian Optimization Using Interior-Point Methods—Applied to Personalized Insulin Dose Guidance. *IEEE Control Systems Letters*, 6:2834–2839, 2022. (Cited on pg. 1.)

[106] Jesper Kristensen and Nicholas J Zabaras. Bayesian uncertainty quantification in the evaluation of alloy properties with the cluster expansion method. *Computer Physics Communications*, 185(11):2885–2892, 2014. (Cited on pg. 125.)

[107] Alex Kulesza and Ben Taskar. Determinantal Point Processes for Machine Learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012. (Cited on pg. 60.)

[108] Fumiaki Kuroda, Satoshi Hagiwara, and Minoru Otani. Structural changes in the lithium cobalt dioxide electrode: A combined approach with cluster expansion and bayesian optimization. *Physical Review Materials*, 7(11):115402, 2023. (Cited on pg. 112.)

[109] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From Word Embeddings to Document Distances. In *Proceedings of the 32th International Conference on Machine Learning*, pages 957–966, 2015. (Cited on pg. 97.)

[110] MD Kuz'min, Yu Skourski, D Eckert, M Richter, K-H Müller, KP Skokov, and IS Tereshina. Spin reorientation in high magnetic fields and the co-gd exchange field in gdco 5. *Physical Review B*, 70(17):172412, 2004. (Cited on pg. 110.)

[111] Jarno Laakso, Milica Todorović, Jingrui Li, Guo-Xu Zhang, and Patrick Rinke. Compositional engineering of perovskites with machine learning. *Physical Review Materials*, 6(11):113801, 2022. (Cited on pg. 110.)

[112] Tze Leung Lai and Herbert Robbins. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985. (Cited on pg. 43.)

[113] Greg Landrum. RDKit: Open-Source Cheminformatics Software. *Open-source software*, 2016. (Cited on pg. 85.)

[114] Jung-Hoon Lee, Adam Jaffe, Yu Lin, Hemamala I Karunadasa, and Jeffrey B Neaton. Origins of the pressure-induced phase transition and metallization in the halide perovskite (ch3nh3) pbi3. *ACS Energy Letters*, 5(7):2174–2181, 2020. (Cited on pgs. 110 and 116.)

[115] David D Lewis and William A Gale. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994. (Cited on pg. 43.)

[116] Christopher H. Lin, Mausam, and Daniel S. Weld. Active Learning with Unbalanced Classes & Example-Generated Queries. In *Proceedings of the 6th AAAI Conference on Human Computation*, 2018. (Cited on pg. 60.)

[117] Dennis V Lindley. On a Measure of the Information Provided by an Experiment . *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956. (Cited on pg. 75.)

[118] Ming Liu, Wray Buntine, and Gholamreza Haffari. Learning How to Actively Learn: A Deep Imitation Learning Approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1874–1883, 2018. (Cited on pg. 76.)

[119] Ming Liu, Wray Buntine, and Gholamreza Haffari. Learning to Actively Learn Neural Machine Translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 334–344, 2018. (Cited on pg. 76.)

[120] Sulin Liu, Xingyuan Sun, Peter J Ramadge, and Ryan P Adams. Task-Agnostic Amortized Inference of Gaussian Process Hyperparameters. In *Advances in Neural Information Processing Systems 33*, pages 21440–21452, 2020. (Cited on pg. 76.)

[121] Tiqing Liu, Yuhmei Lin, Xin Wen, Robert N Jorissen, and Michael K Gilson. BindingDB: A web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic Acids Research*, 35:D198–D201, 2007. (Cited on pgs. 44, 62, and 77.)

[122] Tsung-Wei Liu, Quan Nguyen, Adji Bousso Dieng, and Diego Gomez-Gualdron. Diversity-driven, efficient exploration of a mof design space to optimize mof properties: application to nh3 adsorption. *ChemRxiv preprint*, 2024. (Cited on pg. 8.)

[123] Mina Konakovic Lukovic, Yunsheng Tian, and Wojciech Matusik. Diversity-Guided Multi-Objective Bayesian Optimization With Batch Evaluations. *Advances in Neural Information Processing Systems 33*, 33, 2020. (Cited on pg. 60.)

[124] Lin Ma, Bailu Ding, Sudipto Das, and Adith Swaminathan. Active Learning for ML Enhanced Database Systems. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 175–191, 2020. (Cited on pg. 60.)

[125] David MacKay. The Evidence Framework Applied to Classification Networks. *Neural Computation*, 1992. (Cited on pgs. 75 and 115.)

[126] David MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 1992. (Cited on pgs. 75 and 115.)

[127] Omid Madani, Daniel J Lizotte, and Russell Greiner. Active Model Selection. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 357–365, 2004. (Cited on pg. 60.)

[128] Wesley Maddox, Qing Feng, and Max Balandat. Optimizing High-Dimensional Physics Simulations via Composite Bayesian Optimization. *Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021)*, 2021. (Cited on pg. 13.)

[129] Debasish Das Mahanta, Dennis Robinson Brown, Simone Pezzotti, Songi Han, Gerhard Schwaab, M Scott Shell, and Martina Havenith. Local solvation structures govern the mixing thermodynamics of glycerol–water solutions. *Chemical Science*, 14(26):7381–7392, 2023. (Cited on pg. 110.)

[130] Gustavo Malkomes, Bolong Cheng, Eric H Lee, and Mike Mccourt. Beyond the Pareto Efficient Frontier: Constraint Active Search for Multiobjective Experimental Design. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7423–7434, 2021. (Cited on pgs. 60, 61, 63, and 66.)

[131] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018. (Cited on pg. 22.)

[132] Matthew J McDermott, Brennan C McBride, Corlyn E Regier, Gia Thinh Tran, Yu Chen, Adam A Corrao, Max C Gallant, Gabrielle E Kamm, Christopher J Bartel, Karena W Chapman, et al. Assessing thermodynamic selectivity of solid-state reactions for the predictive synthesis of inorganic materials. *ACS Central Science*, 9(10): 1957–1975, 2023. (Cited on pg. 110.)

[133] Mark McLeod, Stephen Roberts, and Michael A Osborne. Optimization, fast and slow: Optimally switching between local and bayesian optimization. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. (Cited on pg. 128.)

[134] Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, pages 1–6, 2023. (Cited on pg. 110.)

[135] Vanessa Meschke, Prashun Gorai, Vladan Stevanovic, and Eric S Toberer. Search and structural featurization of magnetically frustrated kagome lattices. *Chemistry of Materials*, 33(12):4373–4381, 2021. (Cited on pg. 110.)

[136] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier Than Lazy Greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. (Cited on pg. 59.)

[137] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. (Cited on pg. 83.)

[138] Shayan Monadjemi, Roman Garnett, and Alvitta Ottley. Competing Models: Inferring Exploration Patterns and Information Relevance via Bayesian Model Selection. In *2020 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE Computer Society, 2020. (Cited on pg. 98.)

[139] Shayan Monadjemi, Sunwoo Ha, Quan Nguyen, Henry Chai, Roman Garnett, and Alvitta Ottley. Guided Data Discovery in Interactive Visualizations via Active Search. In *2022 IEEE Visualization and Visual Analytics (VIS)*, pages 70–74. IEEE, 2022. (Cited on pg. 68.)

[140] Tim Mueller and Gerbrand Ceder. Bayesian approach to cluster expansions. *Physical Review B*, 80(2):024103, 2009. (Cited on pg. 125.)

[141] Fatemah Mukadum, Quan Nguyen, Daniel M. Adrion, Gabriel Appleby, Rui Chen, Haley Dang, Remco Chang, Roman Garnett, and Steven A. Lopez. Efficient Discovery of Visible Light-Activated Azoarene Photoswitches with Long Half-Lives Using Active Search. *Journal of Chemical Information and Modeling*, 2021. (Cited on pgs. 8 and 62.)

[142] Sarah Müller, Alexander von Rohr, and Sebastian Trimpe. Local policy search with bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 34, 2021. (Cited on pgs. 13, 15, 22, and 23.)

[143] Elvis Nava, Mojmír Mutnỳ, and Andreas Krause. Diversified Sampling for Batched Bayesian Optimization with Determinantal Point Processes. *arXiv preprint*, 2021. arXiv:2110.11665 [cs.LG]. (Cited on pg. 60.)

[144] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An Analysis of Approximations for Maximizing Submodular Set Functions—I. *Mathematical Programming*, 14(1):265–294, 1978. (Cited on pg. 56.)

[145] Quan Nguyen and Adji Bousso Dieng. Quality-Weighted Vendi Scores And Their Application To Diverse Experimental Design. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. To appear. (Cited on pgs. 8, 84, and 129.)

[146] Quan Nguyen and Roman Garnett. Nonmyopic Multiclass Active Search with Diminishing Returns for Diverse Discovery. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, 2023. (Cited on pgs. 75, 77, 81, and 84.)

[147] Quan Nguyen, Arghavan Modiri, and Roman Garnett. Nonmyopic Multifidelity Acitve Search. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. (Cited on pgs. 58, 59, 75, and 81.)

[148] Andrew Novick, Quan Nguyen, Roman Garnett, Eric Toberer, and Vladan Stevanović. Simulating high-entropy alloys at finite temperatures: An uncertainty-based approach. *Physical Review Materials*, 7(6):063801, 2023. (Cited on pgs. 124 and 125.)

[149] Derick E Ober and Anton Van der Ven. Thermodynamically informed priors for uncertainty propagation in first-principles statistical mechanics. *arXiv preprint arXiv:2309.12255*, 2023. (Cited on pg. 125.)

[150] Hiroaki Okamoto, TB Massalski, et al. Binary alloy phase diagrams. *ASM International, Materials Park, OH, USA*, 12, 1990. (Cited on pg. 110.)

[151] Corey Oses, Eric Gossett, David Hicks, Frisco Rose, Michael J Mehl, Eric Perim, Ichiro Takeuchi, Stefano Sanvito, Matthias Scheffler, Yoav Lederer, et al. Aflow-chull: cloud-oriented platform for autonomous phase stability analysis. *Journal of chemical information and modeling*, 58(12):2477–2490, 2018. (Cited on pg. 110.)

[152] Corey Oses, Cormac Toher, and Stefano Curtarolo. High-entropy ceramics. *Nature Reviews Materials*, 5(4):295–309, 2020. (Cited on pg. 110.)

[153] Shubham Pandey, Jiaxing Qu, Vladan Stevanović, Peter St John, and Prashun Gorai. Predicting energy and stability of known and hypothetical crystals using graph neural network. *Patterns*, 2(11):100361, 2021. (Cited on pg. 110.)

[154] Donald R Paul. *Polymer Blends Volume 1*, volume 1. Elsevier, 2012. (Cited on pg. 110.)

[155] Malcolm Pemberton and Nicholas Rau. *Mathematics for Economists: An Introductory Textbook*. Oxford University Press, 2015. (Cited on pg. 51.)

[156] Tiago Pereira, Maryam Abbasi, Bernardete Ribeiro, and Joel P Arrais. Diversity oriented Deep Reinforcement Learning for targeted molecule generation. *Journal of Cheminformatics*, 13(1):1–17, 2021. (Cited on pg. 50.)

[157] Jason Perry, Christopher D Janneck, Chinua Umoja, and William M Pottenger. Supporting Cognitive Models of Sensemaking in Analytics Systems. Technical report, DIMACS, New Brunswick, NY, 2009. (Cited on pg. 98.)

[158] Victor Picheny, David Ginsbourger, Yann Richet, and Gregory Caplin. Quantile-Based Optimization of Noisy Computer Experiments with Tunable Precision. *Technometrics*, 55(1):2–13, 2013. (Cited on pg. 42.)

[159] Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-Information Source Optimization. In *Advances in Neural Information Processing Systems 30*, pages 4288–4298, 2017. (Cited on pg. 42.)

[160] Matthew J Powell-Palm, Boris Rubinsky, and Wenhao Sun. Freezing water at constant volume and under confinement. *Communications Physics*, 3(1):39, 2020. (Cited on pg. 116.)

[161] Anthony K Rappé, Carla J Casewit, KS Colwell, William A Goddard III, and W Mason Skiff. UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations. *J. Am. Chem. Soc.*, 114(25):10024–10035, 1992. (Cited on pg. 86.)

[162] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. (Cited on pgs. 14, 15, and 72.)

[163] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006. (Cited on pg. 112.)

[164] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA, 2010. (Cited on pg. 101.)

[165] Beth A Reid, Will J Percival, Daniel J Eisenstein, Licia Verde, David N Spergel, Ramin A Skibba, Neta A Bahcall, Tamas Budavari, Joshua A Frieman, Masataka Fukugita, et al. Cosmological constraints from the clustering of the sloan digital sky survey dr7 luminous red galaxies. *Monthly Notices of the Royal Astronomical Society*, 404(1):60–85, 2010. (Cited on pg. 25.)

[166] Lloyd M Robeson. Polymer blends. *A Comprehensive review*, 641, 2007. (Cited on pg. 110.)

[167] David Rogers and Mathew Hahn. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.*, 50(5):742–754, 2010. (Cited on pg. 87.)

[168] Christopher L Rom, Andrew Novick, Matthew J McDermott, Andrey A Yakovenko, Jessica R Gallawa, Gia Thinh Tran, Dominic C Asebiah, Emily N Storck, Brennan C McBride, Rebecca C Miller, et al. Mechanistically guided materials chemistry: Synthesis of ternary nitrides, cazrn2 and cahfn2. *Journal of the American Chemical Society*, 2024. (Cited on pg. 110.)

[169] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning . In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011. (Cited on pgs. 68 and 71.)

[170] Daniel Russo and Benjamin Van Roy. Learning to Optimize via Information-Directed Sampling. In *Advances in Neural Information Processing Systems 27*, 2014. (Cited on pg. 77.)

[171] Daniel Russo and Benjamin Van Roy. Learning to Optimize via Information-Directed Sampling. *Operations Research*, 66:230–252, 2018. (Cited on pg. 77.)

[172] James E Saal, Scott Kirklin, Muratahan Aykol, Bryce Meredig, and Christopher Wolverton. Materials design and discovery with high-throughput density functional theory: the open quantum materials database (oqmd). *Jom*, 65:1501–1509, 2013. (Cited on pg. 110.)

[173] Dominik Sacha, Hansi Senaratne, Bum Chul Kwon, Geoffrey Ellis, and Daniel A. Keim. The role of uncertainty, awareness, and trust in visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):240–249, 2016. ISSN 1077-2626. doi: 10.1109/TVCG.2015.2467591. URL `http://ieeexplore.ieee.org/document/7192716/`. (Cited on pgs. 109 and 110.)

[174] Anindya Sarkar, Nathan Jacobs, and Yevgeniy Vorobeychik. A Partially Supervised Reinforcement Learning Framework for Visual Active Search. In *Advances in Neural Information Processing Systems 36*, pages 12245–12270, 2023. (Cited on pg. 76.)

[175] Anindya Sarkar, Michael Lanier, Scott Alfeld, Jiarui Feng, Roman Garnett, Nathan Jacobs, and Yevgeniy Vorobeychik. A Visual Active Search Framework for Geospatial Exploration. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 8316–8325, 2024. (Cited on pg. 76.)

[176] Jean Scholtz, Mark A Whiting, Catherine Plaisant, and Georges Grinstein. A reflection on seven years of the vast challenge. In *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors-Novel Evaluation Methods for Visualization*, pages 1–8, 2012. (Cited on pg. 103.)

[177] Atsuto Seko and Shintaro Ishiwata. Prediction of perovskite-related structures in a cuo 3- x (a= ca, sr, ba, sc, y, la) using density functional theory and bayesian optimization. *Physical Review B*, 101(13):134101, 2020. (Cited on pg. 112.)

[178] Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *Proceedings of the 6th International Conference on Learning Representations*, 2018. (Cited on pg. 60.)

[179] Burr Settles. Active learning literature survey. 2009. (Cited on pgs. 42, 59, and 111.)

[180] Wanggang Shen and Xun Huan. Bayesian sequential optimal experimental design for nonlinear models using policy gradient reinforcement learning. *Computer Methods in Applied Mechanics and Engineering*, 2023. (Cited on pg. 75.)

[181] Leshi Shu, Ping Jiang, Xinyu Shao, and Yan Wang. A New Multi-Objective Bayesian Optimization Formulation With the Acquisition Function for Convergence and Diversity. *Journal of Mechanical Design*, 142(9):091703, 2020. (Cited on pg. 60.)

[182] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. (Cited on pg. 1.)

[183] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2012. (Cited on pgs. 13 and 42.)

[184] Teague Sterling and John J Irwin. Zinc 15–Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015. (Cited on pgs. 44, 62, and 77.)

[185] Wenhao Sun and Matthew J Powell-Palm. Generalized gibbs' phase rule. *arXiv preprint arXiv:2105.01337*, 2021. (Cited on pg. 110.)

[186] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems 12*, 1999. (Cited on pgs. 81 and 82.)

[187] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. Hybrid Batch Attacks: Finding Black-box Adversarial Examples with Limited Queries. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1327–1344, 2020. (Cited on pg. 28.)

[188] Loren G Terveen. An Overview of Human-Computer Collaboration. *Knowledge-Based Systems*, 8(2-3):67–81, 1995. (Cited on pg. 96.)

[189] Félix Therrien, Eric B Jones, and Vladan Stevanović. Metastable materials discovery in the age of large-scale computation. *Applied Physics Reviews*, 8(3), 2021. (Cited on pg. 110.)

[190] Benjamin I Tingle, Khanh G Tang, Mar Castanon, John J Gutierrez, Munkhzul Khurelbaatar, Chinzorig Dandarchuluun, Yurii S Moroz, and John J Irwin. ZINC-22—A Free Multi-Billion-Scale Database of Tangible Compounds for Ligand Discovery. *Journal of Chemical Information and Modeling*, 63(4):1166–1176, 2023. (Cited on pgs. 68 and 79.)

[191] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. (Cited on pg. 23.)

[192] Jacopo Tomasi, Benedetta Mennucci, and Roberto Cammi. Quantum Mechanical Continuum Solvation Models. *Chem. Rev.*, 105(8):2999–3094, 2005. (Cited on pg. 86.)

[193] Axel van de Walle and Gerbrand Ceder. Automating first-principles phase diagram calculations. *Journal of Phase Equilibria*, 23(4):348, 2002. (Cited on pg. 110.)

[194] Anton Van der Ven, Mehmet K Aydinol, and Gerbrand Ceder. First-principles evidence for stage ordering in li x coo2. *Journal of the Electrochemical Society*, 145(6):2149, 1998. (Cited on pgs. 110 and 116.)

[195] Anton Van der Ven, Zhi Deng, Swastika Banerjee, and Shyue Ping Ong. Rechargeable alkali-ion battery materials: theory and computation. *Chemical reviews*, 120(14):6977–7019, 2020. (Cited on pgs. 110 and 116.)

[196] Hastagiri P Vanchinathan, Andreas Marfurt, Charles-Antoine Robelin, Donald Kossmann, and Andreas Krause. Discovering Valuable Items from Massive Data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1195–1204, 2015. (Cited on pgs. 42, 60, 61, 63, and 66.)

[197] Andrij Vasylenko, Benjamin M Asher, Christopher M Collins, Michael W Gaultois, George R Darling, Matthew S Dyer, and Matthew J Rosseinsky. Inferring energy–composition relationships with bayesian optimization enhances exploration of inorganic materials. *The Journal of Chemical Physics*, 160(5), 2024. (Cited on pg. 112.)

[198] Xingchen Wan, Henry Kenlay, Robin Ru, Arno Blaas, Michael A Osborne, and Xiaowen Dong. Adversarial Attacks on Graph Classifiers via Bayesian Optimisation. In *Advances in Neural Information Processing Systems*, volume 34, 2021. (Cited on pg. 28.)

[199] Amanda Wang, Ryan Kingsbury, Matthew McDermott, Matthew Horton, Anubhav Jain, Shyue Ping Ong, Shyam Dwaraknath, and Kristin A Persson. A framework for quantifying uncertainty in dft energy corrections. *Scientific reports*, 11(1):15496, 2021. (Cited on pg. 125.)

[200] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search. In *Advances in Neural Information Processing Systems*, volume 33, 2020. (Cited on pg. 128.)

[201] Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Active model learning and diverse action sampling for task and motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4107–4114, 2018. (Cited on pg. 60.)

[202] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched Large-scale Bayesian Optimization in High-dimensional Spaces. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018. (Cited on pg. 25.)

[203] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Computational Materials*, 2(1):1–7, 2016. (Cited on pgs. 45 and 77.)

[204] Manfred K Warmuth, Jun Liao, Gunnar Rätsch, Michael Mathieson, Santosh Putta, and Christian Lemmen. Active Learning with Support Vector Machines in the Drug Discovery Process. *Journal of Chemical Information and Computer Sciences*, 43(2): 667–673, 2003. (Cited on pgs. 42, 59, and 75.)

[205] Manfred KK Warmuth, Gunnar Rätsch, Michael Mathieson, Jun Liao, and Christian Lemmen. Active learning in the drug discovery process. In *Advances in Neural Information Processing Systems 15*, pages 1449–1456, 2002. (Cited on pgs. 42, 59, and 75.)

[206] David Weininger. SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, 1988. (Cited on pg. 85.)

[207] Mingjian Wen, Evan Walter Clark Spotte-Smith, Samuel M Blau, Matthew J McDermott, Aditi S Krishnapriyan, and Kristin A Persson. Chemical reaction networks and opportunities for machine learning. *Nature Computational Science*, 3(1):12–24, 2023. (Cited on pg. 110.)

[208] Peter Willett, John M Barnard, and Geoffrey M Downs. Chemical Similarity Searching. *J. Chem. Inf. Comput. Sci.*, 38(6):983–996, 1998. (Cited on pg. 87.)

[209] Hywel D Williams, Natalie L Trevaskis, Susan A Charman, Ravi M Shanker, William N Charman, Colin W Pouton, and Christopher JH Porter. Strategies to address low drug solubility in discovery and development. *Pharmacological reviews*, 65(1):315–499, 2013. (Cited on pg. 110.)

[210] Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical Multi-Fidelity Bayesian Optimization for Hyperparameter Tuning. In *Proceedings*

of the 36th Conference on Uncertainty in Artificial Intelligence, pages 788–798, 2020. (Cited on pg. 42.)

[211] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint*, 2017. arXiv:1708.07747 [cs.LG]. (Cited on pgs. 61, 77, and 82.)

[212] Ziping Xu, Eunjae Shim, Ambuj Tewari, and Paul Zimmerman. Adaptive Sampling for Discovery. In *Advances in Neural Information Processing Systems 35*, pages 1114–1126, 2022. (Cited on pgs. 75 and 77.)

[213] Fei Xue, Yongjun Li, Yijia Gu, Jinxing Zhang, and Long-Qing Chen. Strain phase separation: Formation of ferroelastic domain structures. *Physical Review B*, 94(22): 220101, 2016. (Cited on pg. 110.)

[214] Fei Xue, Yanzhou Ji, and Long-Qing Chen. Theory of strain phase separation and strain spinodal: Applications to ferroelastic and ferroelectric systems. *Acta Materialia*, 133:147–159, 2017. (Cited on pg. 110.)

[215] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. Multi-Class Active Learning by Uncertainty Sampling with Diversity Maximization. *International Journal of Computer Vision*, 113(2):113–127, 2015. (Cited on pg. 59.)

[216] MT Yin and Marvin L Cohen. Theory of static structural properties, crystal stability, and phase transformations: Application to si and ge. *Physical Review B*, 26(10):5668, 1982. (Cited on pgs. 110 and 116.)

[217] Fanghang Yu, Xudong Zhu, Xikai Wen, Zhigang Gui, Zeyu Li, Yulei Han, Tao Wu, Zhenyu Wang, Ziji Xiang, Zhenhua Qiao, Jianjun Ying, and Xianhui Chen. Pressure-induced dimensional crossover in a kagome superconductor. *Phys. Rev. Lett.*, 128: 077001, Feb 2022. doi: 10.1103/PhysRevLett.128.077001. URL `https://link.aps.org/doi/10.1103/PhysRevLett.128.077001`. (Cited on pg. 110.)

[218] Yan Zhao and Donald G Truhlar. The m06 suite of density functionals for main group thermochemistry, thermochemical kinetics, noncovalent interactions, excited states, and transition elements: Two new functionals and systematic testing of four m06-class functionals and 12 other functionals. *Theor. Chem. Acc.*, 120(1):215–241, 2008. (Cited on pg. 86.)

[219] Fedor Zhdanov. Diverse mini-batch Active Learning. *arXiv preprint*, 2019. arXiv:1901.05954 [cs.LG]. (Cited on pg. 60.)

[220] Kangkang Zhou, Kaihu Xian, Ruijie Ma, Junwei Liu, Mengyuan Gao, Saimeng Li, Tao Liu, Yu Chen, Yanhou Geng, and Long Ye. Correlating miscibility, mechanical parameters, and stability of ternary polymer blends for high-performance solar cells. *Energy & Environmental Science*, 16(11):5052–5064, 2023. (Cited on pg. 110.)

[221] Bilin Zhuang, Gabriele Ramanauskaite, Zhao Yuan Koa, and Zhen-Gang Wang. Like dissolves like: A first-principles theory for predicting liquid miscibility and mixture dielectric constant. *Science advances*, 7(7):eabe7275, 2021. (Cited on pg. 110.)

[222] Alex Zunger, S-H Wei, LG Ferreira, and James E Bernard. Special quasirandom structures. *Physical review letters*, 65(3):353, 1990. (Cited on pg. 124.)

Appendix A

Hardness of Multiclass Active Search with Diminishing Returns

We present the proof of Theorem 5.1.1. This is done by constructing a class of AS problems similar to those described in Jiang et al. [88] but with different parameterizations. Consider an AS problem whose setup is summarized in Fig. A.1 and described below. The problem has $n = 16^m$ points, $C = 2^{m+1} + 1$ classes (2^{m+1} positive classes), and the search budget is $T = 2^{m+1}$, where m is a free parameter. We also have the reward function for each class f_i to be the same (arbitrary) positive, increasing, unbounded, and concave f. Each of the n points is classified into two groups: "clumps" and "isolated points."

The former consists of 4^m clumps, each of size T, and is visualized in Fig. A.1b. All points within the same clump share the same label, and exactly one clump contains positives, each of which belongs to a different positive class. In each instance of the problem class being described, this positive clump is chosen uniformly at random among the 4^m clumps. As such, the prior marginal positive probability of any of these points is $p_{\text{clump}} = 4^{-m}$.

As for the isolated points, their labels are independent from one another. The positives among the isolated points only belong to a single positive class. The marginal probability of an isolated is set to be $p_{\text{isolated}} = 1 - 0.5^{\frac{2m^2}{2^m}}$. These isolated points are further separated into two categories:

- A secret set S of size $T/2 = 2^m$, visualized in Fig. A.1a, which encodes the location of the positive clump. The set S is first partitioned into $2m$ subsets $S_1, S_2, \ldots, S_{2m}$, each of size $2^m/2m$. Each subset S_i encodes one virtual bit b_i of information about the location of S, and is further split into m groups of $2^m/2m^2$ points, with each group encoding a virtual bit b_{ij} by a logical OR. The aforementioned virtual bit b_i, on the other hand, is obtained via a logical XOR: $b_i = b_{i1} \oplus b_{i1} \oplus \ldots \oplus b_{im}$.

- The remaining points, denoted as $\mathcal{R}$ and visualized in Fig. A.1c, are completely independent from each other and any other points. We have $|\mathcal{R}| = 16^m - 2(8^m) - 2^m$.

We first make the same two observations as in Jiang et al. [88].

Observation A.0.1. *At least m points from S_i need to be observed in order to infer one bit b_i of information about the location of the positive clump.*

Each b_{ij} has the same marginal probability of being 1:

$$\Pr(b_{ij} = 1) = 1 - \Pr(b_{ij} = 0) = 1 - (1 - p_{\text{isolated}})^{\frac{2^m}{2m^2}} = 0.5.$$

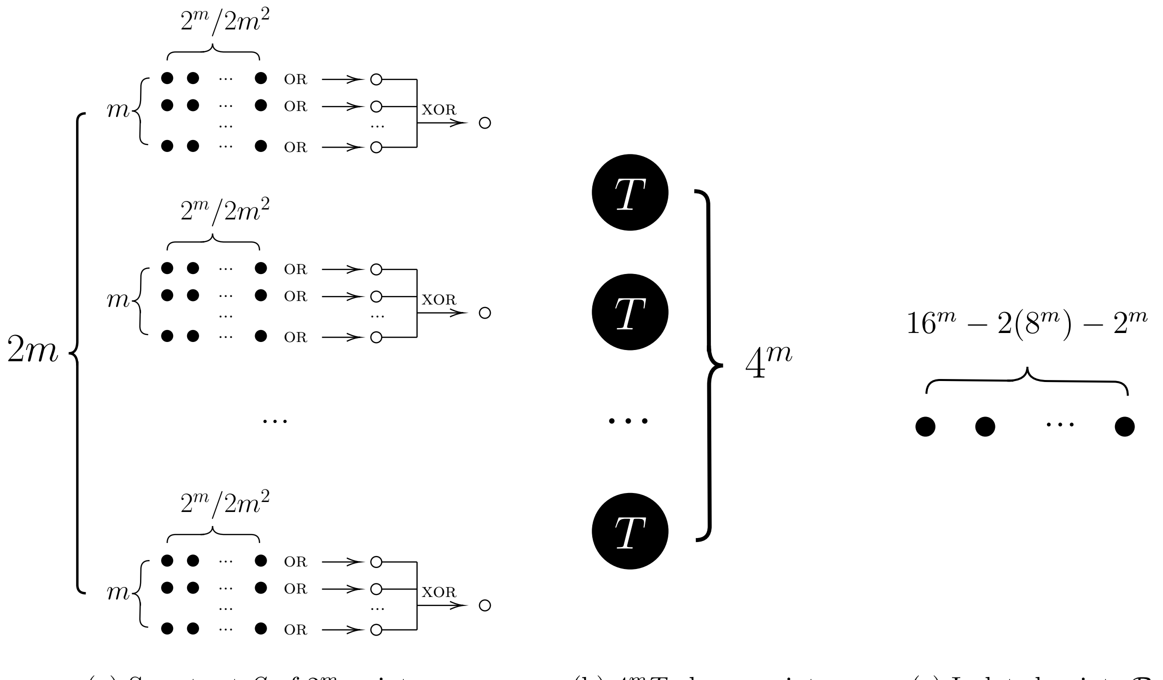
We also have $\Pr(b_i = 1) = 0.5$, as the positive clump is chosen uniformly at random. It is necessary to observe all virtual bits b_{ij} from the same group S_i to infer the bit b_i, since observing a fraction of the inputs of a XOR operator does not change the marginal belief about the output b_i. So, observing $(m-1)$ or fewer points conveys no information about the positive clump.

Observation A.0.2. *Observing any number of clump points does not change the marginal probability of any point in the secret set S.*

The knowledge of b_i does not change the marginal probability of any b_{ij}. This is to say no point in S will have a different probability after observing b_i. This means observing points outside of S does not help distinguish S from the remaining isolated points in $\mathcal{R}$.

With this setup, we now compare the performance of the optimal policy and the expected performance of a given polynomial-time policy. To this end, we first consider the optimal policy with unlimited compute. Before querying any point, the policy computes the marginal probability of an arbitrary fixed clump point, conditioning on observing every possible subset of the isolated points of size m and fantasized positive labels. This set of $O(n^m)$ inference calls will reveal the location of the secret set S, as only points in S will update the probabilities of the fixed clump point.

Now the policy spends the first half of its budget querying S, the labels of which identify the positive clump. The policy now spends the second half of the budget collecting these positive points. The resulting reward, denoted as OPT, is lower-bounded in the worst-case



(a) Secret set S of 2^m points (b) $4^m T$ clump points (c) Isolated points $\mathcal{R}$

Figure A.1: An instance of AS where any efficient algorithm can be arbitrarily worse than the optimal policy.

scenario where S does not contain any positive point:

$$\textsc{opt} \geq \frac{T}{2}\, f(1) = 2^m\, f(1).$$

We now consider a policy $\mathcal{A}$. Let α denote the total number of inference calls performed by $\mathcal{A}$ throughout its run. At the i^{th} inference call, $\mathcal{A}$ uses a training set $\mathcal{D}_i$ of size at most $T = 2^{m+1}$. We will show that $\mathcal{A}$ has a very small chance of collecting a large reward by considering several cases.

We first examine the probability that $\mathcal{A}$ finds the secret set S. By Theorems A.0.1 and A.0.2, $\mathcal{A}$ cannot differentiate between the points in S and those in $\mathcal{R}$ unless $|\mathcal{D}_i \cap S| \geq m$. Suppose that before this inference call, the algorithm has no information about S (which is always true when $i = 1$). The chances of $\mathcal{A}$ choosing $\mathcal{D}_i$ such that $|\mathcal{D}_i \cap S| \geq m$ are no better than a random selection from $n - 2\,(8^m)$ isolated points. We can upper-bound the probability of this event by counting how many subsets of size 2^{m+1} would contain at least m points from

S among all subsets of the $n - 2\left(8^m\right)$ isolated points:

$$\Pr\left(|\mathcal{D}_i \cap S| \geq m\right) \leq \frac{\binom{2^m}{m}\binom{n-2(8^m)-m}{2^{m+1}-m}}{\binom{n-2(8^m)}{2^{m+1}}}.$$

The RHS may further be upper-bounded by considering

$$\frac{\binom{2^m}{m}\binom{n-2(8^m)-m}{2^{m+1}-m}}{\binom{n-2(8^m)}{2^{m+1}}} = \frac{(2^m)!\ (n-2(8^m)-m)!\ (2^{m+1})!}{m!\ (2^m-m)!\ (2^{m+1}-m)!\ (n-2(8^m))!},$$

where

$$\frac{(2^m)!}{(2^m-m)!} < (2^m)^m,$$

$$\frac{(2^{m+1})!}{(2^{m+1}-m)!} < (2^{m+1})^m,$$

$$\frac{(n-2(8^m)-m)!}{m!\ (n-2(8^m))!} < \frac{1}{(n-2(8^m))^m}.$$

The last inequality is due to

$$\frac{(n-2(8^m))^m}{m!} < \frac{(n-2(8^m))!}{(n-2(8^m)-m)!},$$

which is true by observing that for each of the m factors on each side,

$$\frac{n-2(8^m)}{i} < n-2(8^m)-i+1, \forall i = 1,\ldots,m.$$

Overall, we upper-bound the probability that $\mathcal{A}$ hits m points in S with

$$\Pr(|\mathcal{D}_i \cap S| \geq m) < \left(\frac{2^m\ 2^{m+1}}{n-2(8^m)}\right)^m,$$

and union-bound the probability of $\mathcal{A}$ "hitting" the secret set after α inferences, denoted as p_{hit}, with

$$p_{\text{hit}} < \frac{\alpha}{\binom{\frac{n-2(8^m)}{2^m}}{2^{m+1}}^m}.$$

Here, $n - 2(8^m) = 16^m - 2(8^m) = \Theta(16^m)$, so

$$p_{\text{hit}} < \frac{\alpha}{\Theta\left(\left(\frac{16^m}{2(4^m)}\right)^m\right)} = \frac{\alpha}{\Theta\left(4^{m^2}\right)}.$$

Hence, for any $\alpha = O(n^c) = O(16^{cm}) = O(4^{2cm})$, where c is a constant,

$$p_{\text{hit}} < O\left(\frac{4^{2cm}}{4^{m^2}}\right) = O(4^{-m^2}) = O(4^{-\log^2 n}).$$

In other words, the probability that $\mathcal{A}$ does find the secret set S decreases as a function of n. Conditioned on this event, we upper-bound its performance with $2^{m+1} f(1)$, assuming that every query is a hit.

On the other hand, if $\mathcal{A}$ never finds S, we further consider the following subcases: if the algorithm queries an isolated point, no marginal probability is changed; if a clump point is queried, only the marginal probabilities of the points in the same clump are updated. The expected performance in these two cases can be upper-bounded by pretending that the algorithm had a budget of size $2T = 2^{m+2}$, half of which is spent on querying isolated points and half on clump points.

The expected utility after T queries on isolated points is $\mathbb{E}\big[f(X)\big]$, where $X = \sum_{i=1}^{T} X_i$ and $\Pr(X_i = 1) = p_{\text{isolated}}$. We further upper-bound this expectation using Jensen's inequality:

$$\mathbb{E}\big[f(X)\big] < f\big(\mathbb{E}[X]\big) = f\big(T\ p_{\text{isolated}}\big) = f\left(2^{m+1}(1 - 2^{-\frac{2^m}{2m^2}})\right).$$

The expected utility after T queries on clump points is

$$\begin{aligned}\frac{f(T)}{4^m}+\left(1-\frac{1}{4^m}\right)\left(\frac{f(T-1)}{4^m-1}+\left(1-\frac{1}{4^m-1}\right)(\cdots)\right) &= \frac{\sum_{i=1}^{T} f(i)}{4^m}\\ &< \frac{T f(T)}{4^m}\\ &= \frac{f(2^{m+1})}{2^{m-1}}.\end{aligned}$$

Combining the two subcases, we have the expected utility in the case where $\mathcal{A}$ never hits S upper-bounded by

$$f\left(2^{m+1}(1-2^{-\frac{2^m}{2m^2}})\right)+\frac{f(2^{m+1})}{2^{m-1}}.$$

With that, the overall expected utility of $\mathcal{A}$, denoted by $E_{\mathcal{A}}$ is upper-bounded by

$$E_{\mathcal{A}} < 2^{m+1}\, f(1)\, p_{\text{hit}} + f\left(2^{m+1}(1-2^{-\frac{2^m}{2m^2}})\right)+\frac{f(2^{m+1})}{2^{m-1}}.$$

We then consider the upper bound of the approximation ratio

$$\frac{E_{\mathcal{A}}}{\text{OPT}} < 2\, p_{\text{hit}} + \frac{f\left(2^{m+1}(1-2^{-\frac{2^m}{2m^2}})\right)}{2^m\, f(1)} + \frac{(2^{m+1}-1)}{4^m}.$$

The first and third terms are arbitrarily small with increasing m. As for the second term, L'Hôpital's rule shows that $1-2^{-\frac{2^m}{2m^2}}=\Theta\left(\frac{2m^2}{2^m}\right)$, so this term scales like $\Theta\left(\frac{f(4m^2)}{2^m}\right)$, which is $O\left(\frac{4m^2}{2^m}\right)$ and also arbitrarily small with increasing m. As a result, algorithm $\mathcal{A}$ cannot approximate the optimal policy by a constant factor.

Appendix B

Generation of Training Search Problems

We now discuss our procedure of generating active search problems to train the policy network in DAgger, summarized in Alg. 7, where $U\{m, n\}$ denotes a discrete uniform distribution of the integers between m and n (inclusive), while $U[a, b]$ refers to a continuous uniform distribution between a and b.

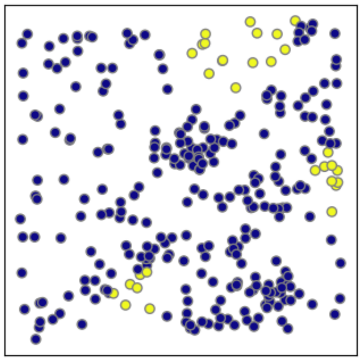


Figure B.1: An example two-dimensional problem generated by Alg. 7, where bright and dark points indicate targets and non-targets, respectively. The search space includes both clusters and more widely dispersed points.

The search space $\mathcal{X}$ of each generated problem exists in a d-dimensional space, where d is a random integer between 2 and 10. Once d is determined, we sample $100d$ points uniformly from the d-dimensional unit hypercube. This set of uniform points is combined with n_{cluster} clusters, where n_{cluster} is a random integer between 10 and $10d$. To generate each cluster, we first sample another random integer between 10 and $10d$, denoted as m, to determine the size of the cluster. We then draw m points from an isotropic Gaussian distribution with the mean vector $\boldsymbol{\mu}$ randomly sampled within the unit hypercube and the diagonal covariance matrix $\sigma^2 I_d$, where σ is drawn from $U[0.1, 0.1d]$. Here, σ, which determines the spread of a given cluster, is constrained to be between 0.1 and $0.1d$ (relatively small numbers) to ensure that the points within this cluster are indeed close to one another.

Again, the union of the uniform points and the clusters make up the entire search space $\mathcal{X}$. We then draw a sample from a Gaussian process (GP) at the points in $\mathcal{X}$. This GP is equipped with a zero mean function and a radial basis function kernel whose length scale ℓ scales linearly with the dimensionality of the space d by a factor of 0.05. This GP sample

Algorithm 7 Generate synthetic search problems

```
1: $d \sim U\{2, 10\}$                                                    ▷ sample dimensionality of search space
2: $\mathcal{X} \leftarrow \big\{x_j : x_j \sim U[0,1]\big\}_{j=1}^{100d}$                          ▷ sample uniform points
3: $n_{\text{cluster}} \sim U\{10, 10d\}$                                       ▷ sample number of clusters
4: for $i = 1$ to $n_{\text{cluster}}$ do
5:     $m \sim U\{10, 10d\}$                                                  ▷ sample cluster size
6:     $\boldsymbol{\mu} = [\mu_j]_{j=1}^{d}$, where $\mu_j \sim U[0,1]$                     ▷ sample cluster center
7:     $\sigma \sim U[0.1, 0.1d]$                                           ▷ sample spread of cluster
8:     $\mathcal{X} \leftarrow \mathcal{X} \cup \big\{x_j : x_j \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 I_d)\big\}_{j=1}^{m}$      ▷ use an isotropic Gaussian distribution
9: end for
10: $\mathbf{f} \sim \mathcal{N}(\mathcal{X}; \mathbf{0}, \boldsymbol{\Sigma})$,
    where $\boldsymbol{\Sigma} = K(\mathcal{X}, \mathcal{X})$ and $K(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\ell^2}\right)$ with length scale $\ell = 0.05d$
11: $p \sim U[0.01, 0.2]$                                            ▷ sample target prevalence
12: $\mathbf{y} = \mathbb{I}\big[\mathbf{f} > 100(1-p)\text{-th quantile in } \mathbf{f}\big]$          ▷ threshold to construct binary labels
13: returns $(\mathcal{X}, \mathbf{y})$
```

yields a vector **f** of real-valued numbers. We then sample uniformly between 0.01 and 0.2 for a prevalence rate p, which determines the proportion of $\mathcal{X}$ corresponds to the targets. As such, we compute the binary labels $\mathbf{y}$ by thresholding $\mathbf{f}$ at the $100(1-p)$-th quantile of the values in $\mathbf{f}$. We keep the prevalence rate p below 20% to ensure that the targets are sufficiently rare. The tuple $(\mathcal{X}, \mathbf{y})$ is finally returned. Fig. B.1 shows an example of one such generated problem in two dimensions, showing a search space with a considerably complex structure with multiple clusters and groups of targets.