WASHINGTON UNIVERSITY IN ST. LOUIS

McKelvey School of Engineering
Department of Computer Science and Engineering

Dissertation Examination Committee:
Roman Garnett, Chair
Ayan Chakrabarti
Sanmay Das
David Duvenaud
Brendan Juba

Efficient Nonmyopic Sequential Experimental Design

by
Shali Jiang

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

August 2020
Saint Louis, Missouri

Contents

List of Tables . . . . . **v**

List of Figures . . . . . **vii**

Acknowledgments . . . . . **x**

Abstract . . . . . **xiii**

1 Introduction **1**
1.1 Declaration of Previous Publications 5

2 Sequential Experimental Design Overview **9**
2.1 Problem Formulation 9
2.2 Bayesian Decision Theory 12
2.3 Bellman Equation 14
2.4 Related Research Literature 17
2.4.1 Markov Decision Process 17
2.4.2 Reinforcement Learning 18
2.4.3 Active Learning 19

3 Budgeted Active Search **20**
3.1 Problem Definition 22
3.2 Bayesian Optimal Policy 22
3.3 Hardness of Approximation 25
3.4 Efficient Nonmyopic Approximation 26
3.4.1 Nonmyopic Behavior 28
3.5 Implementation 29
3.5.1 k Nearest Neighbors 29
3.5.2 Time Complexity 31
3.5.3 Pruning the Search Space 32
3.6 Related Work 33
3.7 Experiments 35
3.7.1 Finding NeurIPS Papers From CiteSeer^x 37
3.7.2 Finding Bulk Metallic Glasses 39
3.7.3 Drug Discovery 39

3.7.4 Compare with Naive Exploration/Exploitation Approaches 42
3.7.5 Compare with UCB-Style Policy 44
3.7.6 Pruning Effectiveness 45
3.8 Conclusion 46

4 Batch Budgeted Active Search 48
4.1 Bayesian Optimal Policy 50
4.2 Adaptivity Gap 52
4.3 Efficient Nonmyopic Approximations 53
4.3.1 Sequential Simulation 55
4.3.2 Greedy Approximation 59
4.3.3 Lazy Evaluation for Pruning 60
4.4 Related Work 62
4.5 Experiments 63
4.5.1 Finding NeurIPS Papers From CiteSeer^x 64
4.5.2 Finding Bulk Metallic Glasses 65
4.5.3 Drug discovery 67
4.5.4 Discussion 69
4.5.5 Pruning Effectiveness 73
4.6 Conclusion 74

5 Cost Effective Active Search 75
5.1 Related Work 77
5.2 Bayesian Optimal Policy 78
5.3 Hardness of Cost Effective Active Search 81
5.4 Efficient Nonmyopic Approximation 82
5.4.1 Negative Poisson Binomial Distribution 83
5.4.2 Approximation of the NPB expectation 84
5.4.3 Approximation of Expected Cost 88
5.4.4 Lazy Evaluation for Pruning 89
5.5 Experiments 90
5.5.1 Finding Bulk Metallic Glasses 91
5.5.2 Drug Discovery 91
5.5.3 Pruning Effectiveness 93
5.6 Conclusion 94

6 Bayesian Optimization and Beyond 95
6.1 Background on Gaussian Processes 96
6.2 BINOCULARS 98
6.2.1 Intuition 98
6.2.2 Non-Adaptive Utility as a Lower Bound of Adaptive Utility 99
6.2.3 Efficient Nonmyopic Approximation Framework 101

6.3 BINOCULARS for Bayesian Optimization . . . 102
6.4 BINOCULARS for Bayesian Quadrature . . . 105
6.5 Practical Considerations . . . 106
6.6 Related Work . . . 107
6.7 Experiments . . . 109
6.7.1 BO Results . . . 111
6.7.2 BQ Results . . . 115
6.8 Multi-Step Lookahead via One-Shot Optimization . . . 117
6.8.1 Previous Work on Two-Step Lookahead BO . . . 117
6.8.2 One-Shot Approach . . . 118
6.8.3 Preliminary Results . . . 119
6.9 Conclusion . . . 121

7 Conclusion and Future Work . . . 122
7.1 Theoretical Guarantees . . . 123
7.2 Learning to Search . . . 124
7.3 Multi-Fidelity Active Search . . . 128

Appendix A Hardness of Budgeted Active Search . . . 130

Appendix B Adaptivity Gap of Batch Budgeted Active Search . . . 137

Appendix C Hardness of Cost Effective Active Search . . . 147

Appendix D Additional Results . . . 152
D.1 Cost Effective Active Search . . . 152
D.2 Bayesian Optimization . . . 152

References . . . 157

List of Tables

3.1 CiteSeer^x (left) and BMG (right) data: Average number of targets found by the one- and two-step myopic policies and ENS with different five budgets, varying from 100 to 900, at specific time steps. The performance of the best method at each time waypoint is in bold. 36

3.2 Number of active compounds found by various active search policies at termination for each fingerprint, averaged over 120 active classes and 20 experiments. Also shown is the difference of performance between ENS and two-step lookahead and the results of the corresponding paired t-test. 39

3.3 Average number of pruned points in each iteration for the two chemical datasets. 46

4.1 Results for CiteSeer^x data: Average number of targets found by various batch policies: greedy-batch, sequential simulation "ss-P-O" and batch-ENS, with batch sizes 5, 10, 15, 20, 25. The average is taken over 100 experiments. Highlighted are the best in each column and those not significantly worse than the best using a one-sided paired t test with significance level $\alpha = 0.05$. . . . 65

4.2 Diversity scores of the chosen batches by all our proposed policies, measured by the average rank of distances from each other in a batch, produced from the results on CiteSeer^x data. Higher value indicates more diversity. 66

4.3 Results for BMG data: average number of targets found by various batch policies: baseline greedy-batch, sequential simulation "ss-P-O" and batch-ENS, with batch sizes 5, 10, 15, 20, 25. The average is taken over 30 experiments. Highlighted are the best in each column and those that are not significantly worse than the best using a one-sided paired t test with significance level $\alpha = 0.05$. 67

4.4 Results for drug discovery data: Average number of positive compounds found by the baseline *uncertain-greedy* batch, greedy-batch, sequential simulation and batch-ENS policies. Each column corresponds to a batch size, and each row a policy. Each entry is an average over 200 experiments (10 datasets by 20 experiments). The budget T is 500. Highlighted are the best (bold) for each batch size and those that are not significantly worse (blue italic) than the best under one-sided paired t-tests with significance level $\alpha = 0.05$. . . . 68

4.5 Diversity scores of the chosen batches by all sequential simulation policies, measured by the average rank of distances from each other in a batch. The results are for $b = 20$ on CiteSeer^x data. Higher values indicate more diversity. For reference, the score for greedy and batch-ENS are 2443 and 3126, respectively. 72

4.6 Results for pruning effectiveness. The numbers are averaged over all iterations of batch-ENS for all batch sizes tested. For drug discovery data, the result is averaged over batch-ENS-16 and batch-ENS-32. 74

5.1 Time cost and quality of various approximations of the expectation of NPB distribution. 87
5.2 Results on materials discovery data. Averaged over 30 experiments. 91
5.3 Averaged results over 30 repetitions and nine drug discovery datasets. 92
5.4 Average pruning rate across all iterations in all experiments for the reported ENCES policies. 94

6.1 Average GAP over 100 repeats on "hard" synthetic functions. 112
6.2 Average GAP over 50 repeats on real functions. 113
6.3 Median fractional error values over 100 repeats on all BQ functions. 116

D.1 Results for all tested policies for the materials discovery dataset. 153
D.2 Results for all tested policies for the nine drug discovery datasets. 154
D.3 Average GAP of 30 repeats on all 31 synthetic functions. 156
D.4 Average GAP of 50 repeats on real functions for all q.EI variants. Note function names are shortcuts for better spacing. 156

List of Figures

3.1 Kernel density estimates of the distribution of points chosen by ENS (top) and 2-step lookahead (bottom) during two different time intervals. The figures on the left show the kernel density estimates for the first 100 locations; the figures on the right, the last 100 chosen locations. 29
3.2 The learning curve of our policy and other baselines on the NeurIPS dataset. 37
3.3 The average difference in cumulative targets found between ENS and the two-step policy, averaged over 120 activity classes and 20 experiments on (a) ECFP4 and (b) GpiDAPH3 fingerprint. 41
3.4 Comparion of active search policies with a naive exploration-exploitation approach called uncertain-then-greedy (UTG), which performs uncertainty sample for the first 100r% of the budget and then greedy sampling in the remaining iterations, where r is a hyperparameter controlling the transition point. (a) CiteSeer^x Dataset. (b) BMGs dataset. (c) Drug discovery datasets. 43
3.5 Number of targets found by the UCB-style policy (3.11), as a function of the hyperparameter p^* as derived in (3.12), averaged over 20 experiments. Note for CiteSeer^x and BMG datasets, the grid size of p^* is 0.01, but for ECFP4, we can only afford grid size of 0.1. To put these results into perspective, we also show the two-step performances by the red horizontal line, indicating two-step performs better than the UCB-style policy on all three domains. All these results are with budget 500. 45

4.1 Illustration of pruning. The x-axis is the index of candidate points in descending order of the upper bounds, and the y-axis is the actual marginal gain of batch-ENS score as in Eq. (6) in the main text. These plots are generated from running batch-ENS on the CiteSeer^x data with budget $T = 500$ and batch size $b = 5$. There are 39 788 points, we only plot the first and last 1 000 points to have a better presentation. (a) At the time of choosing the first point ($k = 1$) of the 99th batch (i.e., $i = 98$) when $T - b - |\mathcal{D}_i| = 5$; here 99.61% of points are pruned. (b) At the time of choosing the first point ($k = 1$) of the 16th (i.e., $i = 15$) batch when $T - b - |\mathcal{D}_i| = 420$; 98.23% of the points are pruned. . . 60
4.2 Number of targets found versus number of samples used for batch-ENS. This is averaged over the results for batch size 50 on 10 drug discovery datasets and 20 experiments each. The text labels show the percentage of improvement and p-value of one-sided t tests comparing against previous numbers, e.g., 8 samples improves over 4 samples by 1.9%, and the p-value is 0.04. 69

4.3 (a) Average performance ratio between sequential policies and batch policies, as a function of batch size, produced using averaged results in Table 4.4. (b) Same plots produced using averaged results (excluding uncertain-greedy) for the CiteSeer dataset (Table 4.1). 70
4.4 Progressive probabilities of the chosen points of greedy and batch-ENS-32, averaged over results for batch size 50 on all 10 drug discovery datasets and 20 experiments each. 73

5.1 Illustration of a probability vector $[p_1, p_2, \ldots, p_n]$ and the corresponding probability mass functions of NPB distribution for different r. Left: the top 1500 posterior marginal probabilities after conditioning on 100 positive and 100 negative points (randomly selected); probabilities are computed using a k-nn model (with $k = 50$) on the CiteSeer^x dataset; middle: $r = 50$; right: $r = 200$. 85
5.2 Approximation errors of various approximation methods for computing the expectation of NPB distribution. y-axis is approximate $\mathbb{E}[m]$ minus exact $\mathbb{E}[m]$. The error for RECIP drops below -10 after about $r = 250$, hence not shown to zoom in the interesting part of the figure. 87
5.3 Average cost versus the number of positives found, averaged over 9 drug discovery datasets. 93

6.1 An illustration of our proposed nonmyopic method applied to BO. (a) A function in $[-1, 1]$ drawn from a GP where the two end points are known to be zero. (b) and (c) show two iterations of BO with the EI acquisition function. (d) EI, 2-EI and 2-step-EI curves with their respective maximizers. (e) and (f) show two iterations of BO where the first point is chosen from the two points maximizing 2-EI, and the second one is chosen by maximizing EI (conditioned on the observation in iteration one). 99
6.2 Average GAP over nine synthetic functions demonstrating the nonmyopic behavior of 12.EI.s. 112
6.3 mean GAP with error bars at termination versus time per iteration (in log scale) averaged over the seven real functions. 114
6.4 Median fractional error over 100 repeats against iterations or time. (a) synthetic functions, (b) real functions, (c) real functions. 116
6.5 Preliminary results of multi-step lookahead Bayesian optimization via one-shot optimization. 120

A.1 An instance of active search where any efficient algorithm can be arbitrarily worse than an optimal policy. 131

B.1 An illustrative example of the constructed instance with $h = 3$ and $2^h = 8$ clumps, where the third clump from the left is positive, corresponding to the correct path, 010. 139

B.2 Illustration of the relevant subtree. The red curve shows the correct path identified so far. If the last node on the correct path is negative, then the node A must also be on the correct path, and the subtree rooted at A is the relevant subtree; otherwise B is on the correct path, and the subtree rooted at B is the relevant subtree. 144

C.1 An instance of active search where any efficient algorithm can be arbitrarily worse than an optimal policy. 148

D.1 Average cost versus the number of positives found for 9 drug discovery datasets. The total number of positives are 553, 378, 506, 1023, 218, 916, 1024, 431, 255, respectively. 155

Acknowledgments

My deepest gratitude goes to my advisor, Roman Garnett. He is among the brightest minds I know. It has been an absolute pleasure working with him. He always gives me the freedom on what to work on but at the same time provides helpful guidance so that I would not get lost. He always knows when to step in so that I would not struggle too much and when to step out so that I get the most out of the experience. I admire him as a wise and knowledgeable professor, and at the same time I can comfortably talk to him as a friend. I enjoyed a lot hanging out with him when attending conferences in Sydney, Montreal, Vancouver etc., and through that I got to know a lot of fellow researchers from around the globe, which helped shape some of the ideas in this thesis.

Thanks to all my collaborators for their contributions. Specially, I would like to thank Benjamin Moseley for all his help on the active search projects. Whenever I struggle with a hard proof I go to Ben, and he would always amaze me with his incredible mathematical intuition and versatile proving techniques. This thesis (especially the theoretical part) would not be possible without him. I would also like to thank my labmate Gustavo Malkomes, who is like a big brother, always there to help, professionally or otherwise. I especially thank him for his help with my first publication on active search. This was a very important milestone for my Ph.D. I would also like to thank my labmate Henry Chai. He is a wonderful teacher and presenter, and always gives great suggestions on my presentations. He is also the first person I go to for English questions. Also, it was a great pleasure working with him on the BINOCULARS project.

Thanks to my committee members, David Duvenaud, Sanmay Das, Brendan Juba, and Ayan Chakrabarti, for asking insightful questions and providing valuable suggestions. Thanks to Prof. Garnett, Das, Juba, and Neumann for teaching the wonderful classes on artificial intelligence, machine learning and multi-agent systems, and also for the inspiring discussions in machine learning seminars.

I would also like to thank my office mates for keeping our office a pleasant working environment and many fun hangouts. Special thanks to Zhuoshu for helping me with intern job search and "dragging" me to the gym, among other things. Special thanks to my classmate/friend/collaborator Muhan Zhang for giving me highly motivational peer pressure by being so hard-working and productive. Thanks to my "hotpot friends" and many others not listed here for the fun times we had together.

Last but not least, I thank my parents for their unconditional love. No matter how far I travel, it gives me peace and strength knowing that there is always a place called home that I can go back to. Thanks to my dear sisters for posting moments of my little nieces. Nothing makes me more relaxed than watching them playing and listening to their talking and laughing (or sometimes hilarious crying). Finally, I would like to thank my beloved fiancée Haipeng Dai. It is her constant understanding, support, encouragement and love that helped me through the most difficult challenges.

Shali Jiang

Washington University in Saint Louis
August 2020

Dedicated to my fiancée and my parents.

ABSTRACT OF THE DISSERTATION

Efficient Nonmyopic Sequential Experimental Design

by

Shali Jiang

Doctor of Philosophy in Computer Science

Washington University in St. Louis, August 2020

Research Advisor: Professor Roman Garnett

Sequential experimental design (SED) problems are abundant in science and engineering. Examples include drug discovery, materials discovery, or machine learning hyperparameter tuning. Such problems can be formulated as sequentially choosing locations to evaluate an expensive black-box function to maximize some notion of utility. The optimal policy under Bayesian decision theory maximizes the expected utility, marginalizing over all future uncertainties in the decision horizon. However, this is usually computationally intractable and needs to be approximated. Existing approximations are often myopic, meaning the lookahead horizon is limited to only very few steps into the future. Though computationally efficient, they often suffer from poor balance between *exploration and exploitation*. In this thesis, we attempt to bridge this gap by designing policies that are both *efficient and nonmyopic*.

The main idea is to *use the non-adaptive expected utility to approximate the adaptive expected utility*. This allows us to efficiently optimize a tighter lower bound of the true expected utility than myopic approximations. This idea is instantiated on a problem known as *active search*,

where we sequentially evaluate items from a large candidate pool to search for targets with a desired property, assumed to be rare. Various settings of active search are considered.

We begin with the budgeted setting, where the goal is to identify maximum number of targets in a given number of iterations. We establish the hardness of the problem, and propose an *efficient nonmyopic search* (ENS) policy by approximating the future adaptive utility after the current step by its non-adaptive counterpart. We then extend to the batch setting, where multiple items are evaluated simultaneously. We prove that, given the same total number of evaluations, the expected utility of the optimal batch policy is worse than that of the optimal sequential policy by at least a linear factor of batch size. ENS can be naturally generalized to the batch setting, but a batch policy requires combinatorial search, for which we propose several effective heuristic solutions such as sequential simulation and greedy selection.

We also study *cost effective active search*, where the goal is to find a given number of targets with minimum number of iterations. We also establish the hardness of this setting. ENS here boils down to approximating the expectation of a *negative Poisson binomial distribution*. In all active search settings, superior performance of our proposed policies is demonstrated on drug and materials discovery, and a desirable nonmyopic exploration/exploitation behavior is often observed.

Finally, we adapt this idea to general continuous SED settings. We first compute an optimal batch (i.e., the non-adaptive solution), then select one point from the batch. We show this procedure maximizes a looser lower bound than ENS, but still much tighter than myopic one-step approximations. We realize this idea for both Bayesian optimization and Bayesian quadrature and demonstrate superior performance on the two drastically different tasks. Finally, we briefly discuss the idea of one-shot optimization for efficient multi-step lookahead Bayesian optimization and show promising preliminary results.

Chapter 1

Introduction

Sequential experimental design (SED) problems are abundant in science and engineering. In scientific discovery, researchers often adaptively design new experiments based on the outcomes of past ones. An example is drug discovery, where one needs to sequentially examine new chemical compounds to find the targets with desired properties. The question is which compounds to examine next so that more targets are identified within a limited budget. In algorithmic engineering, there are often many potential configurations when tuning an algorithm. For example, deep learning has been extremely successful in many domains in recent years, but applying such technologies requires careful design of the network structure and tuning of the hyperparameters such as learning rate, number of hidden nodes, etc.; this can be formulated as an SED problem where we adaptively choose new configurations to evaluate in order to find a satisfactory one. The question is which configuration to evaluate next given limited time and computational resources. In numerical computation, we often encounter analytically intractable integrals with an expensive-to-evaluate integrand; for example when computing the partition function in Bayesian statistics, the integrand is a likelihood function defined by a potentially large data sample. Such a fundamental problem can also be formulated as an SED problem, where we adaptively choose locations to evaluate

the integrand so as to minimize the number of samples required to get an accurate estimate of the integral. The question is, again, which location to evaluate next? In all the examples mentioned above, one common characteristic is the experiments are typically expensive, in terms of monetary, computational, or time cost. Thus, it is of great importance to intelligently choose what locations to evaluate.

In this thesis, we define SED as a paradigm in which we sequentially evaluate an expensive black-box function in order to achieve a certain goal. The key question is: where to evaluate next in order to achieve the goal given limited resources?

We approach this problem with Bayesian decision theory, a framework for decision making under uncertainty. We model the function in a Bayesian fashion, and define a utility function to express preference over different designs and outcomes. The Bayesian optimal policy maximizes the expected utility in each iteration, which marginalizes over all future uncertainties in the decision horizon. The expected utility can be derived in the form of Bellman equation, which is computationally intractable for the problems of interest. Existing work mostly adopts myopic approaches where the decision horizon is limited to only a few steps (e.g., one or two), ignoring the long-term impact beyond the short horizon. Such myopic policies are widely used in practice due to their computational efficiency, but their poor balance of exploration and exploitation often results in suboptimal performance. This thesis attempts to bridge this gap by designing policies that are both *efficient and nonmyopic*.

The main idea behind most of our policies can be summarized in a single sentence: *use the non-adaptive expected utility to approximate the adaptive expected utility*. This simple idea allows us compute a tighter lower bound of the true adaptive expected utility, retaining the nonmyopia of the optimal policy to a much higher degree than myopic approximations, yet remaining computationally efficient.

We implement variants of this idea for several SED problems. The main problem we study is called *active search*, where we sequentially evaluate items from a large candidate pool to search for targets with a desired property, assumed to be extremely rare. Typical examples include drug and materials discovery. This can be seen as an SED problem with the underlying function being binary-valued with a finite domain, and the goal is to identify positive points. Two variants of active search are studied: budgeted setting and a novel cost effective setting.

In budgeted active search (Chapter 3), the goal is to find the maximum number of positive points in a given number of iterations. The idea of our *efficient nonmyopic search* (ENS) policy is to approximate the expected future adaptive utility after the current step by its non-adaptive counterpart. That is, *pretend* all future choices after the outcome of this step would be committed at once. We extend this idea to batch policies, where in each iteration multiple points are selected and evaluated in parallel (Chapter 4). Several heuristics including sequential simulation and greedy selection are proposed to approximately optimize the batch expected utility; exact optimization is not feasible due to its combinatorial nature.

We also apply this idea to the "dual" of the budgeted setting, which we call *cost effective active search* (Chapter 5), where the goal is to identify a given number of positives with minimum number of iterations. This is the first study on this problem with a principled approach in the active search literature. A non-adaptive approximation boils down to estimating the expectation of a *negative Poisson binomial distribution*, i.e., the expected number of non-uniform coins that need to be flipped to get a given number of HEADs.

This idea is also adapted to the continuous setting (Chapter 6), where we directly compute an optimal batch and then select one point from this batch. We show that this essentially optimizes a lower bound of the expected utility, looser than ENS, but much tighter than

myopic one-step approximations. We empirically show that such a simple approximation works well on two drastically different tasks: *Bayesian optimization* and *Bayesian quadrature*.

We also briefly discuss another idea that makes multi-step lookahead Bayesian optimization practical (section 6.8): jointly optimize all the variables in the whole decision tree in *one-shot*, instead of repeatedly solving nested maximizations. We present very promising preliminary results in terms of both optimization performance and computational efficiency.

In each iteration of active search, a standard procedure is to evaluate the approximate expected utility on all candidate points in the pool, and then choose the one with maximum value. This could be very time-consuming since the candidate pool is typically very large (e.g., 100k+). To further improve efficiency, we develop aggressive pruning strategies based on probability bounds for all our active search policies. Typically only a very small portion of points (e.g. 1%) need to be accessed in order to find the best. This allows us to perform experiments on massive datasets.

In almost all our experiments of all problem settings, our proposed nonmyopic policies achieve significant improvement over existing myopic ones, on real datasets such as drug/materials discovery, hyperparameter tuning, etc. More interestingly, we often observe favorable exploration/exploitation behavior: nonmyopic policies initially underperform myopic policies, exploring the space, but catch up at some point and outperform myopic policies towards the end of the budget, exploiting what has been learned and collect the utility.

Theoretically, we for the first time establish the hardness of active search, in both the budgeted and cost effective settings. We prove in the worst case that there are no polynomial time policies that can achieve expected utility within (or expected cost beyond) a constant ratio of that of the optimal policy. We also prove a lower bound on the adaptivity gap that shows

how much we could lose if we use larger batch sizes (hence be less adaptive), given the same number of total evaluations: the optimal batch policy is worse than the optimal purely sequential policy by at least a linear factor of batch size in terms of expected utility. This theoretical result is shown to match the empirical pattern, which could provide guidance on choosing batch sizes in practice.

Finally, we discuss important future directions regarding theoretical guarantees of our nonmyopic approximations, multi-fidelity active search, and possible reinforcement learning or meta learning approaches to problem settings where many similar active search problems are solved repeatedly.

1.1 Declaration of Previous Publications

All the work in this thesis resulted from collaboration with other researchers. Most of them has been published in peer-reviewed conferences. In the following we outline the original work presented in this thesis, with reference to previous publications or unpublished manuscripts, and describe my contribution to each piece of work.

Chapter 2: Sequential Experimental Design Overview

This chapter mainly introduces some well established concepts that this thesis is built on. The presentation is mostly the author's own. The presentation and notations on Bayesian decision theory are partially inspired by a draft of this book:

Garnett, R., *Bayesian Optimization*, in preparation, 2020.

Chapter 3: Budgeted Active Search

The work in this chapter has appeared in the following publication:

Jiang, S., Malkomes, G., Converse, G., Shofner, A., Moseley, B., Garnett, R. Efficient Nonmyopic Active Search. In Proceedings of the International Conference on Machine Learning (ICML), 2017.

Garnett proposed the main idea, implemented a prototype and conducted some preliminary experiments on CiteSeer^x data. Jiang developed and implemented the pruning technique that drastically improved the efficiency, and carried out all the experiments and analysis of the results. Malkomes prepared Figure 3.1 to demonstrate the nonmyopic behavior of the proposed policy and conceived the main parts of the active search instances for the hardness proof, especially the key idea of using XOR operator to encode dependence. Malkomes, Moseley and Jiang jointly completed the proof.

Chapter 4: Batch Budgeted Active Search

The work in this chapter has appeared in the following publication:

Jiang, S., Malkomes, G., Abbott, M., Moseley, B., Garnett, R. Efficient Nonmyopic Batch Active Search. Advances in Neural Information Processing Systems (NeurIPS), 2018.

Garnett suggested sequential simulation and Moseley suggested greedy optimization of the batch utility function. Jiang implemented all the algorithms, built the experimental platform, and carried out all the experiments and analysis of the results. Abbott contributed to the implementation of sequential simulation. Malkomes gave valuable suggestions on testing larger batch sizes to see more consistent patterns. Garnett suggested the presentation of progressive probability trace in Figure 4.4 to demonstrate the nonmyopic behavior. Moseley

conceived the high level idea for proving the lower bound of adaptivity gap; Moseley and Jiang completed the proof jointly.

Chapter 5: Cost Effective Active Search

The work in this chapter has appeared in the following publication:

Jiang, S., Moseley, B., Garnett, R. Cost Effective Active Search. Advances in Neural Information Processing Systems (NeurIPS), 2019.

Garnett conceived the problem setting. Jiang proposed the solution based on approximating expectation of negative Poisson binomial distribution. Jiang also implemented the algorithm and carried out all the experiments and analysis of results. Jiang and Moseley completed the hardness proof jointly.

Chapter 6: Bayesian Optimization and Beyond

This chapter describes a general framework for SED in the continuous setting, called BINOCULARS. The work is based on the content in the following manuscript:

Jiang, S., Chai, H., Gonzalez, J., Garnett, R. BINOCULARS for Efficient, Nonmyopic Sequential Experimental Design. In submission to ICML 2020.

The main idea came up during a meeting of Jiang and Garnett. Jiang derived the mathematical justification that the proposed policy maximizes a lower bound the true expected utility and provided initial convincing results under Bayesian optimization setting. Garnett suggested to try the idea on Bayesian quadrature, which was implemented by Chai. Jiang developed Figure 6.1 to demonstrate the intuition of the proposed algorithm. Jiang carried out the

experiments for Bayesian optimization and Chai carried out the experiments for Bayesian quadrature. Gonzalez helped on implementing the GLASSES baseline.

The work on one-shot optimization, briefly discussed in 6.8, resulted from an ongoing collaboration with Balandat, M., Garnett, R., Jiang, D., and Karrer, B. The idea of one-shot optimization was original proposed in [4] for knowledge gradient. The one-shot multi-step lookahead algorithm is implemented mainly by Balandat. S. Jiang added support for Gauss-Hermite sampling. The preliminary result shown in Figure 6.5 is produced by S. Jiang using the code base built on top of the work for BINOCULARS.

Chapter 2

Sequential Experimental Design Overview

In this chapter, we first formally define the sequential experimental design problem, then introduce Bayesian decision theory and Bellman equation for solving this problem, followed by a discussion of most related research literature.

2.1 Problem Formulation

Suppose there is an unknown function $f\colon \mathcal{X} \mapsto \mathcal{Y}$, typically black-box and expensive to evaluate. We may know the function values on a subset of locations (possibly empty) in the beginning. Denote this initial set of observations as $\mathcal{D} = \mathcal{D}_0$. We sequentially choose locations $x \in \mathcal{X}$ to evaluate (or query) the function and acquire observations $y = f(x)$[1], and accumulate our observation set $\mathcal{D} = \mathcal{D} \cup \{(x, y)\}$, until some termination condition is met. The design problem in this process is to intelligently choose locations to evaluate f so as to

[1]It could be that f is a latent function, and y is generated with some observation model on top of $f(x)$, such as Bernoulli and Gaussian. We adopt a simple presentation here to avoid describing unnecessarily complicated mathematical background on statistical modeling.

Algorithm 1 Sequential Experimental Design

Given input domain $\mathcal{X}$
Given black-box function f
Given initial observations $\mathcal{D} = \mathcal{D}_0 \equiv \{(x_i, y_i)\}_i$
repeat
  $x \leftarrow Policy(\mathcal{D})$ $\{\{x\} \subseteq \mathcal{X}$ could be a batch of points$\}$
  $y \backsim f(x)$ {evaluate x, the outcome(s) could be stochastic}
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x, y)\}$
until termination condition is reached
return $\mathcal{D}$

maximize the "value" of the final observed set $\mathcal{D}$. The concept of "value" will be formalized later. Such sequential experimental design processes are summarized in Algorithm 1.

What differentiates one problem from another is the underlying function f, termination condition and optimization goal. In the following, we instantiate each of the components in this formulation for three example problems we will study later: active search, Bayesian optimization, and Bayesian quadrature.

- Active search: sequentially evaluate items from a large candidate pool to search for targets with a desired property.
  - The design space $\mathcal{X} = \{x_1, \ldots, x_n\}$ is finite. For example, $\mathcal{X}$ could represent a set of chemical compounds we need to search through in the case of drug discovery.
  - The response space $\mathcal{Y} = \{0, 1\}$ is binary, representing positive/negative. For example, in drug discovery, the unknown function f is defined such that $f(x) = 1$ indicates the compound x exhibits sufficient binding activity with a certain biological target. Typically the positive class is extremely rare, rendering this problem challenging.

- The termination condition could be defined by a given querying budget or a predefined number of positives:
  - if the process terminates after a given querying budget T, we call it **budgeted active search**, that is, we can only perform T qeuries. The goal in this setting would be to *maximize the number of positives identified*;
  - if the process terminates when a predefined number of positives are identified, we call it **cost effective active search**. In this case, the goal is to *minimize the number of iterations required.*

- Bayesian optimization:[2] solve $x^* = \arg\max_{x \in \mathcal{X}} f(x)$ with a Bayesian approach.
  - The design space $\mathcal{X}$ in the simplest case could be just $\mathcal{X} \subseteq \mathbb{R}^d$, with each dimension bounded. More typically, it is a mix of continuous, discrete, or even categorical dimensions. For example, in the case of hyperparameter tuning of deep neural networks, each $x \in \mathcal{X}$ is a configuration of the architecture and hyperparameters, e.g., number of hidden nodes in each layer, learning rate, etc.
  - The response space $\mathcal{Y}$ is often $\mathcal{Y} \subseteq \mathbb{R}$. For example, $y = f(x)$ could represent the validation accuracy of a neural network trained with configuration x.
  - The termination condition for Bayesian optimization could also be defined similarly as in active search, but we will focus on budgeted setting. The goal is often to find the best possible input x so that $f(x)$ is maximized. We will formalize these concepts later.

- Bayesian quadrature: solve $Z = \int_{x \in \mathcal{X}} f(x)p(x)dx$ numerically with a Bayesian approach.

[2]There are many variants of Bayesian optimization; what is introduced here is just one of them, which we focus on in this thesis.

- The design and response space can be any continuous space as long as Z is a valid integration. An interesting case is $f(x)$ represents a likelihood model and $\mathcal{X}$ defines the parameter space, and $p(x)$ is a prior belief on the parameters. In this case Z is the model evidence. f is typically expensive if the sample size is large.
- The termination condition could also be budgeted or cost effective. For example, in budgeted setting our goal would be to minimize the variance of Z in a given number of function evaluations. Note Z is a random variable if we model f by a stochastic process.

One of the key challenges in solving SED problems lies in the tradeoff of *exploration and exploitation*; that is, to explore locations where f is highly uncertain, which could considerably improve our knowledge of f, or exploit the current knowledge about f and select locations where we expect to get high immediate reward. Next, we introduce Bayesian decision theory for solving this problem.

2.2 Bayesian Decision Theory

Bayesian decision theory is a framework for decision making under uncertainty. A comprehensive treatment can be found in [5]. In the context of sequential experimental design, Bayesian decision framework operates with two necessary components: Bayesian modeling of the function f and defining a utility function to express preference over different designs $\mathcal{D}$.

- The model of f allows us to reason about the posterior belief about f conditioned on any set of observations $\mathcal{D}$. Under this posterior belief, we have a probability distribution $p(y \mid x, \mathcal{D})$ for any $x \in \mathcal{X}$ and $y = f(x)$.

- The utility function $u(\mathcal{D})$ is defined depending on the task at hand, indicating preference of one set over another. For example, in budgeted active search $u(\mathcal{D})$ would be the number of positive points in $\mathcal{D}$, and in Bayesian optimization, $u(\mathcal{D})$ could be the maximum function value in $\mathcal{D}$.

Suppose we are at a state where we have observed a set $\mathcal{D}$, and our budget allows us to perform k more iterations of querying. If we let $\mathcal{D}_k$ denote the entire set of observations at the end, then $\mathcal{D}_k$ is composed of three parts as shown below:

$$\mathcal{D}_k \equiv \underbrace{\mathcal{D},}_{\text{observed}} \quad \underbrace{x,}_{\text{decision}} \quad \underbrace{y, x_2, y_2, \ldots, x_k, y_k.}_{\text{unobserved}} \tag{2.1}$$

Clearly $\mathcal{D}_k$ is a random quantity due to the uncertain future; hence $u(\mathcal{D}_k)$ is also random. The *Bayesian optimal policy* chooses x maximizing the expected utility

$$x^* = \arg\max_x \mathbb{E}[u(\mathcal{D}_k) \mid x, \mathcal{D}]. \tag{2.2}$$

In most scenarios of interest, this expectation is computationally intractable. In the following, we derive the Bellman equation for computing the expected utility in a dynamic programming approach.

2.3 Bellman Equation

Define $v_k(x \mid \mathcal{D})$ as the expected marginal value of the remaining k steps starting from x, following the optimal policy after choosing x in the first step. That is

$$v_k(x \mid \mathcal{D}) = \mathbb{E}[u(\mathcal{D}_k) - u(\mathcal{D}) \mid x, \mathcal{D}]. \tag{2.3}$$

Recall $\mathcal{D}_k \equiv \mathcal{D} \cup \{(x, y)\} \cup \{(x_2, y_2)\}, \ldots, \{(x_k, y_k)\}$. So the expectation is taken with respect to all unknown quantities: $y, x_2, y_2, \ldots, x_k, y_k$, assuming the sequence is constructed such that each $x_i, i = 2, \ldots, k$, would be selected with the optimal policy upon the revelation of the previous y_{i-1}. Since $u(\mathcal{D})$ is a known constant, (2.2) is equivalent to

$$x^* = \arg\max_x v_k(x \mid \mathcal{D}). \tag{2.4}$$

Define $\mathcal{D}_1 \equiv \mathcal{D} \cup \{(x, y)\}$ and $\mathcal{D}_i \equiv \mathcal{D}_{i-1} \cup \{(x_i, y_i)\}, i = 2, \ldots, k$. We can derive a recursive form of (2.3) by backward induction:

- When $k = 1$,

$$v_1(x \mid \mathcal{D}) = \mathbb{E}_y[u(\mathcal{D}_1) - u(\mathcal{D}) \mid x, \mathcal{D}] = \int_y (u(\mathcal{D}_1) - u(\mathcal{D}))p(y \mid x, \mathcal{D})dy. \tag{2.5}$$

  The integration changes to summation in case y is discrete. For example, when $u(\mathcal{D}) = \sum_{(x,y)\in\mathcal{D}} y$ as in active search, it is easy to see $v_1(x \mid \mathcal{D}) = \Pr(y = 1 \mid x, \mathcal{D})$.

- When $k = 2$, assuming the last step is chosen optimally,

$$\begin{aligned} v_2(x \mid \mathcal{D}) &= \mathbb{E}_{y,x_2,y_2}[u(\mathcal{D}_2) - u(\mathcal{D}) \mid x, \mathcal{D}] \\ &= \mathbb{E}_{y,x_2,y_2}[u(\mathcal{D}_2) - u(\mathcal{D}_1) + u(\mathcal{D}_1) - u(\mathcal{D}) \mid x, \mathcal{D}] \\ &= \mathbb{E}_{y,x_2,y_2}[u(\mathcal{D}_2) - u(\mathcal{D}_1) \mid x, \mathcal{D}] + \mathbb{E}_{y,x_2,y_2}[u(\mathcal{D}_1) - u(\mathcal{D}) \mid x, \mathcal{D}]. \end{aligned} \tag{2.6}$$

The second expectation is simply

$$\mathbb{E}_{y,x_2,y_2}[u(\mathcal{D}_1) - u(\mathcal{D}) \mid x, \mathcal{D}] = \mathbb{E}_y[u(\mathcal{D}_1) - u(\mathcal{D}) \mid x, \mathcal{D}] \equiv v_1(x \mid \mathcal{D}),$$

since $u(\mathcal{D}_1) - u(\mathcal{D})$ is unrelated to x_2, y_2. The first expectation is more complicated:

$$\begin{aligned} &\mathbb{E}_{y,x_2,y_2}[u(\mathcal{D}_2) - u(\mathcal{D}_1) \mid x, \mathcal{D}] \\ &= \int_y \left[\int_{x_2} \left(\int_{y_2} (u(\mathcal{D}_2) - u(\mathcal{D}_1)) p(y_2 \mid x_2, \mathcal{D}_1) dy_2 \right) p(x_2 \mid \mathcal{D}_1) dx_2 \right] p(y \mid x, \mathcal{D}) dy \\ &= \int_y \left[\int_{x_2} v_1(x_2 \mid \mathcal{D}_1) p(x_2 \mid \mathcal{D}_1) dx_2 \right] p(y \mid x, \mathcal{D}) dy \end{aligned} \tag{2.7}$$

We assume the optimal policy is deterministic and always chooses the optimal design $x^* = \arg\max_x v_i(x \mid \mathcal{D})$ for any horizon i and existing observations $\mathcal{D}$. Without loss of generality, we assume such optimal designs are unique (break ties arbitrarily if not). Under the assumption that the last step is optimal, $p(x_2 \mid \mathcal{D}_1)$ is a Dirac delta distribution

$$p(x_2 \mid \mathcal{D}_1) = \delta \left(x_2 - \arg\max_{x'} v_1(x' \mid \mathcal{D}_1) \right).$$

Therefore,

$$\int_{x_2} v_1(x_2 \mid \mathcal{D}_1) p(x_2 \mid \mathcal{D}_1) dx_2 = \max_{x'} v_1(x' \mid \mathcal{D}_1). \tag{2.8}$$

Note $\max_{x'} v_1(x' \mid \mathcal{D}_1)$ is a random variable since $\mathcal{D}_1 \equiv \mathcal{D}, x, y$ is random. Plugging this back into (2.7), we have

$$\mathbb{E}_{y,x_2,y_2}[u(\mathcal{D}_2) - u(\mathcal{D}_1) \mid x, \mathcal{D}] = \mathbb{E}_y \left[\max_{x'} v_1(x' \mid \mathcal{D}_1) \right]. \tag{2.9}$$

Further plugging into (2.6), we have

$$v_2(x \mid \mathcal{D}) = v_1(x \mid \mathcal{D}) + \mathbb{E}_y \left[\max_{x'} v_1(x' \mid \mathcal{D}_1) \right]. \tag{2.10}$$

This is the two-step lookahead value function.

- In general, a k-step lookahead value function is

$$\begin{aligned} v_k(x \mid \mathcal{D}) &= \mathbb{E}_{y,x_2,y_2,\ldots,x_k,y_k}[u(\mathcal{D}_k) - u(\mathcal{D}) \mid x, \mathcal{D}] && (2.11) \\ &= \mathbb{E}_{y,x_2,y_2,\ldots,x_k,y_k}[u(\mathcal{D}_k) - u(\mathcal{D}_1) + u(\mathcal{D}_1) - u(\mathcal{D}) \mid x, \mathcal{D}] && (2.12) \\ &= v_1(x \mid \mathcal{D}) + \mathbb{E}_y[\max_{x'} v_{k-1}(x' \mid \mathcal{D}_1)]. && (2.13) \end{aligned}$$

  This is precisely the format of *Bellman equation* [90].

To expand the recursion, we can write

$$v_k(x \mid \mathcal{D}) = v_1(x \mid \mathcal{D}) + \mathbb{E}_y \Big[\max_{x_2} \{ v_1(x_2 \mid \mathcal{D}_1) + \mathbb{E}_{y_2}[\max_{x_3} \{ v_1(x_3 \mid \mathcal{D}_2) + \cdots \}] \} \Big]. \tag{2.14}$$

We can see the computation of k-step lookahead value function requires alternately nested maximization and expectation. For most real problems of interest, such as the one we study in this thesis, this quantity is computationally intractable.

In practice, what people do is simply one-step lookahead (or two-step lookahead if possible), ignoring the utility beyond this short horizon. Such myopic decisions can be quite suboptimal, translating to considerable loss (or waste) of money and other resources in real applications. The central topic of this thesis is *efficient and nonmyopic* approximations to the Bayesian optimal policies in various sequential experimental design settings.

All above derivation is based on the budgeted setting, where the decision horizon is fixed. However, we may also encounter cost effective setting where the process terminates only when a certain utility threshold is reached (see Chapter 5).

2.4 Related Research Literature

Sequential experimental design is a special case of general sequential decision making, for which extensive literature can be found under various nomenclatures or perspectives. In this section, we discuss a few most related ones.

2.4.1 Markov Decision Process

A Markov decision process (MDP) [43, 90] is defined by a 4-tuple (S, A, P, R), where S is the state space, A is the action space, $P(s, a, s')$ is the probability of transitioning to state s' when taking action a at state s, and $R(s, a, s')$ is a reward function that dictates the reward

of transitioning to state s' after taking action a at state s. There could be a set of start states S_0 and/or terminal states $S_T \subseteq S$. The goal in solving an MDP is to design a policy $\pi : S \mapsto A$, such that the expected reward following the policy from any start state to any terminal state is maximized.

Sequential experimental design as defined previously can be formulated as an MDP. In particular, the state space $S = 2^{\mathcal{X}\times\mathcal{Y}}$ is all possible subsets of Cartesian product of the design space and response space, that is, the observed set $\mathcal{D}$ is the current state of the MDP. The action space is simply the design space $\mathcal{X}$. The transition probability is implied by the Bayesian model. In particular, $P(\mathcal{D}, x, \mathcal{D}_1) = p(y \mid x, \mathcal{D})$. Finally, the reward function $R(\mathcal{D}, x, \mathcal{D}_1) = u(\mathcal{D}_1) - u(\mathcal{D})$ is precisely the marginal utility of evaluating x. If the experimental design has a budget t, then the set of terminal states would be $S_T = \{\mathcal{D} \in S : |\mathcal{D}| = t\}$ (assuming the initial set $\mathcal{D}_0 = \emptyset$, or we can simply exclude $\mathcal{D}_0$ when counting the cardinality of $\mathcal{D}$).

What is special about SED is that the state space S is dauntingly large, or even infinite. Traditional dynamic programming approaches such as value iteration or policy iteration are not feasible in this setting.

2.4.2 Reinforcement Learning

We have shown that SED can be formulated as a Markov decision process, under the assumption that the transition model is known. However, in practice, this is often not the case. In fact, in all SED problems we are going to study here, the transition model is not known. When we do not make assumptions on the transition model of the MDP, we land in the regime of reinforcement learning (RL). In fact, the value function $v_k(x \mid \mathcal{D})$ precisely corresponds to

the action values in RL terminology, often denoted as Q functions. RL is an important field in artificial intelligence, and has enjoyed tremendous success in recent years, especially in game playing [66, 84, 85]. A comprehensive treatment of RL can be found in [90].

While it is possible to solve SED problems with a reinforcement learning approach in some special settings, we focus on model-based policy design in this thesis. We will discuss a setting where an RL approach might be appropriate in the chapter of future work.

2.4.3 Active Learning

Machine learning by default is passive, that is, the learner passively receives whatever examples are presented, typically i.i.d. samples, without control over what examples to learn from. However, in cases where data labeling is expensive, we may want to allocate budget only to labeling the most informative instances. This is the motivation of active learning: the learner "actively" chooses what data to label in an iterative fashion, with the goal of learning the concept with minimum labeling cost. There is extensive research literature on active learning. A notable survey can be found in [80].

Active learning can be perfectly formulated as an SED problem, with a utility function measuring, e.g., the mutual information between the chosen locations in $\mathcal{D}$ and the unknown function f [52]. In fact, the methodology introduced in Chapter 6 is applicable to active learning.

Chapter 3

Budgeted Active Search

In active search, we seek to sequentially inspect data so as to discover members of a rare, desired class. The labels are not known *a priori* but can be revealed by querying a costly labeling oracle. The goal in budgeted setting is to design an policy to sequentially query points to find as many valuable points as possible under a labeling budget. In this chapter, "active search" always refers to budgeted setting. Several real-world problems can be naturally posed in terms of active search; drug discovery, fraud detection, and product recommendation are a few examples. A successful active search policy faces the fundamental dilemma between *exploration* and *exploitation*; i.e., whether to search for new regions of valuable points (exploration) or take advantage of the currently most-promising regions (exploitation).

Previous work developed policies for budgeted active search by appealing to Bayesian decision theory [26, 27]. [27] derived the optimal policy in this framework with a natural utility function. Not surprisingly, realizing this policy in the general case requires exponential computation. To overcome this intractability, the authors of that work proposed using myopic lookahead policies in practice, which compute the optimal policy only up to a limited number of steps into the future. This defines a family of policies ranging in complexity from completely greedy one-step lookahead to the optimal policy, which looks ahead to the depletion of the

entire budget. The authors demonstrated improved performance on active search over the greedy policy even when looking just two steps into the future, including in a drug-discovery setting [25]. The main limitation of these strategies is that they completely ignore what can happen beyond the chosen horizon, which for typical problems is necessarily limited to $\ell \leq 3$, even with aggressive pruning.

The contributions of this chapter are two-fold. First, we prove that no polynomial time policy for active search can have nontrivial approximation ratio with respect to the optimal policy in terms of expected utility. This extends the result in [27] that myopic approximations to the optimal policy cannot approximate the optimal policy. The proof of this theorem is constructive, creating a family of explicitly difficult active search instances and showing that no polynomial time algorithm can perform well compared to the optimal (exponential cost) policy on these.

Second, we introduce a novel *nonmyopic* policy for active search that considers not only the potential immediate contribution of each unlabeled point but also its potential impact on the remaining points that could be chosen afterwards. Our policy *automatically* balances exploitation against exploration consistent with the labeling budget without requiring any parameters controlling this tradeoff. We also develop an effective strategy for pruning unlabeled points by bounding their potential impact on the search problem. We compare our method with several baselines by conducting experiments on numerous real datasets spanning several domains including citation networks, materials science, and drug discovery. Our results thoroughly demonstrate that our policy typically significantly outperforms previously proposed active search approaches.

3.1 Problem Definition

Suppose we are given a finite domain of elements $\mathcal{X} \triangleq \{x_i\}$. We know that there is a rare subset $\mathcal{R} \subset \mathcal{X}$, the members of which are considered valuable, but their identities are unknown *a priori*. We will call the elements of $\mathcal{R}$ *targets* or *positive* items. Assume that there is an oracle that can determine whether a specified element $x \in \mathcal{X}$ is a target, producing the binary output $y \triangleq f(x) = \mathbb{1}\{x \in \mathcal{R}\}$. The oracle, however, is assumed to be expensive and may only be queried t times. We seek to design a policy to sequentially query elements to maximize the number of targets found.

We will express our preference over different sets of observations $\mathcal{D} \triangleq \{(x_i, y_i)\}$ through a simple utility:

$$u(\mathcal{D}) \triangleq \textstyle\sum_{y_i \in \mathcal{D}} y_i, \tag{3.1}$$

which simply counts the number of targets in $\mathcal{D}$. Then, the problem is to sequentially construct a set of t observed points $\mathcal{D}$ with the goal of maximizing $u(\mathcal{D})$. Throughout this work, we use a subscript to specify a set of observed data after $i \leq t$ queries, defining $\mathcal{D}_i \triangleq \{(x_j, y_j)\}_{j=1}^{i}$.

3.2 Bayesian Optimal Policy

Following previous work, we consider the active search problem in the standard Bayesian decision framework. Assume we have a probabilistic classification model that provides the posterior probability of a point x belonging to $\mathcal{R}$, given observed data $\mathcal{D}$: $\Pr(y = 1 \mid x, \mathcal{D})$.

Recall that we are allowed to perform t labeling queries, and suppose we are at some iteration i for $i \leq t$; having already observed $i-1$ examples, $\mathcal{D}_{i-1}$. We wish to submit the ith item to the oracle. Bayesian decision theory compels us to select the item that if we evaluate next maximizes the *expected utility* of the final observed dataset:

$$x_i^* = \underset{x_i \in \mathcal{X} \setminus \mathcal{D}_{i-1}}{\arg\max}\, \mathbb{E}\big[u(\mathcal{D}_t) \mid x_i, \mathcal{D}_{i-1}\big]. \tag{3.2}$$

In other words, we choose a point x_i^* maximizing the expected number of targets found at termination. Unfortunately, as we shall see later, computing $\mathbb{E}\big[u(\mathcal{D}_t) \mid x_i, \mathcal{D}_{i-1}\big]$ is computationally impractical.

To better understand the optimal policy, consider the case $i = t$, so we already have $t-1$ observations $\mathcal{D}_{t-1}$ and there is only one more query left. The expected utility is

$$\mathbb{E}\big[u(\mathcal{D}_t) \mid x_t, \mathcal{D}_{t-1}\big] = \textstyle\sum_{y_t} u(\mathcal{D}_t) \Pr(y_t \mid x_t, \mathcal{D}_{t-1}) = u(\mathcal{D}_{t-1}) + \Pr(y_t = 1 \mid x_t, \mathcal{D}_{t-1}). \tag{3.3}$$

Note $u(\mathcal{D}_{t-1})$ is a constant, since $\mathcal{D}_{t-1}$ was already observed. Thus, when there is one query remaining, the optimal decision is to greedily choose the remaining point with maximum probability of being a target.

When two or more queries are left, the optimal policy is not as trivial. The challenge is that after the first choice, the probability model changes, affecting all future decisions. Below, we show the expected utility for $i = t-1$.

$$\mathbb{E}\big[u(\mathcal{D}_t) \mid x_{t-1}, \mathcal{D}_{t-2}\big] = u(\mathcal{D}_{t-2}) + \Pr(y_{t-1} = 1 \mid x_{t-1}, \mathcal{D}_{t-2}) + \mathbb{E}_{y_{t-1}}\big[\max_{x_t} \Pr(y_t = 1 \mid x_t, \mathcal{D}_{t-1})\big]. \tag{3.4}$$

This expression has an intuitive interpretation. First, we have the reward for the data already observed, $u(\mathcal{D}_{t-2})$. The second term is the expected reward contribution from the point x_{t-1} under consideration, $\Pr(y_{t-1} = 1 \mid x_{t-1}, \mathcal{D}_{t-2})$. Finally, the last term is the expected future reward, which is the expected reward to be gathered on the next step; from our previous analysis, we know that this will be maximized by a greedy selection (3.3). These latter two terms can be interpreted as encouraging exploitation and exploration, respectively, with the optimal second-to-last query.

In general, we can compute expected utility (3.2) at time $i \leq t$ recursively as [27]:

$$\mathbb{E}\big[u(\mathcal{D}_t) \mid x_i, \mathcal{D}_{i-1}\big] = u(\mathcal{D}_{i-1}) + \underbrace{\Pr(y_i = 1 \mid x_i, \mathcal{D}_{i-1})}_{\text{exploitation, } <1} + \underbrace{\mathbb{E}_{y_i}\Big[\max_{x'} \mathbb{E}\big[u(\mathcal{D}_t \setminus \mathcal{D}_i) \mid x', \mathcal{D}_i\big]\Big]}_{\text{exploration, } <t-i}. \quad (3.5)$$

Note if we denote $v_k(x_i \mid \mathcal{D}_{i-1}) \equiv \mathbb{E}[u(\mathcal{D}_t) - u(\mathcal{D}_{i-1}) \mid x_i, \mathcal{D}_{i-1}]$ and $v_{k-1}(x' \mid \mathcal{D}_i) = \mathbb{E}[u(\mathcal{D}_t \setminus \mathcal{D}_i) \mid x', \mathcal{D}_{i-1}]$ for $k = t - i + 1$, we recover the general form of Bellman equation (2.13) as introduced in 2.3. It is easy to show that the time complexity for computing (3.5) is $\mathcal{O}\big((2n)^\ell\big)$, where $\ell = t - i + 1$ is the lookahead and n is the total number of unlabeled points.

This exponential running time complexity makes the Bayeisan optimal policy infeasible to compute, even for small-scale applications. A typical workaround is to pretend there are only a few steps left in the search problem at each iteration, and sequentially apply a myopic policy (e.g., (3.3) or (3.4)). We will refer to these policies as the one-step and two-step myopic policies, respectively, and more generally to the ℓ-step myopic policy, with $\ell < t - i + 1$.

Since these myopic approaches cannot plan more than ℓ steps ahead, they can underestimate the potential benefit of exploration. In particular, the potential magnitude of the exploration term in (3.5) depends linearly on the budget, whereas in an ℓ-step myopic policy, the magnitude of the equivalent term can go no higher than a fixed upper bound of ℓ. In fact, [27] showed via an explicit construction that the expected performance of the ℓ-step policy can be arbitrarily worse than any m-step policy with $\ell < m$, exploiting this inability to "see past" the horizon. When following this suggestion, we must thus trade off the potential benefits of nonmyopia and the rapidly increasing computational burden of lookahead when choosing a policy.

3.3 Hardness of Approximation

We extend the above hardness result to show that no polynomial-time active search policy can be a (constant factor) approximation algorithm with respect to the optimal policy, in terms of expected utility. In particular, under the assumption that algorithms only have access to a unit cost conditional marginal probability $\Pr(y = 1 \mid x, \mathcal{D})$ for any x and $\mathcal{D}$, where $|\mathcal{D}|$ is less than the budget,[3] then:

Theorem 1. *There is no polynomial-time active search policy with a constant factor approximation ratio for optimizing the expected utility.*

We prove this theorem in Appendix A. The main idea is to construct a class of instances where a small "secret" set of elements encodes the locations of a large "treasure" of targets. The probability of revealing the treasure is vanishingly small without discovering the secret

[3]The optimal policy operates under these restrictions.

set; however, it is extremely unlikely to observe any information about this secret set with polynomial-time effort.

Despite the negative result of Theorem 1, we may still search for policies that are empirically effective on real problems. In the next section, we propose a novel alternative approximation to the optimal policy (3.2) that is nonmyopic, computationally efficient, and shows impressive empirical performance.

3.4 Efficient Nonmyopic Approximation

We have seen above how to myopically approximate the Bayesian optimal policy using an ℓ-step-lookahead approximate policy (3.5). Such an approximation, however, effectively assumes that the search procedure will terminate after the next ℓ evaluations, which does not reward exploratory behavior that improves performance beyond that horizon. We propose to continue to exactly compute the expected utility to some fixed horizon (e.g., one-step), but to approximate the remainder of the search differently. We will approximate the expected utility from any remaining portion of the search by assuming that any remaining points, $\{x_{i+1}, x_{i+2}, \ldots, x_t\}$, in our budget will be selected *simultaneously* in one big batch. One rationale is if we assume that after observing $\mathcal{D}_i$, the labels of all remaining unlabeled points are conditionally independent, then this approximation recovers the Bayesian optimal policy exactly. By exploiting linearity of expectation, it is easy to work out the optimal policy for selecting such a simultaneous batch observation: we simply select the points with the highest probability of being valuable. Another perspective is that we are using the optimal non-adaptive policy to approximate the optimal adaptive policy beyond the fixed horizon.

The resulting approximation is

$$\max_{x'} \mathbb{E}\big[u(\mathcal{D}_t \backslash \mathcal{D}_i) \mid x', \mathcal{D}_i\big] \approx \sum\nolimits'_{t-i} \Pr(y = 1 \mid x, \mathcal{D}_i), \tag{3.6}$$

where the summation-with-prime symbol $\sum'_k$ indicates that we only sum the largest k values.

Our proposed policy selects points by maximizing the approximate final expected utility using:

$$\mathbb{E}\big[u(\mathcal{D}_t) \mid x_i, \mathcal{D}_{i-1}\big] \approx u(\mathcal{D}_{i-1}) + \Pr(y_i = 1 \mid x_i, \mathcal{D}_{i-1}) + \underbrace{\mathbb{E}_{y_i}\Big[\sum\nolimits'_{t-i} \Pr\big(y = 1 \mid x, \mathcal{D}_i\big)\Big]}_{\text{exploration, } < t-i}. \tag{3.7}$$

We will call this policy *efficient nonmyopic search* (ENS). As in the optimal policy, we can interpret (3.7) naturally as rewarding both exploitation and exploration, where the exploration benefit is judged by a point's capability to increase the top probabilities among currently unlabeled points. We note further that in (3.7) the reward for exploration *naturally decreases over time* as the budget is depleted, exactly as in the optimal policy. In particular, the very last point x_t is chosen greedily by maximizing probability, agreeing with the true optimal policy. The second-to-last point is also guaranteed to match the optimal policy.

Note that we may also use the approximation in (3.6) as part of a finite-horizon lookahead with $\ell > 1$, producing a family of increasingly expensive but higher-fidelity approximations to the optimal policy, all retaining the same budget consciousness. The approximation in (3.7) is equivalent to a one-step maximization of (3.6). We will see in our experiments that this is often enough to show massive gains in performance, and that even this policy shows clear awareness of the remaining budget throughout the search process, automatically and dynamically trading off exploration and exploitation.

3.4.1 Nonmyopic Behavior

To illustrate the nonmyopic behavior of our policy, we have adapted the toy example presented by [27]. Let $I \triangleq [0,1]^2$ be the unit square. We repeated the following experiment 100 times. We selected 500 points i.i.d. uniformly at random from I to form the input space $\mathcal{X}$. We create an active search problem by defining the set of targets $\mathcal{R} \subseteq \mathcal{X}$ to be all points within Euclidean distance $1/4$ from either the center or any corner of I. We took the closest point to the center (always a target) as an initial training set. We then applied ENS and the two-step-lookahead (3.4) policies to sequentially select 200 further points for labeling.

Figure 3.1 shows a kernel density estimate of the distribution of locations selected by both methods during two time intervals. Figures 3.1(a–b) correspond to our method; Figures 3.1(c–d) to two-step lookahead. Figures 3.1(a, c) consider the distribution of the first 100 selected locations; Figures 3.1(b, d) consider the last 100. The qualitative difference between these strategies is clear. The myopic policy focused on collecting all targets around the center (Figure 3.1(c)), whereas our policy explores the boundaries of the center clump with considerable intensity, as well as some of the corners (Figure 3.1(a)). As a result, our policy is capable of finding some of targets in the corners, whereas two-step lookahead hardly ever can (Figure 3.1(d)). We can also see that the highest probability mass in Figure 3.1(b) is the center, which shows that our policy typically saves many high-probability points until the end. On average, the ENS policy found about 40 more targets at termination than the two-step-lookahead policy.

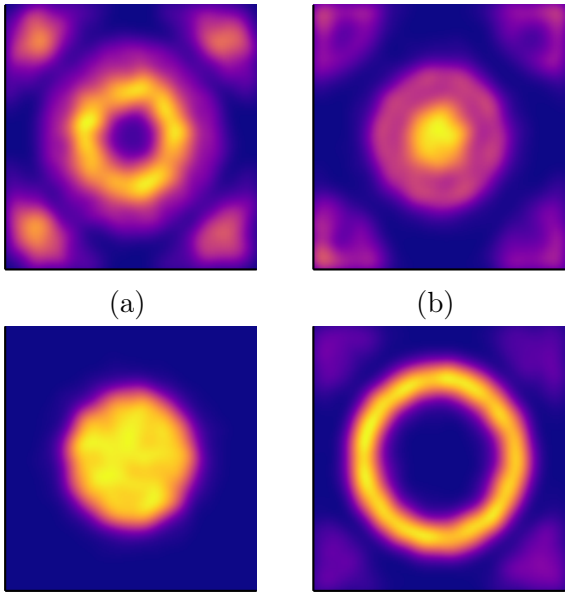


Figure 3.1: Kernel density estimates of the distribution of points chosen by ENS (top) and 2-step lookahead (bottom) during two different time intervals. The figures on the left show the kernel density estimates for the first 100 locations; the figures on the right, the last 100 chosen locations.

3.5 Implementation

In this section, we introduce the model used to estimate the probabilities of a point being positive given a set of observations, and analyze the time complexity of our implementation. We also introduce a pruning technique that we implemented to drastically improve the efficiency of our proposed policy.

3.5.1 k Nearest Neighbors

In active search, the design space $\mathcal{X}$ is finite and response space $\mathcal{Y}$ is binary. We use the k nearest neighbors (k-nn) model [15] to estimate the probability of a point being positive. The

general idea of k-nn for binary classification is very simple: to estimate the probability of a point being positve, we first compute the distances of this point to every point in the observed set $\mathcal{D}$ and determine the k nearest neighbors, among which the proportion of positive points is used as an estimate of the probability.

Our setting is different from a general classification problem, in the sense that $\mathcal{D}$ is usually very small (potentially even smaller than k), and we augment $\mathcal{D}$ in each iteration by labeling points from a large unlabeled pool. We consider the following variant of k-nn model, first introduced in [27]:

- Prior: in the beginning when $\mathcal{D} = \emptyset$, we set a constant α prior belief on the prevalence of the positive points. For example, in a drug discovery task, if chemists believe that there are about 1% of compounds in the pool $\mathcal{X}$ that can bind with the biological target, then we might set $\alpha = 0.01$.

- Posterior: after some observations, the posterior probability of a point $x \in \mathcal{X}$ being positive is estimated as follows: (1) compute the k nearest neighbors among the whole pool; (2) if there are observations among the k points, we count the proportion of positive as the estimate, optionally weighted by similarities; (3) if there are no observations among the k points, we fall back to the prior belief. Mathematically,

$$\Pr(y = 1 \mid x, \mathcal{D}) = \frac{\alpha + \sum_{x' \in \mathrm{LNN}(x)} w_{x'} \cdot y'}{1 + \sum_{x' \in \mathrm{LNN}(x)} w_{x'}}, \tag{3.8}$$

  where $\mathrm{LNN}(x)$ is the set of labeled examples in the k nearest neighbors of x; $w_{x'}$ is the weight of example x', usually proportional to the similarity between x and x'; the weights could simply be all ones for all x'. Note when $\mathrm{LNN}(x)$ is empty, $\Pr(y = 1 \mid x, \mathcal{D}) = \alpha$.

The weights can be precomputed and saved as a sparse matrix $W \subseteq \mathbb{R}^{n \times n}$. This allows fast updates of the posterior probabilities. To update the probability of a point i, we only need to gather the weights with its k nearest neighbors, i.e. the nonzeros of the i'th row $W[i,:]$, and when a point i is observed, we only need to update the probabilities of the points that have i in its k nearest neighbors, i.e., as indicated by $W[:,i] > 0$.

3.5.2 Time Complexity

The complexity of our policy (3.7) is $\mathcal{O}\big(n\big(2(n+n\log n)\big)\big) = \mathcal{O}(n^2 \log n)$, for $n = |\mathcal{X}|$, because we need to compute the approximate expected utility for all n points, evaluate an expectation over its label, conditioning the model and sorting the posterior probabilities in the expectation. However, for some classification models $\Pr(y = 1 \mid x, \mathcal{D})$, observing one point will only affect the probabilities on a small portion of the other points (e.g., in a k-nn model). We can exploit such structure to reduce the complexity of our method by avoiding unnecessary computation.

Specifically, suppose that after observing a point we only need to update the probabilities of at-most m other points. We can avoid repeatedly sorting the probabilities of every unlabeled point when computing the score of each candidate point. Once the *current* probabilities are sorted ($\mathcal{O}(n \log n)$), we only need to update m probabilities and sort these as well ($\mathcal{O}(m \log m)$); now we can merge both lists to get the top $t - i$ posterior probabilities in time $\mathcal{O}(t - i)$, where i is the index of current iteration. In summary, these tricks can reduce the computational complexity to $\mathcal{O}\big(n(\log n + m \log m + t)\big)$. We can see the complexity is about the same as two-step lookahead, which is $\mathcal{O}\big(n(\log n + m)\big)$ when using the same tricks.

3.5.3 Pruning the Search Space

To further reduce the computational complexity, we can use a similar strategy as suggested by [27] to bound the score function (3.7) and prune points that cannot possibly maximize our score. We consider the same two assumptions proposed by these authors. First, observing a new negative point will not raise the probability of any other point being a target. Second, we are able to bound the maximum probability of the unlabeled points after conditioning on a given number of additional targets; that is, we assume there is a function $p^*(n, \mathcal{D})$ such that

$$p^*(n, \mathcal{D}) \geq \max_{x \in \mathcal{X} \setminus \mathcal{D}} \Pr(y = 1 \mid x, \mathcal{D} \cup \mathcal{D}', \textstyle\sum_{y' \in \mathcal{D}'} y' \leq n).$$

That is, the probability of any unlabeled point can become at most $p^*(n, \mathcal{D})$ after further conditioning on n or fewer additional target points.

Consider an unlabeled point x at time i, and define $\pi(x) = \Pr(y = 1 \mid x, \mathcal{D}_i)$ for the remainder of this discussion. The score (3.7), denoted $f(x)$ here for simplicity, can be upper bounded by

$$\begin{aligned} f(x) &\leq \pi(x) \cdot \big(1 + (t - i) p^*(1, \mathcal{D}_i)\big) + \\ &\quad (1 - \pi(x)) \cdot \big(\textstyle\sum'_{t-i} \Pr(y' = 1 \mid x', \mathcal{D}_i)\big) \triangleq U(\pi(x)). \end{aligned}$$

Note this upper bound is only a function of the current probability π. Let x^+ be the point with maximum probability. Then $f(x^+)$ is certainly a lower bound of $\max_x f(x)$. Hence, those points satisfying $U\big(\pi(x)\big) < f(x^+)$ can be safely removed from consideration. Solving this inequality, we have

$$\pi(x) < \frac{f(x^+) - \sum'_{t-i} \Pr(y' = 1 \mid x', \mathcal{D})}{1 + p^*(1, \mathcal{D})(t - i) - \sum'_{t-i} \Pr(y' = 1 \mid x', \mathcal{D})}. \tag{3.9}$$

Then, all points with current probability lower than the RHS of (3.9) can be removed from consideration. We will show empirically that a large fraction of points can often be pruned on massive datasets.

3.6 Related Work

Active search can be seen as a specific realization of active learning (AL). Though highly related, AL and AS have fundamentally different goals: learning an accurate model versus retrieving positive examples. One might argue that AS can be reduced to AL by first learning the decision boundary, then just collecting the predicted positive examples. However, it is often the case that the given budget is far from enough for an accurate model to be learned, and we must have more-elegant approaches to balance exploration and exploitation. Good AL policies could perform poorly in AS: [99] compared several variants of uncertainty sampling (arguably one of the most popular AL policies) with greedy AS policies, and demonstrated that the greedy policies performed much better in terms of retrieving active compounds. In fact, we will show later in the experiments that a k-nn classification model trained on 10 times more data still retrieved significantly fewer positives than a simple greedy AS policy.

The multi-armed bandit (MAB) problem shares some similarities with active search, where selecting an item can understood as "pulling an arm." However, in active search the items are correlated, and, critically, they should never be played twice. Despite the difference, we note that our ENS policy is somewhat similar to the *knowledge gradient* policy introduced by [23].

Active search can be seen as a special case of *Bayesian optimization* (see Chapter 6) with binary observations and cumulative reward. Several nonmyopic policies have been proposed

for Bayesian optimization in the regression setting (e.g., [34, 56, 105]), and our method is spiritually similar to the recently propopsed GLASSES algorithm [35], except they use a heuristic batch as future estimate, but we use an optimal batch.

[92] proposed a method called GP-SELECT to solve a class of problems the authors call "adaptive valuable item discovery," which generalizes active search to the regression setting. GP-SELECT employs a Gaussian process regression model in a manner inspired by the Gaussian process upper confidence bound (GP-UCB) algorithm [88]. A parameter must be specified to balance exploration and exploitation, whereas our method automatically and dynamically trades off these quantities. The method is also critically tied to Gaussian process regression as the underlying model, which is inappropriate for classification. Our decision-theoretic approach does not make any assumptions about the classification model other than being probabilistic.

Active search can also be seen as a special case of (partially observable) Markov decision processes ((PO)MDPs), for which there are known hardness results. [77], for example, defined the class of so-called "purely epistemic" MDPs (EMDPs), where the state does not evolve over time. The authors showed that the optimal policy for these problems cannot admit polynomial-time constant approximations. Unfortunately, these hardness results, for the very rich class of EMDPs are not trivially transferred to the more-specific active search problem.

Our proposed approximation is similar in nature to the active search policy proposed by [95], which only considered the effect of raising probabilities after observing a positive label, and did not consider the budget. Rather, the proposed score always encourages maximal exploration, in opposition to the optimal policy.

There has been some attention to active search in the graph setting where the input domain $\mathcal{X}$ is the nodes of a graph [26, 63, 73, 95]. Our method does not restrict the input space. Further, the classification models used in these settings are often difficult to scale to large datasets, e.g., requiring the pseudoinverse of the graph Laplacian.

Finally, variations on the active search problem have also been considered. [62] proposed the *active area search* problem, wherein a continuous function is sampled to discover regions with large mean value, and [64] extended this idea to define the more-general *active pointillistic pattern search* problem. These settings do not allow querying for labels directly and offer no insight to the core active search problem.

3.7 Experiments

We implemented our approximation to the Bayesian optimal policy with the MATLAB active learning toolbox,[4] and have compared the performance of our proposed ENS policy with several baselines. First we compare with the myopic one-step (greedy) and two-step approximations to the Bayesian optimal policy, presented in (3.3) and (3.4). Note that [27] and [25] thoroughly compared the one- and two-step policies, with the finding that the less-myopic two-step algorithm usually performs better in terms of the number of targets found, as one would expect. In our experiments we will mainly focus on comparing our algorithm with myopic two-step approximate policy.

We also consider a simple baseline which we call RANDOM-GREEDY (RG). Here we randomly select points to query (exploration) during the first half of the budget, and select the remainder using greedy selection (exploitation). Although naïve, this policy adapts to the budget.

[4] https://github.com/rmgarnett/active_learning

Table 3.1: CiteSeer^x (left) and BMG (right) data: Average number of targets found by the one- and two-step myopic policies and ENS with different five budgets, varying from 100 to 900, at specific time steps. The performance of the best method at each time waypoint is in bold.

CiteSeer^x data					
	query number				
policy	100	300	500	700	900
RG	19.7	60.0	104	140	176
IMS	26.3	86.3	147	214	281
one-step	25.5	80.5	141	209	273
two-step	24.9	89.8	155	220	287
ENS–900	25.9	94.3	163	239	**308**
ENS–700	28.0	105	188	**259**	
ENS–500	28.7	**112**	**189**		
ENS–300	26.4	105			
ENS–100	**30.7**				

BMG data					
	query number				
policy	100	300	500	700	900
RG	48.6	144	243	336	427
IMS	93.6	276	451	629	799
one-step	90.8	273	450	633	798
two-step	91.0	273	452	632	802
ENS–900	89.0	270	453	635	**815**
ENS–700	91.3	276	460	**645**	
ENS–500	92.4	279	**466**		
ENS–300	92.8	**279**			
ENS–100	**94.5**				

We further compare with the score function proposed by [95], which we refer to as IMS:

$$\text{IMS}(x) = \Pr(y = 1 \mid x, \mathcal{D})\big(1 + \alpha\, \text{IM}(x)\big); \tag{3.10}$$

where $\text{IM}(x)$ measures the "expected impact", the sum of the raised probabilities x results in if it is positive. Note that it is difficult to determine the tradeoff parameter α without (expensive) cross validation. The empirical results in [95] indicate that $\alpha = 10^{-4}$ performs well on average; we will fix this value in our experiments.

The probability model $\Pr(y = 1 \mid x, \mathcal{D})$ we will adopt is the k-nearest-neighbor (k-NN) classifier as described in 3.5.1. Note IMS was proposed together (but orthogonally) with a graph model for the probability, which is computationally infeasible ($\mathcal{O}(n^3)$) for our datasets. So we also use k-NN model for IMS.

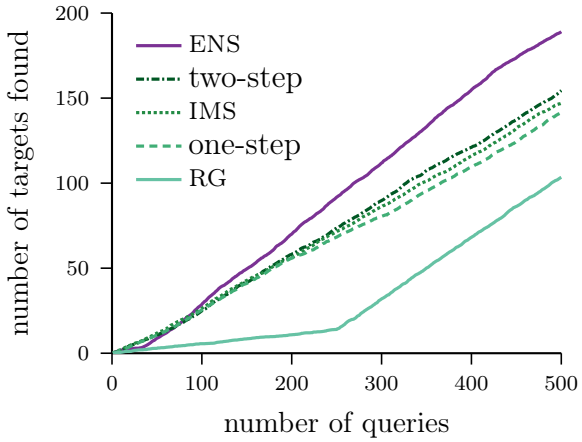


Figure 3.2: The learning curve of our policy and other baselines on the NeurIPS dataset.

3.7.1 Finding NeurIPS Papers From CiteSeer^x

For our first real data experiment, we consider a subset of the CiteSeer^x citation network, first described in [27]. This dataset comprises 39 788 computer science papers published in the top-50 most-popular computer science venues. We form an undirected citation network from these papers. The target class is papers published in the NeurIPS proceedings; there are 2 190 such papers, 5.5% of the whole dataset. Note that distinguishing NeurIPS papers in the citation network is not an easy task, because many other highly related venues such as ICML, AAAI, IJCAI, etc. are also among the most-popular venues. A feature vector for each paper is computed by performing graph principal component analysis [22] on the citation network and retaining the first 20 principal components.

We select a single target (i.e., a NeurIPS paper) uniformly at random to form an initial training set. The budget is set to $t = 500$, and we use $k = 50$ in the k-NN model. These parameters match the choices in [27]. We use each policy to sequentially select t papers for labeling. The experiment was repeated 20 times, varying the initial seed target. Figure 3.2 shows the average number of targets found for each method as a function of the number of

queries. We first observe that the ranking of the performance is ENS, two-step, IMS, one-step, and RG, and our policy outperforms the two-step policy in this task by a large margin. The mean difference in number of targets found at termination vs. two-step is 34.6 (189 vs. 155), an improvement on average of 22%. A two-sided paired t-test testing the hypothesis that the average difference of targets found is zero returns a p-value of $p < 10^{-4}$, and a 95% confidence interval on the increase in number of targets found of $[19.80, 49.30]$.

Another interesting observation is that during the initial $\sim$80 queries, ENS actually performs *worse* on average than all baseline policies except RG, after which it quickly outperforms them. This feature perfectly illustrates an automatic exploration–exploitation transition made by our policy. As we are always cognizant of our budget, we spend the initial stage thoroughly exploring the domain, without immediate reward. Once complete, we exploit what we learned for the remainder of the budget. This tradeoff happens automatically and without any need for an explicit two-stage approach or arbitrary tuning parameters.

Varying the Budget. A distinguishing feature of our method is that it always takes the remaining budget into consideration when selecting a point, so we would expect different behavior with different budgets. We repeated the above experiment for budgets $t \in \{100, 300, 500, 700, 900\}$, and report in Table 3.1 the average number of targets found at these time points for each method. We have the following observations from the table. First, ENS performs better than all other baseline policies for every budget. Second, ENS is able to adapt to the specified budget. For example, when comparing performance after 100 queries, ENS–100 has located many more targets than the ENS methods with greater budgets, which at that time are still strongly rewarding exploration. A similar pattern holds when comparing other pairs of ENS variations, with one minor exception.

Table 3.2: Number of active compounds found by various active search policies at termination for each fingerprint, averaged over 120 active classes and 20 experiments. Also shown is the difference of performance between ENS and two-step lookahead and the results of the corresponding paired t-test.

	policy					t-test results			
fingerprint	100-NN	RG	one-step	two-step	ENS	difference	p-value	95% CI	
ECFP4	189	189	289	297	**303**	5.29	1.76×10^{-3}	2.01	8.56
GpiDAPH3	134	170	255	261	**276**	14.8	3.90×10^{-13}	11.2	18.4

3.7.2 Finding Bulk Metallic Glasses

Our next dataset considers an application from materials science: discovering novel alloys forming bulk metallic glasses (BMGs). BMGs have numerous desirable properties, including high toughness and good wear resistance compared to crystalline alloys. We compiled a database of 118 678 known alloys from the materials literature (e.g., [1, 50]), an extension of the dataset from [97]. Of these, 4 746 ($\sim$4%) are known to exhibit glass-forming ability, which we define to be targets. We conduct the same experiments described for the CiteSeer^x data above and show the results in Table 3.1 (on the right). We can see the results again demonstrate our policy's superior performance over all other methods, and its ability of adapting to the remaining budget.

3.7.3 Drug Discovery

We further conduct experiments on a massive database of chemoinformatic data. The basic setting is to screen a large database of compounds searching for those that show binding activity against some biological target. This is a basic component of drug-discovery pipelines. The dataset comprises 120 activity classes of human biological importance selected from the

Binding DB [61] database. For each activity class, there are a small number of compounds with significant binding activity; the number of targets varies from 200 to 1 488 across the activity classes. From these we define 120 different active search problems. There are also 100 000 presumed inactive compounds selected at random from the ZINC database [89]; these are used as a shared negative class for each of these problems. For each compound, we consider two different feature representations, also known as chemoinformatic fingerprints, called ECFP4 and GpiDAPH3. These fingerprints are binary vectors encoding the relevant chemical characteristics of the compounds; see [25] for more details.[5] So in total we have 240 active search problems, each with more than 100 000 points, and with targets less than 1.5%.

As is standard in this setting, we compute fingerprint similarities via the Jaccard index [44], which are used to define the weight matrix of the k-NN model from above, setting $k = 100$ for all the experiments. For active search policies, we again randomly select one positive as the initial training set, and sequentially query $t = 500$ further points. We also report the performance of a baseline where we randomly sample a stratified sample of size 5% of the database ($\sim$5 000 points, more than 10 times the budget of the active search policies). From this sample, we train the same k-NN model, compute the active probability of the remaining points, and query the 500 points with the highest posterior activity probability. All experiments were repeated 20 times, varying the initial training point. Note we did not test IMS on these data due to computational expense. Our policy nominally has higher time complexity, but our pruning strategy can reduce the computation significantly in practice, as we show in Section 3.7.6.

[5] We did not conduct experiments on the MACCS fingerprint. It was inferior in the findings of [25]. A reviewer of [44] noted that it is no longer used, due to clear underperformance compared to, e.g., ECFP4 and GpiDAPH3.

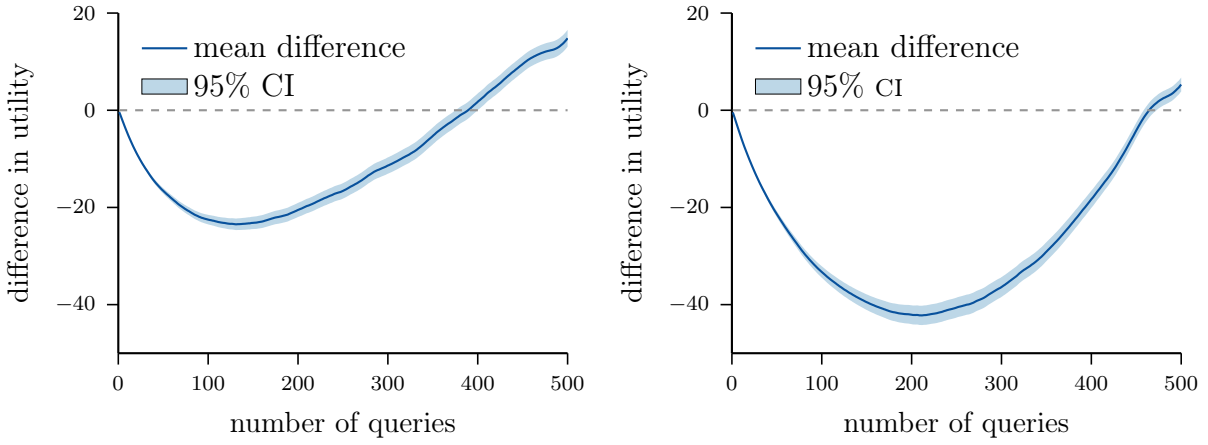


Figure 3.3: The average difference in cumulative targets found between ENS and the two-step policy, averaged over 120 activity classes and 20 experiments on (a) ECFP4 and (b) GPIDAPH3 fingerprint.

Table 3.2 summarizes the results. First we notice that all active search policies perform much better than the recall of a simple classification algorithm, even though they observe less than one-tenth the data. Interestingly, even the naïve random-greedy (RG) policy performs much better than this baseline, albeit much worse than other active search policies. The two-step policy is again better than the greedy policy for both fingerprints, which is consistent with the results reported in [25]. The ENS policy performs significantly better than two-step lookahead; a two-sided paired t-test overwhelmingly rejects the hypothesis that the performance at termination is equal in both cases.

Figure 3.3 shows the mean difference in cumulative targets found between ENS and the two-step policy. Again, we very clearly observe the automatic trade-off between exploration and exploitation by our method. In the initial stage of the search, we explore the space without much initial reward, but around query 100 or 200, our algorithm switches automatically to exploitation, outperforming the myopic policy significantly at termination. The mean difference curves for the other datasets are similar.

3.7.4 Compare with Naive Exploration/Exploitation Approaches

Active search can be seen as a special paradigm of active learning where the key challenge is to carefully balance exploration (learning) and exploitation (search). If we have a fully learned model, then active search is trivial, since we only need to retrieve the points with highest probabilities. In practice, the budget might never be enough for a model to be fully learned, that's why we need to carefully allocate the budget for learning and search. One naive approach is to first spend some budget for learning, then use the remaining budget for search. The results in Table 3.2 have shown that this naive approach is much worse than active search policies even granted 10 times more budget just for learning. Here we conduct more experiments in this regard motivated by the reviewers suggestions [6].

First, we implemented a strategy proposed by one of the reviewers, which we call *uncertain-then-greedy* (UTG). We first perform uncertainty sampling (arguably the most-popular active learning method) for a portion of the budget, then switch to greedy search. The transition point is controlled by a hyperparameter $r \in (0, 1)$: the first $100r\%$ of the budget is used for active learning (exploration), and the remaining $100(1-r)\%$ is used for exploitation. We ran this policy on all three types of datasets described previously for $r \in \{0.1, 0.2, \ldots, 0.9\}$, with a batch size of 1 and budget $T = 500$. We repeated this experiments the same number of times with the same initial random seeds used for the other policies. The results are plotted in Figure 3.4, comparing with one- and two-step lookahead and ENS. We can see that on the CiteSeer^x dataset this naïve approach indeed beats the greedy one-step lookahead policy, and approaches the performance of two-step lookahead when $r = 0.8$, but performs far worse than ENS; on the other two datasets, UTG does not show any advantage. Although we could

[6] This part of work is done during the rebuttal of our submission to NeurIPS 2018 on batch active search, which is the content of Chapter 4. But we only used the sequential setting for this rebuttal, so we move the content to this chapter.

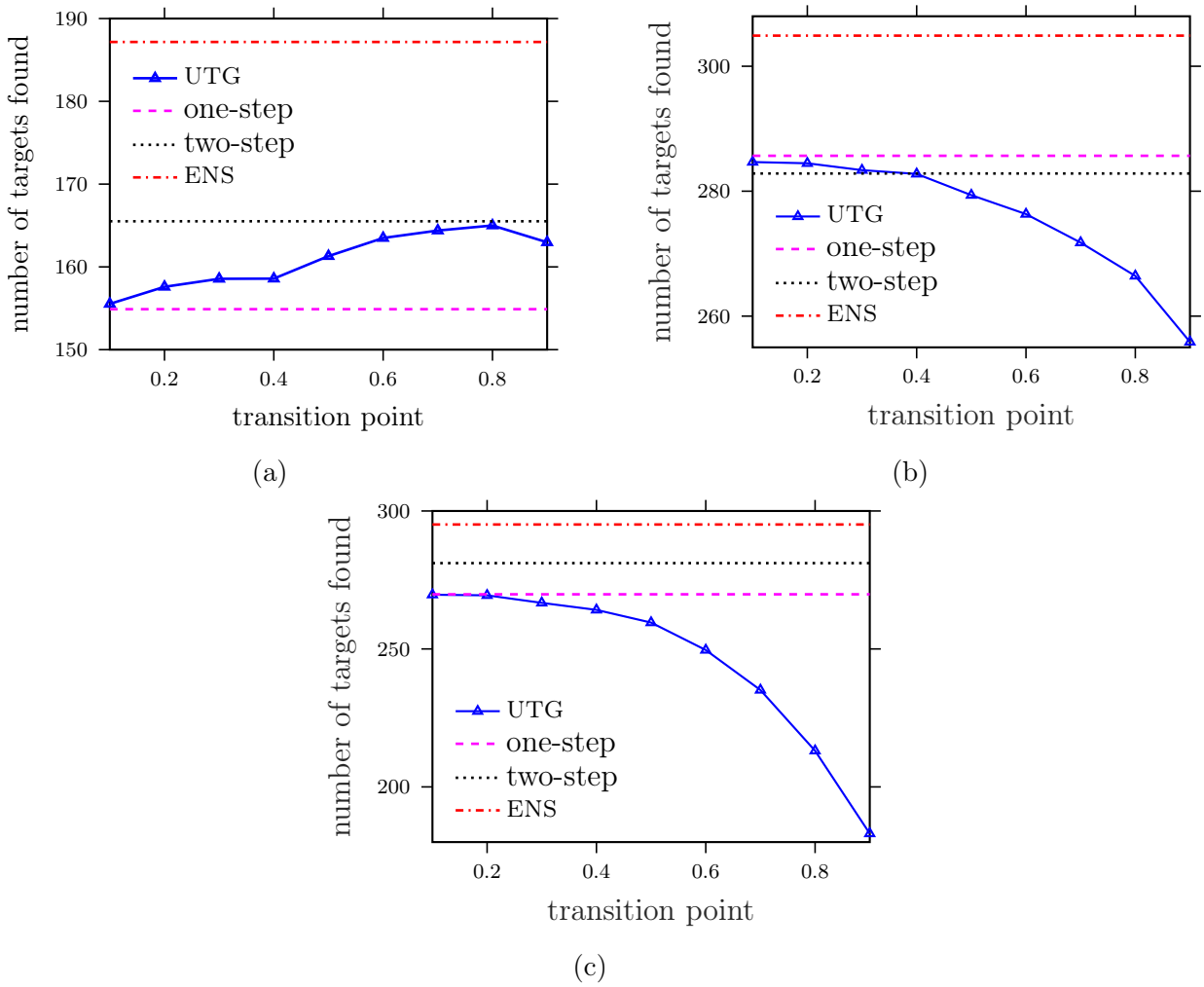


Figure 3.4: Comparion of active search policies with a naive exploration-exploitation approach called uncertain-then-greedy (UTG), which performs uncertainty sample for the first 100r% of the budget and then greedy sampling in the remaining iterations, where r is a hyperparameter controlling the transition point. (a) CiteSeer^x Dataset. (b) BMGs dataset. (c) Drug discovery datasets.

probably strengthen this baseline with more-advanced active learning or active search policies, it will never be clear how to best choose the transition hyperparameter r. Our approach, on the other hand, automatically transitions from exploration to exploitation in line with the optimal policy.

3.7.5 Compare with UCB-Style Policy

Finally, we have also considered the following UCB-style [3] score function:

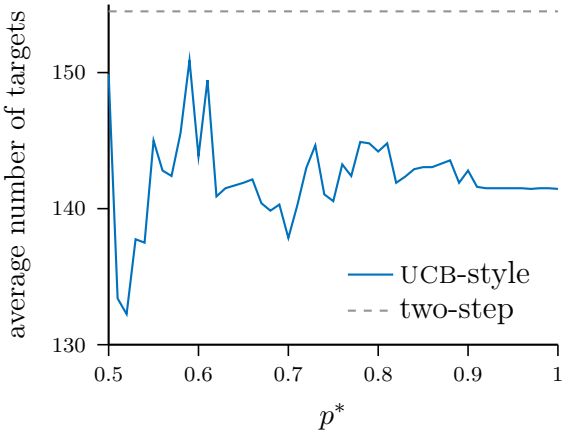
$$\alpha(x, \mathcal{D}) = \pi + \gamma\sqrt{\pi(1-\pi)}, \tag{3.11}$$

where $\pi = \Pr(y = 1 \mid x, \mathcal{D})$ and γ is a tradeoff parameter. The UCB score function is very popular and is the essence of the methods in [88, 92] developed for Gaussian processes, including GP-SELECT. We considered various γ values and our experiments show that it is no better than two-step lookahead.

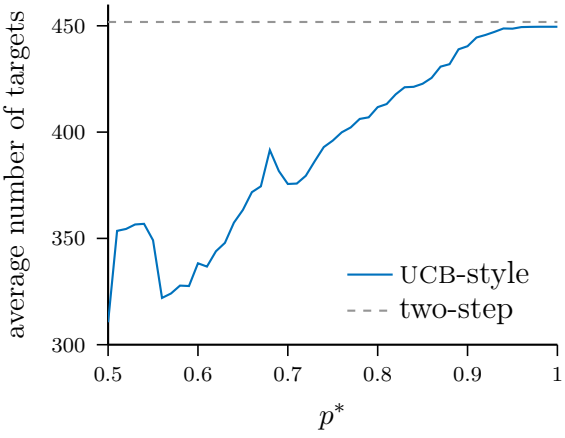
The results of this policy (maximizing $\alpha(x, \mathcal{D})$) varying the hyperparameter γ are shown in Figure 3.5, averaged over 20 experiments. Note with a fixed γ, the score α is maximized at some probability $\pi = p^*$; it is easy to derive that

$$p^* = \frac{1}{2} + \frac{1}{2\sqrt{\gamma^2 + 1}} \tag{3.12}$$

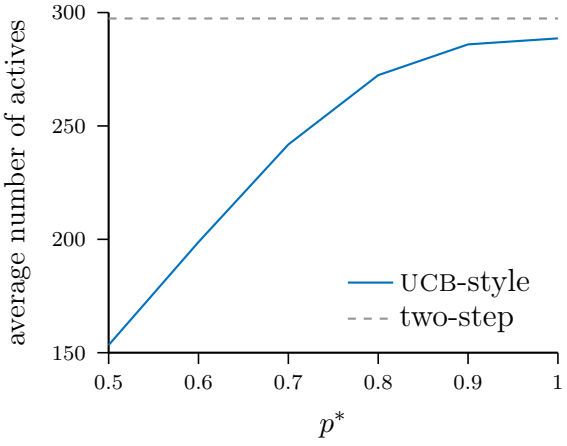
by setting the derivative to zero. To better present the results, we use $p^* \in [0.5, 1]$ as the hyper-parameterization of the score in Figure 3.5. In summary, on the CiteSeer^x dataset, as shown in Figure 3.5(a), the performance is maximized at some p^* near 0.6, but the curve does not seem to be smooth. When we average the performance on 2 400 experiments on the ECFP4 data, as shown in Figure 3.5(c), we see that the α score is monotonically performing better with larger p^* (or smaller γ), and converges to the greedy policy ($p^* = 1$).



(a) CiteSeer^x data



(b) BMG data



(c) ECFP4 data

Figure 3.5: Number of targets found by the UCB-style policy (3.11), as a function of the hyperparameter p^* as derived in (3.12), averaged over 20 experiments. Note for CiteSeer^x and BMG datasets, the grid size of p^* is 0.01, but for ECFP4, we can only afford grid size of 0.1. To put these results into perspective, we also show the two-step performances by the red horizontal line, indicating two-step performs better than the UCB-style policy on all three domains. All these results are with budget 500.

3.7.6 Pruning Effectiveness

To investigate how pruning can improve the efficiency of computing the policy, we computed the average number of pruned points across all $120 \times 20 \times 500 = 3\,000\,000$ iterations of active search, for each fingerprint. Table 3.3 shows the effectiveness of pruning. On average about

Table 3.3: Average number of pruned points in each iteration for the two chemical datasets.

fingerprint	# pruned	# total	pruned %
ECFP4	94 995	100 518	94.5%
GpiDAPH3	93 173	100 518	92.7%

93% of the unlabeled points are pruned, dramatically improving the computational efficiency by approximately a corresponding linear factor. The time for each experiment was effectively reduced from on the order of one day to that of one hour.

3.8 Conclusion

In this chapter we proved the theoretical hardness of active search and proposed an well-motivated and empirically better-performing policy for solving this problem. In particular, we proved that no polynomial-time algorithm can approximate the expected utility of the optimal policy within a constant approximation ratio. We then proposed a novel method, efficient nonmyopic search (ENS), for the active search problem. Our method approximates the Bayesian optimal policy by computing, conditioned on the location of the next point, how many targets are expected at termination, if the remaining budget is spent simultaneously. By taking the remaining budget into consideration in each step, we are able to automatically balance exploration and exploitation. Despite being nonmyopic, ENS is efficient to compute because future steps are flattened into a single batch, in contrast to the recursive simulation required when computing the true expected utility. We also derived an effective pruning strategy that can reduce the number of candidate points we must consider at each step, which can further improve the efficiency dramatically in practice. We conducted a massive empirical

evaluation that clearly demonstrated superior overall performance on various domains, as well as our automatic balance between exploration and exploitation.

Given the hardness result we proved, in general there is little point to require more of an algorithm than superior empirical performance. However, one exciting future direction is to understand, under what conditions (e.g., some assumption about the structure of problem instances) we can find efficient algorithms with guarantees.

Chapter 4

Batch Budgeted Active Search

Chapter 3 and all previous investigations on active search focused on *sequential active search* (SAS), where only one point is queried at a time. However, in many real applications, we can query a *batch* of multiple points simultaneously. For example, modern high-throughput screening technologies for drug discovery can process microwell plates containing 96+ compounds at a time. No policies designed for this batch active search setting are currently available.

In this chapter, we investigate *batch active search* (BAS) in budgeted setting from both theoretical and practical perspectives. We first derive the Bayesian optimal policy for BAS, and show that its time complexity in general is dauntingly high, except in the trivial one-step (myopic) case. We then prove an asymptotic lower bound on the expected performance gap between the optimal sequential and batch policies.

Next we consider practical concerns such as effective policy design. We generalize the efficient nonmyopic sequential policy (ENS) as introduced in Chapter 3 to the batch setting. The nonmyopia of ENS is automatically inherited, but efficiency is lost as the batch version involves combinatorial optimization (i.e., set function maximization). We propose and study two

efficient approximation strategies. The first strategy is a sequential simulation, where we simulate sequential ENS to construct a batch using a fictional labeling oracle. The second strategy is greedily maximizing the marginal gain to our batch ENS score, motivated by our conjecture that the inherent batch score is submodular. We prove that sequential simulation of the one-step Bayesian optimal policy with a *pessimistic* oracle (i.e., one that always outputs negative labels) near-optimally maximizes the probability that *at least one point in the batch* is positive. This theoretical support of pessimism is in contrast to other settings such as Bayesian optimization, where pessimism has been used as a *heuristic* for batch policies.

We also improve the pruning techniques developed in Chapter 3, considerably to reduce the computational overhead of our proposed policies in practice. We demonstrate a connection with lazy evaluation [18] and show that our pruning strategy can provide a speedup of over 50 times in a drug discovery setting.

Finally, we conduct thorough experiments on data from three domains: a citation network, material science, and drug discovery. In total we study 14 policies: the one-step optimal batch policy, 12 sequential simulation policies (three sequential policies combined with four fictional oracles), and greedy maximization of the batch version of ENS. We observe that ENS-based (nonmyopic) policies almost always provide a significant improvement in performance. Two policies are particularly notable: sequential simulation of ENS with a pessimistic oracle and greedy maximization of batch ENS. The latter is shown to be more robust for larger batch sizes.

4.1 Bayesian Optimal Policy

We begin our investigation by generalizing the Bayesian *optimal* policy for sequential active search to batch setting. Recall we express our preference over different datasets $\mathcal{D} = \{(x_i, y_i)\}$ through a natural utility: $u(\mathcal{D}) = \sum_{y_i \in \mathcal{D}} y_i$, which simply counts the number of targets in $\mathcal{D}$. Occasionally we will use the notation $u(Y)$ for $u(\mathcal{D})$ when $\mathcal{D} = (X, Y)$. We now consider the problem of sequentially choosing a set of T (a given budget) points $\mathcal{D}$ with the goal of maximizing $u(\mathcal{D})$. In the batch setting, for each query we must select a batch of b points and will then observe all their labels at the same time. We use $X_i = \{x_{i,1}, x_{i,2}, \dots x_{i,b}\}$ to denote a batch of points chosen during the ith iteration, and $Y_i = \{y_{i,1}, y_{i,2}, \dots y_{i,b}\}$ the corresponding labels. We use $\mathcal{D}_i = \{(X_k, Y_k)\}_{k=1}^{i}$ to denote the observed data after $i \leq t$ batch queries, where $t = \lceil T/b \rceil$.

Again we assume a probability model $\mathcal{P}$ is given, providing the posterior marginal probability $\Pr(y \mid x, \mathcal{D})$ for any point $x \in \mathcal{X}$ and observed dataset $\mathcal{D}$. At iteration $i+1$ (given observations $\mathcal{D}_i$), the Bayesian optimal policy chooses a batch X_{i+1} maximizing the expected utility at termination, recursively assuming optimal continued behavior:

$$X_{i+1} = \arg\max_{X} \mathbb{E}\big[u(\mathcal{D}_t \setminus \mathcal{D}_i) \mid X, \mathcal{D}_i\big]. \tag{4.1}$$

Note that we use slightly different notations from Chapter 3 by excluding $\mathcal{D}_i$ directly from $\mathcal{D}_t$, since the utility function is additive and $u(\mathcal{D}_i)$ is a known constant at this point.

To derive the expected utility, we again adopt the standard technique of backward induction. The base case is when only one batch is left ($i = t - 1$). The expected utility resulting from a

proposed final batch X is then

$$\mathbb{E}\big[u(\mathcal{D}_t \setminus \mathcal{D}_{t-1}) \mid X, \mathcal{D}_{t-1}\big] = \mathbb{E}_{Y|X,\mathcal{D}_{t-1}}\big[u(Y)\big] = \textstyle\sum_{x \in X} \Pr(y = 1 \mid x, \mathcal{D}_{t-1}), \tag{4.2}$$

where $\mathbb{E}_{Y|X,\mathcal{D}_i}$ is the expectation over the joint posterior distribution of Y (the labels of X) conditioned on $\mathcal{D}_i$. In this case, designing the optimal batch (4.1) by maximizing the expected utility is trivial: we select the points with the highest probabilities of being targets, reflecting pure exploitation. This optimal batch can then be found in $\mathcal{O}(n \log b)$ time using, e.g., min-heap of size b.

In general, when $i \leq t-1$, the expected terminal utility resulting from choosing a batch X at iteration $i+1$ and acting optimally thereafter can be written as a Bellman equation as follows:

$$\mathbb{E}\big[u(\mathcal{D}_t \setminus \mathcal{D}_i) \mid X, \mathcal{D}_i\big] = \textstyle\sum_{x \in X} \Pr(y = 1 \mid x, \mathcal{D}_i) + \mathbb{E}_{Y|X,\mathcal{D}_i}\Big[\max_{X'} \mathbb{E}\big[u(\mathcal{D}_t \setminus \mathcal{D}_{i+1}) \mid X', \mathcal{D}_{i+1}\big]\Big], \tag{4.3}$$

where the first term represents the expected utility resulting immediately from the points in X, and the second part is the expected future utility from the following iterations.

The most interesting aspect of the Bayesian optimal policy is that these immediate and future reward components in (4.3) can be interpreted as automatically balancing exploitation (immediate utility) and exploration (expected future utility given the information revealed by the present batch).

However, without further assumptions on the joint label distribution $\mathcal{P}$, exact maximization of (4.3) requires enumerating the whole search tree of the form $\mathcal{D}_i \to X_{i+1} \to Y_{i+1} \to \cdots \to X_t \to Y_t$. The branching factor of the X layers is $\binom{n}{b}$, as we must enumerate all possible

batches. The branching factor of the Y layers is 2^b, as we must enumerate all possible labelings of a given batch. So the total complexity of a naïve implementation computing the optimal policy at iteration $i+1$ would be a daunting $\mathcal{O}\big((2n)^{b(t-i)}\big)$. The running time analysis in [27] is a special case of this result where $b = 1$.

The optimal policy is clearly computationally infeasible, so we must resort to suboptimal policies to proceed in practice. One reasonable and practical alternative is to adopt a myopic lookahead approximation to the optimal policy. A *greedy* (one-step lookahead) approximation, which always maximizes the expected marginal gain in (4.2), constructs each batch by selecting the points with highest probability of being a target. We will refer to this policy as greedy-batch, and this will serve as a natural baseline batch policy for active search.

4.2 Adaptivity Gap

For purely sequential policies (i.e., $b = 1$), every point is chosen based on a model informed by all previous observations. However, for batch policies ($b > 1$), points are typically chosen with less information available. For example, in the extreme case when $b = T$, every point in our budget must be chosen before we have observed anything, hence we might reasonably expect our search performance to suffer. Clearly there must be an inherent cost to batch policies compared to sequential policies due to a loss of adaptivity. How much is this cost?

We have proven the following lower bound on the inherent "cost of parallelism" in active search:

Theorem 2. *There exist active search instances with budget T, such that $\frac{\text{OPT}_1}{\text{OPT}_b}$ is $\Omega\big(\frac{b}{\log T}\big)$, where OPT_x is the expected number of targets found by the optimal batch policy with batch size $x \geq 1$.*

Proof sketch. We construct a special type of active search instance where the location of a large trove of positives is encoded by a binary tree, and a search policy must take the correct path through the tree to decode a treasure map pointing to these points. We design the construction such that a sequential policy can easily identify the correct path by walking down the tree directed by the labels of queried nodes. A batch policy must waste a lot queries decoding the map as the correct direction is only revealed after constructing an entire batch. We show that even the optimal batch policy has a very low probability of identifying the location of the hidden targets quickly enough, so that the expected utility is much less than that of the optimal sequential policy. A detailed proof is given in Appendix B. □

Thus the expected performance ratio between optimal sequential and batch policies, also known as *adaptivity gap* in the literature [2], is lower bounded linearly in batch size. This theorem is not only of theoretical interest: it can also provide practical guidance on choosing batch sizes. Indeed, in drug discovery, modern high-throughput screening technologies provide many choices for batch sizes; understanding the inherent loss from choosing larger batch sizes provides valuable information regarding the tradeoff between efficiency and cost.

4.3 Efficient Nonmyopic Approximations

The greedy-batch policy is myopic in the sense that each decision represents pure exploitation: the future reward is always assumed to be zero, and the remaining budget is not taken into

consideration. Here we generalize the sequential ENS as introduced in Chapter 3 to the batch setting and propose two techniques to approximately compute it.

Our proposed adaptation of ENS to batch setting can be motivated with the following question: how many targets would we expect to find if, after selecting the current batch, we spent the entire remaining budget simultaneously? If this were the case, then the maximum future utility could be computed without recursion:

$$\mathbb{E}[u(\mathcal{D}_t \setminus \mathcal{D}_i) \mid X, \mathcal{D}_i] = \sum_{x \in X} \Pr(y = 1 \mid x, \mathcal{D}_i) + \mathbb{E}_{Y|X,\mathcal{D}_i}\left[\max_{X':|X'|=T-b-|\mathcal{D}_i|} \mathbb{E}\left[u(Y') \mid X', \mathcal{D}_i, X, Y\right]\right]. \quad (4.4)$$

Note the optimal final action simply selects the points with the highest $T-b-|\mathcal{D}_i|$ probabilities, allowing the expected future reward to be computed exactly and efficiently. We may use this insight to rewrite (4.4) as (using $V(X \mid \mathcal{D}_i)$ as shorthand for the expected utility from selecting X given $\mathcal{D}_i$):

$$V(X \mid \mathcal{D}_i) = \sum_{x \in X} \Pr(y = 1 \mid x, \mathcal{D}_i) + \mathbb{E}_{Y|X,\mathcal{D}_i}\left[\sum'_{T-b-|\mathcal{D}_i|} \Pr\left(y' = 1 \mid x', \mathcal{D}_i, X, Y\right)\right]. \quad (4.5)$$

Here we have adopted the notation $\sum'_s$ from ENS in Chapter 3 to denote the sum of the top s probabilities over the unlabeled points, $x' \in \mathcal{X} \setminus (\mathcal{D}_i \cup X)$. Same as ENS, the batch version's underlying assumption is the remaining unlabeled points after this batch are conditionally independent, so that there is no need to recursively enumerate the search tree. This assumption might seem unrealistic at first, but when many *well-spaced* points are observed, we note they might approximately "D-separate" the remaining unlabeled points. Further, ENS naturally encourages the selection of well-spaced points (targeted exploration) in the initial state of the search (see Figure 3.1).

The nonmyopia of (4.5) is automatically inherited in generalizing from sequential to batch setting due to explicit budget awareness. Unfortunately, the efficiency of the sequential ENS policy is not preserved. Direct maximization of (4.5) still requires combinatorial search over all subsets of size b. Moreover, to evaluate a given batch, we need to enumerate all its possible labelings (2^b in total) to compute the expectation in the second term. Accounting for the cost of conditioning and summing the top probabilities, the total complexity would be $\mathcal{O}\big((2n)^b\, n \log T\big)$.

We propose two strategies to tackle these computational problems below.

4.3.1 Sequential Simulation

The cost of computing the proposed batch policy has exponential dependence on the batch size $b > 1$. To avoid this, our first idea is to reduce BAS to SAS ($b = 1$). We select points one at a time to add to a batch by maximizing the sequential ENS score (i.e., (4.5) with $b = 1$). We then use some fictional labeling oracle $\mathcal{L}\colon \mathcal{X} \to \{0, 1\}$ to simulate its label and incorporate the observation into our dataset. We repeat this procedure until we have selected b points. Note that we could use this basic construction replacing ENS by any other sequential policy π, such as the one-step or two-step Bayesian optimal policies [27].

We will see that the behavior of the fictional labeling oracle has large influence on the behavior of resulting search policies. Here we will consider four fictional oracles: (1) sampling, where we randomly sample a label from its marginal distribution; (2) most-likely, where we assume the most-likely label; (3) pessimistic, where we always believe all labels are negative; and (4) optimistic, where always believe all labels are positive.

Sequential simulation is a common *heuristic* in similar settings like batch Bayesian optimization, as we will discuss in detail in the next section. Here we provide some mathematical rationale of this procedure in a special case: the one-step optimal (greedy) search policy combined with the pessimistic oracle.

Proposition 1. *The batch constructed by sequentially simulating the greedy active search policy with a pessimistic oracle near-optimally maximizes the probability that* at least one *of the points in the batch is positive, assuming that marginal target probabilities of unlabeled points are nonincreasing when conditioning on a negative observation.*

Proof sketch. We show that the probability of a batch having at least one positive is a monotone submodular set function, and that sequentially simulating the one-step policy with the pessimistic oracle equivalently maximizes the marginal gain of this function. Therefore, it is near-optimal [69]. A detailed proof is as follows.

Proof. The probability of a batch $\mathcal{B} = \{x_1, x_2, \ldots, x_b\}$ having at least one positive can be written as

$$g(\mathcal{B}) = 1 - \Pr(y_1 = 0 \wedge y_2 = 0 \wedge \cdots \wedge y_b = 0). \tag{4.6}$$

It's easy to see $g(\emptyset) = 0$, since the probability of an empty set having at least one positive is zero. It's also easy to see $g(\mathcal{B})$ is monotone. That is, for any $A \subseteq B \subseteq \mathcal{X}$, $g(A) \leq g(B)$. Now we show $g(\mathcal{B})$ is submodular. For any set $\mathcal{B}$, define $\mathcal{B} = 0$ as the event that $\forall x \in \mathcal{B}$, its label

$y = 0$, i.e., $y_1 = 0 \wedge y_2 = 0 \wedge \cdots \wedge y_b = 0$. The marginal gain of any $x \in \mathcal{X}$ (with label y) is

$$\begin{aligned}
& g(\mathcal{B} \cup \{x\}) - g(\mathcal{B}) \\
=& 1 - \Pr(\mathcal{B} = 0 \wedge y = 0) - (1 - \Pr(\mathcal{B} = 0)) \\
=& \Pr(\mathcal{B} = 0) - \Pr(\mathcal{B} = 0 \wedge y = 0) \\
=& \Pr(\mathcal{B} = 0) - \Pr(y = 0 \mid \mathcal{B} = 0) \Pr(\mathcal{B} = 0) \\
=& \Pr(y = 1 \mid \mathcal{B} = 0) \Pr(\mathcal{B} = 0). \qquad (4.7)
\end{aligned}$$

Let $A \subseteq B \subseteq \mathcal{X}$, $x \in \mathcal{X} \setminus B$ and its label y, we have

$$\begin{aligned}
g(A \cup \{x\}) - g(A) &= \Pr(y = 1 \mid A = 0) \Pr(A = 0); \\
g(B \cup \{x\}) - g(B) &= \Pr(y = 1 \mid B = 0) \Pr(B = 0).
\end{aligned}$$

Since $A \subseteq B$, we have $\Pr(B = 0) \leq \Pr(A = 0)$. We also have $\Pr(y = 1 \mid B = 0) \leq \Pr(y = 1 \mid A = 0)$ due to the (very reasonable) assumption that observing more negative points does not increase the probability of any other point being positive. Hence

$$g(B \cup \{x\}) - g(B) \leq g(A \cup \{x\}) - g(A). \qquad (4.8)$$

Therefore g is a submodular function.

Observing (4.7), we see sequentially simulating one-step Bayesian optimal policy (i.e., choose the point with maximum probability) with a pessimistic (always-0) fictional oracle is exactly greedily maximizing the marginal gain of $g(\mathcal{B})$. By the classical results in [69], we know this greedy solution has approximation ratio $1 - 1/e \approx 0.6321$. □

This proposition is inspired by [93], applying Bayesian active learning to a biological application. The goal of [93] was to find a peptide as short as possible that is substrate for two protein-modifying enzymes. They showed that under their setting, for any set of peptides S and the currently known shortest length ℓ, the probability of improvement (i.e., S contains a positive peptide shorter than ℓ) is a submodular function, and greedy maximization is equivalent to choosing from the peptides shorter than ℓ that has maximum probability of being positive, conditioned on all previously chosen ones being negative.

Note the assumption in this proposition simply means the probability model does not involve negative label correlations; the k-nn model used in our experiments satisfies this assumption.

With this result, it is not hard to see that sequentially simulating the greedy policy with an optimistic oracle greedily maximizes the probability that *all* points in the batch are positive. In this case, however, the corresponding set function is not submodular so we don't know if there are optimality guarantees.

Note we are not claiming that the objective of finding at least one positive serves as a good basis for batch active search; actually as we will see in our experiments, this is often much worse than other nonmyopic batch policies we propose. However, we believe this result provides theoretical insight that could shed light on other batch policies under similar settings. For example, this policy can be considered as an active search counterpart of a batch version of *probability of improvement* for Bayesian optimization [55].

4.3.2 Greedy Approximation

Our second strategy is motivated by our conjecture that (4.5) is a monotone submodular function under reasonable assumptions. If that is the case, then again a *greedy* batch construction returns a batch with near-optimal score [69]. We therefore propose to use a greedy algorithm to sequentially construct the batch by maximizing the marginal gain. That is, we begin with an empty batch $X = \emptyset$. We then sequentially add b points by adding the point maximizing the marginal gain:

$$x = \arg\max_x \Delta_V(x \mid X), \tag{4.9}$$

where

$$\Delta_V(x \mid X) = V(X \cup \{x\} \mid \mathcal{D}_i) - V(X \mid \mathcal{D}_i). \tag{4.10}$$

When b is large, this procedure is still expensive to compute due to the expectation term in (4.5), requiring $\mathcal{O}(2^b)$ operations to compute exactly. Here we approximate the expectation using Monte Carlo sampling with a small set of samples of the labels. Specifically, given a batch of points X, we approximate (4.5) with samples $S = \{\tilde{Y} : \tilde{Y} \sim Y \mid X, \mathcal{D}_i\}$:

$$V(X \mid \mathcal{D}_i) \approx \sum_{x \in X} \Pr(y = 1 \mid x, \mathcal{D}_i) + \frac{1}{|S|} \sum_{Y \in S} \left[\sum'_{T-b-|\mathcal{D}_i|} \Pr\left(y' = 1 \mid x', \mathcal{D}_i, X, Y\right) \right]. \tag{4.11}$$

We will call the batch policy described above batch-ENS. Note batch-ENS using *one* sample of the labels in a batch is similar to sequential simulation of ENS with the sampling oracle, though the two policies are motivated in different ways.

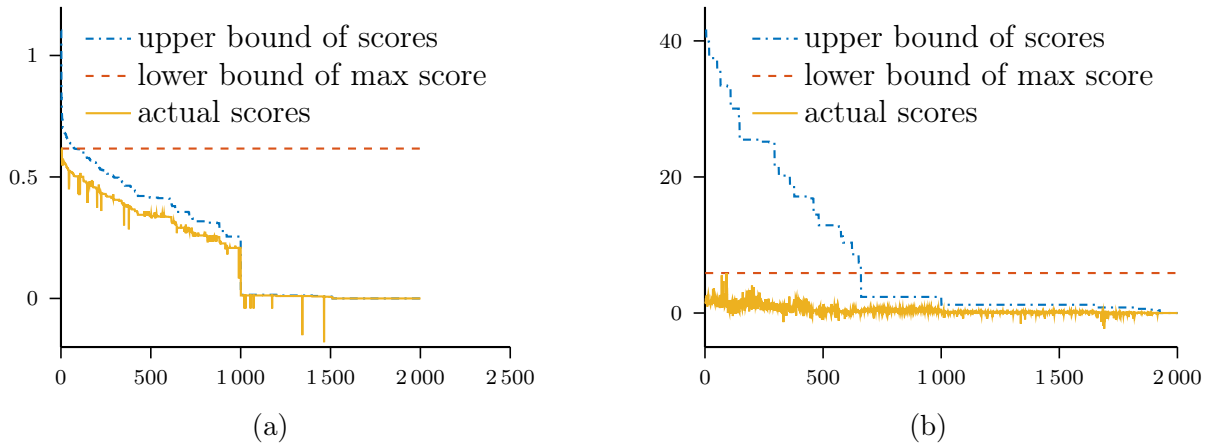


Figure 4.1: Illustration of pruning. The x-axis is the index of candidate points in descending order of the upper bounds, and the y-axis is the actual marginal gain of batch-ENS score as in Eq. (6) in the main text. These plots are generated from running batch-ENS on the CiteSeer^x data with budget $T = 500$ and batch size $b = 5$. There are 39 788 points, we only plot the first and last 1 000 points to have a better presentation. (a) At the time of choosing the first point ($k = 1$) of the 99th batch (i.e., $i = 98$) when $T - b - |\mathcal{D}_i| = 5$; here 99.61% of points are pruned. (b) At the time of choosing the first point ($k = 1$) of the 16th (i.e., $i = 15$) batch when $T - b - |\mathcal{D}_i| = 420$; 98.23% of the points are pruned.

4.3.3 Lazy Evaluation for Pruning

We previously developed a pruning technique for ENS in Chapter 3. The basic idea is: we first compute an upper bound $p^*(\mathcal{D})$ on marginal probabilities after observing $\mathcal{D}$ and one additional positive observation. We use this to compute an upper bound of the ENS score for each candidate point. We also compute the actual ENS score for the point with maximum probability, which serves as a global lower bound of the maximum score. Now any candidate point with upper bound less than the lower bound cannot possibly maximize the score and can be removed from consideration.

Here we further improve the pruning for ENS and generalize it to batch-ENS. First, instead of just computing a maximum probability bound $p^*(\mathcal{D})$, we can compute upper bounds of the highest $r \equiv T - b - |\mathcal{D}_i|$ probabilities $p_1^*(\mathcal{D}), p_2^*(\mathcal{D}), \ldots, p_r^*(\mathcal{D})$ after observing $\mathcal{D}$ and

one additional positive. This generalization is trivial for a k-nn model. We can now upper bound the $\sum'_{T-b-|\mathcal{D}_i|}$ term more tightly by $\sum_{j=1}^{r} p_j^*(\mathcal{D}_i)$ than by $(T-b-|\mathcal{D}_i|)p^*(\mathcal{D}_i)$. This upper-bounding technique for ENS can be straightforwardly generalized to batch-ENS, where we only need to apply this upper bound for each sample and average them.

Besides improving the upper bound, we can also improve the lower bound. As we continue to compute scores for more points, we can update the lower bound when observing a larger score. Tightening the bound enables more-aggressive pruning. To better use this improving lower bound, we first sort the candidates points in decreasing order of their upper bounds, and compute actual ENS scores in this order. This way, we never unnecessarily compute scores for points that might be pruned later. This is similar to the idea of *lazy evaluation* [19] for efficiently maximizing the GP-UCB [88] score.

Figure 4.1a illustrates the pruning in a representative iterations of batch-ENS on CiteSeer^x data with budget $T = 500$ and batch size $b = 5$. We see when the remaining budget is small ($T - b - |\mathcal{D}_i| = 5$), the upper bound is extremely tight, and in this example 99.61% of the points are pruned.

We give another illustration of pruning when the remaining budget ($T - b - |\mathcal{D}_i|$) is large in Figure 4.1b. We can see in this case the upper bound is much looser, but we are still able to prune 98.23% of the points.

4.4 Related Work

Active search (AS) and its variants have been the subject of a great deal of recent work [25, 27, 46, 63, 64, 92, 95, 98, 99, 100]; nevertheless, to the best of our knowledge, this is the first study on batch active search under this particular setting.

[98, 99] considered a different goal of batch active search: to find all or a given number of actives as soon as possible. Our goal, in contrast, is to find as many actives as possible in a given budget, which encourages more nonmyopic planning. Their proposed batch policy is to pick the most-likely positive points (those farthest from an SVM hyperplane), which is quite different from our more-principled approach using Bayesian decision theory. Their policy is an analog of the one-step (greedy) myopic policy in our treatment, which performs poorly as we will show in Section 6.7.

Batch policies have been studied extensively in active learning [10, 14, 16, 42]. In particular, [14] proposed an adaptive submodular objective function, and chose points greedily by maximizing the marginal gain. This algorithm is similar in spirit to our batch-ENS policy, though it is not known whether the batch-ENS function is submodular. They also proved a result similar in spirit to our Theorem 2 (also called "adaptivity gap" in [2]) to show that the price of parallelism is bounded irrespective of batch sizes. This theorem holds under the stochastic submodular maximization setting where the outcomes of variables are independent, which certainly does not apply in our case.

As mentioned before, AS can be considered as Bayesian optimization (BO) with binary observations and cumulative reward maximization on a finite domain. Numerous batch BO policies have been studied [30, 31, 33, 102] [30, 31] proposed q-EI, in which q points are selected simultaneously to maximize the expected improvement. They also used sequential

simulation to optimize the q-EI objective, and proposed two heuristic "fictional oracles" called the Kriging believer (KB) and constant liar (CL). KB sets the label of a chosen point to its posterior mean, and CL sets the label to be a chosen constant, such as the maximum, mean, or minimum of the observed values so far. This is similar to our pessimistic or optimistic oracles.

In the Gaussian process (GP) bandit optimization setting, [19] proposed GP-BUCB, a batch extension of the GP-UCB policy [88]. They also construct the batch by sequentially simulating the GP-UCB policy, where the values of the selected points are "hallucinated" with the posterior mean, equivalent to the Kriging believer heuristic for q-EI. A similar strategy was adopted also in [100] to identify the compounds with the top-k continuous-valued binding activities against an identified biological target. These approaches don't directly apply to our setting, where the target values are binary. In fact, in Chapter 3 we showed that a UCB-style policy adapted to the Bernoulli setting performs worse than a myopic two-step policy on a range of problems in their supplementary material.

4.5 Experiments

In this section, we comprehensively compare all our 14 proposed policies: (1) greedy-batch, coded as "greedy"; (2–13) sequential simulation, coded as "ss-P-O", where P (for policy) could be "one" (for one-step), "two" (for two-step), or "ENS", and O (for oracle) could be "s" (sampling), "m" (most-likely), "0" (pessimistic, i.e., always-0), or "1" (optimistic, i.e., always-1); (14) batch-ENS. Suggested by one of the the anonymous reviewers, we also compare these policies against another naïve baseline, which we call *uncertain-greedy* batch (UGB), where we build batches that simultaneously encourage exploration and exploitation by combining

the most uncertain points and the highest probability points. We use a hyperparamter $r \in (0, 1)$ to control the proportion, choosing the most uncertain points for $100r\%$ of the batch, and greedy points for the remaining $100(1-r)\%$ of the batch. We run this policy for $r \in \{0.1, 0.2, \ldots, 0.9\}$, and show the best result among them. As in Chapter 3, we consider data from three application domains: a citation network, material science, and drug discovery. We use k nearest neighbor (k-nn) with $k = 100$ as our probability model for the drug discovery datasets, and $k = 50$ for the other two datasets, same as in the sequential setting.

4.5.1 Finding NeurIPS Papers From CiteSeer^x

We use experimental settings matching Chapter 3. Specifically, we randomly select a single target (i.e., a NeurIPS paper) to form the initial observations $\mathcal{D}_0$, and repeat the experiment 100 times for each batch size and each policy. We set the budget T to 500. We show the average number of NeurIPS papers found at termination for batch sizes $b \in \{5, 10, 15, 20, 25\}$ in Table 4.1. If b does not divide T (e.g., $b = 15$), we take $t = \lceil T/b \rceil$, and only count the first T points. We also add the results for batch size 1 for reference. For batch-ENS, we use 16 samples; therefore for batch size 5 the expected future utility is computed exactly. We highlight the best result for each batch size in boldface. We conduct a paired t-test [7] for each other policy against the best one, and also emphasize those that are not significantly worse than the best with significance level $\alpha = 0.05$ in blue italics. We use this convention in all tables.

We summarize the patterns as follows: (1) for this dataset the uncertain-greedy batch (UGB) is actually a strong baseline; it performed mostly better than the myopic policies. But keep in mind we are reporting the best result for UGB varying the hyperparameter; in practice, it

[7] Wilcoxon signed rank tests give similar results.

Table 4.1: Results for CiteSeer^x data: Average number of targets found by various batch policies: greedy-batch, sequential simulation "ss-P-O" and batch-ENS, with batch sizes 5, 10, 15, 20, 25. The average is taken over 100 experiments. Highlighted are the best in each column and those not significantly worse than the best using a one-sided paired t test with significance level $\alpha = 0.05$.

	1	5	10	15	20	25
UGB	-	159.2	153.5	*155.5*	149.3	147.9
greedy	154.9	154.2	149.1	149.2	148.4	*151.1*
ss-one-1	-	152.1	141.7	126.2	121.4	117.9
ss-one-m	-	152.4	141.7	132.0	127.8	127.4
ss-one-s	-	149.2	147.7	146.7	142.5	132.8
ss-one-0	-	154.1	148.5	149.2	148.5	*151.1*
ss-two-1	165.9	154.6	143.3	134.2	122.8	122.2
ss-two-m	-	156.1	142.2	139.8	128.7	126.1
ss-two-s	-	157.1	153.5	145.8	141.1	142.0
ss-two-0	-	156.6	154.7	*152.4*	*151.6*	*154.1*
ss-ENS-1	187.2	154.7	142.6	135.0	131.9	122.1
ss-ENS-m	-	154.2	143.0	138.1	135.7	126.6
ss-ENS-s	-	165.5	155.8	147.5	142.8	139.3
ss-ENS-0	-	*169.1*	**165.6**	*155.1*	*152.9*	149.8
batch-ENS	-	**170.0**	*163.1*	**157.0**	**154.2**	**154.5**

is hard to know which hyperparameter is the best. However, it is still far worse than ss-ENS-0 and batch-ENS. (2) As a general trend, the performance decreases as batch size increases. (3) ss-ENS-0 and batch-ENS (both nonmyopic) are either the best or not significantly worse than the best for all batch sizes except 25. (4) Sequential simulation is almost always better with the pessimistic oracle, except with two-step for $b = 5$.

4.5.2 Finding Bulk Metallic Glasses

We conduct the same experiments as for CiteSeer^x data. This dataset is much larger, so we only repeat the experiment 30 times, randomizing the initial seed. We also set $T = 500$. The results are reported in Table 4.3.

Table 4.2: Diversity scores of the chosen batches by all our proposed policies, measured by the average rank of distances from each other in a batch, produced from the results on CiteSeer^x data. Higher value indicates more diversity.

	5	10	15	20	25
greedy	1515	1970	2336	2443	2535
ss-one-1	303	437	599	677	761
ss-one-m	866	810	1000	976	1111
ss-one-s	1221	1134	1458	1374	1692
ss-one-0	1524	1983	2339	2450	2532
ss-two-1	413	507	603	735	772
ss-two-m	927	982	987	1221	1322
ss-two-s	1344	1331	1467	1538	1483
ss-two-0	1768	1933	2221	2540	2576
ss-ens-1	968	1254	1328	1299	1400
ss-ens-m	1323	1458	1751	1683	1887
ss-ens-s	2131	2258	2370	2402	2602
ss-ens-0	1987	2281	2542	2587	2725
batch-ENS	2266	2585	2842	3126	3225

We have the following observations: (1) The uncertain-greedy batch policy is again mostly worse than all our proposed batch active search policies. (2) The performance often decreases as batch size increases. But we see more exceptions than in the results for CiteSeer^x data, probably due to fewer trials. (3) The nonmyopic policies (i.e., ENS based policies) consistently perform best for all batch sizes, and all myopic policies are significantly worse than the best policy. (4) Sequential simulation with the pessimistic oracle are mostly the best for one-step and two-step policies; however, we do not see the same pattern for ENS. At this moment we are not sure whether this is only because of lack of repetition or if this particular dataset has different properties.

Table 4.3: Results for BMG data: average number of targets found by various batch policies: baseline greedy-batch, sequential simulation "ss-P-O" and batch-ENS, with batch sizes 5, 10, 15, 20, 25. The average is taken over 30 experiments. Highlighted are the best in each column and those that are not significantly worse than the best using a one-sided paired t test with significance level $\alpha = 0.05$.

	1	5	10	15	20	25
UGB	-	269.6	265.8	265.9	255.7	247.9
greedy	283.7	278.1	272.5	269.6	264.2	263.6
ss-one-1	-	262.8	244.2	235.4	227.6	224.1
ss-one-m	-	269.0	252.2	236.7	231.3	225.6
ss-one-s	-	283.0	272.7	263.1	255.6	246.1
ss-one-0	-	278.1	272.5	269.8	264.3	263.6
ss-two-1	282.0	270.7	242.7	241.4	231.4	225.6
ss-two-m	-	272.5	250.9	243.6	242.5	226.4
ss-two-s	-	274.4	267.8	262.3	250.9	251.6
ss-two-0	-	277.1	275.8	274.1	264.7	266.1
ss-ENS-1	304.9	*301.7*	**298.4**	290.2	277.5	280.2
ss-ENS-m	-	**304.3**	*294.5*	290.3	284.7	283.1
ss-ENS-s	-	290.3	283.5	288.0	**299.2**	*289.2*
ss-ENS-0	-	*301.4*	281.7	283.8	279.5	275.7
batch-ENS	-	*300.6*	*296.2*	**306.7**	287.6	**294.8**

4.5.3 Drug discovery

We conduct our main investigation on a drug discovery application. Recall in this application, our goal is to find chemical compounds that exhibit binding activity with a target protein. Each target protein defines an active search problem. We only consider the first ten of the 120 datasets and only the ECFP4 fingerprint, which showed the best performance in the sequential setting (see Chapter 3). These datasets share a pool of 100 000 negative compounds randomly selected from the ZINC database [89]. The number of positives of the ten datasets varies from 221 to 1024, with mean 553.

Table 4.4: Results for drug discovery data: Average number of positive compounds found by the baseline *uncertain-greedy* batch, greedy-batch, sequential simulation and batch-ENS policies. Each column corresponds to a batch size, and each row a policy. Each entry is an average over 200 experiments (10 datasets by 20 experiments). The budget T is 500. Highlighted are the best (bold) for each batch size and those that are not significantly worse (blue italic) than the best under one-sided paired t-tests with significance level $\alpha = 0.05$.

	1	5	10	15	20	25	50	75	100
UGB	-	257.6	257.9	258.3	250.1	246.0	218.8	206.2	172.1
greedy	269.8	268.1	264.1	261.6	258.2	257.0	240.1	227.2	208.2
ss-one-1	269.8	260.7	254.6	245.2	233.6	223.4	200.8	182.9	178.9
ss-one-m	269.8	264.5	257.7	250.0	244.4	236.5	211.7	195.4	179.4
ss-one-s	269.8	266.8	261.3	256.7	248.7	244.1	214.9	202.4	181.3
ss-one-0	269.8	268.1	264.1	261.6	258.2	257.0	240.1	227.2	208.2
ss-two-1	281.1	237.1	219.8	210.8	212.1	196.2	172.1	158.8	152.9
ss-two-m	281.1	252.6	246.4	237.2	232.9	225.1	200.2	181.6	167.2
ss-two-s	281.1	248.9	242.5	235.3	226.6	219.2	196.7	175.3	158.3
ss-two-0	281.1	252.5	247.6	247.9	244.4	240.4	225.6	213.8	199.1
ss-ENS-1	**295.1**	269.4	247.9	227.2	223.1	210.3	185.3	152.6	148.7
ss-ENS-m	*295.1*	293.8	290.2	285.3	281.6	274.4	249.4	217.2	203.1
ss-ENS-s	*295.1*	289.9	278.3	269.8	262.6	255.0	220.8	185.5	161.2
ss-ENS-0	*295.1*	293.6	289.1	288.1	*287.5*	280.7	269.2	257.2	241.0
batch-ENS-16	*295.1*	**300.8**	**296.2**	293.9	**292.1**	*288.0*	275.8	*272.3*	252.9
batch-ENS-32	*295.1*	*300.8*	*295.5*	**297.9**	*290.6*	**288.8**	**281.4**	**275.5**	**263.5**

For each dataset, we start with one random initial positive seed observation and repeat the experiment 20 times. We test for batch sizes $b \in \{5, 10, 15, 20, 25, 50, 75, 100\}$, we also show the results for sequential search ($b = 1$) as a reference. The budget is set as $T = 500$. We test batch-ENS with 16 and 32 samples, coded as batch-ENS-16 and batch-ENS-32. We show the number of positive compounds found in Table 4.4, averaged over the 10 datasets and 20 experiments each, so each entry in the table is an average over 200 experiments. We highlight the best result for each batch size in boldface.

We highlight the following observations, similar to that of previous datasets. (1) The performance decreases as the batch size increases. (2) Nonmyopic policies are consistently better than myopics ones; in particular, batch-ENS is a clear winner. (3) For sequential simulation policies, the pessimistic oracle is almost always the best.

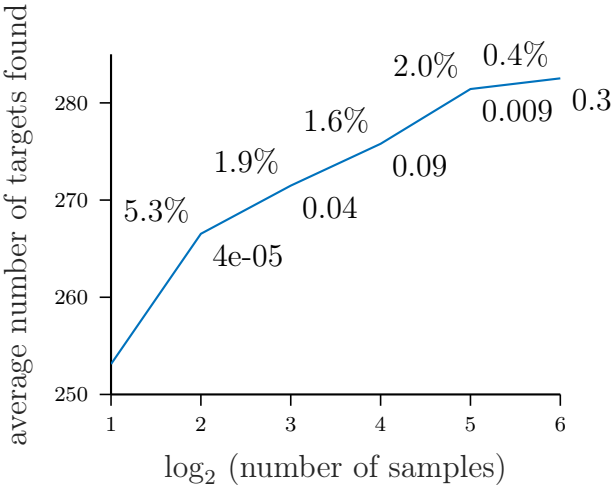


Figure 4.2: Number of targets found versus number of samples used for batch-ENS. This is averaged over the results for batch size 50 on 10 drug discovery datasets and 20 experiments each. The text labels show the percentage of improvement and p-value of one-sided t tests comparing against previous numbers, e.g., 8 samples improves over 4 samples by 1.9%, and the p-value is 0.04.

For batch-ENS, we find batch-ENS with 32 samples often performs better than with 16, especially for larger batch sizes. We have run batch-ENS for $b = 50$ with $N \in \{2, 4, 8, 16, 32, 64\}$, and find that the performance improves considerably as the number of samples increases, but the magnitude of this improvement tends to decrease with larger numbers, as shown in Figure 4.2. We believe 32 label samples offers a good tradeoff between efficiency and accuracy for $b = 50$. An interesting future work is theoretical analysis providing guidance for choosing N.

4.5.4 Discussion

We now discuss our observations in more detail. First we see all our proposed policies perform better than the heuristic uncertain-greedy batch, even if we optimistically assume the best hyperparameter of this policy (not to mention we hardly know what the best hyperparameter

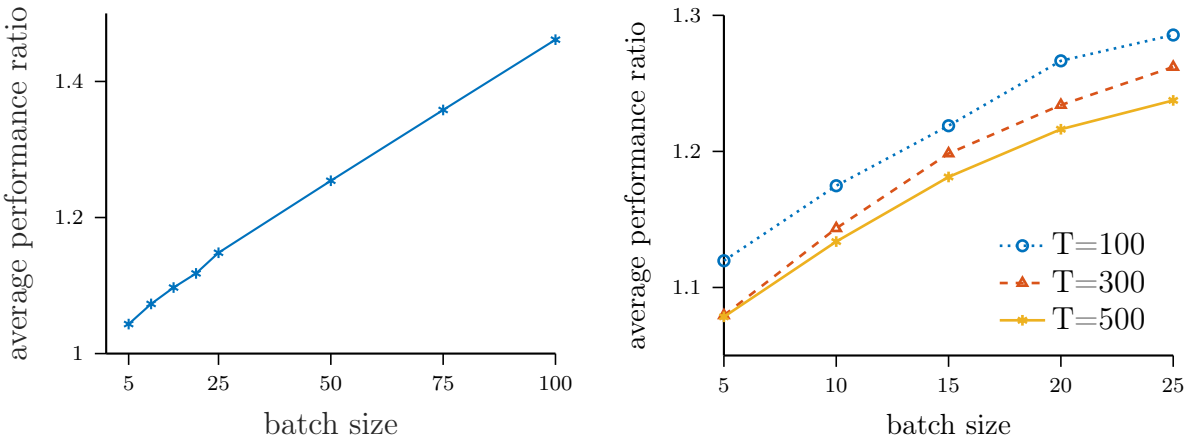


Figure 4.3: (a) Average performance ratio between sequential policies and batch policies, as a function of batch size, produced using averaged results in Table 4.4. (b) Same plots produced using averaged results (excluding uncertain-greedy) for the CiteSeer dataset (Table 4.1).

should be in practice). Our framework based on Bayesian decision theory offers a more principled approach to batch active search (especially batch-ENS); and our methods are effectively hyperparameter-free (except the number of samples used in batch-ENS). In the following, we elaborate on the three observations.

Empirical adaptivity gap. Regardless of what policy is used, the performance in general degrades as the batch size increases. But how fast? We average the results in Table 4.4 over all policies for each batch size b as an empirical surrogate for OPT_b in Theorem 2, and plot the resulting surrogate value of $\frac{\text{OPT}_1}{\text{OPT}_b}$ as a function of b in Figure 4.3a. Although these policies are not optimal, the empirical performance gap matches our theoretical linear bound surprisingly well. To verify the empirical linear trend also holds for different budgets and on other datasets, we plot the same curves for the CiteSeer^x dataset for budget $T = 100, 300, 500$ in Figure 4.3b. We can see these lines all appear to be linear; more interestingly, given the same batch size, the gap is smaller for larger T, which is also indicated by Theorem 2. As

future work, we will further investiage whether this gap will decrease as $1/\log T$. These results could provide valuable guidance on choosing batch sizes.

Despite the overall trends in our results, we see some interesting exceptions. That is, batch-ENS with batch size 5 is significantly better than that with batch size 1, with a p-value of 0.02 under a one-sided paired t-test. This is counterintuitive based on our analysis regarding the adaptivity gap. We conjecture that batch-ENS with larger batch sizes forces more (but not too much) exploration, potentially improving somewhat on sequential ENS in practice.

Why is the pessimistic oracle better? Among the four fictional oracles, the pessimistic one usually performs the best for sequential simulation. When combined with a greedy policy, we have provided some mathematical rationale in Proposition 1: sequential simulation then near-optimally maximizes the probability of unit improvement, which is a reasonable criterion. Intuitively, by always assuming the previously added points to be negative, the probabilities of nearby points are lowered, offering a repulsive force compelling later points to be located elsewhere, leading to a more diverse batch. This mechanism could help better explore the search space.

To verify our hypothesis that the pessimistic oracle performances better is due to encouraging diversity, we show in Table 4.2 the diversity scores of the chosen batches using the results on CiteSeer^x data. The diversity is measured as follows: for each batch $\mathcal{B}$, for any $x_i, x_j \in \mathcal{B}$, we compute the rank of x_j according to increasing distance to x_i, and average the ranks for all pairs as the diversity score of this batch. We use rank instead of distance for invariance to scale.

Table 4.5: Diversity scores of the chosen batches by all sequential simulation policies, measured by the average rank of distances from each other in a batch. The results are for $b = 20$ on CiteSeer^x data. Higher values indicate more diversity. For reference, the score for greedy and batch-ENS are 2443 and 3126, respectively.

	optimisitic	most likely	sample	pessimistic
one-step	677	976	1374	2450
two-step	735	1221	1538	2540
ENS	1299	1683	2402	2587

To better contrast, we extract the diversity scores for batch size 20, and present in Table 4.5. There is an increasing trend in each row, with "pessmistic" (always negative) the highest; also in each column, with ENS the highest.

If comparing Table 4.2 and Table 4.1 closely, one could find that the diversity scores and active search performances align remarkably well. This indicates diversity could be an important consideration for batch policy design.

Note this coincides with the idea of explicitly using repulsion to create a diverse batch, which has been adopted in similar settings such as Bayesian optimization [33].

Myopic vs. nonmyopic behavior. Nonmyopic policies (ENS-based) almost always perform better than myopic policies. This certainly matches our expectation as nonmyopic policies are always cognizant of the budget and hence can better trade off exploration and exploitation. To gain some insight into the nature of this myopic/nonmyopic behavior, in Figure 4.4 we plot the probabilities of the points chosen (at the iteration of being chosen) by the greedy and batch-ENS-32 policies for batch size 50 across the drug discovery datasets. The behavior of other myopic policies such as sequential simulation of one- or two-step) is similar to greedy-batch, whereas that of the other ENS based policies is similar to batch-ENS-32.

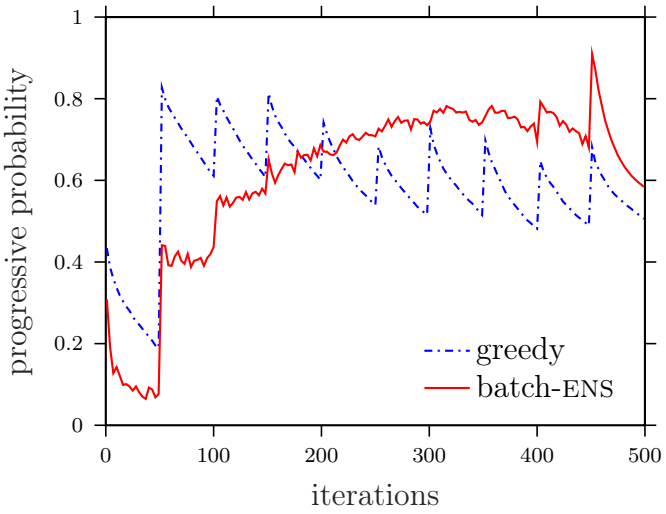


Figure 4.4: Progressive probabilities of the chosen points of greedy and batch-ENS-32, averaged over results for batch size 50 on all 10 drug discovery datasets and 20 experiments each.

First, in each batch, the trend for greedy is not surprising, since every batch represents the top-50 points ordered by probabilities. For batch-ENS, there is no such trend except in the last batch, where batch-ENS naturally degenerates to greedy behavior. Second, along the whole search process, greedy has a decreasing trend, likely due to over-exploitation in early stages. On the other hand, batch-ENS has an increasing trend. This could be partly due to more and more positives being found. More importantly, we believe this trend is in part a reflection of the nonmyopia of batch-ENS: in early stages, it tends to explore the search space, so low probability points might be chosen. As the remaining budget diminishes, it becomes more exploitive; in particular, the last batch is purely exploitive.

4.5.5 Pruning Effectiveness

We present the results of pruning for all three types of datasets in Table 4.6. On all three types of datasets, the majority of points in each iteration are pruned. Especially on drug

discovery datasets, on average over 98% of the points are pruned; that is over 50 times speed-up.

Table 4.6: Results for pruning effectiveness. The numbers are averaged over all iterations of batch-ENS for all batch sizes tested. For drug discovery data, the result is averaged over batch-ENS-16 and batch-ENS-32.

datasets	#total	#pruned	pruning rate
CiteSeer^x	39546	27422.1	68.37%
BMG	111360	82343.6	73.94%
Drug discovery	100316	98612.6	98.30%

4.6 Conclusion

We have completed the first study on batch active search, where the goal is to find as many positives as possible in a given labeling budget. We derived the Bayesian optimal policy for batch active search, and proved a lower bound, linear in batch size, on the performance gap between optimal sequential and batch policies. This was shown to match empirical results.

We then generalized a recently proposed efficient nonmyopic search (ENS) policy to the batch setting and proposed two approaches to approximately solving the batch version of ENS: sequential simulation with fictional labeling oracles and greedy set function maximization. We conducted comprehensive experments on data from three application domains evaluating all fourteen proposed policies. Results show that nonmyopic policies perform significantly better than myopic ones. By analyzing the results, we gained a deeper understanding of the nonmyopic behavior and find diversity to be an importantant consideration for batch policy design.

Chapter 5

Cost Effective Active Search

In Chapter 3 and 4, we focused on the budgeted active search, where the goal is to find as many positive points as possible in a given budget B, the total number of points one can query, known *a priori*. In this paper, we consider the "dual" problem: *how to find a given number of positives with minimum cost*. Formally, given a large pool of n points $\mathcal{X} = \{x_i\}_{i=1}^n$ with corresponding unknown labels $y_i \in \{0, 1\}$ indicating whether x_i is positive or not, we want to sequentially choose a set $\mathcal{D} \subset \mathcal{X}$ of points to evaluate to identify a given target number T of positives in as few iterations as possible[8]. In this chapter we use T to denote the "target" number of positives. We call this problem *cost effective active search* (CEAS). To contrast with the budgeted setting, we present both formulations:

Budgeted: $\arg\max_{\mathcal{D}\subset\mathcal{X}}, \sum_{x_i\in\mathcal{D}} y_i,$ s.t. $|\mathcal{D}| = B$; **CEAS**: $\arg\min_{\mathcal{D}\subseteq\mathcal{X}}|\mathcal{D}|,$ s.t. $\sum_{x_i\in\mathcal{D}} y_i \geq T$.

One might wonder if we could reduce CEAS to the budgeted case, for which many effective policies are readily available. For example, given a CEAS problem with target T, we could

[8]Here we abuse the notation $\mathcal{D}$ to actually mean the input locations in $\mathcal{D}$ when we write $\mathcal{D} \subseteq \mathcal{X}$

instead solve a budgeted problem with an estimated budget B. However, this "dual" transformation is not one-to-one: given a budget B, the expected utility is no longer T. In fact, both Pr(utility | B) and Pr(cost | T) are highly complicated distributions and estimating their expectations is extremely intractable, as we will show later. Therefore it is necessary to develop specific policies for CEAS.

CEAS is a direct model of practical cases where we must achieve a certain target with minimum cost; for example, during initial screening for drug discovery, we may seek a certain number of active compounds to serve as lead compounds to be refined later in the discovery process. There have been several previous investigations into the the CEAS setting (e.g., [98, 99]), but the proposed policies are often based on heuristics, and to the best of our knowledge, few theoretical results have been established regarding the hardness of this problem.

In this Chapter, we study the CEAS problem under the Bayesian decision-theoretic framework. We first derive the Bayesian optimal policy and establish a strong hardness result showing that the optimal policy is extremely inapproximable. In particular, we show any efficient algorithm is at least $\Omega(n^{0.16})$ times worse than the optimal policy in terms of the expected cost needed.

We then propose nonmyopic approximations to the Bayesian optimal policy and develop efficient implementation and pruning techniques that make nonmyopic search in large spaces possible. To that end, we discuss an understudied distribution called the "negative Poisson binomial" and propose a simple and fast approximation to its expectation, an essential component of our efficient nonmyopic policy. We conduct comprehensive experiments on benchmark datasets in various domains such as drug and materials discovery and demonstrate that our policy is superior to widely used baselines.

5.1 Related Work

[14] studied minimum-cost active learning under the version space reduction utility and proved that a greedy algorithm is near-optimal due to adaptive submodularity. [99] studied "active learning in the drug discovery process." They used perceptron and SVM as predictive models and tried several variants of uncertainty sampling (sampling points closest to the separating hyperplane) and greedy sampling (choosing points furthest from the hyperplane). The authors found that greedy sampling performed better, but offered no theoretical justification, in contrast to our decision theoretic approach.

CEAS is a special case of the *adaptive stochastic minimum cost cover* problem [32], for which an inapproximability bound was proved using a spiritually similar instance construction to our own, including the idea of "treasure hunting" and using XOR for encoding. However, their worst case construction is much more complicated and does not directly correspond to active search. Furthermore, their bound only holds conditionally (PH $\neq \Sigma_2^P$). Our proof is much simpler and does not rely on any complexity theoretic hypothesis.

CEAS is also closely related to the *total recall* problem in information retrieval [37, 75, 104] or the so-called technology-assisted review progress for evidence discovery in legal settings [36], where the goal is to retrieve (nearly) *all* relevant documents at a reasonable labeling cost. Despite the similarity to our setting, retrieving *all* positives could be drastically different (and arguably much harder). One immediate difference is that we often do not know the total number of positives in a dataset *a priori,* thus a considerable amount of research is devoted to deciding when to stop in the total recall procedure [37]. Our definition of CEAS avoids this complication.

5.2 Bayesian Optimal Policy

We again assume there is a model that provides the posterior probability of a point x being positive, conditioned on previously observed data $\mathcal{D}$, i.e., $\Pr(y = 1 \mid x, \mathcal{D})$. The Bayesian optimal policy then chooses the point that minimizes the expected number of further iterations required. Formally, in iteration $(i + 1)$, where we have observed $\mathcal{D}_i$, then

$$x^* = \arg\min_x \mathbb{E}[c^r \mid x, \mathcal{D}_i], \tag{5.1}$$

where c^r denotes the further cost incurred (i.e., the size of the chosen set) when r more positives are identified after $\mathcal{D}_i$, and $r = T - \sum_{(x,y)\in\mathcal{D}_i} y$ is the remaining target yet to be achieved.

The expected cost can be written as a Bellman equation:

$$\mathbb{E}[c^r \mid x, \mathcal{D}_i] = 1 + \Pr(y = 1 \mid x, \mathcal{D}_i) \cdot \min_{x'} \mathbb{E}[c^{r-1} \mid x', x, y = 1, \mathcal{D}_i] + \Pr(y = 0 \mid x, \mathcal{D}_i) \cdot \min_{x'} \mathbb{E}[c^r \mid x', x, y = 0, \mathcal{D}_i]. \tag{5.2}$$

However, this recursion is not mathematically well defined since there is no exit. To see this, consider the base case where $r = 1$, i.e., only one more positive left to find. With probability $\Pr(y = 1 \mid x, \mathcal{D}_i)$, x is positive, and we finish with cost 1; with probability $1-\Pr(y = 1 \mid x, \mathcal{D}_i)$, x is negative, and then we need to update the probability conditioned on $y = 0$ and repeat this process. That is,

$$\mathbb{E}[c^1 \mid x, \mathcal{D}_i] = \Pr(y = 1 \mid x, \mathcal{D}_i) \cdot 1 + (1 - \Pr(y = 1 \mid x, \mathcal{D}_i)) \cdot \min \mathbb{E}[c^1 \mid x, \mathcal{D}_i]. \tag{5.3}$$

So even for $r = 1$, the recursion never exits. Note this is very different from the budgeted setting [27], where the base case with budget 1 is well defined and trivial to compute.

In practice, the recursion must stop after exhausting the whole pool. We assume the search stops after at most t steps; we stress that this is only for derivation purposes, and should not be confused with the budgeted setting with budget t. If we set $t = |\mathcal{X}|$, then that means we keep searching until the target is achieved or there are no more points left to evaluate. Let c_t^r be the cost incurred after r positives are found or t evaluations are completed. Now we can derive the base case of the recursion. To simplify notations, we will omit the conditioning on $\mathcal{D}_i$ in the following and assume all probabilities are implicitly conditioned on current observations. Let $p(x) = \Pr(y = 1 \mid x, \mathcal{D}_i)$. We start with the simplest case $r = 1$:

When $t = 1$, the expected cost would be 1 no matter what.

When $t = 2$, $\mathbb{E}[c_2^1 \mid x] = p(x) \cdot 1 + (1 - p(x)) \cdot 2 = 2 - p(x)$. In this case, the greedy policy choosing the point with highest probability is optimal.

When $t = 3$,

$$\begin{aligned}\mathbb{E}[c_3^1 \mid x] &= p(x) \cdot 1 + (1 - p(x)) \cdot \min_{x'} \mathbb{E}[c_2^1 \mid x', x, y = 0] \\ &= 2 - p(x) - (1 - p(x)) \max_{x'} p'(x'). \end{aligned} \tag{5.4}$$

where $p'(x') = \Pr(y' = 1 \mid x', x, y = 0)$. We can see

$$\arg\min_x \mathbb{E}[c_3^1 \mid x] \Leftrightarrow \arg\max_x p(x) + (1 - p(x)) \max_{x'} p'(x'). \tag{5.5}$$

Note the form on the right hand side of (5.5) has a clear exploration vs. exploitation explanation. It balances between choosing a point with high probability as governed by

$p(x)$ (exploit) and a point that has low probability but would lead to a high probability point if it turns out to be negative as governed by $(1 - p(x)) \max_{x'} p'(x')$ (explore). This is counter-intuitive since one would imagine with only one positive left to find, the best thing to do should be greedy, but we have shown if we are granted three or more iterations, greedy might not be optimal in terms of minimizing expected cost. Therefore, we argue that nonmyopic planing is crucial for CEAS.

We draw a connection between (5.5) and the two-step score in budgeted setting:

$$\arg\max_x p(x) + p(x) \max_{x'} p(y' = 1 \mid x', y = 1) + (1 - p(x)) \max_{x'} p(y' = 1 \mid x', y = 0); \quad (5.6)$$

we can see in (5.6), the future utility is averaged over the positive and negative case, whereas in (5.5) only the negative case is considered.

The recursion is now well-defined for $r = 1$:

$$\mathbb{E}[c_t^1 \mid x] = p(x) \cdot 1 + (1 - p(x)) \cdot \min_{x'} \mathbb{E}[c_{t-1}^1 \mid x', x, y = 0]. \quad (5.7)$$

Note it takes $\mathcal{O}(n^t)$ time to compute even for the simplest case $r = 1$. In general ($r \geq 1$), we have

$$\begin{aligned} \mathbb{E}[c_t^r \mid x] = 1 + p(x) \cdot \min_{x'} \mathbb{E}[c_{t-1}^{r-1} \mid x', x, y = 1] + \\ (1 - p(x)) \cdot \min_{x'} \mathbb{E}[c_{t-1}^r \mid x', x, y = 0]. \end{aligned} \quad (5.8)$$

5.3 Hardness of Cost Effective Active Search

The above derivation indicates that we can not efficiently compute the expected cost exactly. In fact, the optimal policy is not only hard to compute, it is even hard to approximate. As we show in the following theorem, any efficient algorithm is at least $\Omega(n^{0.16})$ times worse than the optimal policy in terms of average cost:

Theorem 3. *Any algorithm $\mathcal{A}$ with computational complexity $o\left(n^{n^\epsilon}\right)$ has an approximation ratio $\Omega(n^\epsilon)$, for $\epsilon = 0.16$; that is,*

$$\frac{\mathbb{E}[\text{cost}_{\mathcal{A}}]}{\text{OPT}} = \Omega\left(n^\epsilon\right), \tag{5.9}$$

where $\mathbb{E}[\text{cost}_{\mathcal{A}}]$ is the expected cost of $\mathcal{A}$, and OPT *is that of the optimal policy.*

Note that this lower bound is very tight since $\mathcal{O}(n)$ is a trivial upper bound. We prove the theorem by constructing a class of active search instances similar to that of the hardness proof for the budgeted setting. Specifically, there is a small secret set of points, the labels of which encode the location of a clump of positive points. The optimal policy (with unlimited computational power) could easily identify this secret set by enumerating all possible subsets of the same size and feeding them into the inference model, thereby revealing the positive clump and completing the search quickly. However, for an algorithm with limited computational power, the probability of revealing this secret set is extremely low, and hence it can not do any better than randomly guessing, which results in a much higher expected cost.

This proof has two key differences compared to that of the budgeted setting. First, it results in an *exponentially* stronger bound ($n^{0.16}$ vs $\sqrt{\log n}$). Second, the improved bound requires a different methodology on how to hide the set of profitable points from the algorithm. In

particular, a new counting technique is used for bounding the probability of identifying the "secret set". In the budgeted setting, one key argument considered "If an efficient policy selects a subset of B points, how likely is it to identify the secret set?", which was straightforward to compute since the chosen budget B defined the cardinality of the selected subset. Such reasoning does not apply to CEAS setting. In fact, we leverage the underlying algorithmic challenge of optimizing the newly considered objective where the algorithm has to continue searching until it reaches the target. A formal proof is given in the Appendix C.

5.4 Efficient Nonmyopic Approximation

The fundamental difficulty in computing the expected cost is that we need to recursively update the probabilities of the remaining points conditioned on possible outcomes of the previous points. A natural way to relieve this burden is to assume *conditional independence* (CI) at some point in the recursion, e.g., after observing the label of the point under consideration. This idea has been applied successfully in the budgeted setting, where it showed excellent empirical performance in practice. We propose to adapt this idea to CEAS. However, compared to the budgeted setting, it is less straightforward to approximate the expected remaining cost for CEAS – even assuming conditional independence – as the stopping time is a random variable.

We model the remaining cost as a *negative Poisson binomial* (NPB) distribution, i.e., the number of coins (with nonuniform biases) that need to be flipped (independently) to get a given number of HEADs. This is in contrast to the utility being modeled as a *Poisson binomial* (PB) distribution in the budgeted setting. The NPB distribution is a natural generalization of the *negative binomial* (NB) distribution where all coins have the same biases. While both NB

and PB distributions are well studied, few references can be found for NPB. Some informal discussions about NPB can be found in the Physics Forum.[9] [11] defines a distribution they also call NPB, but it is actually a sum of geometric variables. [59] defines NPB as the number of "failures given successes" for predicting the outcome of darts tournaments. In the following, we formally define the NPB distribution and propose novel and fast approximations to its expectation, based on which we derive our efficient nonmyopic policy for CEAS.

5.4.1 Negative Poisson Binomial Distribution

We define the NPB distribution in an intuitive manner as follows:

Definition 1 (Negative Poisson binomial distribution)**.** *Let there be an infinite number of ordered coins with* HEADs *probabilities* $p_1 \geq p_2 \geq p_3 \geq \cdots$*; given a number* r *of* HEADs *required, we toss the coins one by one in this order until* r HEADs *occur. The number of coins tossed* m *follows a negative Poisson binomial distribution:* $m \backsim \text{NPB}(r, [p_1, p_2, p_3, \cdots])$.

Note that this distribution has support for any integer $m \geq r$ assuming the probabilities are nonzero. Here we study a truncated version where we have a finite number of coins, n, and we assume n is large enough so that $\Pr(m \geq n)$ is negligible. This is typically the case in active search since $r \ll n$, where n the size of unlabeled pool and r is the target to achieve.

The PMF of this distribution can be derived using the PMF of a Poisson binomial (PB) distribution $r \backsim \text{PB}([p_1, p_2, \cdots, p_n])$, which is the number of HEADs observed if we independently toss n coins with HEADs probabilities $p_1, p_2, \ldots, p_n$. Denote $\Pr_{\text{PB}}(i, j)$ as the probability of j HEADs when there are i coins with probabilities $p_1, p_2, \ldots, p_i$. Then $\Pr_{\text{PB}}(n, r)$ can be

[9] `https://www.physicsforums.com/threads/negative-poisson-binomial-distribution.759630/`

computed via dynamic programming (DP):

$$\Pr_{\text{PB}}(n, r) = \begin{cases} \prod_{i=1}^{n}(1 - p_i), & \text{if } r = 0; \\ p_n \Pr_{\text{PB}}(n-1, r-1) + (1 - p_n)\Pr_{\text{PB}}(n-1, r), & \text{if } 0 < r <= n; \end{cases} \tag{5.10}$$

The DP table of $\Pr_{\text{PB}}(n, r)$ is of size $\mathcal{O}(nr)$. [12] also derived other formulas for computing the PMF of the PB distribution.

Given the PMF of the PB distribution, the PMF of $m \backsim \text{NPB}(r, [p_1, \cdots, p_n])$ is, $\forall\ m \geq r$:

$$\Pr_{\text{NPB}}(m, r) = p_m \Pr_{\text{PB}}(m-1, r-1), \tag{5.11}$$

and the expectation is

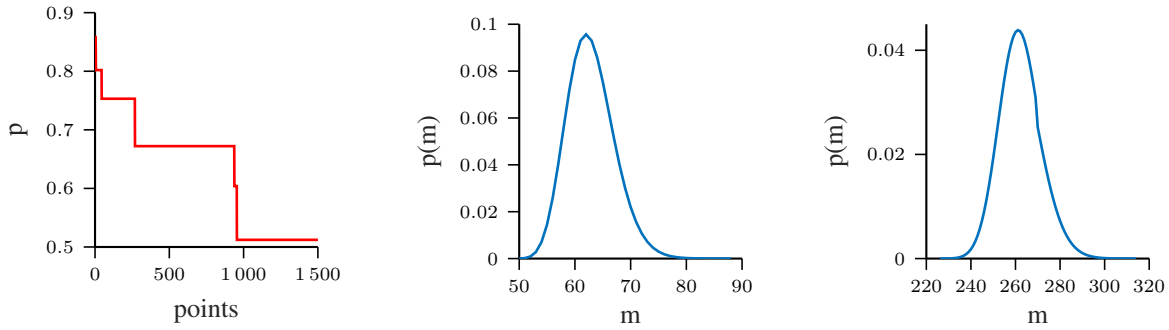
$$\mathbb{E}[m] = \textstyle\sum_{i=r}^{n} \Pr_{\text{NPB}}(i, r) \cdot i. \tag{5.12}$$

Note in computing $\Pr_{\text{NPB}}(n, r)$, all $\Pr_{\text{NPB}}(m, r)$ for $m < n$ are also computed. So the complexity for computing the expectation is also $\mathcal{O}(nr)$.

Figure 5.1 shows some examples of negative Poisson binomial distributions using the posterior marginal probabilities shown in 5.1a. We sort the probabilities in decreasing order since this is obviously the order that leads to minimum $\mathbb{E}[m]$, which is what we care in CEAS.

5.4.2 Approximation of the NPB expectation

The complexity $\mathcal{O}(nr)$ for computing the expectation is prohibitively high. To reduce its complexity, we observe in practice that $\Pr_{\text{NPB}}(m, r)$ will be very close to zero for $m \gg r$. So



(a) (b) (c)

Figure 5.1: Illustration of a probability vector $[p_1, p_2, \ldots, p_n]$ and the corresponding probability mass functions of NPB distribution for different r. Left: the top 1500 posterior marginal probabilities after conditioning on 100 positive and 100 negative points (randomly selected); probabilities are computed using a k-nn model (with $k = 50$) on the CiteSeer^x dataset; middle: $r = 50$; right: $r = 200$.

we can stop at $\bar{m}$ when $\sum_{i=r}^{\bar{m}} \Pr_{\text{NPB}}(i, r) \geq 1 - \varepsilon$ for, e.g., $\varepsilon = 10^{-6}$. Hence, an almost exact solution can be computed in $\mathcal{O}(\bar{m}r)$. We call this approach ε-DP.

The complexity of this approximation $\mathcal{O}(\bar{m}r)$ for computing the expectation in (5.12) may still be high since we might need to compute this expectation for every candidate point in each iteration. We propose another cheap but accurate approximate method with only $\mathcal{O}(\mathbb{E}[m])$ complexity. The idea is simple: coin toss i contributes p_i HEADs in expectation, so we accumulate these contributions until r HEADs occur. That is,

$$\mathbb{E}[m] \approx \arg\min_k \sum_{i=1}^{k} p_i \geq r. \tag{5.13}$$

We call this approximation ACCU (short for "accumulate"). Note that ACCU always returns an integer, while the true expectation might not be integral. To fix this, we subtract a correction term. Let $\hat{m} = \arg\min_k \sum_{i=1}^{k} p_i \geq r$. We check what portion of $p_{\hat{m}}$ was needed for the sum

to be exactly r, and remove the extra portion:

$$\mathbb{E}[m] \approx \hat{m} - \frac{\left(\sum_{i=1}^{\hat{m}} p_i\right) - r}{p_{\hat{m}}}. \tag{5.14}$$

We call this approximation ACCU'. In the special case of the negative binomial distribution (i.e., $p = p_1 = p_2 = \ldots = p_n$), ACCU' recovers the true expectation r/p.

One might be tempted to approximate the expectation by a natural generalization of the expectation of a negative binomial distribution $r/p = 1/p + 1/p + \cdots + 1/p$; that is, $\mathbb{E}[m] \approx 1/p_1 + 1/p_2 + \cdots + 1/p_r$. We call this RECIP. Note that this approximation is also exact when $p_1 = p_2 = \cdots = p_n = p$. However, we will see that this approximation can be very poor.

We run simulations to demonstrate the fidelity of these approximations. We parameterize the NPB distribution using the posterior marginal probabilities $[p_1, \ldots, p_n]$ computed in a typical active search iteration using a k-nn model (see Figure 5.1a). We plot the approximation errors against the EXACT expectation computed via (5.12) for $r = 1, \ldots, 500$, shown in Figure 5.2.

We can see ε-DP has basically zero error everywhere. ACCU' also has zero error almost everywhere except two locations. As expected, ACCU constantly overestimates the exact value by a small fraction, whereas RECIP considerably underestimates the true value especially for large r. The root mean square error (RMSE) and total time cost for computing the 500 expectations are shown in Table 5.1.

After a closer look at the locations where the errors are higher (e.g., around $r = 220$, for which $\mathbb{E}[m] \approx 300$), we find that such locations exactly correspond to r values such that the

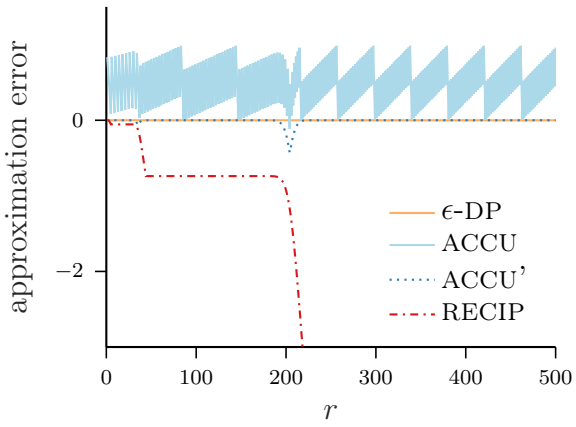


Figure 5.2: Approximation errors of various approximation methods for computing the expectation of NPB distribution. y-axis is approximate $\mathbb{E}[m]$ minus exact $\mathbb{E}[m]$. The error for RECIP drops below -10 after about $r = 250$, hence not shown to zoom in the interesting part of the figure.

Table 5.1: Time cost and quality of various approximations of the expectation of NPB distribution.

	EXACT	ϵ-DP	ACCU'	ACCU	RECIP
RMSE	—	0.0004	0.0438	0.5723	7.8276
time(s)	97.4841	0.1851	0.0029	0.0029	0.0002

probability around $\mathbb{E}[m]$ drops abruptly (refer to Figure 5.1a). This makes perfect sense since ACCU' and RECIP are not aware of such changes after $\mathbb{E}[m]$. RECIP suffers from this problem more severely since it only looks at r probability values but ACCU' looks at $\mathbb{E}[m]$ values. Overall, we can see ACCU' provides a reasonable time–quality tradeoff, and we will use this approximation for our policy. It is an interesting question whether we can derive error bounds for ACCU'.

5.4.3 Approximation of Expected Cost

We can approximate the expected cost (2.13) as follows:

$$\mathbb{E}[c^r \mid x, \mathcal{D}_i] \approx 1 + \Pr(y = 1 \mid x, \mathcal{D}_i)\mathbb{E}[m^+] + \Pr(y = 0 \mid x, \mathcal{D}_i)\mathbb{E}[m^-] \equiv f(x), \tag{5.15}$$

where $\mathbb{E}[m^+] \approx \mathbb{E}[c^{r-1} \mid x, y = 1, \mathcal{D}_i]$, $\mathbb{E}[m^-] \approx \mathbb{E}[c^r \mid x, y = 0, \mathcal{D}_i]$, and $\Pr(c^{r-1} \mid x, y = 1, \mathcal{D}_i)$ and $\Pr(c^r \mid x, y = 1, \mathcal{D}_i)$ are assumed to be NPB distributions defined by posterior marginal probabilities (in descending order) after the respective conditioning . We use ACCU' to compute the approximate expectations. In every iteration, we choose x minimizing $f(x)$. We will call this policy *efficient nonmyopic cost effective search,* or ENCES.

We further propose to adapt our policy by treating the remaining target r as a tuning parameter. In particular, we argue that it is better to set this parameter smaller than the actual remaining target. The rationale is three-fold: (1) the approximation based on the NPB distribution typically overestimates the actual expected cost, since we are pretending the probabilities will not change after the current iteration, whereas actually the top probabilities defining the NPB distribution will increase considerably as we discover more positives. So the actual expected cost should be much smaller. (2) even if the CI assumption is correct, planning too far ahead might hurt if the model is not accurate enough; after all, "all models are wrong." (3) setting r smaller makes the bounds on the expectation much tighter, hence pruning is much more effective (see 5.4.4 about pruning).

We consider two schemes: setting r to a constant or proportional to the remaining target. For example, ENCES-10 means we always set $r = 10$ if the remaining target is greater than 10, otherwise we use the actual remaining target; and ENCES-0.2 means we set r to be 20% of the actual remaining target.

5.4.4 Lazy Evaluation for Pruning

To further improve the computational efficiency, we develop similar pruning techniques as for policies in the budgeted case. Assume we have upper bounds $\hat{p}$ of the probabilities after one additional positive observation: that is $\forall x \in \mathcal{X} \setminus \mathcal{D}_i$,

$$\hat{p}(x) \geq \Pr(y \mid x, \mathcal{D}_i, x', y' = 1), \forall x' \in \mathcal{X}. \tag{5.16}$$

We can compute a lower bound of $\mathbb{E}[m^+]$ using $\hat{p}$. If we also assume observing a negative does not increase the probability of any other point (the k-nn model satisfies this condition), then a natural probability upper bound after observing a negative point is simply the current probability

$$\Pr(y \mid x, \mathcal{D}_i) \geq \Pr(y \mid x, \mathcal{D}_i, x', y' = 0), \tag{5.17}$$

and we can compute a lower bound of $\mathbb{E}[m^-]$ using $\Pr(y \mid x, \mathcal{D}_i)$. Combining both lower bounds we get a lower bound of $f(x), \forall x \in \mathcal{X} \setminus \mathcal{D}_i$. It is easy to see the bound is tighter with smaller r. This bound can be used to prune points in a similar fashion as in [45].

To find the point $x^* = \arg\min_x f(x)$, we evaluate the candidate points in increasing order of the lower bound, and maintain the minimum of currently evaluated points as an upper bound of $\min f(x)$; we can stop whenever the lower bound becomes greater than the upper bound. We will show that often only a very small percentage (e.g. 1%) of the candidate points need to be evaluated in each iteration.

5.5 Experiments

We conduct experiments to compare our proposed policy against three baselines:

GREEDY: always chooses a point with the highest probability. It is an equivalent of choosing the point furthest from the separating hyperplane in the case of an SVM model [99]. It has been shown that this baseline is hard to beat in the total recall setting [37].

TWO-STEP: the budgeted two step lookahead policy as defined in (5.6). Note we use (5.6) instead of (5.5) since (5.5) looks ahead by considering the maximum probability a point would lead to if it is negative, so it does not make much sense to use this policy with k-nn, which only models positive correlations.

ENS: the efficient nonmyopic search policy recently developed in Chapter 3 for the budgeted setting, This policy was shown to perform remarkably well in various domains due to its adaptation to the remaining budget. However, to apply it in the CEAS setting, we have to make a simple modification since these is no concept of budget. We also experiment with two schemes of setting b: constant or proportional to the remaining target. We test ENS-10, 30, 50, 70 and ENS-0.1, 0.3, 0.5, 0.7 and compare our method against the best one. Such b's underestimate the true budget, but the same rationale given in Sec. 5.4.3 applies.

We run our policies with $r = 10, 20, 30, 40, 50$ and $0.1, 0.2, 0.3, 0.4, 0.5$ and also report the results of the best one (on average) in each scheme. Full results can be found in Appendix D.1. We report results on drug and materials discovery data. In all our experiments, we start the search with a single randomly selected positive point and repeat the experiment 30 times.

Table 5.2: Results on materials discovery data. Averaged over 30 experiments.

	50	100	200	300	400	500	1000	1500	average
GREEDY	84.5	175.0	347.7	522.5	721.8	924.1	2025.9	2981.7	972.9
TWO-STEP	86.0	179.1	349.0	533.2	735.0	938.1	1973.1	3019.4	976.6
ENS-70	*81.5*	*165.1*	*337.7*	524.6	720.0	*910.0*	1790.6	*2757.0*	910.8
ENS-0.7	**80.2**	167.8	**328.4**	**509.7**	708.6	*887.4*	1798.5	*2773.6*	906.8
ENCES-50	85.2	**156.4**	*330.8*	*518.4*	*701.4*	**885.7**	**1745.4**	*2753.2*	*897.1*
ENCES-0.5	87.1	164.3	*336.9*	*513.7*	**683.9**	*891.9*	*1774.1*	**2724.0**	**897.0**

5.5.1 Finding Bulk Metallic Glasses

We conduct experiments on the materials discovery dataset as introduced in previous chapters. We test $T = 50, 100, 200, 300, 400, 500, 1000, 1500$. Table 5.2 shows the average cost. We again highlight the entries with lowest cost in boldface, and those not significantly worse than the best in blue *italic*, under one sided paired t-tests with $\alpha = 0.05$.

We summarize the results as follows: (1) all of the nonmyopic policies outperform GREEDY, and TWO-STEP performs on par with GREEDY. (2) ENCES variants are mostly the best or not significantly worse than the best. (3) ENS is a strong baseline, being the best or not significantly worse than ENCES for several cases. (4) ENCES with r being half of the remaining target performs the best on average, but $r = 50$ is not significantly worse.

5.5.2 Drug Discovery

We use the first nine ECFP4 drug discovery datasets described in previous chapters [10]. The number of positives in the nine datasets are 553, 378, 506, 1023, 218, 916, 1024, 431, and 255, with a shared pool of 100 000 negatives. We set $T = 50, 100, 150, 200$. The average costs

[10]The 10th dataset has 221 positives and it takes a long time to finish the experiment with $T = 200$

Table 5.3: Averaged results over 30 repetitions and nine drug discovery datasets.

	50	100	150	200	average
GREEDY	215.7	414.4	503.2	587.4	430.2
TWO-STEP	71.7	156.0	243.2	322.4	198.4
ENS-30	*58.8*	134.9	208.3	283.3	171.3
ENS-0.7	*59.1*	132.8	212.0	284.2	172.0
ENCES-20	**56.3**	**112.7**	**184.5**	**255.1**	**152.2**
ENCES-0.2	*72.9*	116.0	194.8	298.9	170.7

are shown in Table 5.3. Each entry in this table is averaged over the nine datasets and 30 experiments each, comprising a total of 270 experiments for each policy and target T.

We see a consistent winner: ENCES-20. On average it outperforms all baselines by a large margin. In particular, it improves over GREEDY by a 56–73% reduction in average cost. ENCES–0.2 also performs very well and is at least not significantly worse than ENCES–20 when $T = 50$. Though we only presented results of our method with the best parameters, we point out that it always outperforms GREEDY by a large margin for all other parameters. Full results can be found in Table D.2. Also note that the tested ENS variants – which solve the CEAS problem by reducing it to the budgeted setting – are much better than the myopic methods, but still significantly worse than our proposed method, which suggests it is beneficial to design specific policies for cost effective setting.

To gain insight into the behavior of these policies, in Figure 5.3 we plot the average cost for each policy until $T = 200$ positives are found. The individual curves for each of the nine datasets are shown in Figure D.1. We only show ENCES–20 and ENS–30 in this plot; the curves of ENCES–0.2 and ENS–0.7 are very similar. We see the average cost of GREEDY exhibits a "piecewise linear" shape. This is likely due to its greedy behavior [46]: the method exploits high probability points around a discovered positive neighborhood until exhausted. Afterwards it has to spend a long time to stumble upon another neighborhood, as its greedy

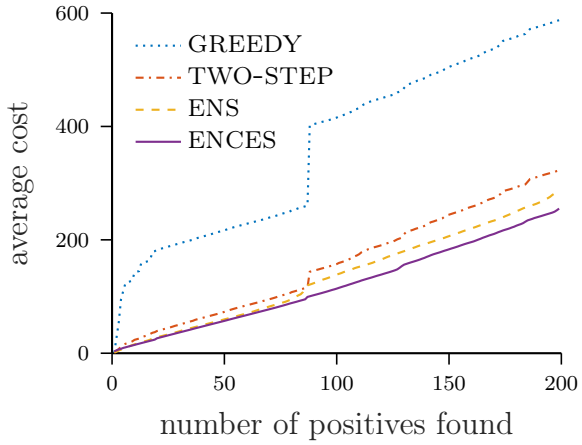


Figure 5.3: Average cost versus the number of positives found, averaged over 9 drug discovery datasets.

behavior did not reveal much information about the space. These episodes of exploitation followed by random search result in the observed discontinuity of the learning curve. In contrast, TWO-STEP and ENS exhibit much smoother behavior with minimal discontinuity. In fact ENCES has almost perfectly *linear cost* w.r.t. T, with the slope only slightly greater than one. This is a very desirable property for cost effective active search.

5.5.3 Pruning Effectiveness

We also show the effectiveness of the pruning technique in Table 5.4. The third row shows the average percentage of pruned points over all candidates in each iteration; the average is taken over all iterations and all experiments. We see the pruning is very effective on all tested datasets; most of the time only 1% of the points need to be evaluated in each iteration.

Table 5.4: Average pruning rate across all iterations in all experiments for the reported ENCES policies.

BMGs		drug discovery	
ENCES-50	ENCES-0.5	ENCES-20	ENCES-0.2
98.64%	98.01%	99.03%	99.06%

5.6 Conclusion

In this chapter, we introduced and studied cost effective active search under the Bayesian decision-theoretic framework. This is the first principled study of the problem in the literature. We derived the Bayesian optimal policy and proved a novel hardness result: the optimal policy is extremely inapproximable, with approximation ratio bounded below by $\Omega(n^{0.16})$. We then proposed an efficient strategy to approximate the optimal policy using the negative Poisson binomial distribution and proposed efficient approximations for its expectation. We demonstrated the superior performance of our proposed policy to several baselines, including the widely used greedy policy and the state-of-the-art nonmyopic policy adapted from budgeted active search. The performance on drug discovery was especially encouraging, with a 56–73% cost reduction on average compared to greedy sampling.

Regarding the tuning parameter in our policy, one rule-of-thumb is to set it to be relatively small (e.g., ≤ 50 or $\leq 50\%$ of the remaining target). How to adapt it in a more principled way is an interesting future direction. Another future direction is to extend the proposed method to batch setting, where multiple points are evaluated simultaneously.

Chapter 6

Bayesian Optimization and Beyond

In this chapter, we propose a novel *efficient and nonmyopic* approximation framework for general SED, called BINOCULARS: **B**atch-**I**nformed **NO**nmyopic **C**hoices, **U**sing **L**ong-horizons for **A**daptive, **R**apid **S**ED. BINOCULARS is inspired by the fact that the optimal batch (or non-adaptive) design is an approximation to the optimal sequential (or adaptive) design. In fact, the optimal adaptive and non-adaptive designs are exactly the same in some notable cases where the data utility does not depend on the observed outcomes, such as maximizing information gain for a fixed Gaussian process (GP) [52]. Even when this is not the case, we show that the optimal batch expected utility is a lower bound of the optimal sequential expected utility. Furthermore, it is tighter than the one-step optimal policy's implied expected utility. Motivated by this insight, BINOCULARS iteratively computes an optimal batch of designs, then chooses one point from this batch. While many existing methods construct batch policies by simulating a sequential policy [18, 31, 45], BINOCULARS goes the other way and "reduces" sequential design to batch design.

BINOCULARS is a general framework applicable to any SED problem, but we only focus on continuous domain in this Chapter. We realize this framework on two important yet *fundamentally different* SED tasks: Bayesian optimization (BO) [55, 67, 82] and Bayesian

quadrature (BQ) [20, 57, 70]. In BO, an agent repeatedly queries an expensive function seeking its global optimum, whereas in BQ the goal is to estimate an intractable integral of the function.

For both problems, many popular policies are myopic: examples include expected improvement (EI) for BO [67] and uncertainty sampling (UNCT) for BQ [38]. These are all one-step optimal for maximizing particular utility functions in expectation. While they are computationally efficient and give reasonably good empirical results, they are liable to suffer from myopia and over-exploitation. Nonmyopic alternatives have recently been applied to BO [35, 56, 105], and although results are promising, these are typically costly to compute.

Our contributions in this chapter can be summarized as follows: (1) We propose a general framework for efficient and nonmyopic SED with finite horizons, inspired by the close connection between optimal sequential and batch designs. (2) We realize the framework on two important SED problems: Bayesian optimization and Bayesian quadrature. This represents the first nonmyopic policy proposed for BQ. (3) We conduct thorough experiments demonstrating that the proposed method significantly outperforms the myopic baselines and is competitive with (if not better than) state-of-the-art nonmyopic alternatives, while being much more efficient.

At last, we also briefly discuss a more advanced technique called *one-shot* optimization for multi-step lookahead Bayesian optimization, and show promising preliminary results.

6.1 Background on Gaussian Processes

In this chapter, we will make extensive use of Gaussian processes, a powerful probabilistic modeling technique. A Gaussian process (GP) defines a distribution over functions, such that

the function values of any finite set of locations follow a multi-variate Gaussian distribution. A GP is fully specified by a mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and a covariance function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ (also known as kernel function). For any finite set of locations $X = [x_1, x_2, \ldots, x_n]$, the function values $Y = [f(x_1), f(x_2), \ldots, f(x_n)]$ follow a joint Gaussian distribution defined as follows:

$$\begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix} \backsim \mathcal{N}\left(\underbrace{\begin{pmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_n) \end{pmatrix}}_{\mu(X)}, \underbrace{\begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & \ldots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \ldots & k(x_2,x_n) \\ \ldots & \ldots & \ldots & \ldots \\ k(x_n,x_1) & k(x_n,x_2) & \ldots & k(x_n,x_n) \end{pmatrix}}_{k(X,X)} \right). \tag{6.1}$$

This is the prior belief. After observing $\mathcal{D} \equiv (X, Y)$, the posterior belief about f is still a Gaussian process. In particular, for any set of locations $X_* \subseteq \mathcal{X}$, their function values Y_* follows a joint Gaussian distribution with mean vector and covariance matrix as follows:

$$\mu_{f|\mathcal{D}}(X_*) = \mu(X_*) + k(X_*, X)k(X, X)^{-1}k(X, X_*)(Y - \mu(X)), \tag{6.2}$$

$$k_{f|\mathcal{D}}(X_*, X_*) = k(X_*, X_*) - k(X_*, X)k(X, X)^{-1}k(X, X_*), \tag{6.3}$$

where $\mu(X_*)$ and $k(X_*, X)$ are defined similarly as $\mu(X)$ and $k(X, X)$.

A common kernel function is squared exponential:

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right),$$

where σ_f^2 is called signal variance and ℓ is input length scale. These are hyperparameters of the GP. A common approach to determine these hyperparameters is type II maximum likelihood

estimation. That is, $\theta = \arg\max_\theta \log \Pr(Y \mid X)$, where $\Pr(Y \mid X)$ is the multi-variate Gaussian density function as defined in (6.1). More details can be found in the GPML book [74].

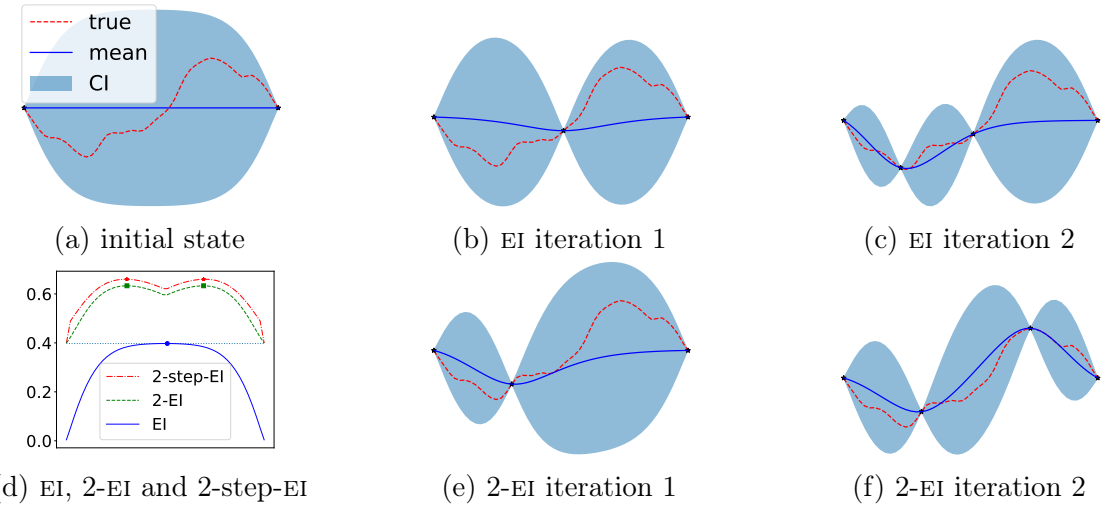
6.2 BINOCULARS

We will first illustrate the intuition behind BINOCULARS and provide explicit mathematical justification. We will then realize BINOCULARS for two specific SED scenarios: BO and BQ. Throughout the rest of this work, we will make extensive use of Gaussian processes (GPs) to model the underlying function f.

6.2.1 Intuition

Consider the BO example in Figure 6.1, where we wish to maximize a one-dimensional objective function over an interval, conditioned on initial observations at the boundary. Suppose we are allowed to design two further function evaluations. The myopic EI policy (introduced later) would greedily pick the middle point first, followed by a point bisecting the left half of the domain. The resulting choices completely ignore the right half, where the maximum happens to lie.

Now consider the following alternative for designing the observations: we first construct the optimal *batch* of size two (2-EI). These points can be determined relatively efficiently as recursion is not required and reflect a better approximation of the remainder of the optimization than just looking one step ahead. We then pick any point from this batch and use EI to choose the final point given the result. This policy results in well-distributed queries



(a) initial state (b) EI iteration 1 (c) EI iteration 2

(d) EI, 2-EI and 2-step-EI (e) 2-EI iteration 1 (f) 2-EI iteration 2

Figure 6.1: An illustration of our proposed nonmyopic method applied to BO. (a) A function in $[-1, 1]$ drawn from a GP where the two end points are known to be zero. (b) and (c) show two iterations of BO with the EI acquisition function. (d) EI, 2-EI and 2-step-EI curves with their respective maximizers. (e) and (f) show two iterations of BO where the first point is chosen from the two points maximizing 2-EI, and the second one is chosen by maximizing EI (conditioned on the observation in iteration one).

and better performance. We can compare these decisions with the *optimal* (but expensive) policy maximizing the full lookahead expected utility (2-step-EI in Figure 6.1d): our choices are nearly perfect.

6.2.2 Non-Adaptive Utility as a Lower Bound of Adaptive Utility

Recall in a state where $\mathcal{D}$ is observed and there are k more steps to go, the general Bellman equation (2.13) for the k-step lookahead expected (marginal) utility is:

$$v_k(x \mid \mathcal{D}) = v_1(x \mid \mathcal{D}) + \mathbb{E}_y[\max_{x'} v_{k-1}(x' \mid \mathcal{D} \cup \{(x, y)\})]. \tag{6.4}$$

The optimal adaptive policy maximizes this expected utility.

Imagine the remaining k experiments $X = \{x_1, \ldots, x_k\}$ must be designed *simultaneously* given current observations $\mathcal{D}$. The expected marginal utility of the resulting observations is

$$V(X \mid \mathcal{D}) = \mathbb{E}_Y[u(\mathcal{D} \cup \{(x_j, y_j)\}_{j=1}^k) - u(\mathcal{D})], \tag{6.5}$$

where we capitalized v to denote batch utility, and the expectation is taken over the joint distribution of $Y = \{y_1, \ldots, y_k\}$, $p(Y \mid X, \mathcal{D})$. Rewriting (6.5) by decomposing X into x_j and X_{-j} where $X_{-j} = X \setminus \{x_j\}$, we have (by telescoping sum)

$$V(X \mid \mathcal{D}) = V(x_j \mid \mathcal{D}) + \mathbb{E}_{y_j}\Big[V\big(X_{-j} \mid \mathcal{D} \cup \{(x_j, y_j)\}\big)\Big]. \tag{6.6}$$

Let $X^* \in \arg\max_X V(X \mid \mathcal{D})$ be an optimal batch of experiments. For any $x_j^* \in X^*$, it follows that

$$\mathbb{E}_{y_j^*}\Big[V\big(X_{-j}^* \mid \mathcal{D} \cup \{(x_j^*, y_j^*)\}\big)\Big] = \max_{X_{-j}} \mathbb{E}_{y_j^*}\Big[V\big(X_{-j} \mid \mathcal{D} \cup \{(x_j^*, y_j^*)\}\big)\Big], \tag{6.7}$$

as otherwise we can construct a batch with higher utility than $V(X^* \mid \mathcal{D})$. Therefore, given that the expected reward of the entire batch can be decomposed using (6.6), choosing any experiment $x^* \in X^*$ is equivalent to solving the following optimization: $x^* \in \arg\max_x v_k'(x \mid \mathcal{D})$ where

$$v_k'(x \mid \mathcal{D}) = v_1(x \mid \mathcal{D})] + \max_{X' : |X'| = k-1} \mathbb{E}_y\Big[V\big(X' \mid \mathcal{D} \cup \{(x, y)\}\big)\Big]. \tag{6.8}$$

Comparing (6.8) and the Bellman equation (6.4), we see two differences: 1) the expectation and maximization are exchanged in the future utility term and 2) the adaptive utility is replaced by a non-adaptive counterpart. As such, (6.8) is clearly a *lower bound of the true*

expected utility:

$$\begin{aligned}
&\max_{X':|X'|=k-1} \mathbb{E}_y\Big[V\big(X' \mid \mathcal{D} \cup \{(x,y)\}\big)\Big] \\
&\qquad \leq \mathbb{E}_y\left[\max_{X':|X'|=k-1} V\big(X' \mid \mathcal{D} \cup \{(x,y)\}\big)\right] \qquad (6.9) \\
&\qquad \leq \mathbb{E}_y\left[\max_{x'} v_{k-1}\big(x' \mid \mathcal{D} \cup \{(x,y)\}\big)\right]. \qquad (6.10)
\end{aligned}$$

This is illustrated in Figure 6.1d: 2-step-EI corresponds to (6.4), and 2-EI to (6.8). An interesting open question is the tightness of this bound, closely related to the so-called *adaptivity gap* [45, 52]. The similarity between these formulations provides mathematical justification for using (6.8) to approximate the optimal policy. Note that (6.8) is exactly equal to (6.4) if the remaining experiments become conditionally independent given the observed data, in which case there is no advantage to adaptation.

We also draw a connection between BINOCULARS and ENS. If we relate (6.9) and (3.7), we can see the future utility approximation in (3.7) has exactly the same form as (6.9), albeit with different utility definitions. Therefore, if the utility function is the same, BINOCULARS maximizes a looser lower bound of the true expected utility than ENS.

6.2.3 Efficient Nonmyopic Approximation Framework

BINOCULARS is summarized in Algorithm 2. The primary computational cost comes from computing the optimal batch, a high-dimensional optimization problem. For the examples considered below (BO and BQ), this optimization can be done using gradient-based methods and we show empirically that BINOCULARS runs much faster than previously proposed nonmyopic methods (see section 6.7). Note that while we do use a batch method, it is only

as a subroutine. Algorithm 2 is for *sequential* experimental design: in each iteration, we only observe the outcome of one experiment.

Algorithm 2 BINOCULARS

Input: design space $\mathcal{X}$, response space $\mathcal{Y}$, model $p(y \mid x, \mathcal{D})$, utility function $u(y \mid x, \mathcal{D})$, budget T
Output: $\mathcal{D}$, a sequence of experiments and observations
for $i \leftarrow 0$ to $T-1$ **do**
  Compute the optimal batch X^* of size $T-i$
  Choose an experiment $x^* \in X^*$ and observe response y^* {see 6.5 on how to choose}
  Augment $\mathcal{D} = \mathcal{D} \cup \{(x^*, y^*)\}$

6.3 BINOCULARS for Bayesian Optimization

Consider the task: $x^* = \arg\max_{x \in \mathcal{X}} f(x)$, where f is modeled as a GP. Suppose we have a budget of T function evaluations. Once the budget has been expended, we recommend the point with the highest observed value as the maximizer of f. In this setting, our goal is to *sequentially* select a set $X = \{x_1, x_2, \ldots, x_T\}$ of T points from $\mathcal{X}$ such that $\max\{y_j\}$ is maximized, where $y_j = f(x_j)$[11].

A natural utility function for this problem is

$$u(\mathcal{D}) = \max_{(x,y) \in \mathcal{D}} y. \tag{6.11}$$

Expected improvement. The well known *expected improvement* (EI) acquisition function is precisely the one-step value function as defined in (2.5) under utility definition (6.11). Let

[11] Typically y is a noisy observation of $f(x)$. Here we assume a noiseless setting for simplicity of presentation. Our method naturally generalizes to noisy setting if plug in appropriate noise model and utility function.

$\mathcal{D}$ be current observations, $y_{\mathcal{D}} = u(\mathcal{D})$ be the best value observed so far. If we plug in the utility function and we have

$$\begin{aligned} v_1(x \mid \mathcal{D}) &= \int_y (u(\mathcal{D}_1) - u(\mathcal{D})) \Pr(y \mid x, \mathcal{D}) dy \\ &= \int_y (\max\{y, y_{\mathcal{D}}\} - y_{\mathcal{D}})) \Pr(y \mid x, \mathcal{D}) dy \\ &= \int_y (y - y_{\mathcal{D}})^+ N(y; m(x), \sigma^2(x)) dy, \end{aligned} \tag{6.12}$$

where $(a)^+ = \max(a, 0)$, and $N(y; m(x), \sigma^2(x))$ is the normal density function with mean $m(x) = \mu_{f|\mathcal{D}}(x)$ and variance $\sigma^2(x) = k_{f|\mathcal{D}}(x, x)$ dictated by the posterior Gaussian process defined in (6.2) and (6.3).

The integral in (6.12) is analytic, we can derive the formula for EI [7]:

$$EI(x) = v_1(x \mid \mathcal{D}) = (m(x) - y_{\mathcal{D}}) \cdot \Phi(Z) + \sigma(x) \cdot \phi(Z), \tag{6.13}$$

where $Z = \frac{m(x) - y_{\mathcal{D}}}{\sigma(x)}$. Though myopic, $EI(x)$ increases with higher posterior mean $m(x)$ or higher the variance $\sigma^2(x)$, balancing exploitation and exploration to some degree. EI is computational efficient and performs reasonably well, so it is arguably the most widely used BO policy.

Note the two-step lookahead value function $v_2(x \mid \mathcal{D})$ is already analytically intractable as it requires an expensive numerical integration: the integrand is $\max_{x'} v_1(x' \mid \mathcal{D}, x, y)$ and entails global optimization!

Batch expected improvement. EI can be extended to the batch setting, known as qEI [31], where a batch of q points are selected in each iteration. Given a batch of q points $X = \{x_1, \ldots, x_q\}$, let $y_i = f(x_i)$ be the corresponding (unknown) function values, and denote

$Y = \{y_1, \ldots, y_q\}$. Similar to (6.12),

$$qEI(X) = \int_y (\max\{Y\} - y_{\mathcal{D}})^+ N(Y; m(X), \Sigma(X)) dy, \tag{6.14}$$

where we replaced the function value y of a single candidate point x by the maximum of a batch of function values $\max\{Y\}$, and the single value Gaussian density by a multivariate Gaussian density function. This integral is not analytically tractable anymore. We have to resort to numerical approximation such as Monte Carlo:

$$qEI(X) \approx \frac{1}{n} \sum_{i=1}^{n} (\max\{Y_i\} - y_{\mathcal{D}})^+, \tag{6.15}$$

where $Y_i, i = 1, \ldots, n$ are n samples from $N(Y; m(X), \Sigma(X))$. However, such approximation loses dependence on X, hence we can not use gradient-based methods to optimize it. A recently popularized reparameterization trick allows us to retain the dependence on X in a differentiable way [94, 101]. Specifically, let $\xi_i \in \mathbb{R}^q$ be a sample from the standard q dimensional Gaussian distribution $N(0, I_q)$; let $L(X)$ be the matrix square root of $\Sigma(X)$, i.e., $\Sigma(X) = L(X)L(X)^T$, then

$$Y_i = m(X) + L(X)\xi_i \tag{6.16}$$

is a sample from $N(m(X), \Sigma(X))$. We can plug in such reparameterized Y_i samples in (6.15), and the approximation is differentiable w.r.t. X as long as $m(X)$ and $L(X)$ are differentiable. Typically, $m(X)$ is polynomial and $L(X)$ is the Cholesky decomposition of $\Sigma(X)$; they are both differentiable w.r.t. X [68].

BINOCULARS for BO. To apply BINOCULARS to BO, we optimize the batch EI objective, qEI, and then pick a point from the optimal batch; how to pick this point is discussed later.

BINOCULARS trivially extends to other utility functions such as knowledge gradient [102], probability of improvement [55] and predictive entropy [81] by replacing qEI appropriately.

6.4 BINOCULARS for Bayesian Quadrature

Consider a non-analytic integral of the form $Z = \int f(x)\pi(x)\,\mathrm{d}x$, where $f(x)$ is a likelihood function and $\pi(x)$ is a prior. Such integrals frequently occur in Bayesian inference (e.g., Bayesian model selection and averaging). Bayesian quadrature operates by placing a GP on the integrand and then minimizing the posterior variance of Z:

$$\mathrm{Var}[Z \mid X] = \iint K_X(x, x')\pi(x)\pi(x')\,\mathrm{d}x\,\mathrm{d}x', \tag{6.17}$$

where $X = \{x_1, x_2, \ldots, x_T\}$ is a set of T points that needs to be optimized, and $K_X(x, x')$ is the posterior covariance after conditioning on observations at X. If the GP hyperparameters are fixed, the optimal design $X^* = \mathrm{argmin}_X \mathrm{Var}[Z \mid X]$ can be precomputed, as the posterior covariance of a GP does not depend on the observed values $f(X)$; this effectively eliminates the need for sequential experimental design in this setting.

However, in general the hyperparameters are not fixed *a priori*, but are instead learned iteratively in light of new observations. Furthermore, when the integrand is known to be *positive* (e.g., a likelihood function), it is often a good practice to place a GP on some non-linear transformation of f, such as $\sqrt{f}$ or $\log(f)$ [8, 38, 71]. As a result, the posterior GP must be approximated (e.g., by moment matching), which causes the posterior covariance to depend on the observed values. In these cases adaptive sampling becomes critical.

The adaptive version of $\mathrm{Var}[Z \mid X]$ is computationally expensive to evaluate so [38] proposed the use of *uncertainty sampling* (UNCT) [58, 80] as a surrogate, i.e. sequentially evaluating the location with the largest variance. This greedily minimizes the entropy of the integrand instead of the integral.

Formally, we use the differential entropy of the multivariate Gaussian $f(X)$ as the utility function:

$$H(Y \mid X) = \tfrac{1}{2} \log \Big(\det \big(2\pi e \, K(X, X) \big) \Big). \tag{6.18}$$

Using the chain rule for differential entropy, this quantity can be expressed in the same form as (2.13):

$$H(Y \mid X) = H(y_j \mid x_j) + \mathbb{E}_{y_j}[H(Y_{-j} | X_{-j}, x_j, y_j)]. \tag{6.19}$$

Note that $\arg\max_{x_j} H(y_j \mid x_j)$ corresponds to the sequential uncertainty sampling policy. To apply BINOCULARS for BQ, we must find $\arg\max_X H(Y \mid X)$, which is the mode of a determinantal point process (DPP) [53] defined over $q = |X|$ points. This can be done using gradient-based optimization. Note that this formulation can be applied to active learning of GPs, where uncertainty sampling is also a common strategy.

6.5 Practical Considerations

Some practical issues arise when applying BINOCULARS to real problems. First, given an optimal batch, how should one go about selecting a point from this batch? We considered several options: selecting the point with the highest expected immediate reward or randomly selecting a point, either proportional to their expected immediate reward or simply uniformly.

Empirically, we found that "best" and "proportional sampling" perform similarly while "uniform sampling" is worse than the other two methods.

Second, given that BINOCULARS is only an approximation to the optimal policy, it is not necessarily true that setting q to the exact remaining budget is the best. In theory, if the model is perfect, then full lookahead is optimal. However, in practice, the model is always wrong and thus planning too far ahead could hurt the empirical performance [105]. Further, smaller values of q result in more efficient computation. We empirically study the choice of q in section 6.7.

6.6 Related Work

Bayesian optimization (BO) is a sample-efficient global optimization method that dates back to the 1960s [54, 55]. It has been popularized in recent years due to its promise in optimizing machine learning hyperparameters [86]. Most of the research in this area focuses on the design of policies, often induced by maximizing so-called *acquisition functions*. Notable examples include probability of improvement [55], expected improvement [67, 78], knowledge gradient [23], upper confidence bound [87], Thompson sampling [49, 79, 91], (predictive) entropy search [39, 40], etc. We refer to [82] for a literature survey.

On the subject of nonmyopic BO, [72] derived the optimal policy for BO, and demonstrate that it is possible to approximately compute (with great effort) the two-step lookahead policy for low-dimensional functions and that it generally performs better than the one-step policy. [29] also derived the optimal policy and gave an explicit example where two-step EI is better than one-step EI in expectation with a desired degree of statistical significance. [35] proposed

a nonmyopic approximation of the optimal policy, known as GLASSES, by simulating future decisions using a batch BO method.[12] [45, 46] proposed a nonmyopic policy for (batch) active search, which can be understood as a special case of BO with cumulative reward, using a similar idea. [56] proposed to use rollout for BO, which is a classic approximate DP method [6]. [105] presented theoretical justification for rollout, and gave theoretical and practical guidance on how to choose the rollout horizon. [60] proposed a branch-and-bound near-optimal policy for GP planning assuming that the reward function is Lipschitz continuous, and applied it to BO and active learning. [103] proposed a gradient-based optimization of two-step EI, but each evaluation of two-step EI still requires a quadrature subroutine with an expensive integrand: optimization of one-step EI.

Of these, GLASSES and rollout are most related to BINOCULARS. GLASSES's acquisition function shares almost the same form as (6.8), except the future batch X' is constructed using a heuristic batch policy, instead of optimized with the qEI objective. The batch policy adds points one by one by optimizing the sequential EI function penalized at locations already added to the batch [33], and the expected utility of the chosen batch is estimated using expectation propagation.

Rolling out two steps using EI as the heuristic policy is exactly equivalent to the two-step lookahead policy, up to quadrature error. Mathematically, the rollout acquisition function can also be written in a similar form as (6.8), except X' is adaptively constructed, depending on sampled values of y instead of globally (irrespective of y) constructed or optimized as in GLASSES and BINOCULARS. Both rollout and GLASSES are very expensive to compute.

While we are unaware of any existing work on nonmyopic BQ, there has been some prior work on nonmyopic active learning of GPs. [52] derived the adaptivity gap for active learning

[12] The name BINOCULARS is inspired by GLASSES.

of GPs under two utility functions. They also proposed a nonmyopic method for active learning of GPs which separates the process into an exploration phase and an exploitation phase. They considered different acquisition functions for the exploration phase; notably, the implicit exploration (IE) method is comparable to the uncertainty sampling baseline in subsection 6.7.2. [41] developed a method for active learning of GPs that does away with separate exploration and exploitation phases and instead naturally trades off between the two. Their proposed policy, ε-BAL, approximates the solution to the Bellman formulation of the active GP learning problem using a truncated sampling method. They analyzed the theoretical performance of their method and also developed a pruning-based anytime version of their method.

The setting of our BQ work (integration of non-negative integrands) and active learning of GPs appear related yet are fundamentally different. The cited works focus exclusively on learning the hyperparameters of the GP. In our setting, the use of a transformation to model non-negativity introduces adaptivity beyond the GP hyperparameters: even if the true GP hyperparameters are known *a priori*, the nonlinear transformation causes the approximate GP posterior to depend on the observed values.

6.7 Experiments

We designed our experiments to broadly test the performance and computational cost of BINOCULARS relative to notable myopic and nonmyopic baselines for BO and BQ. We also conducted a thorough exploration of the BINOCULARS design choices: the number of steps to look ahead and how to select a point from the optimal batch.

The primary takeaways of our experimental results are that BINOCULARS outperforms myopic baselines while running only slightly slower and is at least as good as previously proposed nonmyopic methods while running orders of magnitude faster. This places it *on the Pareto front* of the running time–performance tradeoff in policy design. Furthermore, BINOCULARS clearly demonstrates *distinctively nonmyopic behavior* on both BO and BQ tasks, two entirely different SED problems.

We use the following nomenclature to describe BINOCULARS: our nonmyopic BO method will be denoted as "q.EI.s" or "q.EI.b", where q is the batch size and "s" represents sampling from the batch while "b" means choosing the "best." For BQ, we replace "EI" with "DPP." In addition to the myopic methods, EI and UNCT, we also compare against rollout for both tasks and GLASSES for BO.[13] Each rollout method is denoted as "q.R.n", where q represents the number of steps to roll out, and n is the number of samples used to estimate the expectations encountered in each step. Each GLASSES method is denoted as "q.G" where q represents the size of the simulated batch. We use DIRECT [48] to optimize the GLASSES and rollout acquisition functions, following [35]. For all nonmyopic methods, when the remaining budget $r < q$, we set $q = r$. Thus the final decision is always made (optimally) with one-step lookahead.

For all experiments, we start with $2d$ randomly-sampled observations and perform $20d$ further iterations, where d is the function's dimensionality. Unless otherwise noted, all results presented are aggregated over 100 repeats with different random initializations. For all tabulated results, the best method is indicated in bold and the entries not significantly worse than the best (under a one-sided paired Wilcoxon signed-rank test with $\alpha = 0.05$) are in blue italics.

[13] We did not compare against a BQ-equivalent of GLASSES as no such method has been published.

6.7.1 BO Results

We implemented our nonmyopic BO policy and all baselines using `BoTorch`,[14] which contains efficient EI and qEI implementations. We present experiments for two rollout variants: "2.R.10" and "3.R.3." As we will see, rolling out with horizon two is already very expensive even for just ten y samples. Gauss–Hermite quadrature is used for rollout as in [56]. We also present experiments for two GLASSES variants: "2.G" and "3.G". We implemented a slightly different version of GLASSES in `BoTorch`, using quasi Monte Carlo instead of expectation propagation to estimate the expected improvement of the batch, a standard practice for computing qEI in state of the art BO packages such as `BoTorch`.

We use GPs with a constant mean and a Matérn $5/2$ ARD kernel to model the objective function, the default in `BoTorch`. We tune hyperparameters every iteration by maximizing the marginal likelihood using L-BFGS-B. We also maximize the qEI acquisition function with L-BFGS-B. We use the GAP measure to evaluate the performance: $\text{GAP} = (y_i - y_0) \,/\, (y^* - y_0)$, where y_i's are maximum observed values and y^* is the true optimal value; we convert all problems to maximization problems by negating if necessary.

Synthetic functions. In this section, we focus on demonstrating the superior performance of our method over EI on nine "hard" benchmark functions. These nine functions are selected by first running experiments on 31 functions[15] with 30 repeats (see Table D.3). We then select the ones where EI terminates with average GAP < 0.9. We believe nonmyopic methods are more advantageous on challenging functions; by first identifying these hard problems, we will gain more insight into the various policies. To put the BO performance into perspective, we also include a comparison against a random baseline, "Rand."

[14] `https://github.com/pytorch/botorch`
[15] `https://www.sfu.ca/~ssurjano/optimization.html`

Table 6.1: Average GAP over 100 repeats on "hard" synthetic functions.

	Rand	EI	2.EI.b	2.EI.s	3.EI.b	3.EI.s	4.EI.b	4.EI.s	10.EI.b	10.EI.s	12.EI.s	15.EI.s
eggholder	0.498	0.613	0.614	0.633	0.604	0.657	0.646	*0.694*	0.622	*0.704*	**0.738**	*0.694*
dropwave	0.486	0.439	0.507	0.531	0.473	*0.552*	0.467	0.514	0.397	*0.591*	**0.598**	*0.585*
shubert	0.355	0.408	0.366	*0.441*	*0.394*	**0.507**	0.388	*0.484*	0.305	*0.455*	*0.479*	*0.465*
rastrigin4	0.374	0.801	0.769	0.775	0.817	*0.821*	**0.840**	0.805	0.797	0.804	0.793	0.799
ackley2	0.358	0.821	0.825	0.823	0.819	*0.869*	0.812	*0.872*	0.801	**0.892**	*0.886*	*0.888*
ackley5	0.145	0.509	0.544	0.509	0.601	0.550	0.596	0.592	**0.636**	0.606	*0.627*	*0.626*
bukin	0.600	*0.849*	0.856	0.855	*0.872*	*0.859*	0.864	*0.865*	**0.878**	0.850	0.829	*0.853*
shekel5	0.038	0.286	0.311	0.320	0.330	0.343	0.342	0.344	*0.374*	*0.373*	*0.358*	**0.395**
shekel7	0.045	0.268	0.346	0.313	0.349	0.325	0.352	0.370	*0.399*	0.358	**0.412**	*0.386*
Average	0.322	0.555	0.571	0.578	0.584	0.609	0.590	0.616	0.579	*0.626*	**0.635**	*0.632*

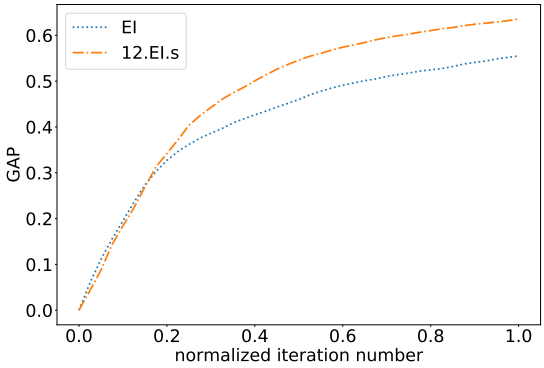


Figure 6.2: Average GAP over nine synthetic functions demonstrating the nonmyopic behavior of 12.EI.s.

Table 6.1 shows the average GAP at termination. We summarize the results as follows: (1) All q.EI.s variants perform significantly better than EI on average, with 12.EI.s being the best and outperforming EI by a large margin. (2) The q.EI.s variants are consistently better than the q.EI.b variants (for better spacing we did not show results for 12.EI.b and 15.EI.b). (3) The performance of our method generally improves as we increase q, up to 12.

Perhaps more interestingly, we can clearly observe the nonmyopic behavior of 12.EI.s as shown in Figure 6.2: it is initially outperformed by the myopic EI as it explores the space. However, our method catches up to EI at $\sim$20% of the budget (on average) as it transitions to exploiting its findings until finally, it outperforms EI by a large margin. This behavior

Table 6.2: Average GAP over 50 repeats on real functions.

	EI	2.EI.s	3.EI.s	4.EI.s	6.EI.s	8.EI.s	2.G	3.G	2.R.10	3.R.3
SVM	0.738	*0.913*	**0.940**	*0.911*	*0.937*	0.834	*0.881*	0.898	*0.930*	*0.928*
LDA	0.956	**1.000**	*0.996*	*0.993*	0.982	*0.995*	*1.000*	*0.999*	*0.999*	*1.000*
LogReg	0.963	*0.998*	*1.000*	*0.999*	*0.999*	**1.000**	0.989	0.911	0.965	0.948
NN Boston	*0.470*	*0.467*	*0.478*	*0.460*	*0.502*	*0.467*	0.455	**0.512**	*0.503*	*0.482*
NN Cancer	0.665	0.627	0.654	0.686	0.700	0.686	**0.806**	*0.755*	0.708	0.698
Robotpush3	0.928	*0.960*	*0.962*	*0.957*	**0.962**	*0.961*	*0.955*	0.951	*0.955*	*0.954*
Robotpush4	*0.730*	0.726	0.695	0.695	0.736	0.697	*0.765*	**0.786**	*0.770*	*0.745*
Average	0.779	*0.813*	*0.818*	0.815	*0.831*	0.806	**0.836**	*0.830*	*0.833*	*0.822*

indicates that our method seamlessly navigates the exploration/exploitation tradeoff without the need for any external intervention.

Real world functions. In this section, we compare our method against popular nonmyopic baselines: rollout and GLASSES. We present results on hyperparameter tuning functions used by [65, 86, 96]. These functions are evaluated on a predefined grid, so we first compute all policies (except EI) using continuous optimization, then pick the closest point from the grid.

Table 6.2 shows the results averaged over 50 repeats. We only show the "sampling" variants of our method; full results can be found in Table D.4. First we see again all q.EI.s variants outperform EI by a large margin, with $q = 6$ achieving the best results. Comparing 6.EI.s with the nonmyopic baselines, 2.G is the best, but the difference of 0.005 is negligible; the p-value under a one-sided paired signed-rank test for 6.EI.s against 2.G is 0.4257.

We now focus on comparing the time cost of the tested methods. Figure 6.3 shows the average GAP versus average time per iteration; the average is taken over 350 experiments (seven functions with 50 repeats each); error bars are also plotted. We again see that our methods are not significantly different from rollout and GLASSES in terms of GAP performance, but are considerably faster in terms of average time cost per iteration (note the log scale on the

time axis). Clearly, our method lies on the Pareto front in terms of computational cost and performance.

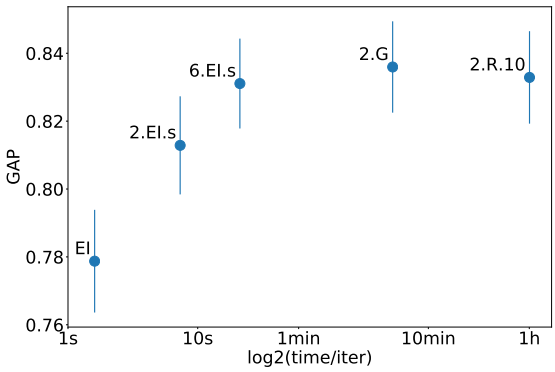


Figure 6.3: mean GAP with error bars at termination versus time per iteration (in log scale) averaged over the seven real functions.

We also attempted to compare with the recently published practical two-step EI method [103], which is intended to be a more efficient version of our 2.R.n. As far as we understand, the difference is first- versus zeroth-order optimization of the acquisition function. In fact, our implementation of rollout already supports gradient-based optimization thanks to automatic differentiation. However, we did not find it considerably faster than using DIRECT. We leave it to future work to optimize the implementation and compare with our method.

It is also possible to further improve rollout's performance using an adaptive rolling horizon in light of a recent study [105], but it would be even more expensive to compute. In fact, Figure 1 in [105] shows that with their adaptive horizon approach, the most frequently chosen horizon was two.

6.7.2 BQ Results

We implemented our nonmyopic BQ policy and all baselines using the GPML MATLAB package.[16] For all BQ experiments, we use the framework of [8]: we place GP priors on the log of the integrands as they are all non-negative. We use GPs with a constant mean and a Matérn 3/2 ARD kernel to model the integrands. We tune the GP hyperparameters after each observation by maximizing the marginal likelihood using L-BFGS-B. We also use L-BFGS-B to maximize the DPP likelihood. Complete details of our implementation can be found in our attached code.

We perform experiments on five standard benchmark synthetic functions[17] as well as one additional synthetic benchmark and two real model likelihood functions used by [9]. The additional synthetic benchmark is: $f(x) = \prod_{i=1}^{d} \frac{\sin(x_i)+\cos(3x_i))^2/2}{x_i^2/4+0.3}$; this function was included because of its multi-modal (MM) nature. We evaluate the performance of all methods using their fractional error: $|Z - \hat{Z}|/Z$ where $\hat{Z}$ is the estimate of the integral.

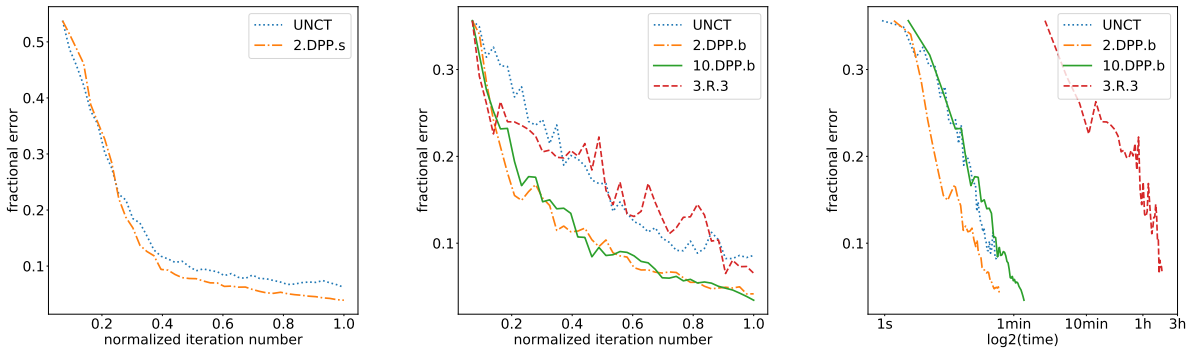
Figure 6.4a indicates that 2.DPP.s exhibits the same nonmyopic behavior as 12.EI.s: it initially lags behind but eventually overtakes the myopic UNCT, again suggesting a superior and automatic tradeoff of exploration and exploitation.

Table 6.3 shows the median fractional error at termination for all BQ experiments. Figure 6.4 shows the convergence of the fractional error as a function of both iterations and time (in log scale). These results corroborate many of the findings from our BO experiments: (1) All nonmyopic methods outperform UNCT on average. (2) Our proposed nonmyopic methods are competitive with, if not better than, rollout while running orders of magnitude faster.

[16] `http://gaussianprocess.org/gpml/code/matlab`
[17] `https://www.sfu.ca/~ssurjano/integration.html`

Table 6.3: Median fractional error values over 100 repeats on all BQ functions.

	UNCT	2.DPP.b	3.DPP.b	10.DPP.b	2.DPP.s	3.DPP.s	10.DPP.s	2.R.10	3.R.3
cont	0.045	0.052	0.055	0.059	0.039	0.037	**0.029**	0.036	0.045
corner	0.265	0.206	0.137	0.065	**0.047**	0.078	0.132	0.074	0.063
discont	*0.523*	*0.511*	*0.488*	**0.446**	0.572	0.610	0.590	0.537	0.577
Gauss	*0.004*	0.004	0.005	0.006	**0.003**	*0.003*	*0.003*	0.004	*0.003*
MM	0.254	0.207	0.203	0.207	0.221	0.161	0.177	*0.110*	**0.086**
prod	0.007	0.007	*0.007*	0.007	0.007	**0.006**	*0.006*	0.012	0.012
GP	0.231	0.082	**0.057**	0.077	*0.069*	*0.073*	0.116	0.283	0.248
DLA	0.019	*0.013*	0.025	*0.013*	*0.016*	*0.016*	0.033	*0.019*	**0.011**
Average	0.068	0.056	0.055	*0.041*	**0.037**	*0.043*	0.055	0.049	0.051



(a) (b) (c)

Figure 6.4: Median fractional error over 100 repeats against iterations or time. (a) synthetic functions, (b) real functions, (c) real functions.

We also note that in general, q.DPP.s variants tend to outperform q.DPP.b variants and increasing the batch size q does not consistently improve the performance.

The primary conclusion here is the same as for BO: BINOCULARS significantly and consistently outperforms myopic policies while only slightly increasing computational cost.

6.8 Multi-Step Lookahead via One-Shot Optimization

Before we conclude this chapter, we introduce an ongoing effort that makes multi-step lookahead Bayesian optimization possible. We will introduce the idea and show some preliminary results, on top of BINOCULARS. The approach presented in this section is also applicable to any continuous SED problems, but we focus on Bayesian optimization.

We illustrate the idea with the simplest case: two-step lookahead. Recall the two-step value function for Bayesian optimization (or any SED problem) is as follows:

$$v_2(x \mid \mathcal{D}) = v_1(x \mid \mathcal{D}) + \mathbb{E}_y\Big[\max_{x'} v_1(x' \mid \mathcal{D}_1)\Big]. \tag{6.20}$$

To evaluate $v_2(x \mid \mathcal{D})$, we must resort to numerical integration since the expectation in the second term is not analytically tractable. Let $y_1, \ldots, y_m$ be m samples drawn from $\Pr(y \mid x, \mathcal{D})$, then a Monte Carlo approximation would be

$$v_2(x \mid \mathcal{D}) \approx v_1(x \mid \mathcal{D}) + \frac{1}{m}\sum_{i=1}^{m} \max_{x'} v_1(x' \mid \mathcal{D}, x, y_i) \equiv g(x). \tag{6.21}$$

Note each evaluation of $g(x)$ involves m optimization problems, hence very expensive.

6.8.1 Previous Work on Two-Step Lookahead BO

There has been several attempts for two-step lookahead BO [29, 56, 72], but it was not until recently that it was claimed practical due to [103]. An approximate gradient estimate method is proposed in [103], which allows us to optimize the two-step lookahead value function with gradient-based methods. The method to differentiate $g(x)$ is based on envelope theorem.

Basically, for each y_i, we first perform optimization to get $x_i^* = \arg\max_{x'} v_1(x' \mid \mathcal{D}, x, y_i)$, and then plugging it back into $g(x)$ we have

$$g(x) = v_1(x \mid \mathcal{D}) + \frac{1}{m} \sum_{i=1}^{m} v_1(x_i^* \mid \mathcal{D}, x, y_i). \tag{6.22}$$

Assuming x_i^* is a global maximizer of $v_1(x' \mid \mathcal{D}, x, y_i)$, then $g(x)$ is differentiable w.r.t. x (see Theorem 1 in [103] for more detailed requirements). [103] reported superior efficiency and BO performance using stochastic gradient ascent to optimize $g(x)$.

This approach is certainly an advance towards nonmyopic BO. However, it is still problematic in two aspects: first, though less iterations (than zeroth-order optimization) are needed to optimize $g(x)$ as claimed in [103], each evaluation of $g(x)$ still involves many optimizations, hence still expensive; second, the assumption that x_i^*'s are global maximizers may not be practical, especially when the $v_1(x' \mid \mathcal{D}, x, y_i)$ surfaces are multi-modal. Note this assumption must be satisfied for every $x_i^*, i = 1, \ldots, m$; any violation will lead to discontinuity of $g(x)$, not to mention differentibility.

6.8.2 One-Shot Approach

A key observation is: we do not need to optimize $v'(x' \mid \mathcal{D}, x, y_i)$ to full extent to get an ascent direction of $g(x)$. In abstract terms, $g(x)$ is a function of the form

$$g(x) \equiv \max_{x'_1, \ldots, x'_m} h(x, x'_1, \ldots, x'_m).$$

Observe that

$$\max_x g(x) = \max_{x,x'_1,\ldots,x'_m} h(x, x'_1, \ldots, x'_m). \tag{6.23}$$

Instead of optimizing $x'_1, \ldots, x'_m$ to full extent to get an ascent direction of x, why not optimize x and the $x'_1, \ldots, x'_m$ jointly? Note h is differentiable w.r.t. both x and $x'_1, \ldots, x'_m$ for commonly used value functions such as expected improvement. This is precisely the idea of *one-shot optimization*, first proposed to efficiently optimize the knowledge gradient acquisition function [4], and here we adapt it to optimizing multi-step lookahead value function. We solve the optimization on the right side of (6.23) instead of the left side, and the partial solution x^* from the full one $[x^*, x_1^*, \ldots, x_m^*]$ would be the maximizer of $g(x)$. Intuitively this may take more iterations than the envelope approach as discussed above, but each iteration is much cheaper since no nested maximizations are required.

It is straightforward to extend this idea to general k-step lookahead by *jointly optimizing over the whole decision tree*. If we draw m samples in each stage, there will be $(1 + m + m^2 + \ldots + m^{k-1})d = \frac{m^k-1}{m-1}d$ variables to jointly optimize, where d is the dimension of each x. The dimension of the joint optimization problem grows exponentially w.r.t. the lookahead horizon k, so it is still impractical for large k. But we are able to manage $k = 4$. Note $k \geq 3$ was never attempted before due to its dauntingly high computational complexity.

6.8.3 Preliminary Results

We show some preliminary results in Figure 6.5, averaged over the nine "hard" synthetic functions (see Table 6.1) and 100 repetitions. We can see multi-step EI improves over EI by a very large margin, and is even much better than the best variant of BINOCULARS, 12.EI.S, as

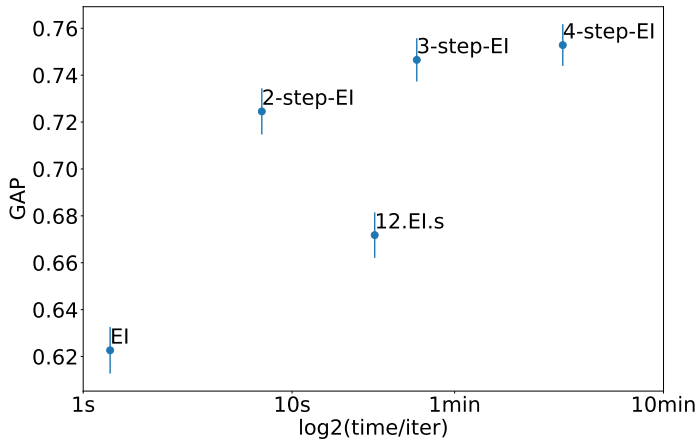


Figure 6.5: Preliminary results of multi-step lookahead Bayesian optimization via one-shot optimization.

reported in Table 6.1.[18] First we see two-step-EI only takes about 10 seconds per iteration on average, and four-step-EI can be optimized well within 10 minutes. We also see the marginal improvement of k-step-EI over $(k-1)$-step-EI is diminishing. We conjecture that looking more steps ahead intrinsically has diminishing return, but currently we can not rule out the possibility that this was (at least partially) because the optimization of a larger horizon value function becomes harder.

Remark 1. *The one-shot optimization can be naturally extended to batch policies, and each stage could have a different batch size. In particular, for the two-step case, if we set the batch size of the first stage to one, and that of the second stage to be the remaining budget minus one, then this is precisely an implementation of* ENS *(see Chapter 3) for Bayesian optimization.*

[18]This part of work is done with an updated version of `BoTorch`, so we reran EI and 12.EI.s. The average numbers shown in Figure 6.5 might be slightly different from what are reported in Table 6.1, but the patterns about BINOCULARS remain the same.

6.9 Conclusion

We proposed BINOCULARS: an efficient, nonmyopic approximation framework for finite-horizon sequential experimental design. BINOCULARS computes an optimal batch, then picks a point from the batch. We gave an intuitive understanding and a mathematical justification for why this is a good approximation. We applied BINOCULARS to Bayesian optimization and Bayesian quadrature, two entirely different problems, and empirically demonstrated that it significantly outperforms commonly used myopic policies while being much more efficient than popular nonmyopic alternatives.

We also briefly discussed an ongoing work that makes multi-step lookahead Bayesian optimization practical, which naturally generalizes to any continuous SED problems. The basic idea is to jointly optimize all the variables in the whole decision tree in "one-shot", instead of repeatedly solving nested maximizations. Preliminary results show very promising BO performance achieved with high computational efficiency.

As suggested by [105], it would be useful to derive theories to guide the choice of lookahead horizon q for our method. Another interesting theoretical question is whether we can provide explicit bounds for the adaptivity gap for general continuous SED problems.

Chapter 7

Conclusion and Future Work

In this thesis, we studied several important problems in sequential experimental design (SED), with a focus on developing *efficient and nonmyopic* policies. The key idea of our nonmyopic policies is to *use the non-adaptive expected utility to approximate the adaptive expected utility*, which results in policies that maximize a lower bound of the true expected utility, much tighter than myopic approximations yet computationally efficient. We implemented variants of this idea for several SED settings such as active search, Bayesian optimization and Bayesian quadrature. We demonstrated superior empirical performance in important application domains such as drug discovery, materials discovery, model hyperparameter tuning, etc. We often observe desirable behavior of our nonmyopic policies that well balances exploration and exploitation in our experiments. We also established the theoretical hardness of active search for both the budgeted and cost effective settings, and derived a lower bound on the adaptivity gap for batch budgeted active search, which was shown to match with the empirical results. This result could provide guidance on choosing batch sizes in practice. Finally, we discussed a novel one-shot optimization technique that shows promising performance for multi-step lookahead Bayesian optimization, which has never been attempted before. We believe our

contributions represent a significant advance towards efficient and nonmyopic sequential experimental design, from both a theoretical and practical perspective.

In the remainder of the thesis, we discuss some future directions.

7.1 Theoretical Guarantees

We proposed many nonmyopic approximations for various problems throughout the thesis. Despite their remarkable empirical performance, these approximations did not come with theoretical guarantees. In fact, for active search we proved the performance gap between any efficient policies and the optimal policy can be arbitrarily large. However, such pessimistic results are for worst-case scenarios. The proof relies on constructing active search instances that have very atypical dependence among the points, which makes the optimal policy hardly approximable. Another extreme case is no dependence among the points at all, in which even a random policy is optimal. The most interesting and practical cases lie in between. It is thus natural to ask: under what practical regularity assumption on the structure of the problem can we derive theoretical guarantees, and how? A possible starting point is to confine the family of joint distributions over the label space, which is essentially multivariate Bernoulli [17], so that the correlations among the Bernoulli variables reflect typical active search instances in practice.

The approximation guarantee is related to the other theoretical result we had: adaptivity gap. Since almost all our proposed nonmyopic solutions are based on the idea of "using the non-adaptive expected utility to approximate the adaptive counterpart", a natural first step to bound the performance gap is to bound the adaptivity gap. What we derived is a lower

bound, but what we need is an upper bound. Note that the lower bound we derived is also for the worst-case scenario, which relies on constructing active search instances structured in a binary tree with extremely skewed probabilities for the dichotomizing branches. This is also not typical in practice. With some regularity assumption on the probability distribution, it should be possible to derive an upper bound. It is worth mentioning that the lower bound matches with our empirical results, indicating the lower bound might also be an upper bound. Therefore, a linear factor in batch size might be a good target bound to start with.

It is also possible to derive the adaptivity gap (and hence approximation ratio) in the continuous case. A naive approach would be to discretize the space and generalize the binary tree construction to multi-way tree, and further bound the error due to discretization.

7.2 Learning to Search

We revisit active search in this section, but with a specific application scenario, where there are many similar active search problems to solve. This is typical in drug discovery, where each biological target defines a new active search problem, and we often need to search for useful compounds for many different targets in a fixed or similar search space. After we solve many similar active search problems, we have accumulated considerable valuable experience that could be useful for solving a new problem in a similar domain. We ask the question: "can we transfer previous active search experience to solving a new problem?"

Meta-learn a better model. While we focused on policy design in this thesis, accurate modeling of the underlying function is also of crucial importance. This is especially true for nonmyopic policies due to its increasing reliance on the model with more lookahead. For all

the drug discovery experiments in this thesis, we used the k-nn model, which works quite well as we have shown repeatedly. However, can we do even better by leveraging many available (partially) labeled datasets from solving past active search problems?

One possible approach is a recently proposed meta-learning method known as *conditional neural processes* [24]. We adapt this approach to our setting, where the Gaussian process generating the training data is replaced by the empirical distribution composed of a family of active search datasets, and of course the regression loss is replaced by classification loss. We summarize the high level ideas as follows. Suppose we have m datasets $\{(X_i, Y_i)\}_{i=1}^m$, where dataset i contains the labeling information after solving active search for biological target i, and $X_i \in \mathbb{R}^{n_i \times d}$ and $Y_i \in \{0,1\}^{n_i}$. For now we assume all unlabeled compounds are negative.

A machine learning algorithm typically takes in a training set (or *context set*) $\mathcal{D} = (X_c, Y_c)$, and output a model $\hat{f}$, and then $\hat{f}$ is used to predict on some unseen *target* input X_t, such that $\hat{Y}_t = \hat{f}(X_t)$. This process can be summarized in a single function F:

$$\hat{Y}_t = F(X_t, X_c, Y_c). \tag{7.1}$$

If we have many such context and target set pairs, we can train a universally good model F, which we can then apply to an unseen context and target set. This is exactly the idea of conditional neural processes: we draw many context and target set pairs from a family of datasets in the same domain, and learn a *distribution* over F by minimizing the loss $\sum_{(x,y)\in(X',Y')} \ell(y, F(x, X_c, Y_c))$, where the loss is defined on both the context and target points: $X' = [X_c, X_t], Y' = [Y_c, Y_t]$.

A possible implementation of training a conditional neural process with stochastic gradient descent is summarized in Algorithm 3. It is also possible to improve this method with an

Algorithm 3 Conditional Neural Process Training

Given m datasets $\{(X_i, Y_i)\}_{i=1}^m$
repeat
  Randomly draw an index i from $1, \ldots, m$;
  Draw a random subset (X', Y') of some random size from dataset i;
  Partition the subset into context set and target set $(X', Y') = ((X^c, Y^c), (X^t, Y^t))$;
  Encode the context set: $s = h_{w_1}(X^c, Y^c)$, where h_{w_1} is a deep set architecture (see (7.2));
  Construct predictor: $\Pr(y = 1 \mid x, X', Y') = g_{w_2}(x, s)$;
  Compute logistic loss $L(w_1, w_2) = \sum_{(x,y)\in(X',Y')} \ell(y, g(x, s))$;
  Compute gradient $\nabla L(w_1, w_2)$ with back-propagation;
  Update w_1, w_2 with a gradient step to minimize the loss;
until convergence
return meta-model: $\Pr(y = 1 \mid x, \mathcal{D}) = g_{w_2^*}(x, h_{w_1^*}(\mathcal{D}))$, for any $x, \mathcal{D}$.

attention mechanism, so that the model learns to focus on different context points for different target input [51].

Meta-learn a better policy. As discussed in 2.4.2, it is also possible to adopt a model-free reinforcement learning approach to directly learn a policy. The past experience can be represented in a format such as "how much utility we got in the end if we choose x when set $\mathcal{D}$ is observed". This is precisely the definition of action values $Q(\mathcal{D}, x)$, which reminds us of Q-learning [90]. One key question is: how to represent the Q function, or essentially, the state $\mathcal{D}$?

A good representation of the observed set $\mathcal{D}$ relies on the good representation of each individual point in $\mathcal{D}$. For the specific application of drug discovery, the individual points would be molecules. Commonly used representations in cheminformatics are based on Weisfeiler-Lehman (WL) algorithm operating on a molecular graph [83], such as Extended-Connectivity Fingerprints (ECFP) [76]. They are basically a hashing-based indicator vectors indicating the presence of molecular substructures of different radii.

Recently, *graph neural networks* [21, 28] has become a popular and powerful tool for learning molecular representations, which can be seen as a differentiable version of the WL algorithm. The molecular graph is mapped into a continuous vector in a differentiable way via message passing [28], so that a data-driven representation can be learned via back-propagation. We adopt a recently proposed graph-based variational autoencoder called JTVAE [47], and use the latent mean as a representation of a molecule. Preliminary results show that this representation achieves much better performance than ECFP4 for active search on the drug discovery datasets we have been using throughout this thesis.

Given a representation of each point x, we still need to represent $\mathcal{D}$. Previous work on meta-learning policies for Bayesian optimization [13] treats $\mathcal{D}$ as a sequence and encodes it via a recurrent neural network. While we could adapt this work to active search, it might be more appropriate to use a set function to represent the state $\mathcal{D}$, since $\mathcal{D}$ is a set and the order does not matter. One option is deep sets [106]. The idea is to represent state $\mathcal{D}$ as

$$s = \rho\left(\sum_{(x,y)\in\mathcal{D}} \phi(x,y)\right), \tag{7.2}$$

with transformations ρ and ϕ represented as deep neural networks.

As discussed in 2.4.1, active search is an MDP where the observation set $\mathcal{D}$ alone is not enough to represent the state. The locations and geometry of all unobserved points should also be part of the state. This is easy to understand: given the same observations $\mathcal{D}$, the expected utility of the same point might be very different if it is a cluster center than if it is just a singleton point. How to incorporate such "background" information into the state representation in a meaningful and computationally efficient way is an interesting problem.

That said, explicitly representing the "background" may not be necessary if we always search in a *fixed space*; such geometry information should be implicitly embedded in the network.

7.3 Multi-Fidelity Active Search

In scientific discovery, the candidate pool (e.g., molecules, alloys, etc.) is usually put through several stages of screening to narrow down the targets. These screening procedures could be computational simulation, laboratory tests, or human inspection, and later stages usually come with higher cost. How to search for desired targets in such settings with a cost effective approach is an important problem.

This motivates *multi-fidelity active search*! Suppose there is a family of functions $\mathcal{F} = \{f_i \colon \mathcal{X} \mapsto \mathcal{Y}, i = 1, \ldots, m\}$ with increasing *fidelity* to the true label, and the cost of evaluating $f_i(x)$, c_i, is increasing such that $c_1 \leq c_2 \leq \ldots \leq c_m$. How to define "fidelity" could depend on the task at hand. A typical case is that F defines a family of nested subsets containing the true targets $\mathcal{R} \subseteq \mathcal{X}$, that is, $\{x \in \mathcal{X} \colon f_1(x) = 1\} \supseteq \{x \in \mathcal{X} \colon f_2(x) = 1\} \supseteq \cdots \supseteq \{x \in \mathcal{X} \colon f_m(x) = 1\} = \mathcal{R}$, essentially formalizing the concept of "narrowing down." Given observations $\mathcal{D} = \{(x_j, y_j)\}$, where each $y_j \in \{0, 1, ?\}^m$ and question mark denotes "unknown", the problem is to design a policy that recommends which point to evaluate with which function. The procedure in budgeted setting is summarized in Algorithm 4.

Suppose we still adopt Bayesian decision theory to solve this problem. The first challenge we face is modeling both intra- and inter-function correlations. In particular, for any $x \in \mathcal{X}$, $k < i$, we have: $f_i(x) = 1$ implies $f_k(x) = 1$, $f_k(x) = 0$ implies $f_i(x) = 0$, $f_k(x) = 1$ would

Algorithm 4 Multi-fidelity active search

Given candidate pool $\mathcal{X} = \{x_1, \ldots, x_n\}$;
Given functions $f_1, \ldots, f_m$ with increasing fidelity and costs $c_1, \ldots, c_m$;
Given total cost budget C;
Initialize all $y_j = [?, ?, \ldots, ?], j = 1, \ldots, n$ (vector of length m);
Initialize $\mathcal{D} = \{(x_j, y_j)\}$;
repeat
  $i, j = Policy(\mathcal{D})$; {Goal: maximize $\sum \mathbb{1}\{y_j[m] == 1\}$}
  Evaluate: $y_j[i] = f_i(x_j)$;
  Deduct cost: $C = C - c_i$;
  Update $\mathcal{D}$ with y_j;
until $C \leq 0$

likely increase the probability of $f_i(x) = 1$, and $f_i(x) = 0$ would likely decrease the probability of $f_k(x) = 1$.

Given a model, the derivation of Bayesian optimal policy is also much more challenging than simple active search where the budget is simply defined as the number of iterations. It is possible to borrow ideas from dynamic programming for *knapsack problem*. Also it may be wise to start with a restricted policy class. For example, we do not allow coming back to an earlier stage after passing it. That is, commit actions only in the first stage, observe, then the second stage, observe, etc., till the last stage. We can imagine the optimal policy is still challenging to derive, even if we further restrict the sub-policy in each stage to be non-adaptive. The good news is we usually only deal with two or three stages in practice.

Appendix A

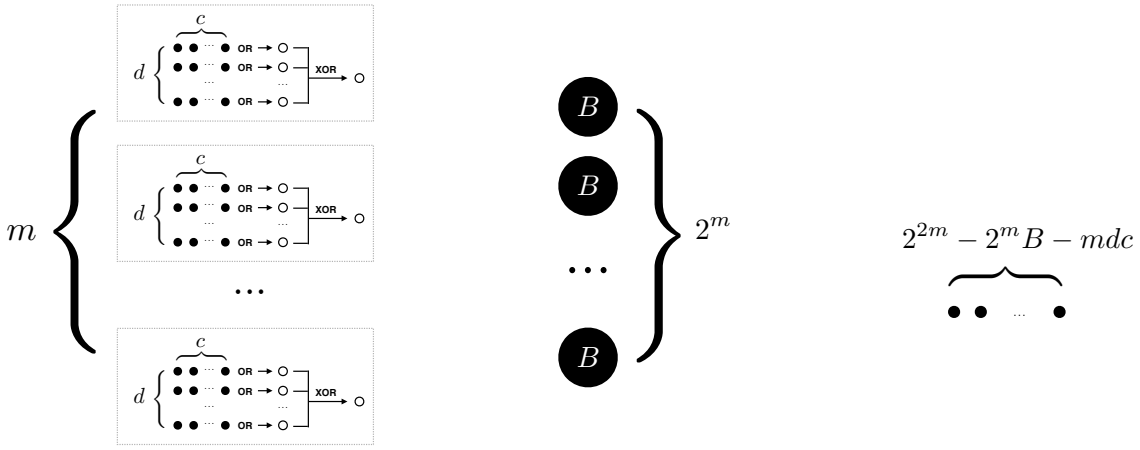
Hardness of Budgeted Active Search

In this section, we present the proof of Theorem 1. We assume that active search policies have access to the correct marginal probabilities $f(x; \mathcal{D}) = \Pr(y = 1 \mid x, \mathcal{D})$, for any given point x and labeled data $\mathcal{D}$, which may include "ficticious" observations. Further, the computational cost will be analyzed as the number of calls to f, i.e., $f(x; D)$ has unit cost. Note that the optimal policy operates in such a computational model, with exponentially many calls (in terms of $|\mathcal{X}|$) to the marginal probability function f.

Our proof technique consists of constructing an explicit active search instance where a small "secret" set of points S encodes the location of a larger "hidden" group of positive points. A particular feasibly exponential-cost policy identifies this small set first, and then obtains a large reward by collecting the revealed targets. We will show that an algorithm with limited computational power (i.e., polynomial in $n = |\mathcal{X}|$) will not be able to identify the set S. As a consequence, its performance will be arbitrarily worse than an optimal solution as the size of the instance increases.

The crux of the proof is to construct a class of instances $\mathcal{H}$ that we present next. Figure A.1 shows a schematic representation of an example instance $\mathcal{I} \in \mathcal{H}$. The instances in $\mathcal{H}$ differ from each other by a permutation of the labels. An instance has $n = |\mathcal{X}| = 2^{2m}$ points, where m is a parameter of the instance. The search budget is defined to be $B = m^2$. The points in each instance can be categorized as follows.

"Clumps." These points are partitioned into 2^m groups, which we will call "clumps," each of size B. All points in a clump share the same label. Additionally, exactly one of the clumps comprises all positive points; the remaining points are all negative. The clump containing



(a) Secret set S of mdc isolated points.

(b) $2^m B$ points from clumps C_j.

(c) Isolated and independent points R.

Figure A.1: An instance of active search where any efficient algorithm can be arbitrarily worse than an optimal policy.

the positive points is chosen uniformly at random; therefore, the prior marginal probability for any point x_c in this category is $f(x_c; \emptyset) = p_c = 2^{-m}$. We denote the clump of all positive points C_*, where $*$ can be regarded as a m-bit integral index $1 \leq * \leq 2^m$. Figure A.1(b) illustrates these points.

"Isolated points." The remaining points share the property that observing any *single* one of them does not change the marginal probabilities of any other point. These points are illustrated in Figures A.1(a) (black dots) and A.1(c). The marginal probabilities for any point x_b in this category is defined to be $f(x_b; \emptyset) = p_b = 1 - \sqrt[c]{1/2}$, where we define $c = \sqrt{m}/2$ for convenience. These points can be further classified into two categories:

- A "secret set," denoted by S; see Figure A.1(a) (black dots). These points encode which of the clumps C_* contains the positive points, using a scheme we describe below. For ease of exposition, we partition the set S into m subsets $S_1, \ldots, S_m$, each of size dc, where we define $d = \sqrt{m}$. Thus $|S| = mdc = {}^{m^2}/_2 = {}^{B}/_2$; the size of this secret set is exactly half the budget.

  The key of this construction is that each subset S_i encodes one bit b_i of information about which clump C_* contains the positive points, using a simple encoding scheme: the binary representation of the positive clump C_*'s index is $* = b_1 b_2 \ldots b_m$. Each bit

is encoded with a two-step mechanism. First, each S_i is partitioned into d groups of c points, each group encoding a "meta" bit of information $b_{ij}, 1 \leq i \leq m, 1 \leq j \leq d$, by a logical OR. These meta-bits, not in the problem instance, are illustrated by the white dots in A.1(a). Finally, the meta-bits associated with S_i encode the bit b_i via a logical XOR,[19] $b_i = b_{i1} \oplus \cdots \oplus b_{id}$.

- Independent points. The remaining $2^{2m} - 2^m B - mdc$ points are totally independent from the others; revealing them conveys no information for any other point. We denote this set of points by R.

Observation 1. *At least d points from S_i need to be observed in order to infer one bit b_i of information about the positive clump.*

A virtual bit b_i is computed by the XOR operation of the d associated meta-bits $b_i = b_{i1} \oplus \cdots \oplus b_{id}$. Notice that each b_{ij} has same marginal probability of being positive, i.e., $\forall i, j, \Pr(b_{ij} = 1) = (1 - p_b)^c = 1/2$. We also have $\forall i, \Pr(b_i = 1) = 1/2$. It is necessary to observe all d meta-bits b_{ij} from the same group S_i to infer the bit b_i, since observing a fraction of the inputs of an XOR does not change the marginal belief about its outcome. So observing $d - 1$ or fewer points from S conveys no information about the positive clump. Equivalently, $\forall x, \Pr(y = 1 \mid x, \mathcal{D}) = \Pr(y = 1 \mid x)$ if $|\mathcal{D} \cap S| \leq d - 1$.

Observation 2. *Observing any number of clump points does not change the marginal probability of any point in the secret set S.*

We need to make sure that no external information can help to identify the secret set S. Notice that the knowledge of b_i does not change the marginal probability of any b_{ij}; hence, no point x in S will have a different probability after observing b_i. This means that observing points outside S does not help distinguish S from the remaining isolated points R.

Now we formally restate Theorem 1 and provide its proof. The theorem implies that a polynomial time algorithm cannot achieve a constant approximation ratio.

Theorem 4 (Formal statement of Theorem 1)**.** *Any (possibly randomized) policy for active search that performs $o\big(n^{\sqrt{\frac{1}{2}\log n}}\big)$ inference calls $f(x, \mathcal{D})$, with $|\mathcal{D}| \leq B$, has approximation*

[19] Sum of the bits modulo 2.

ratio with respect to the expected utility of the optimal policy of $\mathcal{O}\left(\frac{1}{\sqrt{\log n}}\right)$ *where* $|\mathcal{X}| = n$ *is the number of points, and* B *is the budget.*[20]

Proof. Consider a random instance $\mathcal{I} \in \mathcal{H}$ and fix a policy $\mathcal{A}$. Let α be the total number of inference calls performed by $\mathcal{A}$ throughout its execution. At the ith inference call $\Pr(y = 1 \mid x, \mathcal{D}_i)$, $\mathcal{A}$ might use an arbitrary training set $\mathcal{D}_i$ of size at most B. We will show that $\mathcal{A}$ has a very small probability of collecting a large reward on $\mathcal{I}$.

Before analyzing the algorithm $\mathcal{A}$, we present a lower bound on the performance of an optimal policy. Consider the following policy with unlimited computational power: In the first iteration, compute the marginal probability of an arbitrary fixed clump point, conditioning on observing every possible subset of the isolated points of size d with labels all equal to 1. This set of $\mathcal{O}(n^d) = \mathcal{O}(n^{\sqrt{\frac{1}{2}\log n}})$ inference calls will reveal the location of the secret set S: exactly those points will modify the probabilities of the fixed clump point. Now the policy spends the first half of its budget querying the points in S (recall $|S| = B/2$). These outcomes reveal the hidden clump of positives C_*. The policy now spends the second half of the budget querying (collecting) these positive points. The expected performance of this strategy is $B/2 + p_b \; B/2 > B/2$. Certainly this is a lower bound on the optimal performance; hence

$$\textsc{opt} > \frac{B}{2}. \tag{A.1}$$

Now consider the algorithm $\mathcal{A}$ at the ith inference. By Observations 1 and 2, $\mathcal{A}$ cannot differentiate between the points in S and those in R unless $|\mathcal{D}_i \cap S| \geq d$. Suppose that before the ith inference, the algorithm has no information about S. Then the chance of $\mathcal{A}$ choosing a $\mathcal{D}_i$ such that $|\mathcal{D}_i \cap S| \geq d$ is no better than that of a random selection from n' points, where $n' = n - 2^m B$ is the number of isolated points. We can upper bound the probability of $\mathcal{A}$ choosing a dataset $\mathcal{D}_i$ such that $|\mathcal{D}_i \cap S| \geq d$, by counting how many subsets would contain at least d points from S, among all subsets of the n' points of size at most B:

$$\Pr\big(|\mathcal{D}_i \cap S| \geq d\big) \leq \frac{\binom{B/2}{d}\binom{n'-d}{B-d}}{\binom{n'}{B}}. \tag{A.2}$$

[20]Note we used the little-o notation for the number of inference calls, and the big-O notation of the approximation ratio.

We only consider the isolated points because an algorithm $\mathcal{A}$ that only queries the isolated points has a higher probability of hitting the secret set. Also note that technically the denomenator in (A.2) should be $\binom{n'}{B} - i + 1$ since one would not choose the same subsets $\mathcal{D}_j, j < i$ as those before the ith inference. But asymptotically $i \leq \alpha$ (assuming $\alpha = \mathcal{O}(2^n)$ for now) is of much lower order than $\binom{n'}{B}$, therefore $\binom{n'}{B} - \alpha + 1 = \Theta\Big(\binom{n'}{B}\Big)$. Denote p_h as the probability of algorithm $\mathcal{A}$ ever "hiting" the secret set after α inferences; then p_h can be union-bounded:

$$\begin{aligned} p_h &\leq \frac{\alpha\binom{B/2}{d}\binom{n'-d}{B-d}}{\binom{n'}{B}} \\ &< \frac{\alpha\left(\frac{B}{2}\right)^d B^d}{(n')^d} \\ &= \frac{\alpha}{\left(\frac{2n'}{B^2}\right)^d} \end{aligned}$$

Note $B = m^2 = \frac{1}{4}\log^2 n$ and $n' = n - 2^m B = n - \sqrt{n}\frac{1}{4}\log^2 n = \Theta(n)$, so

$$\begin{aligned} \left(\frac{2n'}{B^2}\right)^d &= \Theta\left(\left(\frac{2n}{B^2}\right)^d\right) \\ &= \Theta\left(\left(\frac{n}{\frac{1}{32}\log^4 n}\right)^{\sqrt{\frac{1}{2}\log n}}\right) \\ &= \Theta\left(n^{\sqrt{\frac{1}{2}\log n}}\right). \end{aligned}$$

We have now derived

$$p_h < \frac{\alpha}{\Theta(n^{\sqrt{\frac{1}{2}\log n}})}.$$

So for any $\alpha = \mathcal{O}\big(n^{\sqrt{\frac{1}{2}\log n}-\varepsilon}\big)$, where ε is a positive constant, we have

$$p_h < \mathcal{O}\left(\frac{1}{n^\varepsilon}\right). \tag{A.3}$$

If $\mathcal{A}$ ever hits the secret set S, for simplicity, we will assume that it achieves maximal performance B. If $\mathcal{A}$ never finds the secret set, we can further consider the following two cases. If the algorithm queries points $x \in R$, no marginal probabilities are changed; if a

point $x \in C_j$ is queried, for any clump j, only the marginal probabilities of the clumps are changed. The expected performance in these two cases can be upper bounded by pretending that the algorithm had a larger budget of size $2B$; in which half the budget (i.e., B) is spent on querying points in R, and the other half on querying the clump points. The expected number of targets found after B queries on R is

$$Bp_b = B(1 - \sqrt[c]{1/2}) = B\left(1 - 2^{-\frac{2}{\sqrt{m}}}\right). \tag{A.4}$$

The expected number of targets found after B queries on the clump points is

$$\begin{aligned} &\frac{B}{2^m} + \left(1 - \frac{1}{2^m}\right)\left(\frac{B-1}{2^m - 1} + \left(1 - \frac{1}{2^m - 1}\right)(\cdots)\right) \\ &\qquad = \frac{B(B+1)}{2^{m+1}}. \end{aligned} \tag{A.5}$$

Combining (A.4) and (A.5), we get that the expected performance in the case when $\mathcal{A}$ never hits S can be upper bounded by

$$\frac{B(B+1)}{2^{m+1}} + B\big(1 - 2^{-\frac{2}{\sqrt{m}}}\big).$$

The overall expected performance of $\mathcal{A}$ can be upper bounded by

$$E_{\mathcal{A}} < Bp_h + \frac{B(B+1)}{2^{m+1}} + B\big(1 - 2^{-\frac{2}{\sqrt{m}}}\big),$$

where we have used the trivial upper bound $1 > (1 - p_h)$. Finally, combining with the lower bound of OPT in (A.1), the approximation ratio can be upper bounded by

$$\begin{aligned} \frac{E_{\mathcal{A}}}{\text{OPT}} &< \frac{Bp_h + \frac{B(B+1)}{2^{m+1}} + B\big(1 - 2^{-\frac{2}{\sqrt{m}}}\big)}{B/2} \\ &= 2p_h + \frac{B+1}{2^m} + 2\big(1 - 2^{-\frac{2}{\sqrt{m}}}\big) \\ &= \mathcal{O}\left(\frac{1}{n^{\varepsilon}} + \frac{\log^2 n}{4\sqrt{n}} + 2\Big(1 - 2^{-\frac{2\sqrt{2}}{\sqrt{\log n}}}\Big)\right) \\ &= \mathcal{O}\left(\frac{1}{\sqrt{\log n}}\right). \end{aligned}$$

for any $\alpha = \mathcal{O}\big(n^{\sqrt{\frac{1}{2}\log n}-\varepsilon}\big) = o\big(n^{\sqrt{\frac{1}{2}\log n}}\big)$. Note that it is easy to verify that $2\big(1-2^{-\frac{2\sqrt{2}}{\sqrt{\log n}}}\big) = \Theta\left(\frac{1}{\sqrt{\log n}}\right)$ with L'Hôpital's rule. □

Appendix B

Adaptivity Gap of Batch Budgeted Active Search

Theorem 2 is restated as follows:

There exist active search instances with budget T, such that

$$\frac{\mathrm{OPT}_1}{\mathrm{OPT}_b} = \Omega\left(\frac{b}{\log T}\right),$$

where OPT_x is the expected number of targets found by the optimal batch policy with batch size $x \geq 1$.

Proof. We begin the proof by constructing a class of active search instances $\mathcal{I}$, parameterized by a given budget T and batch size b, illustrated in Figure B.1. In these instances, there are two types of points. Points of the first type are organized in a complete binary tree of height h, which is a parameter we will specify later. There are $2^h - 1$ such points, each with marginal probability p of being positive, where p is a also parameter we will fix later.

The second type of points is "clump points." There are 2^h "clumps" of points attached to the leaves of the binary tree, and each clump has size $T - h$. The labels in each clump are perfectly correlated; that is, either all labels in a clump are positive or all are negative. We construct the problem instance such that exactly one of these clumps contains positive points.

We denote the clumps as $\{C_j\}_{j=1}^{2^h}$, where each clump C_j corresponds to a path $\mathcal{P}_j = D_1 D_2 \cdots D_h$ from the root of the binary tree to the clump, where $D_i \in \{\mathcal{L}, \mathcal{R}\}$ indicates progressing from a parent to its left ($\mathcal{L}$) or right ($\mathcal{R}$) child.

The positive clump can be identified by the following rule: start from the root of the binary tree and progress left if its label is negative and right if its label is positive. Repeat this procedure until a clump is reached. This clump is defined to be the positive clump, and we will refer to the single path leading to the positive clump as the *correct path*.

For example, in Figure B.1, if the labels of the points on the red dashed path are 0, 1, and 0, respectively, then the third clump from the left would be the positive clump, and all others would contain negative points only.

To better understand the correlations among the labels of the tree nodes and the clumps, we define a notion of consistency between a labeling and a path.

Definition 2. *The labels $\{y_i\}_{i=1}^{h}$ of the nodes along a given path $\mathcal{P} = D_1 D_2 \cdots D_h$ corresponding to a clump C are* consistent *with $\mathcal{P}$ if we have $y_i = 0$ when $D_i = \mathcal{L}$ and $y_i = 1$ when $D_i = \mathcal{R}$ for all labels along the path.*

There are 2^h possible labelings for the h tree nodes on each path, but only one of them is consistent with the path. Among all the 2^h paths, the one with consistent labeling would correspond to the positive clump. Therefore, identifying the positive clump exactly specifies all the labels of the nodes on the correct path and also constrains all other clumps to contain negative points only.

However, identifying a clump as negative only implies that the joint probability of the consistent labeling of the nodes on its path is zero; but any other labeling of these points is still possible. The marginal probability of any point on this path does not change given this information unless there is only one point on this path remaining unobserved. The probability of its closest unobserved sibling clumps would also be increased given this information. For example, if the leftmost clump in Figure B.1 is observed as being negative, this does not imply its immediate parent tree node is positive, but it does imply that the probability of its closest unobserved sibling clump would be increased by an appropriate factor. We will characterize this joint probability distribution more in Lemma 5.

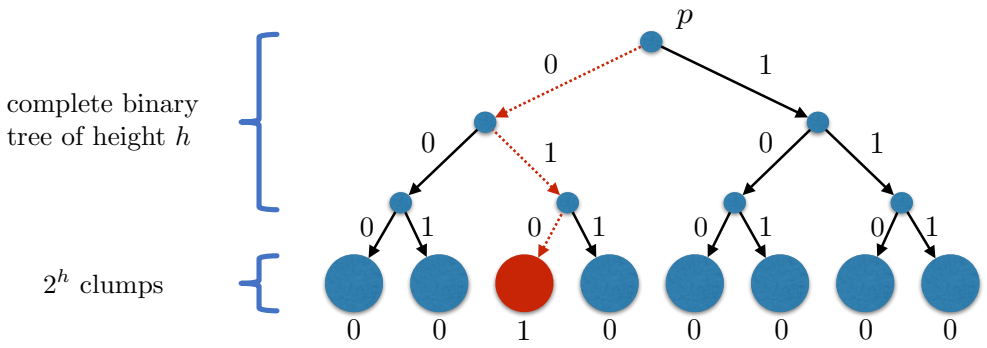


Figure B.1: An illustrative example of the constructed instance with $h = 3$ and $2^h = 8$ clumps, where the third clump from the left is positive, corresponding to the correct path, 010.

Parameter settings. We set the height of the tree $h = T/2$, and hence the clump size is also $T/2$. We set $p = \frac{100 \log T}{b}$, and here we assume b is of higher order than $\log T$.

Lower bound the optimal sequential policy.

Lemma 1. $\text{OPT}_1 > T/2$.

Proof. Consider the following sequential policy. Query the tree along a path from the top down, selecting each point based on the label of the parent to maintain consistency. This policies identifies the positive clump in $h = T/2$ steps. The remaining budget, $T/2$, can then be spent querying the points in the positive clump. The expected performance of this policy certainly lower bounds that of the optimal one. Hence

$$\text{OPT}[\text{SAS}] \geq T/2 \cdot p + T/2 > T/2. \tag{B.1}$$

□

Upper bound the optimal batch policy.

Lemma 2. $\text{OPT}_b = \mathcal{O}\left(\frac{T \log T}{b}\right)$.

First we claim that an optimal batch policy should never alternately query tree points and clump points before the positive clump is identified. Let the trace of an optimal batch policy

be $B_1, B_2, \ldots, B_t$, where $t = T/b$, B_i is the ith batch. Assume the positive clump is identified at the kth iteration; if it is never identified, let $k = t + 1$. We claim that $B_1, \ldots, B_{k-1}$ should never alternate between tree points and clump points. That is, if B_j is the first batch that contains clump points, then $B_{j+1}, \ldots, B_{k-1}$ would only contain clump points; otherwise, say $B_{j'}(j < j' \leq k-1)$ contains tree points, then it's only better to move the tree points in $B_{j'}$ to the position of B_j. This is because the immediate utility of querying tree points is always[21] the same before the positive clump is identified, no matter earlier or later; but earlier can only result in higher expected future utility for clump points. Once the positive clump is identified ($k \leq t$), then after B_k, we know all other clumps are negative, and we also know the labels of all the nodes on path corresponding to the positive clump, and all remaining tree points become totally independent. Hence the optimal batches $B_{k+1}, B_{k+2}, \ldots, B_t$ would be just collecting all the known positives; if there is still budget remaining after that, query the tree points randomly (no difference).

So there are only two possible cases for the optimal policy: (1) query tree points first, then clumps points; if the positive clump is identified, possibly query more tree points; (2) query clumps points only; if the positive clump is identified, possibly query some tree points.

Based on this observation, we upper bound the expected utility of the optimal batch policy by allowing budget $2T$, split by sub-policies $\mathcal{P}$ and $\mathcal{Q}$, where $\mathcal{P}$ only queries tree points with budget T, and $\mathcal{Q}$ only clump points also with budget T; whenever the positive clump is identified, we automatically grant utility T.

Let u_1 be the utility from tree nodes, u_2 be that from the clumps nodes. Let the expected utility of $\mathcal{P}$ be $\mathbb{E}_{\mathcal{P}}[u_1]$, and the expected utility of $\mathcal{Q}$ be $\mathbb{E}_{\mathcal{Q}}[u_2]$. We have

$$\textsc{opt}[\textsc{bas}] \leq \max_{\mathcal{P},\mathcal{Q}} \big\{ \mathbb{E}_{\mathcal{P}}[u_1] + \mathbb{E}_{\mathcal{P}}\big[\mathbb{E}_{\mathcal{Q}}[u_2]\big], \mathbb{E}_{\mathcal{Q}}[u_2] + \mathbb{E}_{\mathcal{Q}}\big[\mathbb{E}_{\mathcal{P}}[u_1]\big] \big\}. \tag{B.2}$$

We will prove Lemma 2 by upper bounding both cases. First we upper bound the easier case, where one performs $\mathcal{Q}$ first and then $\mathcal{P}$, then the harder case, first $\mathcal{P}$ then $\mathcal{Q}$.

When one performs $\mathcal{Q}$ first and then $\mathcal{P}$, we have the following:

[21] Except when the tree point is the only unobserved one along an otherwise consistent path corresponding to a clump known to be negative; this happens with extremely small probability as we will also see in Lemma 6.

Lemma 3. *For any $\mathcal{P}$ and $\mathcal{Q}$, $\mathbb{E}_{\mathcal{Q}}[u_2] + \mathbb{E}_{\mathcal{Q}}\big[\mathbb{E}_{\mathcal{P}}[u_1]\big] = \mathcal{O}\left(\frac{T \log T}{b}\right)$.*

To prove this lemma, we upper bound the two parts separately.

Lemma 4. *For any $\mathcal{Q}$, $\mathbb{E}_{\mathcal{Q}}[u_2] = o(1)$.*

To prove this bound, we prove the following more general lemma, which will be also used later.

Lemma 5. *Given a tree of height h', a policy $\mathcal{Q}$ with budget T that only queries clump points has expected utility upper bounded by $U \equiv \frac{1}{2}T(T+1)(1-p)^{h'-\log(T+1)-1}$. Furthermore, the probability of identifying the positive clump is upper bounded by $\mathcal{O}(\frac{1}{T})$ if $h' = h$.*

Proof. For a tree of height h', the probabilities of the clumps are non-increasing from left to right: $(1-p)^{h'} \equiv p_{\max}, (1-p)^{h'-1}p, (1-p)^{h'-1}p, (1-p)^{h'-2}p^2, \ldots, (1-p)p^{h'-1}, p^{h'}$.

An optimal policy $\mathcal{Q}$ (possibly $b = 1$) only querying clump points goes like this: select points from some clumps; if any one turns out to be positive, then spend the remaining budget only querying that clump and the positives on the corresponding path (as mentioned in the beginning, identifying the positive clump also reveals the labels of all points on its corresponding path); if all selected points turn out to be negative, continue to try other clumps in the same way.

The expected utility of this process can be upper bounded via the following strategy. We first compute an upper bound p^* of the posterior probability of any clump conditioned on T negative observations, then compute the expected utility of this process as if all clumps were independent and of probabilities p^*.

We first derive the maximum possible probability p^* after any T negative observations. For clumps under a subtree of height k, the maximum increase of probability is at most by a factor of $1/(1-p)^k$ by eliminating all $2^k - 1$ clumps except the leftmost one. In terms of increasing the maximum posterior probability by the most amount, the best strategy is to focus on a minimum subtree covering at least $T+1$ clumps, and spend all the budget T eliminating all other clumps $(2^k - 1)$ except the leftmost one. Let k be the minimum integer such that $T \leq 2^k - 1$, we get $k = \lceil \log(T+1) \rceil \leq \log(T+1) + 1$. So observing any T clumps (if no hit) can increase

the probability of any other clump to at most $(1-p)^{h'}/(1-p)^k \leq (1-p)^{h'-\log(T+1)-1} \equiv p^*$. So the expected utility of any "clumps-only" policy can be upper bounded by (for any $b \geq 1$)

$$\begin{aligned}
&p^*T + p^*(T-b) + p^*(T-2b) + \cdots + p^*b \\
&\leq \frac{1}{2}T(T+1)p^* \\
&\leq \frac{1}{2}T(T+1)(1-p)^{h'-\log(T+1)-1} \equiv U.
\end{aligned}$$

The probability of identifying the positive clump $\Pr(hit)$ can also be upper bounded using p^*:

$$\Pr(\text{hit}) < 1 - (1-p^*)^T. \tag{B.3}$$

One can show that

$$\lim_{T\to\infty} \frac{1-(1-p^*)^T}{1/T} = 0. \tag{B.4}$$

So $\Pr(\text{hit}) = o(1/T)$.

□

Proof of Lemma 4. Using Lemma 5, we can bound $\mathbb{E}_{\mathcal{Q}}[u_2] \leq U$ with $h' = h$, and it's easy to derive U is $o(1)$ in this case. □

Lemma 6. *For any $\mathcal{P}, \mathcal{Q}$, $\mathbb{E}_{\mathcal{Q}}\big[\mathbb{E}_{\mathcal{P}}[u_1]\big] = \mathcal{O}\left(\frac{T\log T}{b}\right)$.*

Proof. Consider two cases after $\mathcal{Q}$ finishes:

- First, $\mathcal{Q}$ identified the positive clump, with probability at most $\mathcal{O}(1/T)$ by Lemma 5. In this case the correct path would also be identified, and there are at most h positives on the path. We can simply upper bound the utility by T. So $\mathbb{E}_{\mathcal{P}}[u_1] \leq \frac{1}{T} \cdot T = 1$;
- Second, $\mathcal{Q}$ did not identify the positive clump, in which case each tree point almost always has expected utility p. A subtle situation is when all but one point are queried on a path corresponding to a known negative clump, in which case the label of this point would be known already. So it's possible to get utility 1 instead of p when

querying a tree point. However, this happens with extremely small probability (at most $(1-p)^{T/2-1}$), since all but one point on the path (length $T/2$) must have consistent labels with the path. So we can simply ignore the expected utility of this case. Hence $\mathbb{E}_{\mathcal{P}}[u_1] \leq Tp = \mathcal{O}\left(\frac{T\log T}{b}\right)$.

If we sum these two cases, we have $\mathbb{E}_{\mathcal{Q}}\left[\mathbb{E}_{\mathcal{P}}[u_1]\right] = \mathcal{O}\left(\frac{T\log T}{b}\right)$.

□

Proof of Lemma 3. By Lemma 4 and 6, we have $\mathbb{E}_{\mathcal{Q}}[u_2] + \mathbb{E}_{\mathcal{Q}}\left[\mathbb{E}_{\mathcal{P}}[u_1]\right] = \mathcal{O}\left(\frac{T\log T}{b}\right)$. □

Now consider the first case where one performs $\mathcal{P}$ first and then $\mathcal{Q}$, we also have

Lemma 7. *For any $\mathcal{P}, \mathcal{Q}$, $\mathbb{E}_{\mathcal{P}}[u_1] + \mathbb{E}_{\mathcal{P}}\left[\mathbb{E}_{\mathcal{Q}}[u_2]\right] = \mathcal{O}\left(\frac{T\log T}{b}\right)$.*

We also upper bound the two parts separately.

Lemma 8. *For any $\mathcal{P}$, $\mathbb{E}_{\mathcal{P}}[u_1] = \mathcal{O}\left(\frac{T\log T}{b}\right)$.*

Proof. This is trivial to show due to independence of the nodes: $\mathbb{E}_{\mathcal{P}}[u_1] \leq Tp = T \cdot \frac{100\log T}{b} = \mathcal{O}\left(\frac{T\log T}{b}\right)$. □

Lemma 9. *For any $\mathcal{P}, \mathcal{Q}$, $\mathbb{E}_{\mathcal{P}}\left[\mathbb{E}_{\mathcal{Q}}[u_2]\right] = \mathcal{O}(1)$.*

Note a key property of the constructed instance is that one has to wait until the previouly chosen points are observed to determine which direction to go. However, in BAS, a batch of points has to be observed simultaneously, hence in each batch, various number of observations could be wasted depending on the how many points turn out to be on the correct path. This is exactly why there could be a gap between the performances of SAS and BAS. In the following, we will upper bound the expected performance in this case by bounding the probability of reaching a certain level on the correct path after all T/b batch queries. We first introduce some useful definitions:

Definition 3. *For any observed node in the tree, if it is positive, then we call its right subtree* relevant*, and its left subtree* irrelavant*; or the other way around if it is negative.*

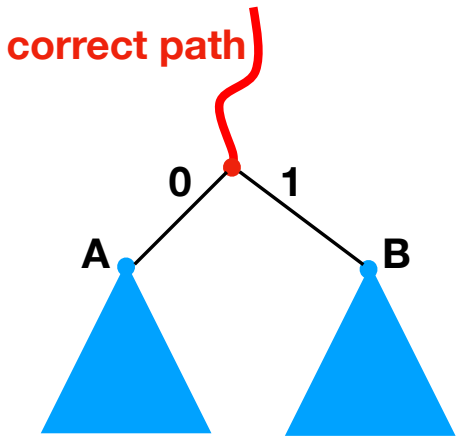


Figure B.2: Illustration of the relevant subtree. The red curve shows the correct path identified so far. If the last node on the correct path is negative, then the node A must also be on the correct path, and the subtree rooted at A is the relevant subtree; otherwise B is on the correct path, and the subtree rooted at B is the relevant subtree.

Definition 4. *The* relevant subtree *at any step is acquired by trimming all observed nodes and their irrelevant subtrees. That is, removing any observed node and its irrelevant subtree, and reconnecting its parent to its relevant subtree.*

Let $P^* = \arg\max_{\mathcal{P}} E_{\mathcal{P}}\left[\max \mathbb{E}_{\mathcal{Q}}[u_2]\right]$ be the optimal sub-policy querying the tree. It's easy to see that P^* should always only query points in the relevant tree, since querying anywhere else has same u_1 but reveals no information about the identity of the positive clump.

Definition 5. *A sequence S of length ℓ is a set of ℓ points $x_1, \ldots, x_\ell$ in a relevant subtree such that x_i is an ancestor of x_{i+1} for all $i = 1, \ldots, \ell - 1$.*

More intuitively, a set of points is a sequence if they are contained in some path.

Proof of Lemma 9. Recall $p = \frac{100 \log T}{b} \to 0$ as $T \to \infty$ since we assumed b is of higher order than $\log T$, so p is close to 0 (hence $p \ll 1/2$) for large enough T.

For any sequence S of length ℓ, the probability of S lying on the correct path is upper bounded by

$$(1-p)^\ell = \left[\left(1-\frac{1}{1/p}\right)^{-1/p}\right]^{-p\ell} \approx \exp(-p\ell) = \exp\left(-\frac{100\log T}{b}\frac{b}{10}\right) = \frac{1}{T^{10}}.$$

Given any batch of size b, there are at most b sequences of length ℓ. So the probability that this batch contains ℓ nodes on the correct path can be union-bounded by

$$\frac{1}{T^{10}}b \le \frac{1}{T^{10}}T = \frac{1}{T^9}.$$

The probability of any T/b batches contains $\frac{T}{b}\cdot\frac{b}{10} = \frac{T}{10}$ nodes on the correct path can be union-bounded by

$$\frac{1}{T^9}\cdot\frac{T}{b} = \frac{1}{T^8 b} \le \frac{1}{T^8}. \tag{B.5}$$

For any policy (e.g., $\mathcal{P}^*$) iteratively choosing a set A of T points with batch size b, two cases can happen:

- case 1: A contains $T/10$ or more points on the correct path. We have shown the probability of this happening is at most $1/T^8$. In this case we simply upper bound $\mathbb{E}_{\mathcal{Q}}[u_2]$ by T.
- case 2: A contains less than $T/10$ points on the correct path. Then the remaining unidentified points on the correct path is at least $h' = T/2 - T/10 = 2T/5$. That is, the shallowest leaf in the relevant subtree has height at least h'. The distribution of the remaining clumps is dominated by that of a brand new active search instance $\mathcal{A}$ with the tree height h', in the sense that the optimal expected utility u_2' of $\mathcal{Q}$ on $\mathcal{A}$ can only be greater, that is, $\mathbb{E}_{\mathcal{Q}}[u_2] \le \mathbb{E}_{\mathcal{Q}}[u_2']$. Applying Lemma 5 with $h' = 2T/5$, we can find $\mathbb{E}_{\mathcal{Q}}[u_2'] = o(1)$.

Therefore, there exist a constant c and large enough T, such that

$$\begin{aligned}\mathbb{E}_{\mathcal{P}}\left[\mathbb{E}_{\mathcal{Q}}[u_2]\right] \leq & \Pr(\text{case 1}) \cdot T + \Pr(\text{case 2}) \cdot c \\ \leq & \frac{1}{T^8} \cdot T + 1 \cdot c,\end{aligned}$$

which is $\mathcal{O}(1)$. □

Proof of Lemma 7. Combining Lemma 8 and 9, we have $\mathbb{E}_{\mathcal{P}}[u_1] + \mathbb{E}_{\mathcal{P}}\left[\mathbb{E}_{\mathcal{Q}}[u_2]\right] = \mathcal{O}\left(\frac{T \log T}{b}\right)$. □

Proof of Lemma 2. By (B.2) and Lemma 3 and 7, we have

$$\text{OPT}_b = \mathcal{O}\left(\frac{T \log T}{b}\right). \tag{B.6}$$

□

Combining Lemma 1 and Lemma 2, we have

$$\frac{\text{OPT}_1}{\text{OPT}_b} = \Omega\left(\frac{b}{\log T}\right).$$

□

Appendix C

Hardness of Cost Effective Active Search

We restate the hardness theorem for cost effective active search as follows:

Theorem 5. *Any algorithm $\mathcal{A}$ with computational complexity $o\left(n^{n^\epsilon}\right)$ has an approximation ratio $\Omega(n^\epsilon)$, for $\epsilon = 0.16$; that is,*

$$\frac{\mathbb{E}[\text{cost}_{\mathcal{A}}]}{\text{OPT}} = \Omega\left(n^\epsilon\right), \tag{C.1}$$

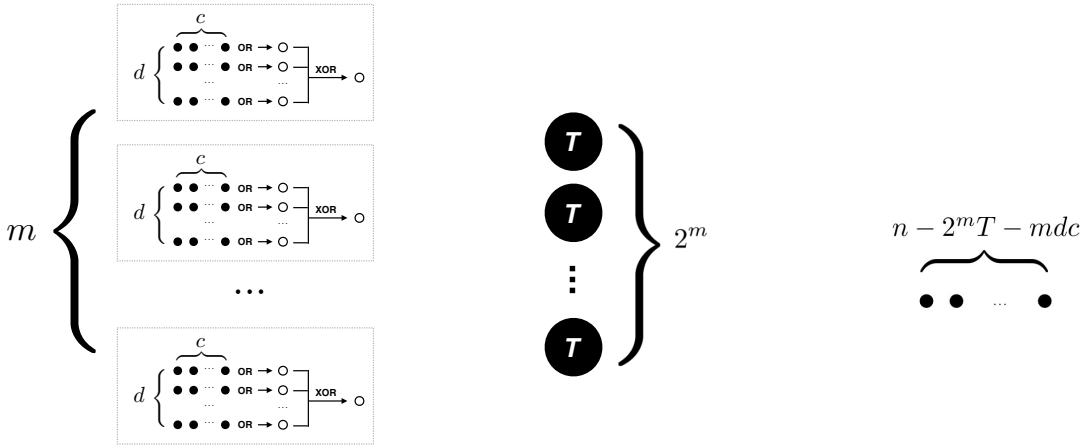
where $\mathbb{E}[\text{cost}_{\mathcal{A}}]$ is the average cost of $\mathcal{A}$, and OPT *is that of the optimal policy.*

Proof. We begin our proof by constructing a very similar class of instances $\mathcal{H}$ as in the proof for budgeted setting (see Appendix A), with different parameter settings. We reproduce the instance illustration here in Figure C.1, and briefly summarize the essences. Each instance in the class has n points with two types: "clumps" and "isolated points".

"Clumps." As shown in Figure C.1b, there are $n^{4\epsilon}$ clumps, and each one has $T = n^{2\epsilon}$ points with the same labels, where ϵ is a small constant such as 0.1. There is exactly one positive clump. So *a priori* the marginal probability of each clump point being positive is $p_c = 1/n^{4\epsilon}$.

"Isolated points." The remaining $n - n^{6\epsilon} = \Theta(n)$ points are isolated points, they are all independent to each other; these points are further categorized into two classes: a "secret set" and totally independent points.

- The secret set, denoted by $\mathcal{S}$ (Figure C.1a), encodes the location of the positive clump in the following way: $\mathcal{S}$ contains $m = \log_2(n^{4\epsilon}) = 4\epsilon \log n$ groups $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$, each of



(a) Secret set $\mathcal{S}$ of mdc isolated points.

(b) $2^m T$ points from clumps.

(c) Isolated and independent points $\mathcal{R}$.

Figure C.1: An instance of active search where any efficient algorithm can be arbitrarily worse than an optimal policy.

size $n^{2\epsilon}$; each $\mathcal{S}_i$ are further partitioned into $d = n^\epsilon/(1-5\epsilon)$ groups [22], with each group having $c = n^\epsilon$ points. Each of the jth ($j = 1, \ldots, d$ group in $\mathcal{S}_i, i = 1, \ldots, m$ encodes one virtual bit b_{ij} by taking the OR operation on the c labels (i.e. $b_{ij} = 1$ iff at least one of the c points are positive); then the d bits $b_{i1}, \ldots, b_{id}$ encode a virtual bit b_i using XOR, i.e., $b_i = b_{i1} \oplus \cdots \oplus b_{id}$. We set probability of each point in $\mathcal{S}$ as $p_s = 1 - \frac{1}{2^{1/c}}$ so that $\Pr(b_{ij} = 1) = (1 - p_b)^c = \frac{1}{2}$, which also leads to $\Pr(b_i = 1) = \frac{1}{2}$. Finally the binary string $b_1 b_2 \cdots b_m$ determines the index of the positive clump.

- The remaining $n - T2^m - mdc$ points, denoted as set $\mathcal{R}$, are totally independent to each other and any other points. The probability of any point in $\mathcal{R}$ is also p_s.

The goal is to find T points with minimum labeling cost.

The two observations given in the proof for budgeted setting still hold, as restated as following.

Observation 3. *At least d points from $\mathcal{S}_i$ (for any i) need to be observed in order to infer one bit b_i of information about the positive clump.*

[22] here dividing by $(1-5\epsilon)$ is not essential; only for the purpose of getting a simpler formula in our theorem.

Observation 4. *Observing any number of clump points does not change the marginal probability of any point in the secret set $\mathcal{S}$.*

Consider a random instance $\mathcal{I} \in \mathcal{H}$. We assume any policy has access to the correct marginal probability $\Pr(y \mid x, \mathcal{D})$ where $\mathcal{D}$ may contain current observations and/or some "lookahead" points, and we limit the lookahead amount to be d since an optimal policy operates under such condition.

Upper bound of an optimal policy. With unlimited computational power, a policy can first compute the marginal probability of an arbitrary fixed clump point, conditioning on every possible subset of the isolated points of size d with labels all equal to 1. This set of $\mathcal{O}(n^d)$ inference calls will reveal the location of the secret set, since exactly those points that could change the marginal probability of any fixed clump point are the secret points. Then the policy could query the identified positive clump and is guaranteed to achieve the target T in time T. So the total cost of an optimal policy is upper bounded by $|\mathcal{S}| + T$, i.e.,

$$\text{OPT} < mdc + T = \frac{4\epsilon}{1 - 5\epsilon} n^{2\epsilon} \log n + n^{2\epsilon} = \mathcal{O}(n^{2\epsilon}). \tag{C.2}$$

In our asymptotic notations, all log factors will be omitted.

Lower bound of any policy with limited computational power. Fix a policy $\mathcal{A}$. Our goal is to show that with $o\left(n^{n^\epsilon}\right)$ inference calls, the expected cost of $\mathcal{A}$ is lower bounded by $\Omega\left(n^{3\epsilon}\right)$. The key is to bound the probability of $\mathcal{A}$ revealing the secret set throughout its execution. By Observation 3 and 4, $\mathcal{A}$ can make inference calls $\Pr(y \mid x, \mathcal{D})$ to distinguish points in $\mathcal{S}$ from those in $\mathcal{R}$ only when $|\mathcal{D} \cap S| \geq d$. Suppose that before the ith inference call, $\mathcal{A}$ has no information about $\mathcal{S}$. Then the chance of $\mathcal{A}$ choosing a set $\mathcal{D}$ such that $|\mathcal{D} \cap S| \geq d$ is no better than that of a random selection from $n' = n - 2^m T = n - n^{6\epsilon} = \Theta(n)$ points. Since our goal is to prove a lower bound of the cost in the order of $\Omega(n^{3\epsilon})$, we allow $\mathcal{A}$ to make inference calls $\Pr(y \mid x, \mathcal{D})$ with $|\mathcal{D}| \leq n^{3\epsilon}$. Note this is much larger than the lookahead limit d. We can upper bound the probability of $\mathcal{A}$ choosing a dataset $\mathcal{D}$ such that $|\mathcal{D} \cap S| \geq d$, by counting how many subsets would contain at least d points from $\mathcal{S}$, among all subsets of

the n' points of size at most $\beta = n^{3\epsilon}$:

$$\Pr\left(|\mathcal{D} \cap S| \geq d\right) \leq \frac{\binom{mdc}{d}\binom{n'-d}{\beta-d}}{\binom{n'}{\beta}} \tag{C.3}$$

$$< \left(\frac{\beta mdc}{n'}\right)^d \tag{C.4}$$

$$= \mathcal{O}\left(\frac{1}{n^{n^\epsilon}}\right) \tag{C.5}$$

With α such inference calls, the probability p_h of at least one of them hitting the secret set can be union bounded by

$$p_h < \mathcal{O}\left(\frac{\alpha}{n^{n^\epsilon}}\right). \tag{C.6}$$

Hence for any $\alpha \leq n^{n^\epsilon - \delta}$ (for any positive constant δ),

$$p_h < \mathcal{O}\left(\frac{1}{n^\delta}\right). \tag{C.7}$$

If $\mathcal{A}$ ever hits the secret set $\mathcal{S}$, we simply assume it will find all T positives with zero cost. If not, then $\mathcal{A}$ can do no better than random selection, and its expected cost is T/p_s (note $p_s > p_c$, and querying $n^{3\epsilon}$ clump points would not make the remaining clump points' probabilities higher than p_s). So the overall expected cost is lower bounded by $p_h \cdot 0 + (1 - p_h) \cdot \frac{T}{p_s} = \Theta(n^{3\epsilon})$. Here we used the fact that $p_s = 1 - \frac{1}{2^{1/c}} = \Theta(\frac{1}{c})$, which is easy to verify by L'Hôpital's rule.

Therefore,

$$\frac{\mathbb{E}[\text{cost}_{\mathcal{A}}]}{\text{OPT}} > \frac{\Omega(n^{3\epsilon})}{\mathcal{O}(n^{2\epsilon})} = \Omega\left(n^\epsilon\right). \tag{C.8}$$

That is, any policy $\mathcal{A}$ for cost effective active search with $\alpha = o(n^{n^\epsilon})$ inference calls would have expected cost at least $\Omega(n^\epsilon)$ times more than the optimal cost. The proof holds for ϵ such that $n' = n - n^{6\epsilon} = \Theta(n)$ indicating $6\epsilon < 1$, and for (C.5) to hold, we have $5\epsilon < 1$. We can set $\epsilon = 0.16$, which is less than $1/6$.

□

Remark 2. *The parameters of the constructed instance are set to satisfy the following constraints:*

- *Make* OPT *linear in* T*, which means* $T = \Omega(|S|)$*.*
- *Our goal is to prove an* $\Omega(n^\epsilon)$ *bound, so the probability of the secret points* $p_s = 1 - \frac{1}{2^{1/c}} = \Theta(1/c)$ *should be* $O(\frac{1}{n^\epsilon})$*. This is because* OPT *is set to be linear in* T *and the cost upper bound of a uniform random policy* T/p_s *should be at least* $\Omega(n^\epsilon) \cdot T$*.*
- *Make the probability of secret points larger than that of the clump points, i.e.,* $p_s > p_c$*, otherwise the cost upper bound would be* T/p_c*. So we have* $\frac{1}{n^\epsilon} > \frac{1}{2^m}$*. That makes* $m > \epsilon \log n$*.*
- *The number of clump points should be less than the total number of points. That is,* $n > T \cdot 2^m$*, which leads to* $m < (1 - 2\epsilon) \log n$*.*
- d *controls the scale of the computational complexity bound and lookahead limit, the larger the tighter would be the bound.*

Appendix D

Additional Results

D.1 Cost Effective Active Search

In Chapter 5 experiments, we only showed the results for ENS and CEAS with their best parameters. Table D.1 and D.2 show the full set of results for all tested policies. Note we did not test CEAS-50 and CEAS-0.5 for drug data since we expect them to be worse, according to Table D.2.

Figure D.1 show the cost curves for each individual drug discovery datasets. The average of the nine plots is shown in the main text.

D.2 Bayesian Optimization

In the main text, we presented BO results for nine synthetic functions. These nine functions are selected from the 31 functions shown in Table D.3, with GAP of EI less than 0.9. We only run up to 10.EI for all functions, so 12.EI.s and 15.EI.s are not shown. We argue that by identifying this set of "hard" functions, we are able to consistently see the advantage of nonmyopic BO methods. In Table D.3, we can see all variants of our method perform better than EI on average, but other interesting patterns are weak, possibly because they are averaged out by the "easy" functions.

Table D.1: Results for all tested policies for the materials discovery dataset.

	50	100	200	300	400	500	1000	1500	average
GREEDY	84.5	175.0	347.7	522.5	721.8	924.1	2025.9	2981.7	972.9
TWO-STEP	86.0	179.1	349.0	533.2	735.0	938.1	1973.1	3019.4	976.6
ENS-10	*81.7*	167.8	339.2	*520.4*	721.9	939.8	1896.1	2836.5	937.9
ENS-30	*78.6*	164.0	*335.8*	515.2	724.7	927.4	1795.8	*2799.3*	917.6
ENS-50	**78.0**	162.5	*329.6*	*517.0*	729.6	926.6	1793.0	2812.2	918.6
ENS-70	*81.5*	165.1	*337.7*	524.6	720.0	910.0	1790.6	*2757.0*	910.8
ENS-0.1	84.2	177.2	343.2	520.6	717.7	946.3	1804.7	*2765.1*	919.9
ENS-0.3	*83.6*	171.1	340.2	518.2	*689.5*	917.1	1815.9	*2739.4*	909.4
ENS-0.5	82.3	162.7	*335.4*	535.3	*693.2*	*897.9*	1812.8	*2736.4*	907.0
ENS-0.7	*80.2*	167.8	**328.4**	*509.7*	708.6	*887.4*	1798.5	*2773.6*	906.8
ENCES-10	87.3	173.8	345.0	518.3	719.0	930.0	1825.5	2886.7	935.7
ENCES-20	88.8	167.5	*335.4*	518.5	715.1	925.0	1779.7	*2761.6*	911.4
ENCES-30	88.4	164.1	*332.7*	*511.5*	703.7	*898.4*	**1734.7**	*2768.2*	*900.2*
ENCES-40	*79.3*	**154.6**	*330.3*	523.8	713.2	910.0	*1748.8*	*2757.6*	*902.2*
ENCES-50	85.2	*156.4*	*330.8*	*518.4*	*701.4*	*885.7*	*1745.4*	*2753.2*	*897.1*
ENCES-0.1	86.2	173.9	341.5	517.0	716.1	945.5	1844.1	*2775.3*	925.0
ENCES-0.2	84.5	167.6	*334.9*	**506.7**	720.9	919.5	1845.1	2805.7	923.1
ENCES-0.3	85.5	170.3	*333.6*	*524.2*	709.1	*884.5*	1823.6	*2767.0*	912.2
ENCES-0.4	83.4	*158.6*	*331.2*	532.0	704.5	**881.4**	1797.0	2808.3	912.0
ENCES-0.5	87.1	164.3	*336.9*	*513.7*	**683.9**	*891.9*	*1774.1*	**2724.0**	**897.0**

Table D.4 shows the average results of 50 repeats of EI and both "sampling" and "best" variants of q.EI on the real world functions. Different from the results on synthetic functions, we do not see "sampling" being consistently better than "best" or the other way around.

Table D.2: Results for all tested policies for the nine drug discovery datasets.

	50	100	150	200	average
GREEDY	215.7	414.4	503.2	587.4	430.2
TWO-STEP	71.7	156.0	243.2	322.4	198.4
ENS-10	*59.3*	133.0	211.9	291.4	173.9
ENS-30	*58.8*	134.9	208.3	283.3	171.3
ENS-50	*58.6*	137.3	205.1	286.3	171.8
ENS-70	80.2	197.4	207.3	288.8	193.5
ENS-0.1	61.3	142.2	219.1	297.5	180.1
ENS-0.3	59.5	133.0	215.3	292.8	175.2
ENS-0.5	*59.2*	132.6	215.0	287.9	173.7
ENS-0.7	*59.1*	132.8	212.0	284.2	172.0
ENCES-10	*57.3*	119.5	196.2	273.6	161.7
ENCES-20	**56.3**	**112.7**	**184.5**	**255.1**	**152.2**
ENCES-30	75.1	*128.4*	*196.5*	*261.5*	165.4
ENCES-40	102.4	165.9	221.5	*282.3*	193.0
ENCES-0.1	*58.2*	197.3	195.0	271.4	180.5
ENCES-0.2	*72.9*	116.0	194.8	298.9	170.7
ENCES-0.3	*57.1*	121.5	267.6	318.9	191.3
ENCES-0.4	56.8	*134.4*	275.5	319.8	196.6

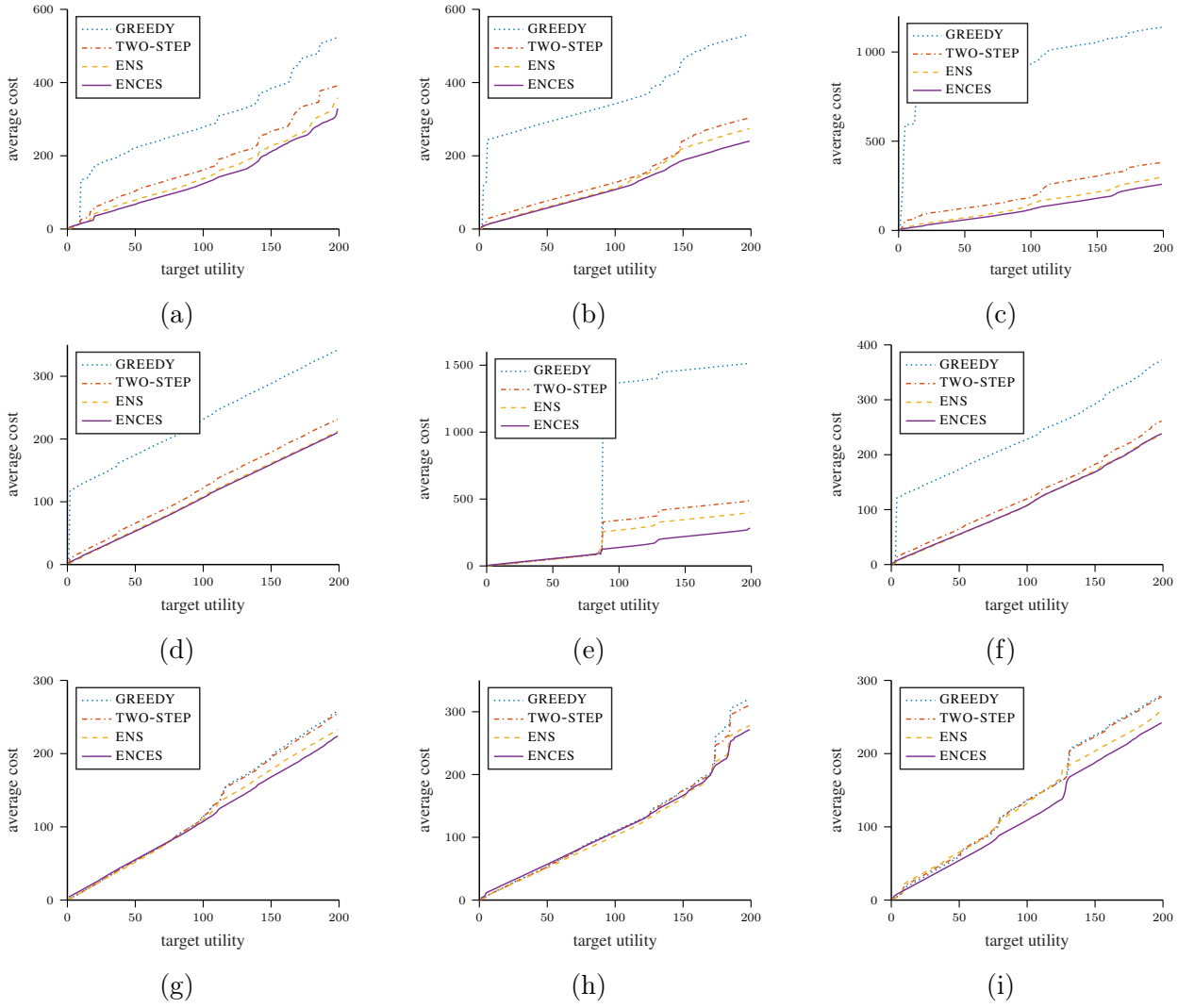


Figure D.1: Average cost versus the number of positives found for 9 drug discovery datasets. The total number of positives are 553, 378, 506, 1023, 218, 916, 1024, 431, 255, respectively.

Table D.3: Average GAP of 30 repeats on all 31 synthetic functions.

	EI	2.EI.b	2.EI.s	3.EI.b	3.EI.s	4.EI.b	4.EI.s	5.EI.b	5.EI.s	10.EI.b	10.EI.s
branin	*1.000*	1.000	0.999	1.000	0.999	*1.000*	1.000	*1.000*	1.000	**1.000**	0.999
rosenbrock2	*0.989*	*0.978*	*0.985*	*0.990*	*0.981*	0.971	*0.979*	*0.969*	**0.996**	*0.981*	0.973
rosenbrock4	*0.989*	*0.989*	0.988	0.990	*0.990*	*0.991*	*0.990*	**0.992**	0.988	*0.991*	*0.989*
rosenbrock6	*0.989*	0.989	*0.990*	**0.992**	*0.990*	*0.990*	*0.990*	*0.991*	*0.990*	*0.991*	0.985
hartmann3	1.000	*1.000*	*1.000*	1.000	*1.000*	**1.000**	*1.000*	*1.000*	*1.000*	*0.999*	*1.000*
hartmann6	0.957	0.966	0.964	0.970	0.965	0.974	0.970	*0.976*	0.974	**0.978**	0.971
eggholder	*0.605*	*0.606*	*0.589*	*0.603*	*0.612*	*0.649*	*0.638*	0.554	*0.620*	*0.600*	**0.651**
dropwave	0.455	0.489	0.524	0.475	*0.599*	*0.538*	0.550	0.435	*0.613*	0.448	**0.651**
beale	*0.920*	*0.903*	*0.910*	**0.935**	*0.915*	*0.927*	0.874	*0.901*	0.902	*0.912*	0.900
shubert	0.323	0.299	*0.440*	*0.387*	**0.551**	0.382	*0.500*	*0.464*	0.371	0.285	*0.458*
6humpcamel	*0.996*	0.994	0.992	*0.994*	0.991	**0.997**	0.990	*0.995*	0.988	0.990	0.992
holder	*0.936*	0.873	*0.913*	*0.941*	*0.930*	**0.965**	*0.949*	0.950	*0.948*	*0.883*	*0.936*
3humpcamel	**0.988**	*0.981*	*0.978*	*0.970*	0.978	*0.981*	0.949	0.975	0.931	0.971	0.930
rastrigin2	**0.917**	*0.903*	0.882	*0.884*	*0.891*	*0.899*	*0.884*	*0.877*	*0.910*	0.847	0.836
rastrigin4	*0.806*	0.759	0.773	*0.830*	**0.838**	*0.834*	*0.815*	0.769	*0.800*	0.766	0.775
ackley2	0.850	0.772	0.838	0.802	**0.918**	0.832	*0.869*	*0.774*	0.783	0.811	*0.896*
ackley5	0.528	0.557	0.555	0.579	0.562	0.602	0.594	0.604	*0.620*	**0.671**	0.621
levy2	0.925	*0.949*	*0.927*	*0.933*	0.915	0.960	0.961	*0.958*	0.913	**0.963**	0.929
levy3	*0.960*	0.948	*0.962*	*0.954*	*0.962*	*0.951*	0.961	*0.960*	*0.968*	**0.969**	0.951
levy4	*0.968*	*0.959*	*0.970*	*0.970*	*0.974*	*0.962*	*0.950*	*0.976*	**0.976**	*0.970*	*0.972*
griewank2	*0.960*	*0.963*	0.952	*0.958*	**0.966**	*0.954*	0.955	*0.962*	*0.958*	*0.961*	*0.960*
griewank5	0.981	0.984	0.983	*0.985*	*0.984*	*0.985*	0.983	**0.986**	*0.984*	*0.985*	0.983
stybtang2	0.999	0.970	*0.999*	*1.000*	0.999	0.999	0.999	0.999	0.992	**1.000**	0.999
stybtang4	**0.937**	*0.911*	0.897	*0.916*	0.884	*0.915*	0.901	0.900	0.908	0.893	0.883
powell4	*0.976*	*0.965*	*0.973*	*0.975*	*0.972*	*0.977*	0.965	**0.978**	*0.971*	*0.966*	0.957
dixonprice2	*0.988*	*0.985*	**0.990**	*0.989*	*0.963*	*0.967*	*0.953*	*0.959*	*0.945*	*0.982*	0.953
dixonprice4	*0.987*	*0.986*	0.985	*0.958*	0.981	*0.982*	*0.986*	*0.982*	*0.985*	**0.987**	0.971
bukin	0.822	*0.864*	*0.865*	0.844	*0.860*	0.851	*0.861*	0.852	*0.850*	**0.885**	0.826
shekel5	0.273	*0.383*	0.400	*0.414*	*0.413*	*0.402*	*0.405*	*0.425*	0.366	*0.401*	**0.439**
shekel7	0.280	*0.414*	0.330	*0.397*	0.341	*0.380*	0.369	0.378	*0.406*	**0.445**	*0.387*
michal2	0.990	0.999	0.983	*0.977*	*1.000*	*1.000*	*0.982*	*0.967*	*0.984*	**1.000**	*0.961*
Average	0.842	*0.844*	*0.850*	*0.853*	**0.861**	*0.859*	*0.856*	*0.850*	*0.853*	*0.851*	*0.858*

Table D.4: Average GAP of 50 repeats on real functions for all q.EI variants. Note function names are shortcuts for better spacing.

	EI	2.EI.b	2.EI.s	3.EI.b	3.EI.s	4.EI.b	4.EI.s	6.EI.b	6.EI.s	8.EI.b	8.EI.s
SVM	0.738	*0.926*	*0.913*	*0.930*	**0.940**	*0.914*	*0.911*	*0.892*	*0.937*	*0.929*	0.834
LDA	0.956	**1.000**	*1.000*	*0.998*	*0.996*	0.996	*0.993*	*0.999*	0.982	*0.995*	*0.995*
LR	0.963	*1.000*	*0.998*	*0.999*	*1.000*	*0.999*	*0.999*	**1.000**	*0.999*	*1.000*	*1.000*
Bos	*0.470*	*0.491*	*0.467*	*0.490*	*0.478*	*0.495*	*0.460*	*0.460*	**0.502**	*0.455*	*0.467*
Can	*0.665*	*0.652*	0.627	0.625	*0.654*	*0.640*	*0.686*	0.625	**0.700**	0.609	*0.686*
R3d	0.928	*0.959*	*0.960*	*0.944*	*0.962*	*0.956*	0.957	0.960	*0.962*	**0.967**	*0.961*
R4d	*0.730*	*0.725*	*0.726*	*0.720*	0.695	**0.764**	0.695	*0.760*	*0.736*	*0.732*	0.697
Ave	0.779	*0.821*	*0.813*	*0.815*	*0.818*	*0.823*	*0.815*	*0.813*	**0.831**	*0.812*	*0.806*

References

[1] ASM Alloy Center Database.

[2] Arash Asadpour, Hamid Nazerzadeh, and Amin Saberi. Stochastic Submodular Maximization. In C. Papadimitriou and S. Zhang, editors, *International Workshop on Internet and Network Economics (*WINE *2008)*, volume 5385 of *Lecture Notes in Computer Science*, pages 477–489. Springer–Verlag, 2008.

[3] Peter Auer. Using Confidence Bounds for Exploitation–Exploration Trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[4] Maximilian Balandat, Brian Karrer, Daniel R Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: Programmable Bayesian Optimization in PyTorch. *arXiv preprint arXiv:1910.06403*, 2019.

[5] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer–Verlag, 2 edition, 1980.

[6] Dimitri P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific, 2017.

[7] 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. *CoRR*, abs/1012.2599, 2010.

[8] Henry Chai and Roman Garnett. Improving quadrature for constrained integrands. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (*AISTATS*)*, 2019.

[9] Henry Chai, Jean-François Ton, Michael A. Osborne, and Roman Garnett. Automated model selection with Bayesian quadrature. In *Proceedings of the 36th International Conference on Machine Learning (*ICML*)*, 2019.

[10] Shayok Chakraborty, Vineeth Balasubramanian, and Sethuraman Panchanathan. Adaptive Batch Mode Active Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8):1747–1760, 2015.

[11] Constantinos Charalambous. On the evolution of particle fragmentation with applications to planetary surfaces, 2014. Chapter 5.

[12] Sean X Chen and Jun S Liu. Statistical applications of the Poisson-binomial and conditional Bernoulli distributions. *Statistica Sinica*, pages 875–892, 1997.

[13] Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando De Freitas. Learning to learn without gradient descent by gradient descent. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 748–756. JMLR. org, 2017.

[14] Yuxin Chen and Andreas Krause. Near-optimal Batch Mode Active Learning and Adaptive Submodular Optimization. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (*ICML *2013)*, volume 28 of *Proceedings of Machine Learning Research*, pages 160–168, 2013.

[15] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[16] Nguyen Viet Cuong, Wee Sun Lee, Nan Ye, Kian Ming A. Chai, and Hai Leong Chieu. Active Learning for Probabilistic Hypotheses Using the Maximum Gibbs Error Criterion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26 (*NIPS *2013)*, pages 1457–1465, 2013.

[17] Bin Dai, Shilin Ding, Grace Wahba, et al. Multivariate Bernoulli distribution. *Bernoulli*, 19(4):1465–1483, 2013.

[18] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 15:3873–3923, 2014.

[19] Thomas Desautels, Andreas Krause, and Joel W. Burdick. Parallelizing Exploration-Exploitation Tradeoffs in Gaussian Process Bandit Optimization. *Journal of Machine Learning Research*, 15(Dec):4053–4103, 2014.

[20] Persi Diaconis. Bayesian numerical analysis. *Statistical Decision Theory and Related Topics*, 4(1):163–175, 1988.

[21] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015.

[22] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19:355–369, 2007.

[23] Peter I Frazier, Warren B Poweel, and Savas Dayanik. A Knowledge-Gradient Policy for Sequential Information Collection. SIAM *Journal on Control and Optimization*, 47(5):2410–2439, 2008.

[24] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional Neural Processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[25] 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(4):305–314, 2015.

[26] Roman Garnett, Yamuna Krishnamurthy, Donghan Wang, Jeff Schneider, and Richard Mann. Bayesian Optimal Active Search on Graphs. In *9th Workshop on Mining and Learning with Graphs*, 2011.

[27] Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff G. Schneider, and Richard P. Mann. Bayesian Optimal Active Search and Surveying. In *Proceedings of the 29th International Conference on Machine Learning (*ICML *2012)*, 2012.

[28] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[29] David Ginsbourger and Rodolphe Le Riche. Towards Gaussian process-based optimization with finite time horizon. In *Advances in Model-Oriented Design and Analysis (*MODA*) 9*, pages 89–96, 2010.

[30] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. Technical report, 2008. hal-00260579.

[31] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is Well-Suited to Parallelize Optimization. In Y. Tenne and Goh C. K., editors, *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Adaptation Learning and Optimization*, pages 131–162. Springer–Verlag, 2010.

[32] 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.

[33] Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian optimization via local penalization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (*AISTATS*)*, 2016.

[34] Javier González, Michael Osborne, and Neil D Lawrence. Glasses: Relieving the myopia of Bayesian optimisation. *arXiv preprint arXiv:1510.06299*, 2015.

[35] Javier González, Michael Osborne, and Neil D. Lawrence. GLASSES: Relieving The Myopia Of Bayesian Optimisation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, number 51 in Proceedings of Machine Learning Research, pages 790–799, 2016.

[36] Maura R Grossman and Gordon V Cormack. Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review. *Rich. JL & Tech.*, 17:1, 2010.

[37] Maura R. Grossman, Gordon V. Cormack, and Adam Roegiest. TREC 2016 Total Recall Track Overview. *TREC*, 2016.

[38] Tom Gunter, Michael A. Osborne, Roman Garnett, Philipp Hennig, and Stephen J. Roberts. Sampling for inference in probabilistic models with fast Bayesian quadrature. In *Advances in Neural Information Processing Systems (*NEURIPS*) 27*, 2014.

[39] Philipp Hennig and Christian J Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

[40] Hernández-Lobato, Matthew W José M, Hoffman, and Zoubin Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. In *Advances in Neural Information Processing Systems 27 (*NIPS *2014)*, pages 918–926, 2014.

[41] Trong Nghia Hoang, Kian Hsiang Low, Patrick Jaillet, and Mohan Kankanhalli. Nonmyopic ϵ-Bayes-optimal active learning of Gaussian processes. In *Proceedings of the 31st International Conference on Machine Learning (*ICML*)*, 2014.

[42] Steven C.H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch Mode Active Learning and Its Application to Medical Image Classification. In W. Cohen and A. Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning (*ICML *2006)*, pages 417–424, 2006.

[43] Ronald A Howard. Dynamic Programming and Markov Processes. 1960.

[44] Swarit Jasial, Ye Hu, Martin Vogt, and Jürgen Bajorath. Activity-relevant similarity values for fingerprints and implications for similarity searching. *F1000Research*, 5(Chem Inf Sci):591, 2016.

[45] Shali Jiang, Gustavo Malkomes, Matthew Abbott, Benjamin Moseley, and Roman Garnett. Efficient nonmyopic batch active search. In *Advances in Neural Information Processing Systems (*NEURIPS*) 31*, pages 1099–1109, 2018.

[46] 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 (*ICML*)*, 2017.

[47] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction Tree Variational Autoencoder for Molecular Graph Generation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2323–2332, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[48] Donald R. Jones. Direct global optimization algorithm. In *Encyclopedia of optimization*, pages 431–440. 2009.

[49] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised Bayesian optimisation via Thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.

[50] Yoshiyuki Kawazoe, Jing-Zhi Yu, An-Pang Tsai, and Tsuyoshi Masumoto, editors. *Nonequilibrium Phase Diagrams of Ternary Amorphous Alloys*, volume 37A of *Condensed Matter*. 1997.

[51] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019.

[52] Andreas Krause and Carlos Guestrin. Nonmyopic active learning of Gaussian processes: an exploration-exploitation approach. In *Proceedings of the 24th International Conference on Machine Learning (*ICML*)*, 2007.

[53] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.

[54] Harold J Kushner. A versatile stochastic model of a function of unknown and time varying form. *Journal of Mathematical Analysis and Applications*, 5(1):150–167, 1962.

[55] Harold Joseph Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *ASME. J. Basic Eng*, 86(1):97–106, 1964.

[56] Remi Lam, Karen Willcox, and David H. Wolpert. Bayesian optimization with a finite budget: an approximate dynamic programming approach. In *Advances in Neural Information Processing Systems (*NEURIPS*) 29*, 2016.

[57] F. M. Larkin. Gaussian measure in Hilbert space and applications in numerical analysis. *Rocky Mountain Journal of Mathematics*, 2(3):379–422, 1972.

[58] 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*, 1994.

[59] Steffen Liebscher and Thomas Kirschstein. Predicting the outcome of professional darts tournaments. *International Journal of Performance Analysis in Sport*, 17(5):666–683, 2017.

[60] Chun Kai Ling, Kian Hsiang Low, and Patrick Jaillet. Gaussian process planning with Lipschitz continuous reward functions: towards unifying Bayesian optimization, active learning, and beyond. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[61] 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(suppl 1):D198–D201, 2007.

[62] Yifei Ma, Roman Garnett, and Jeff G. Schneider. Active Area Search via Bayesian Quadrature. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

[63] Yifei Ma, T-K Huang, and Jeff G. Schneider. Active Search and Bandits on Graphs using Sigma-Optimality. In M. Meila and T. Heskes, editors, *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (*UAI *2015)*, pages 542–551, 2015.

[64] Yifei Ma, Dougal J. Sutherland, Roman Garnett, and Jeff Schneider. Active Pointillistic Pattern Search. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, number 38 in Proceedings of Machine Learning Research, pages 672–680, 2015.

[65] Gustavo Malkomes and Roman Garnett. Automating Bayesian optimization with Bayesian optimization. In *Advances in Neural Information Processing Systems (*NEURIPS*) 31*, 2018.

[66] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[67] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1974.

[68] Iain Murray. Differentiation of the Cholesky decomposition. *arXiv preprint arXiv:1602.07527*, 2016.

[69] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An Analysis of Approximations for Maximizing Submodular Set Functions. *Mathematical Programming*, 14(1):265–294, 1978.

[70] Anthony O'Hagan. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29:245–260, 1991.

[71] Michael A. Osborne, Roman Garnett, Zoubin Ghahramani, David Duvenaud, Stephen J. Roberts, and Carl E. Rasmussen. Active learning of model evidence using Bayesian quadrature. In *Advances in Neural Information Processing Systems (*NEURIPS*) 25*, 2012.

[72] Michael A Osborne, Roman Garnett, and Stephen J Roberts. Gaussian processes for global optimization. In *The 3rd International Conference on Learning and Intelligent Optimization (LION3)*, 2009.

[73] Joseph J Pfeiffer III, Jennifer Neville, and Paul N Bennett. Active Exploration in Networks: Using Probabilistic Relationships for Learning and Inference. In *Proceedings of the 23rd* ACM *International Conference on Information and Knowledge Management*, pages 639–648, 2014.

[74] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[75] Jean-Michel Renders. Active Search for High Recall: A Non-stationary Extension of Thompson Sampling. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval*, pages 722–728, 2018.

[76] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

[77] Régis Sabbadin, Jérôme Lang, and Nasolo Ravoanjanahary. Purely Epistemic Markov Decision Processes. In *Proceedings of the 22nd* AAAI *Conference on Artificial Intelligence*, pages 1057–1062, 2007.

[78] Vydūnas R. Šaltenis. One Method of Multiextremum Optimization. *Avtomatika i Vychislitel'naya Tekhnika (Automatic Control and Computer Sciences)*, 5(3):33–38, 1971.

[79] Steven L Scott. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[80] Burr Settles. Active learning literature survey. *Computer Sciences Technical Report*, 2010.

[81] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems (*NEURIPS*) 28*, 2015.

[82] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: a review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, Jan 2016.

[83] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011.

[84] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[85] 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.

[86] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (*NEURIPS*) 25*, pages 2951–2959, 2012.

[87] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1015–1022, USA, 2010. Omnipress.

[88] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias W. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (*ICML *2010)*, pages 1015–1022, 2010.

[89] Teague Sterling and John J. Irwin. ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

[90] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[91] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[92] Hastagiri P. Vanchinathan, Andreas Marfurt, Charles-Antoine Robelin, Donald Kossmann, and Andreas Krause. Discovering Valuable Items from Massive Data. In *Proceedings of the 21st* ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining (*KDD *2015)*, pages 1195–1204, 2015.

[93] Jialei Wang. *Bayesian Optimization with Parallel Function Evaluations and Multiple Information Sources: Methodology with Applications in Biochemistry, Aerospace Engineering, and Machine Learning*. PhD thesis, Operations Research and Information Engineering, Cornell University, 2017.

[94] Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel Bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*, 2016.

[95] Xuezhi Wang, Roman Garnett, and Jeff G. Schneider. Active Search on Graphs. In R Ghani, T E Senator, P Bradley, R Parekh, and J He, editors, *Proceedings of the 19th* ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining (*KDD *2013)*, pages 731–738, 2013.

[96] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *International Conference on Machine Learning (ICML)*, 2017.

[97] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A General-Purpose Machine Learning Framework for Predicting Properties of Inorganic Materials. 2016. arXiv preprint arXiv:1606.09551 [cond-mat.mtrl-sci].

[98] 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.

[99] Manfred K Warmuth, Gunnar Rätsch, Michael Mathieson, Jun Liao, and Christian Lemmen. Active Learning in the Drug Discovery Process. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (*NIPS *2001)*, pages 1449–1456, 2001.

[100] Kevin Williams, Elizabeth Bilsland, Andrew Sparkes, Wayne Aubrey, Michael Young, Larisa N Soldatova, Kurt De Grave, Jan Ramon, Michaela de Clare, Worachart Sirawaraporn, et al. Cheaper faster drug development validated by the repositioning of drugs against neglected tropical diseases. *Journal of the Royal Society Interface*, 12(104):20141289, 2015.

[101] James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *Advances in Neural Information Processing Systems 31*, pages 9905–9916. 2018.

[102] Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch Bayesian optimization. In *Advances in Neural Information Processing Systems (*NEURIPS*) 29*, 2016.

[103] Jian Wu and Peter Frazier. Practical two-step lookahead Bayesian optimization. In *Advances in Neural Information Processing Systems (*NEURIPS*) 32*, 2019.

[104] Zhe Yu and Tim Menzies. Total recall, language processing, and software engineering. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, pages 10–13, 2018.

[105] Xubo Yue and Raed Al Kontar. Why non-myopic Bayesian optimization is promising and how far should we look-ahead? A study via rollout. *arXiv*, 2019. Accepted to AISTATS 2020.

[106] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401. 2017.