Efficient nonmyopic active search

Shali Jiang, Gustavo Malkomes
Washington University in St. Louis
{jiang.s,luizgustavo}@wustl.edu

Geoff Converse
Simpson College
geoff.converse@my.simpson.edu

Alyssa Shofner
University of South Carolina
alyssa.shofner@icloud.com

Benjamin Moseley, Roman Garnett
Washington University in St. Louis
{bmoseley,garnett}@wustl.edu

Abstract

Active search is a learning paradigm with the goal of actively identifying as many members of a given class as possible. Many real-world problems can be cast as an active search, including drug discovery, fraud detection, and product recommendation. Previous work has derived the Bayesian optimal policy for the problem, which is unfortunately intractable due to exponential complexity. In practice, myopic approximations are used instead, only looking a small number (e.g., 1–3) of steps ahead in the decision process. We propose a novel active search policy that always considers the entire remaining budget and is thus nonmyopic, yet remains efficient. Our approach automatically and dynamically balances exploration and exploitation in a manner consistent with the budget, without relying on a tradeoff parameter. We also develop a bounding technique to achieve greater efficiency when using certain natural probability models. Experimental results show superior performance of our method over myopic approximations to the optimal policy.

1 Introduction

In active search (AS), 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 is to design an algorithm able to sequentially query points to find as many valuable points as possible under a budget constraint. Real-world problems can be naturally posed in terms of active search; drug discovery [3], fraud detection, and product recommendation [14] are a few examples.

Previous work [4, 5] developed Bayesian optimal policies for active search with a natural utility function. Not surprisingly, this policy requires exponential computation. To overcome this computational intractability, myopic lookahead policies are used in practice, which compute the optimal policy only up to a limited number of steps into the future, ignoring what can happen beyond that horizon.

We introduce a novel nonmyopic policy for active search that considers not only the potential immediate contribution of each unlabeled data point but also the potential impact on the remaining points which 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 in each step by bounding their potential impact to the search problem. Our results demonstrate that our approach significantly outperforms myopic lookahead policies and can dynamically adapt to the remaining budget.

Related work. Active search falls into the broader framework of active learning [7, 12], but with drastically different goal. Active search is similar in spirit to the multi-armed bandit (MAB) problem, but the crucial difference is in AS the "arms" (i.e., unlabeled points) are correlated and the rewards are deterministic. Despite the difference, we note that our policy is similar in spirit to the *knowledge* *gradient* policy introduced in [2]. [15] proposed a method called GP-SELECT specifically for Gaussian process models, inspired by the GP-UCB algorithm [13]. Our policy doesn't make any assumption about the underlying model. Readers interested in active search specifically on graphs can refer to [4, 16, 11, 9]. Variations of AS such as *active area search* [8] and *active pointillistic pattern search* [10] have also been considered. Active search can also be interpreted as a realization of Bayesian optimization with binary rewards and cumulative regret. Interestingly, we also note our policy is similar to a recently proposed nonmyopic Bayesian optimization algorithm, GLASSES [6], in the sense that we both consider the remaining budget when choosing which point to evaluate next.

2 Efficient nonmyopic active search

Suppose we are given a finite domain of elements $\mathcal{X} \triangleq \{x_i\}$. 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 $y \triangleq \mathbb{1}\{x \in \mathcal{R}\}$. The oracle, however, is assumed to be expensive and may only be queried t times. We seek to design a policy that sequentially queries elements to maximize the number of targets discovered.

We will express our preference over different sets of observations $\mathcal{D} = \big\{(x_i, y_i)\big\}$ through a natural utility: $u(\mathcal{D}) = \sum y_i$, which simply counts the number of targets in $\mathcal{D}$. Then, the problem is to sequentially construct a set of t points $\mathcal{D}$ with the goal of maximizing $u(\mathcal{D})$. We use $\mathcal{D}_i = \big\{(x_j, y_j)\big\}_{j=1}^{i}$ to denote the observed data after $i \leq t$ queries.

As we have mentioned before, myopic ℓ-step-lookahead approximations of Bayesian optimal policies [5] assume 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, 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. By exploiting linearity of expectation, it is easy to work out the optimal policy for selecting such a batch: we simply select the points with the highest probability of being valuable. The resulting approximation is

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

where the sum-with-prime symbol $\sum'_k$ indicates summing the largest k values (here $x \in \mathcal{X} \setminus \mathcal{D}_i$).

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

$$\mathbb{E}\big[u(\mathcal{D}_t) \mid x_i, \mathcal{D}_{i-1}\big] \approx u(\mathcal{D}_{i-1}) + \Pr(y_i = 1 \mid x_i, \mathcal{D}_{i-1}) + \underbrace{\mathbb{E}_{y_i}\Big[{\sum}'_{t-i} \Pr\big(y = 1 \mid x, \mathcal{D}_i\big)\Big]}_{\text{exploration, } < t-i}. \tag{2}$$

We will call this policy *efficient nonmyopic search* (ENS). As in the optimal policy, we can interpret (2) 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 (2) 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.

Note that we may also use the approximation in (1) 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 (2) is equivalent to a one-step maximization of (1). 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.

Nonmyopic behavior. To illustrate the nonmyopic behavior of our policy, we have adapted the toy example presented in [5] (Section 7.1). Let $I \triangleq [0, 1]^2$ be the unit square. We repeated the following experiment 100 times. We selected 500 points iid uniformly at random from I to form the $\mathcal{X}$ input space. 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 any of five points: $(0, 0), (0, 1), (1, 0), (1, 1), (1/2, 1/2)$ (the four corners

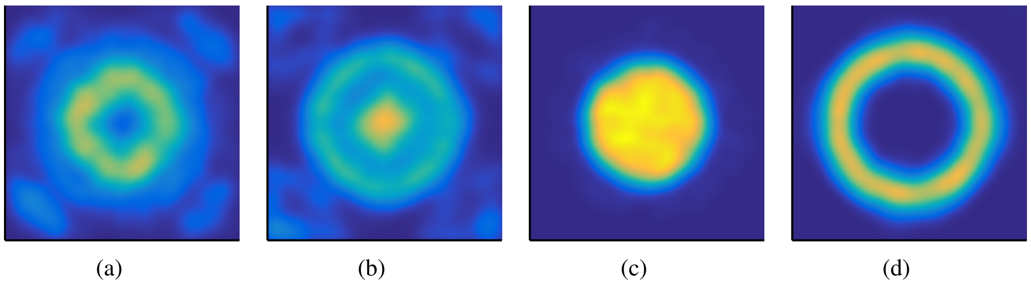

Figure 1: Kernel density estimates of the distribution of points chosen by ENS (a–b) and 2-step lookahead [5] (c–d) during two different time intervals. The figures (a) and (c) show the kernel density estimates for the first 100 locations; the figures (b) and (d), the last 100 chosen locations.

and the center). The closest point to the center (with overwhelming probability a target) and a random element of $\mathcal{X} \setminus \mathcal{R}$ forms an initial training set. We then applied ENS and the two-step-lookahead [5] policies to sequentially select 200 further points for labeling.

Figure 1 shows a kernel density estimate of the distribution of locations selected by both methods during two different time intervals. Figures 1(a–b) correspond to our method; Figures 1(c–d) to two-step lookahead. Figures 1(a, c) consider the distribution of the first 100 selected locations; Figures 1(b, d) consider the distribution of the last 100. The qualitative difference between these strategies is clear. The myopic policy focused on collecting all targets around the center (Figure 1(c)), whereas our policy explores the boundaries of the center clump (with considerably intensity), as well as some of the corners (Figure 1(a)). As a result, our policy is capable of finding some of target points in the corners, whereas two-step lookahead cannot (Figure 1(d)). More interestingly, we can see that the highest probability mass in Figure 1(b) is the square center, which shows that our policy typically saves the points with highest probability until the end. On average, the ENS policy found about 20 more targets at termination than the two-step-lookahead policy.

Implementation and time complexity. Suppose after observing a point we only need to update the probabilities of at-most m other points (e.g., for a k-nn model). We can achieve a computational complexity of $\mathcal{O}\big(n(\log n + m \log m + t)\big)$. which is about the same as two-step lookahead, which is $\mathcal{O}\big(n(\log n + m)\big)$ when implemented cleverly.

Pruning the search space. To further reduce the computational complexity, we can use a similar strategy as in [5] to bound the score function (2) and prune the point that cannot possibly maximize the expected utility. We empirically observe that over 92% of points can often be pruned on massive drug discovery datasets [3]. Unfortunately we must omit the derivation of pruning strategy and these massive empirical results due to the space limit.

3 Experiments

We implemented our approximation to the Bayesian optimal policy with the MATLAB active learning toolbox,[1] and have compared the performance of our proposed ENS policy with the myopic one-step (greedy) and two-step approximations to the Bayesian optimal policy, presented in [5]. Note that [5] and [3] thoroughly compared the one- and two-step policies, with the finding that the less-myopic two-step algorithm usually performs better in terms of targets found, as one would expect. The probability model $\Pr(y = 1 \mid x, \mathcal{D})$ we will adopt is the k-nearest-neighbor (k-nn) classifier as described in Section 7 of [5], along with the corresponding probability bound.

CiteSeer^x data. We consider a subset of the CiteSeer^x citation network, first described in [5]. This dataset comprises 39 788 [2] 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 NIPS proceedings; there are 2 190 such papers, 5.5% of the whole dataset. Note that distinguishing NIPS 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

[1] `https://github.com/rmgarnett/active_learning`

[2] This dataset is constantly evolving by improving preprocessing, so numbers may change.

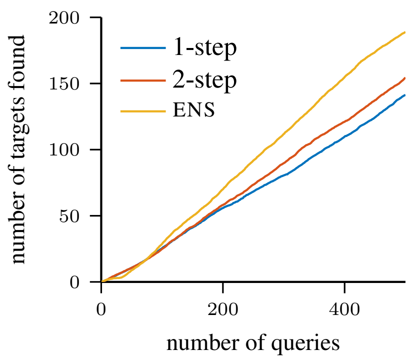

Figure 2: The learning curve of the one-step and two-step myopic polices and our policy on the NIPS dataset, averaged over 20 trials, each starting with a random positive example.

	query number				
policy	100	300	500	700	900
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**				

Table 1: Average number of targets found by the one- and two-step myopic policies and ENS with different five budgets. The performance of the best method at each time waypoint is in bold.

vector for each paper is computed by performing graph principal component analysis [1] on the citation network and retaining the first 20 principal components.

We select a single target (i.e., a NIPS 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 [5]. We use each policy to sequentially select t papers for labeling. The experiment was repeated 20 times, varying the initial seed target. Figure 2 shows the average number of NIPS papers found for each method as a function of the number of queries. We first observe that the two-step policy outperforms the greedy one-step policy, as expected, and matching the results from [5]. Second, 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 of $[19.80, 49.30]$.

Another interesting observation is that during the initial $\sim$80 queries, ENS actually performs *worse* on average than both myopic policies, 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 to control this tradeoff.

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 1 the average number of NIPS papers found at these time points for each method. We have the following observations from the table. First, ENS performs better than the myopic 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.

4 Conclusion

In this paper we 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. Experimental results demonstrate our superior performance and automatic balance between exploration and exploitation.

5 Acknowledgments

SJ, GM, and RG were supported by the National Science Foundation (NSF) under award number IIA–1355406. Additionally, GM was also supported by the Brazilian Federal Agency for Support and Evaluation of Graduate Education (CAPES). GC and AS were supported by NSF under award number CNS–1560191. BM was supported in part by a Google Research Award, a Yahoo Research Award, and by NSF under award number CCF–1617724.

References

[1] F. 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.

[2] 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.

[3] 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.

[4] Roman Garnett, Yamuna Krishnamurthy, Donghan Wang, Jeff Schneider, and Richard Mann. Bayesian optimal active search on graphs. In *Ninth Workshop on Mining and Learning with Graphs*, 2011.

[5] 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*, 2012.

[6] Javier González, Michael Osborne, and Neil D Lawrence. Glasses: Relieving the myopia of Bayesian optimisation. *arXiv preprint arXiv:1510.06299*, 2015.

[7] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International* ACM SIGIR *conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[8] Yifei Ma, Roman Garnett, and Jeff G. Schneider. Active Area Search via Bayesian Quadrature. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 595–603, 2014.

[9] Yifei Ma, Tzu-Kuo Huang, and Jeff G. Schneider. Active Search and Bandits on Graphs using Sigma-Optimality. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 542–551, 2015.

[10] Yifei Ma, Dougal J. Sutherland, Roman Garnett, and Jeff G. Schneider. Active pointillistic pattern search. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015.

[11] 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.

[12] Burr Settles. Active learning literature survey. *Computer Sciences Technical Report*, 1648, 2010.

[13] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias W. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1015–1022, 2010.

[14] Dougal J. Sutherland, Barnabás Póczos, and Jeff Schneider. Active Learning and Search on Low-Rank Matrices. In *Proceedings of the 19th* ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining (*KDD *2013)*, pages 212–220, 2013.

[15] Hastagiri P. Vanchinathan, Andreas Marfurt, Charles-Antoine Robelin, Donald Kossmann, and Andreas Krause. Discovering valuable items from massive data. In *Proceedings of the 21th* ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1195–1204, 2015.

[16] Xuezhi Wang, Roman Garnett, and Jeff G. Schneider. Active search on graphs. In *The 19th* ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining*, pages 731–738, 2013.