Efficient nonmyopic batch active search

Shali Jiang, Gustavo Malkomes, Matthew Abbott, Benjamin Moseley, Roman Garnett
Washington University in St. Louis
{jiang.s,luizgustavo,mbabbott,bmoseley,garnett}@wustl.edu

Abstract

Active search is a learning paradigm for actively identifying as many members of a given class as possible. One could see active search as a special case of Bayesian optimization with binary rewards and cumulative regret. Many real-world problems can be cast as an active search, including drug discovery, fraud detection, and product recommendation. All existing works focus on sequential policies, i.e., selecting one point to query at a time. However, in many real applications (e.g. drug discovery), it is possible to evaluate *multiple* elements simultaneously. In this paper we investigate batch active search, and this is the first such study we know in the literature. We first derive the Bayesian optimal policy for batch active search, then propose two strategies to approximate it efficiently and nonmyopically. The results on a citation network and drug discovery data demonstrate that our methods are very promising.

1 Introduction

In active search (AS), we seek to sequentially inspect data so as to discover members of a rare, desired class. Formally, suppose we are given a finite domain of n elements $\mathcal{X} \triangleq \{x_i\}_{i=1}^n$, among which there is a rare, valuable subset $\mathcal{R} \subset \mathcal{X}$, which we call *targets* or *positive* items. The identities are unknown *a priori*, but can be identified by querying an expensive oracle (i.e., the oracle can tell $y \triangleq \mathbb{1}\{x \in \mathcal{R}\}$ for any $x \in \mathcal{X}$, but with a price). We seek to design a policy that sequentially queries elements to maximize the number of targets identified, under a given labeling budget (See Algorithm 1). Many real-world problems can be naturally posed in terms of active search; drug discovery [4], fraud detection, and product recommendation [18] are a few examples.

Previous work [5, 6] has developed Bayesian optimal policies for active search with a natural utility function. Not surprisingly, this policy requires exponential computation. In fact, Jiang *et al.* (2017) have shown that it is impossible to α-approximate the optimal policy for any constant α. To overcome this computational intractability, in practice, the optimal policy is computed with a limited horizon (e.g. one or two-step ahead). We refer to these policies as myopic due to their shortsighted horizon. An *nonmyopic* policy was proposed in [11], called efficient nonmyopic search (ENS), and demonstrated remarkable empirical performance on various domains, including citation network, material science and drug discovery.

All these work focused on sequential active search (SAS), that is, selecting one point at a time. However, in many real applications, we can query the identities of multiple points simultaneously. For example, modern virtual screening technologies can process multiples of 96 compounds at a time. In this paper, we deliver the first investigation of batch active search (BAS). In particular, we generalize the Bayesian optimal policy for SAS to BAS, and propose two strategies to approximate it efficiently and nonmyopically. We present our preliminary results on a CiteSeer^x citation network and drug discovery data, and demonstrate that our proposed methods are very promising for batch active search.

Related work. Active search and its variants have been the subject of many recent work [6, 19, 14, 15, 11], nevertheless, to the best of our knowledge, this paper is the first study of batch active search.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Batch policies have been studied in similar contexts, such as *active learning* [10, 8, 2], *multi-armed bandits* [3, 13, 12] and *Bayesian optimization* [1, 17, 7, 9]. In particular, one of our policies is similar to the active learning method in [2], which also constructs batches in a greedy fashion.

2 Bayesian optimal policy and its nonmyopic approximations

In this section, we first derive the Bayesian optimal policy for batch active search, then propose two efficient and nonmyopic approximations to the optimal policy based on recent work in [11].

Algorithm 1 Batch active search

Require: $\mathcal{X}$, $\mathcal{P}$, budget B, batch size b, $\mathcal{D}_0$, batch policy $\Pi : (\mathcal{D}, B, b) \mapsto X$
Ensure: $\mathcal{D}_t$, where $t = B/b$
1: **for** $i = 1$ to t **do**
2: $\quad X = \Pi(\mathcal{D}_i, B, b)$
3: $\quad$ Query labels Y of X from oracle
4: $\quad \mathcal{D}_i = \mathcal{D}_{i-1} \cup \{(X, Y)\}$
5: **end for**
6: **return** $\mathcal{D}_t$

Bayesian optimal policy. We will express our preference over different sets of observations $\mathcal{D} = \{(x_i, y_i)\}$ through a natural utility: $u(\mathcal{D}) = \sum y_i$, which simply counts the number of targets in $\mathcal{D}$ (sometimes we also use $u(Y)$ for $\mathcal{D} = \{(X, Y)\}$). Then, the problem is to sequentially construct a set of B (a given budget) points $\mathcal{D}$ with the goal of maximizing $u(\mathcal{D})$. In the batch setting, each time we need to pick a batch of b points, and query their labels at the same time. We use $X_i = \{x_i^{(j)}\}_{j=1}^b$ to denote the batch of points chosen at the i-th iteration, and $Y_i = \{y_i^{(j)}\}_{j=1}^b$ 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 $t = B/b$.

We assume a Bayesian probability model $\mathcal{P}$ is given, providing $\Pr(y \mid x, \mathcal{D})$ for any $x \in \mathcal{X}$ and $\mathcal{D} \subseteq \mathcal{X}$ (possibly $\mathcal{D} = \emptyset$). The expected utility of choosing a batch X of size b at iteration $i+1$, given observation $\mathcal{D}_i$, can be written as a Bellman equation as follows:

$$\mathbb{E}[u(\mathcal{D}_t \setminus \mathcal{D}_i) \mid X, \mathcal{D}_i] = \mathbb{E}_{Y|X,\mathcal{D}_i}\left[u(Y) + \max_{X':|X'|=b} \mathbb{E}[u(\mathcal{D}_t \setminus \mathcal{D}_{i+1}) \mid X', \mathcal{D}_i, X, Y]\right], \quad (1)$$

where the expectation is taken over the joint distribution of Y (labels of X) conditioned on $\mathcal{D}_i$, and first part is the expected utility of points in the batch X, and the second part is the expected future utility if we choose $X_i = X$ and continue to choose later batches optimally. These two parts can balance the exploration (future reward) and exploitation (immediate reward of this batch).

Without further assumptions on $\mathcal{P}$, the complexity for computing (1) is dauntingly $\mathcal{O}\left(\left(\binom{n}{b}2^b\right)^{t-i}\right)$ to enumerate the search tree of depth $t - i$. The derivation in [6] is a special case when $b = 1$. One simple work-around is myopic approximations of the optimal policy. A greedy (one-step lookahead) approximation, choosing the b points with highest probabilities, maximizes $\mathbb{E}_{Y|X,\mathcal{D}_i}[u(Y)]$. It ignores all future utilities, but can be computed in $\mathcal{O}(n \log n)$. We will refer to this policy as greedy-batch.

Nonmyopic approximation. The idea of ENS in [11] can also be generalized to batch case. The approach can be motivated with the following question: how many targets do we expect to see if, after selecting the current batch, all the remaining budget is spent simultaneously? If it were really the case, then the expected utility could be approximated by

$$\mathbb{E}[u(\mathcal{D}_t \setminus \mathcal{D}_i) \mid X, \mathcal{D}_i] = \mathbb{E}_{Y|X,\mathcal{D}_i}\left[u(Y) + \max_{X':|X'|=B-b-|\mathcal{D}_i|} \mathbb{E}[u(Y') \mid X', \mathcal{D}_i, X, Y]\right], \quad (2)$$

where the second part can be computed by summing the highest $B-b-|\mathcal{D}_i|$ probabilities. Specifically, (2) can be written as (use $f(X \mid \mathcal{D}_i)$ as a shortcut)

$$f(X \mid \mathcal{D}_i) = \sum_{x \in X} \Pr(y = 1 \mid x, \mathcal{D}_i) + \mathbb{E}_{Y|X,\mathcal{D}_i}\left[\sum'_{B-b-|\mathcal{D}_i|} \Pr(y' = 1 \mid x', \mathcal{D}_i, X, Y)\right], \quad (3)$$

where we used a notation $\sum'_s$ to denote the sum over the top s probabilities for all $x' \in \mathcal{X} \setminus (\mathcal{D}_i \cup X)$. Note (3) is actually approximating (1) by assuming that 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 more *well-spaced* points are observed, they might approximately "D-separate" the remaining unlabeled points. And as demonstrated in Figure 1 of [11], ENS actually encourages the selection of well-spaced points (exploration) in the initial state of the search. We will term this policy (maximizing (3)) batch-ENS. Note ENS in [11] is a special case

of batch-ENS for $b = 1$, and its remarkable performance has been empirically validated by massive experiments on various domains [11].

The nonmyopia of batch-ENS is automatically inherited in the generalization from sequential to batch setting, but not efficiency. Direct maximization of (3) still requires combinatorial search over all subsets of size b, and for a given batch, we need to enumerate all its possible labelings (2^b) to compute the second expectation part, plus summing the top probabilities; the total complexity would be $\mathcal{O}\left(\binom{n}{b}2^b \cdot n \log n\right)$. We propose two strategies to tackle these computational problems.

Sequential simulation. Note the exponential complexity comes from the fact that $b > 1$. So our first idea is to reduce BAS to SAS ($b = 1$). To be specific, we select points one by one to add into the batch with ENS, and then use some fictional label oracle $\mathcal{L} : \mathcal{X} \mapsto \{0, 1\}$ to simulate its label, and repeat until we have b points. Four fictional oracles are considered: (1) sampling, i.e., randomly sample $y^{(k)}$ from $\Pr(y^{(k)} \mid x^{(k)}, \mathcal{D}_i, X^{(k-1)})$; (2) most-likely, i.e., $y^{(k)} = \arg\max_{y\in\{0,1\}} \Pr(y^{(k)} = y \mid x^{(k)}, \mathcal{D}_i, X^{(k-1)})$; (3) pessimistic, i.e., always set $y^{(k)} = 0$; (4) optimistic, i.e., always set $y^{(k)} = 1$. Note in this framework can be replaced by any other sequential policy $\pi : \mathcal{D} \mapsto \mathcal{X}$, such as the myopics ones [6]. We call this type of policies seq-sim-batch, summarized in Algorithm 2.

Algorithm 2 seq-sim-batch

procedure SEQ-SIM-BATCH($\mathcal{X}$, $\mathcal{D}_i$, B, b, π, $\mathcal{L}$)
  Initialize: $X^{(0)} = \{\}$
  for $k = 1$ to b **do**
    Select $x^{(k)} = \pi(\mathcal{D}_i \cup \{(X^{(k-1)}, Y^{(k-1)})\})$
    "Query" label $y^{(k)}$ of $x^{(k)}$ from $\mathcal{L}$
    $X^{(k)} = X^{(k-1)} \cup \{x^{(k)}\}$
    $Y^{(k)} = Y^{(k-1)} \cup \{y^{(k)}\}$
  end for
  return $X^{(b)}$
end procedure

Algorithm 3 batch-ENS-greedy

procedure BATCH-ENS-GREEDY($\mathcal{X}$, $\mathcal{D}_i$, B, b)
  Initialize: $X^{(0)} = \{\}$
  for $k = 1$ to b **do**
    $x^{(k)} = \arg\max_x \Delta_f(x \mid X^{(k-1)})$ as in (4)
    $X^{(k)} = X^{(k-1)} \cup \{x^{(k)}\}$
  end for
  return $X^{(b)}$
end procedure

Greedy approximation. The second strategy is motivated by our conjecture that (3) is a monotone submodular function. If that's the case, a *greedy* solution is guaranteed not much worse [16]. So we propose to use *greedy* algorithm to sequentially construct the batch by maximizing the marginal gain. That is, for $k = 1, \ldots, b$,

$$x^{(k)} = \arg\max_x \Delta_f(x \mid X^{(k-1)}) \equiv f(X^{(k-1)} \cup \{x\} \mid \mathcal{D}_i) - f(X^{(k-1)} \mid \mathcal{D}_i). \tag{4}$$

In cases when b is large, this still expensive to compute due to the expectation term ($\mathcal{O}(2^b)$). We use the Monte Carlo method to estimate the expectation in (3) using a small set of samples of the labels. Specifically, given a batch of points $X : |X| \leq b$, (3) is approximated with samples $S = \{\tilde{Y} : \tilde{Y} \sim Y \mid X, \mathcal{D}_i\}$ as following: $f(X \mid \mathcal{D}_i) \approx$

$$\sum_{x\in X} \Pr(y = 1 \mid x, \mathcal{D}_i) + \tfrac{1}{|S|} \sum_{Y\in S} \left[\sum\nolimits'_{B-b-|\mathcal{D}_i|} \Pr\left(y' = 1 \mid x', \mathcal{D}_i, X, Y\right)\right]. \tag{5}$$

We use 16 samples (i.e. $|S| = 16$) in our experiments. Our preliminary results show that our method is not very sensitive to the number of samples after a certain point (e.g. $|S| \geq 8$). Indeed, since our policy is itself an approximation to the optimal policy, more accurate estimation of (3) need not result in better final performance.

To further reduce the computational burden, the implementation tricks as described in 3.2 of [11] can be adapted. We also adapted and improved the pruning technique developed in [11], which can most of the time prune over 90% of the points in the search space in each iteration. But we save the details due to space. We call this greedy approximation batch-ENS-greedy, summarized in Algorithm 3.

3 Experiments

In this section, we present some preliminary results comparing our policies with several baselines. Since there are no batch active search policies in the literature yet (as far as we know), the main baseline we consider is the *greedy-batch* policy mentioned in last section, i.e., choose the b points with highest probabilities. We also consider seq-sim-batch combined with other sequential policies: one-step and two-step [6], which are myopic. For the probability model $\mathcal{P}$, we use k-nn with $k = 50$

Table 1: Average number of NIPS papers found from CiteSeer^x citation network by various BAS policies, over 20 experiments. The best results are **bold**. seq-sim-batch uses "pessimistic" oracle.

batch sizes (b)	1	5	10	15	20	25
greedy-batch	155.30	154.35	152.05	148.55	150.45	150.70
batch-ENS-greedy	**188.40**	167.60	161.35	**162.15**	**155.65**	**156.95**
seq-sim-batch (one-step)	155.30	154.35	152.05	148.55	150.75	150.70
seq-sim-batch (two-step)	165.80	152.60	155.40	154.50	152.55	153.80
seq-sim-batch (ENS)	**188.40**	**168.55**	**166.70**	154.05	148.40	148.45

for all experiments, as in [6]. For the four fictional oracles, preliminary results show that "pessimistic" one often outperforms others; so we only present the results for this one for seq-sim-batch due to space limit.

Finding NIPS papers from a CiteSeer^x citation network. In this experiment, we consider a subset of the CiteSeer^x citation network (first described in [6]) comprised of 39 788 computer science papers published in the top-50 most popular computer science venues. Among them there are 2190 NIPS paper (5.5%), and our active search task is to find those NIPS papers. Note this is a challenging task since many very similar venues such as ICML, AAAI, IJCAI etc. are also in this network. We use experimental settings matching [11]. Specifically, we randomly select a single target (i.e. a NIPS paper) to form the initial observations $\mathcal{D}_0$, and set budget $B = 500$. We show the average number of NIPS papers found at end for batch sizes 1, 5, 10, 15, 20, 25 in Table 1.

Drug discovery. We also conduct experiments on a massive database of chemoinformatic data as did in [11]: 120 datasets for two fingerprints (ECFP4 and GpiDAPH3), each of which has 100 000 negative points and 200 to 1 488 positive points. For space limitation, we refer the readers to Section 5.3 of [11] for detailed description of these datasets. Here we only report the results for the first 10 datasets for ECFP4 (the remaining not available yet), and only batch sizes 1, 5, 10, 20, in Table 2.

Table 2: Number of active compounds found by various BAS policies, averaged over 10 datasets and 20 experiments each (so 200 experiments). The best results are **bold**.

batch sizes (b)	1	5	10	20
greedy-batch	269.84	249.16	246.35	240.59
batch-ENS-greedy	**295.10**	270.42	262.69	261.66
seq-sim-batch (one-step)	269.84	249.16	247.26	240.59
seq-sim-batch (two-step)	281.05	234.94	232.61	228.08
seq-sim-batch (ENS)	**295.10**	**270.93**	**270.33**	**266.31**

Discussion. We can make the following observations: (1) The performance of batch active search generally degrades as the batch size increases. In the sequential setting ($b = 1$), every decision is based on the most updated information acquired from the true label oracle, whereas in batch setting, most decisions are made based on fictional labels (e.g. sampling); the larger the batch size, the higher the chance of a mismatch between the fictional labels and the true labels, which could to lead to worse decisions. (2) batch-ENS-greedy and seq-sim-batch with ENS performs the best among all policies (including seq-sim-batch with all other three fictional oracles, not shown here due to space). We believe this is because these two policies are nonmyopic (taking remaining budget into consideration when making a decision), whereas others are not. However, the results for these two policies are inconsistent for batch size 20 on the two types of data. More experiments are needed to explain this. (3) Another interesting observation is that seq-sim-batch with the pessimistic oracle often performs better than that with other oracles (not shown here). One explanation could be that being pessimistic actually encourages selecting a more diverse batch, since this way the next point is unlikely nearby the points already chosen in the batch, resulting in better exploration. This coincides with the idea for choosing a diverse batch for Bayesian optimization, e.g., via Determinantal point processes [13].

4 Conclusion

In this paper, we delivered the first investigation on batch active search. Several efficient and nonmyopic policies are proposed based on state-of-the-art sequential nonmyopic policies. Results demonstrated their superior empirical performance on various datasets. One interesting theoretical aspect of this problem is: As we have noticed, there are performance gaps between different batch sizes. Can we theoretically (upper and/or lower) bound the gaps (between optimal policies)?

References

[1] Javad Azimi, Alan Fern, and Xiaoli Z. Fern. Batch Bayesian Optimization via Simulation Matching. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 109–117. Curran Associates, Inc., 2010.

[2] Yuxin Chen and Andreas Krause. Near-optimal batch mode active learning and adaptive submodular optimization. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages I–160–I–168. JMLR.org, 2013.

[3] Thomas Desautels, Andreas Krause, and Joel W. Burdick. Parallelizing Exploration-Exploitation Tradeoffs in Gaussian Process Bandit Optimization. *Journal of Machine Learning Research*, 15:4053–4103, 2014.

[4] 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.

[5] 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.

[6] 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.

[7] Javier Gonzalez, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian Optimization via Local Penalization. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 648–657, Cadiz, Spain, 09–11 May 2016.

[8] Yuhong Guo and Dale Schuurmans. Discriminative batch mode active learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 593–600. Curran Associates, Inc., 2008.

[9] José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1470–1479, 2017.

[10] Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 417–424, New York, NY, USA, 2006. ACM.

[11] Shali Jiang, Gustavo Malkomes, Geoff Converse, Alyssa Shofner, Benjamin Moseley, and Roman Garnett. Efficient nonmyopic active search. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1714–1723, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[12] Kwang-Sung Jun, Kevin Jamieson, Robert Nowak, and Xiaojin Zhu. Top arm identification in multi-armed bandits with batch arm pulls. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 139–148, Cadiz, Spain, 09–11 May 2016.

[13] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched Gaussian Process Bandit Optimization via Determinantal Point Processes. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4206–4214, 2016.

[14] 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.

[15] 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.

[16] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[17] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3330–3338. Curran Associates, Inc., 2015.

[18] 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.

[19] 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.