Efficient Nonmyopic Active Search

Shali Jiang [1] **Gustavo Malkomes** [1] **Geoff Converse** [2] **Alyssa Shofner** [3] **Benjamin Moseley** [1] **Roman Garnett** [1]

Abstract

Active search is an active learning setting with the goal of identifying as many members of a given class as possible under a labeling budget. In this work, we first establish a theoretical hardness of active search, proving that no polynomial-time policy can achieve a constant factor approximation ratio with respect to the expected utility of the optimal policy. We also propose a novel, computationally efficient active search policy achieving exceptional performance on several real-world tasks. Our policy is nonmyopic, always considering the entire remaining search budget. It also automatically and dynamically balances exploration and exploitation consistent with the remaining budget, without relying on a parameter to control this tradeoff. We conduct experiments on diverse datasets from several domains: drug discovery, materials science, and a citation network. Our efficient nonmyopic policy recovers significantly more valuable points with the same budget than several alternatives from the literature, including myopic approximations to the optimal policy.

1. Introduction

In many real-world applications, the process of analyzing data incurs some cost; for example, obtaining a label may require a laborious experiment, a human action, or depletion of some other expensive resource. In these scenarios, carefully selecting data to label can often help us achieve our goals more efficiently than, e.g., random sampling. This is the motivation behind *active learning.*

Naturally, which data examples are the most useful to label might change according to our objective. Traditionally, much of the active learning literature has focused on training a model to have high generalization performance with few training examples. Here, we consider a special and atypical realization of active learning: the *active search* problem (Garnett et al., 2012). 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 is to design an policy to sequentially query points to find as many valuable points as possible under a labeling budget. 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 active search by appealing to Bayesian decision theory (Garnett et al., 2011; 2012). Garnett et al. (2012) 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 (Garnett et al., 2015). 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 paper 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 by Garnett et al. (2012) 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

---

[1]Washington University in St. Louis, St. Louis, MO, USA [2]Simpson College, Indianola, IA, USA [3]University of South Carolina, Columbia, SC, USA. Correspondence to: Shali Jiang <jiang.s@wustl.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

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.

2. Active Search and the Optimal Policy

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 $y \triangleq \mathbb{1}\{x \in \mathcal{R}\}$. 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 $\mathcal{D} \triangleq \big\{(x_i, y_i)\big\}$ through a simple utility:

$$u(\mathcal{D}) \triangleq \textstyle\sum_{y_i \in \mathcal{D}} y_i, \tag{1}$$

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 $\mathcal{D}_i \triangleq \big\{(x_j, y_j)\big\}_{j=1}^{i}$.

2.1. The Bayesian Optimal Policy

Following previous work, we consider the active search problem in the standard Bayesian 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:

$$x_i^* = \operatorname*{arg\,max}_{x_i \in \mathcal{X} \setminus \mathcal{D}_{i-1}} \mathbb{E}\big[u(\mathcal{D}_t) \mid x_i, \mathcal{D}_{i-1}\big]. \tag{2}$$

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

$$\begin{aligned}\mathbb{E}\big[u(\mathcal{D}_t) \mid x_t, \mathcal{D}_{t-1}\big] &= \textstyle\sum_{y_t} u(\mathcal{D}_t) \Pr(y_t \mid x_t, \mathcal{D}_{t-1}) \\ &= u(\mathcal{D}_{t-1}) + \Pr(y_t = 1 \mid x_t, \mathcal{D}_{t-1}).\end{aligned} \tag{3}$$

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 $i = t-1$.

$$\begin{aligned}\mathbb{E}\big[u(\mathcal{D}_t) \mid x_{t-1}, \mathcal{D}_{t-2}\big] = u(\mathcal{D}_{t-2}) &+ \\ \Pr(y_{t-1} = 1 \mid x_{t-1}, \mathcal{D}_{t-2}) &+ \\ \mathbb{E}_{y_{t-1}}\big[\max_{x_t} \Pr(y_t = 1 \mid x_t, \mathcal{D}_{t-1})\big]&.\end{aligned} \tag{4}$$

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). 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 (2) at time $i \leq t$ recursively as (Garnett et al., 2012):

$$\begin{aligned}\mathbb{E}\big[u(\mathcal{D}_t) \mid x_i, \mathcal{D}_{i-1}\big] = u(\mathcal{D}_{i-1}) &+ \\ \underbrace{\Pr(y_i = 1 \mid x_i, \mathcal{D}_{i-1})}_{\text{exploitation}, \, < 1} &+ \\ \underbrace{\mathbb{E}_{y_i}\Big[\max\nolimits_{x'} \mathbb{E}\big[u(\mathcal{D}_t \setminus \mathcal{D}_i) \mid x', \mathcal{D}_i\big]\Big]}_{\text{exploration}, \, < t-i}&.\end{aligned} \tag{5}$$

It is easy to show that the time complexity for computing Eq. (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) or (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 (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, Garnett et al. (2012) 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.

2.2. 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,[1] 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 the appendix. 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 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 (2) that is nonmyopic, computationally efficient, and shows impressive empirical performance.

[1]The optimal policy operates under these restrictions.

3. Efficient Nonmyopic Active Search

We have seen above how to myopically approximate the Bayesian optimal policy using an ℓ-step-lookahead approximate policy (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, 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, $\{x_{i+1}, x_{i+2}, \dots, x_t\}$, 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. The resulting approximation is

$$\max_{x'} \mathbb{E}\big[u(\mathcal{D}_t \backslash \mathcal{D}_i) \mid x', \mathcal{D}_i\big] \approx {\textstyle\sum'_{t-i}} \Pr(y = 1 \mid x, \mathcal{D}_i), \tag{6}$$

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:

$$\begin{aligned}\mathbb{E}\big[u(\mathcal{D}_t) \mid x_i, \mathcal{D}_{i-1}\big] \approx u(\mathcal{D}_{i-1}) &+ \\ \Pr(y_i = 1 \mid x_i, \mathcal{D}_{i-1}) &+ \\ \underbrace{\mathbb{E}_{y_i}\Big[{\textstyle\sum'_{t-i}} \Pr\big(y = 1 \mid x, \mathcal{D}_i\big)\Big]}_{\text{exploration}, \, < t-i}&.\end{aligned} \tag{7}$$

We will call this policy *efficient nonmyopic search* (ENS). As in the optimal policy, we can interpret (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 (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 (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 (7) is equivalent to a one-step maximization of (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.

3.1. Nonmyopic Behavior

To illustrate the nonmyopic behavior of our policy, we have adapted the toy example presented by Garnett et al. (2012). 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 (4) 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 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 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 considerable intensity, as well as some of the corners (Figure 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 1(d)). We can also see that the highest probability mass in Figure 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.

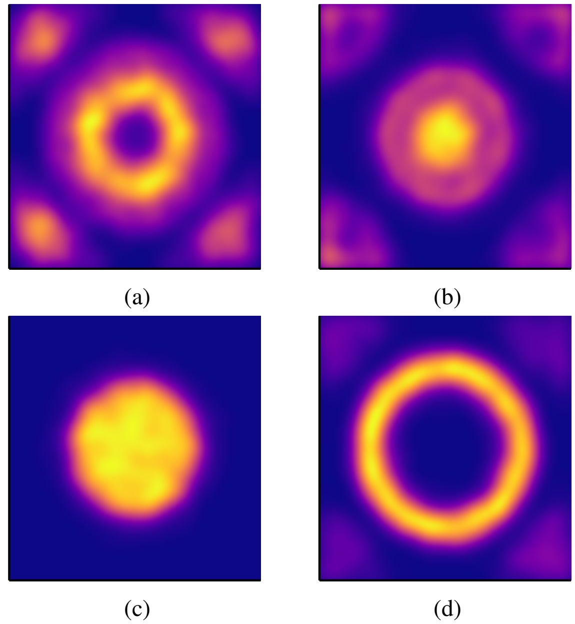

Figure 1: Kernel density estimates of the distribution of points chosen by ENS (top) and 2-step lookahead (bottom) during two different time intervals. The figures on the left show the kernel density estimates for the first 100 locations; the figures on the right, the last 100 chosen locations.

3.2. Implementation and Time Complexity

The complexity of our policy (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.

3.3. Pruning the Search Space

To further reduce the computational complexity, we can use a similar strategy as suggested by Garnett et al. (2012) to bound the score function (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 (7), denoted $f(x)$ here for simplicity, can be upper bounded by

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

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

$$\pi(x) < \frac{f(x^+) - \sum'_{t-i} \Pr(y' = 1 \mid x', \mathcal{D})}{1 + p^*(1, \mathcal{D})(t-i) - \sum'_{t-i} \Pr(y' = 1 \mid x', \mathcal{D})}. \tag{8}$$

Then, all points with current probability lower than the RHS of (8) can be removed from consideration. We will show empirically that a large fraction of points can often be pruned on massive datasets.

4. Related Work

Our method falls into the broader framework of active learning. The particular setting of finding elements of a valuable class is rather unusual in active learning, which typically considers the goal of training a high-fidelity model (Lewis & Gale, 1994). For an exhaustive introduction to active learning, we refer the reader to Settles (2010).

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 can never be played twice. Despite the difference, we note that our ENS policy is somewhat similar to the *knowledge gradient* policy introduced by Frazier et al. (2008).

Active search can be seen as a special case of *Bayesian optimization* (Brochu et al., 2010; Snoek et al., 2012) with binary observations and cumulative reward. Several nonmyopic policies have been proposed for Bayesian optimization in the regression setting (e.g., Ling et al. (2016)), and our method is spiritually similar to the recently propopsed GLASSES algorithm (González et al., 2016).

Vanchinathan et al. (2015) 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 (Srinivas et al., 2010). 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.

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. Sabbadin et al. (2007), 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 Wang et al. (2013), 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 (Garnett et al., 2011; Wang et al., 2013; Pfeiffer III et al., 2014; Ma et al., 2015a). 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. Ma et al. (2014) proposed the *active area search* problem, wherein a continuous function is sampled to discover regions with large mean value, and Ma et al. (2015b) 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.

5. Experiments

We implemented our approximation to the Bayesian optimal policy with the MATLAB active learning toolbox,[2] 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 (3–4). Note that Garnett et al. (2012) and Garnett et al. (2015) 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. 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-

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

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.

We further compare with the score function proposed by Wang et al. (2013), which we refer to as IMS:

$$\text{IMS}(x) = \Pr(y = 1 \mid x, \mathcal{D})\big(1 + \alpha\, \text{IM}(x)\big); \qquad (9)$$

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 (Wang et al., 2013) indicate that $\alpha = 10^{-4}$ performs well on average; we will fix this value in our experiments.

Finally, we have also considered the following UCB-style (Auer, 2002) score function: $\alpha(x, \mathcal{D}) = \pi + \gamma\sqrt{\pi(1-\pi)}$, 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 (Vanchinathan et al., 2015; Srinivas et al., 2010) 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, so we present these results in the appendix due to space.

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 (Garnett et al., 2012). This model, while being rather simple, shows reasonable generalization error, is nonparameteric, and can be rapidly updated given new training data, an important property in the active setting we consider here. We will also adopt the probability bound (8) for this model described in that work. 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.

5.1. CiteSeer^x Data

For our first real data experiment, we consider a subset of the CiteSeer^x citation network, first described in (Garnett et al., 2012). 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 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 vector for each paper is computed by performing graph principal component analysis (Fouss et al., 2007) 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 (Garnett et al., 2012). 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 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 $[19.80, 49.30]$.

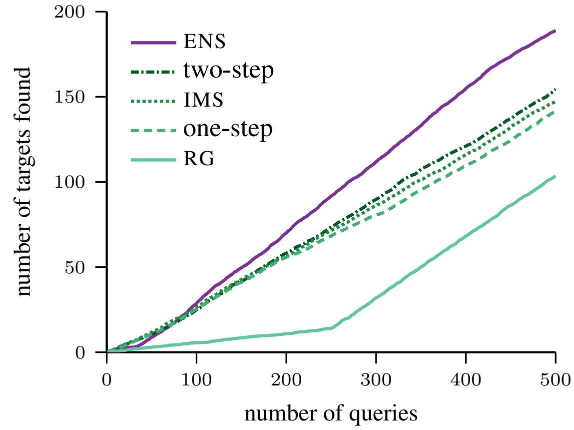

Figure 2: The learning curve of our policy and other baselines on the CiteSeer^x dataset.

Another interesting observation is that during the initial ∼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 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

Table 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**				

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.

5.2. 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., (Kawazoe et al., 1997; all)), an extension of the dataset from (Ward et al., 2016). Of these, 4 746 (∼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 1. 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.

5.3. Virtual Drug Screening Data

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 (Liu et al., 2007) 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 (Sterling & Irwin, 2015); 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 (Garnett et al., 2015) for more details.[3] 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 (Jasial et al., 2016), 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 (∼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 5.4.

Table 2 summarizes the results. First we notice that all

[3] We did not conduct experiments on the MACCS fingerprint. It was inferior in the findings of Garnett et al. (2015). A reviewer of (Jasial et al., 2016) noted that it is no longer used, due to clear underperformance compared to, e.g., ECFP4 and GpiDAPH3.

Table 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

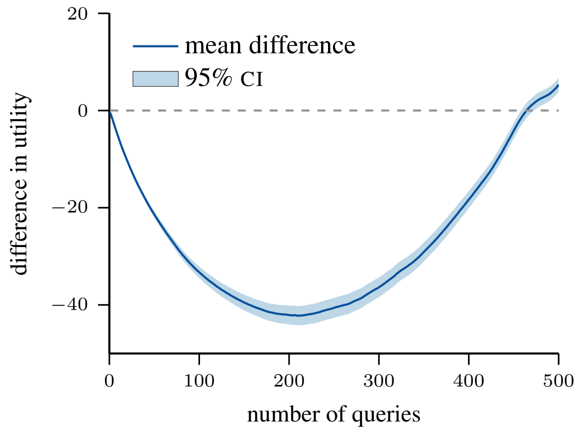

Figure 3: The average difference in cumulative targets found between ENS and the two-step policy, averaged over 120 activity classes and 20 experiments on the ECFP4 fingerprint.

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 (Garnett et al., 2015). 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 shows the mean difference in cumulative targets found between ENS and the two-step policy for the ECFP4 fingerprint. 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 200, our algorithm switches automatically to exploitation, outperforming the myopic policy significantly at termination. The mean difference curves for the other fingerprint is similar, and can be found in the appendix, along with the individual learning curves of the first six activity classes of ECFP4.

5.4. Effect of Pruning

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. On average about 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. See the appendix for detailed results.

6. Conclusion

In this paper 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.

Acknowledgments

We would like to thank Brendan Juba for insightful discussion. SJ, GM, and RG were supported by the National Science Foundation (NSF) under award number IIA–1355406. 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 by a Google Research Award, a Yahoo Research Award, and by NSF under award number CCF–1617724.

References

ASM Alloy Center Database. URL http://mio.asminternational.org/ac/.

Auer, Peter. Using Confidence Bounds for Exploitation–Exploration Trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

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

Fouss, Francois, Pirotte, Alain, Renders, Jean-Michel, and Saerens, Marco. 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.

Frazier, Peter I, Poweel, Warren B, and Dayanik, Savas. A Knowledge-Gradient Policy for Sequential Information Collection. SIAM *Journal on Control and Optimization*, 47(5):2410–2439, 2008.

Garnett, Roman, Krishnamurthy, Yamuna, Wang, Donghan, Schneider, Jeff, and Mann, Richard. Bayesian Optimal Active Search on Graphs. In *9th Workshop on Mining and Learning with Graphs*, 2011.

Garnett, Roman, Krishnamurthy, Yamuna, Xiong, Xuehan, Schneider, Jeff G., and Mann, Richard P. Bayesian Optimal Active Search and Surveying. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Garnett, Roman, Gärtner, Thomas, Vogt, Martin, and Bajorath, Jürgen. Introducing the 'active search' method for iterative virtual screening. *Journal of Computer-Aided Molecular Design*, 29(4):305–314, 2015.

González, Javier, Osborne, Michael, and Lawrence, Neil D. 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, pp. 790–799, 2016.

Jasial, Swarit, Hu, Ye, Vogt, Martin, and Bajorath, Jürgen. Activity-relevant similarity values for fingerprints and implications for similarity searching. *F1000Research*, 5(Chem Inf Sci):591, 2016.

Kawazoe, Yoshiyuki, Yu, Jing-Zhi, Tsai, An-Pang, and Masumoto, Tsuyoshi (eds.). *Nonequilibrium Phase Diagrams of Ternary Amorphous Alloys*, volume 37A of *Condensed Matter*. Springer–Verlag, 1997.

Lewis, David D. and Gale, William A. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International* ACM SIGIR *Conference on Research and Development in Information Retrieval*, pp. 3–12, 1994.

Ling, Chun Kai, Low, Kian Hsiang, and Jaillet, Patrick. Gaussian process planning with Lipschitz continuous reward functions: Towards unifying Bayesian optimization, active learning, and beyond. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 1860–1866, 2016.

Liu, Tiqing, Lin, Yuhmei, Wen, Xin, Jorissen, Robert N, and Gilson, Michael K. BindingDB: A web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic Acids Research*, 35(suppl 1): D198–D201, 2007.

Ma, Yifei, Garnett, Roman, and Schneider, Jeff. Active Area Search via Bayesian Quadrature. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, number 33 in Proceedings of Machine Learning Research, pp. 595–603, 2014.

Ma, Yifei, Huang, Tzu-Kuo, and Schneider, Jeff. Active Search and Bandits on Graphs using Sigma-Optimality. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pp. 542–551, 2015a.

Ma, Yifei, Sutherland, Dougal J., Garnett, Roman, and Schneider, Jeff. Active Pointillistic Pattern Search. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, number 38 in Proceedings of Machine Learning Research, pp. 672–680, 2015b.

Pfeiffer III, Joseph J, Neville, Jennifer, and Bennett, Paul N. Active Exploration in Networks: Using Probabilistic Relationships for Learning and Inference. In *Proceedings of the 23rd* ACM *International Conference on Information and Knowledge Management*, pp. 639–648, 2014.

Sabbadin, Régis, Lang, Jérôme, and Ravoanjanahary, Nasolo. Purely Epistemic Markov Decision Processes. In *Proceedings of the 22nd* AAAI *Conference on Artificial Intelligence*, pp. 1057–1062, 2007.

Settles, Burr. Active Learning Literature Survey. Technical report, University of Wisconsin–Madison, 2010. Computer Sciences Technical Report 1648.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems 25*, pp. 2951–2959, 2012.

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

Sterling, Teague and Irwin, John J. ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

Vanchinathan, Hastagiri P., Marfurt, Andreas, Robelin, Charles-Antoine, Kossmann, Donald, and Krause, Andreas. Discovering Valuable Items from Massive Data. In *Proceedings of the 21st* ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining*, pp. 1195–1204, 2015.

Wang, Xuezhi, Garnett, Roman, and Schneider, Jeff. Active Search on Graphs. In *Proceedings of the 19th* ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining*, pp. 731–738, 2013.

Ward, Logan, Agrawal, Ankit, Choudhary, Alok, and Wolverton, Christopher. A General-Purpose Machine Learning Framework for Predicting Properties of Inorganic Materials. 2016. arXiv preprint arXiv:1606.09551 [cond-mat.mtrl-sci].