Nonmyopic Multiclass Active Search with Diminishing Returns for Diverse Discovery

Quan Nguyen **Roman Garnett**

Washington University in St. Louis

Abstract

Active search is a setting in adaptive experimental design where we aim to uncover members of rare, valuable class(es) subject to a budget constraint. An important consideration in this problem is diversity among the discovered targets – in many applications, diverse discoveries offer more insight and may be preferable in downstream tasks. However, most existing active search policies either assume that all targets belong to a common positive class or encourage diversity via simple heuristics. We present a novel formulation of active search with multiple target classes, characterized by a utility function chosen from a flexible family whose members encourage diversity via a diminishing returns mechanism. We then study this problem under the Bayesian lens and prove a hardness result for approximating the optimal policy for arbitrary positive, increasing, and concave utility functions. Finally, we design an efficient, nonmyopic approximation to the optimal policy for this class of utilities and demonstrate its superior empirical performance in a variety of settings, including drug discovery.

1 Introduction

A theme underlying many real-world applications is the need to rapidly discover rare, valuable instances from massive databases in a budget-efficient manner. For example, both drug discovery and fraud detection entail searching through huge spaces of candidates for rare instances exhibiting desired properties: binding activity against a biological target for drug discovery, or fraudulent behavior for fraud detection. In both of these cases, labeling a given data point is expensive: synthesis and characterization in a laboratory for drug discovery, and human intervention (and possibly lost sales) for fraud detection. This cost of labeling rules out exhaustive scanning and raises a challenge in experimental design. The *active search* (AS) framework frames such tasks in terms of active learning, where one iteratively queries an expensive oracle to determine whether chosen data points are valuable. The goal of AS is to adaptively design queries for labeling in order to identify as many valuable data points as possible under a given budget.

AS has enjoyed a great deal of study (Garnett et al., 2012; Jiang et al., 2017, 2018, 2019; Nguyen et al., 2021), and strong theoretical results and efficient algorithms are known. Most relevant to this work, Jiang et al. (2017) established a strong hardness result for AS. Namely, *no* computationally tractable policy (i.e., one that runs in polynomial time with respect to its querying budget) can guarantee recovery within any constant factor of the (exponential-time) optimal policy in the worst case. This proof was by an explicit construction of arbitrarily hard problems. Nonetheless, Jiang et al. (2017) were able to develop an efficient, nonmyopic policy that achieves impressive empirical results.

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

After defining this problem, we study it through the lens of Bayesian decision theory. We begin by outlining how we

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

can capture the notion of diminishing returns in marginal discovery through an (arbitrary) positive, increasing, and concave utility function. We then extend the hardness of approximation result by Jiang et al. (2017) from the linear utility to this much larger family, demonstrating that search is fundamentally difficult for a broad range of natural utility functions. We then propose an approximation to the optimal policy for problems in this this class that is both computationally efficient and nonmyopic. We show that the resulting algorithms effectively encourage diversity among discoveries, and, similar to nonmyopic policies from previous work, leverage budget-awareness to dynamically balance exploration and exploitation. We demonstrate the superior empirical performance of our approach through an exhaustive series of experiments, including in a challenging drug discovery setting. Across the board, our proposed policy recovers both better balanced and richer data sets than a suite of strong baselines.

Figure 1: Discoveries by region relative to a uniform target distribution by the state-of-the-art policy ENS, in a toy search problem where points within the visible regions in the unit square are considered search targets. ENS over-samples the center region, the most common and easiest-to-identify target class, and collects a highly unbalanced data set.

2 Problem Definition

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

Our high-level goal is to design a policy that decides which elements of $\mathcal{X}$ should be queried in order to uncover as many targets as possible. Preferences over different data sets (experimental outcomes) are expressed via a *utility function;* previous work (Garnett et al., 2012; Jiang et al., 2017) has used a linear utility in the binary setting $C = 2$:

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

which effectively groups all targets in a common positive class and assigns equal reward to each discovery. In many practical scenarios, however, once a target class has been thoroughly investigated, the marginal utility of finding yet more examples decreases and we would prefer to either expand a rarely sampled class or discover a new one. For example, in drug discovery – one of the main motivating applications for AS – screening procedures optimized for hit rate tend to propose very structurally similar compounds and lead to an overall decline in usefulness of these discoveries downstream (Galloway et al., 2010). This has lead to efforts to artificially encourage diversity when generating new screening experiments, as a way to induce the desired search behavior (Benhenda, 2017; Pereira et al., 2021).

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

$$u(\mathcal{D}) \triangleq \sum_{c>1} f_c\Big(\sum_{y \in \mathcal{D}} \mathbb{1}\{y = c\}\Big) = \sum_{c>1} f_c(m_c), \tag{2}$$

where f_c is a positive, increasing, and concave function quantifying our reward given the number of found targets from a given class c. The term m_c denotes the number of targets of class c in data set $\mathcal{D}$; we also use $m_{c,t}$ to denote the corresponding number in $\mathcal{D}_t$ at time t. Any utility encoding decreasing marginal gains (that is, concave) is an appropriate choice for this setting, and we will see later in Sect. 3 that the key element of our algorithm is valid with any concave utility. In our experiments, we use the logarithmic function $f_c(x) = \log(x + 1)$ (which has seen a wide range of applications, namely modeling utility of wealth (Bernoulli, 1954) or production output (Pemberton and Rau, 2015) in economics) as a reasonable default with which to generate our main results. We also present results showing that our methods generalize well to other possibilities such as the square root utility $f_c(x) = \sqrt{x}$.

Under this model, the marginal gain of an additional discovery decreases with the size of the corresponding class in $\mathcal{D}$. Consider a toy problem illustrated in Fig. 1, where we wish to search for points close to the center and corners of the

unit square, each representing a target class. The state-of-the-art ENS policy (Jiang et al., 2017), which uses a linear utility, recovers many targets but over-exploits the center region. This is undesirable behavior when we would prefer to have balanced discoveries across all classes; ideally, we would like to achieve a uniform target distribution (an equal number of hits across all classes for maximum diversity), where all five regions in the plot are transparent. We will develop policies that can achieves such balance in the next section.

2.1 The Bayesian Optimal Policy

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

$$x^*_{t+1} = \operatorname*{arg\,max}_{x_{t+1} \in \mathcal{X} \setminus \mathcal{D}_t} \mathbb{E}\big[u(\mathcal{D}_T) \mid x_{t+1}, \mathcal{D}_t\big]. \tag{3}$$

This expected optimal utility may be computed using backward induction (Bellman, 1957). In the base case where $t = T-1$ and we are faced with the very last query x_T,

$$\mathbb{E}\big[u(\mathcal{D}_T) \mid x_T, \mathcal{D}_{T-1}\big] = \sum_{c \in [C]} u(\mathcal{D}_T)\, p_c(x_T \mid \mathcal{D}_{T-1}). \tag{4}$$

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

$$\Delta(x_T \mid \mathcal{D}_{T-1}) \triangleq \sum_{c>1} p_c(x_T \mid \mathcal{D}_{T-1}) \times \Big[f_c(m_{c,T-1}+1) - f_c(m_{c,T-1})\Big]. \tag{5}$$

For each class c, this quantity not only increases as a function of the positive probability p_c, but also decreases as a function of the number of targets already found in that class. Therefore, even at this very last step, the optimal decision balances between hit probability and discovery/extension of a rare class. When more than one query remains in our budget, the expected optimal utility in Eq. (3) expands into

$$\mathbb{E}\big[u(\mathcal{D}_T) \mid x_{t+1}, \mathcal{D}_t\big] = u(\mathcal{D}_t) + \Delta(x_{t+1} \mid \mathcal{D}_t) + \mathbb{E}_{y_{t+1}}\Big[\max_{x_{t+2}} \mathbb{E}\big[u(\mathcal{D}_T) \mid x_{t+2}, \mathcal{D}_{t+1}\big] - u(\mathcal{D}_{t+1})\Big], \tag{6}$$

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

A potential solution to this problem is to limit the lookahead horizon by pretending that ℓ is small, thus myopically approximating the optimal policy. The simplest form of this is to set $\ell = 1$ and greedily optimize for the one-step expected marginal utility gain in Eq. (5). We refer to the resulting policy as the *one-step* policy. Since our utility function has elements with diminishing returns, a question naturally arises as to whether the results from the submodularity (Krause and Guestrin, 2007) and adaptive submodularity (Golovin and Krause, 2011) literature apply here, and whether the greedy strategy of the one-step policy could approximate the optimal policy well. In the next subsection, we present the perhaps surprising result that *no* polynomial-time policy can approximate the optimal policy by any constant factor.

2.2 Hardness of Approximation

Assuming access to a unit-cost conditional probability $p_c(x \mid \mathcal{D})$ for any point $x \in \mathcal{X}$ and data set $\mathcal{D}$, we obtain the same hardness result of Jiang et al. (2017) for the broad range of utility functions considered here:

Theorem 2.1. *There is no polynomial-time policy providing any constant factor approximation to the optimal expected utility in the worst case.*

Our proof strategy follows that of Jiang et al. (2017). We construct a family of hard problem instances, where in each instance a secret set of points encodes the location of a larger "treasure" of targets. The probability of discovering this treasure is extremely small without observing the secret set first, which in itself is vanishingly unlikely to happen in polynomial time. Further, the average hit rate outside of the treasure set is vanishingly low, making it infeasible to compete with the optimal policy. Remarkably, we can construct such hard problem instances for any utility that is positive, increasing, unbounded, and concave in the number of discoveries in each class. The complete proof is included in Appx. A.

Despite this negative result, we hope to design *empirically* effective policies. Previous work has demonstrated that nonmyopic policies offer both theoretical and empirical benefits when working with the linear utility function, and that budget-awareness is in particular can be especially bene-

ficial for a policy to effectively balance exploration and exploitation. In the next section, we propose an efficient, nonmyopic approximation to the optimal policy for the class of utility functions we consider here, which we will show later also improves practical performance.

3 Efficient Nonmyopic Approximation

We propose a batch rollout approximation to the optimal policy similar to the ENS algorithm for binary AS (Jiang et al., 2017) and the GLASSES algorithm for Bayesian optimization (González et al., 2016). The key idea is to assume that after a proposed query in the current iteration, all remaining budget will be spent *simultaneously* on a single batch of queries exhausting the budget. This assumption simplifies the decision tree we must analyze, reducing its depth to 2 while expanding the branching factor of the last layer. Under the linear utility model, the expected marginal utility of a final batch of queries conveniently decomposes into a sum of positive probabilities of individual batch members. The optimal final batch therefore consists of the points with the highest probabilities, which may be computed efficiently. By matching the size of the following batch to the number of queries remaining, we can effectively account for our remaining budget when computing the expected utility of a given putative query. Unfortunately, the linear decomposition enabling rapid computation in ENS does not hold in our setting due to our nonlinear utility (2), and designing an effective batch policy requires more care.

3.1 Making a Batch of Queries

We first temporarily consider the subproblem of designing a batch of b queries X given a data set $\mathcal{D}$ to maximize the expected utility of the combined observation set, i.e., $\mathbb{E}_Y\left[u\left(\mathcal{D}\cup X, Y\right)\right]$, where the expectation is taken over Y,[1] the labels of X. As labels may be conditionally dependent and the utility function u is not linear, exact computation of this expectation requires iterating over all C^b realizations of the label set Y. This is infeasible unless b is very small, and represents the primary challenge we must overcome.

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

$$\mathbb{E}_Y\left[u\left(\mathcal{D}\cup X, Y\right)\right] = \sum_{c>1}\mathbb{E}_Y\left[f_c\left(m'_c\right)\right], \tag{7}$$

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

$$\sum_{c>1}\mathbb{E}_Y\left[f_c\left(m'_c\right)\right] \le \sum_{c>1} f_c\Big(\mathbb{E}_Y\left[m'_c\right]\Big). \tag{8}$$

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

$$\mathbb{E}_Y\left[m'_c\right] = m_c + \sum_{x\in X} p_c\left(x \mid \mathcal{D}\right). \tag{9}$$

We then upper-bound the overall expected utility:

$$\begin{aligned}\mathbb{E}_Y\left[u\left(\mathcal{D}\cup X, Y\right)\right] &\le \overline{u}(X)\\ &\triangleq \sum_{c>1} f_c\Big(m_c + \sum_{x\in X} p_c(x \mid \mathcal{D})\Big).\end{aligned} \tag{10}$$

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

Our goal now is to find the batch X that maximizes $\overline{u}$, as an approximation to the batch maximizing the expected one-step utility. Naïvely maximizing $\overline{u}$ requires iterating over $\binom{n}{b}$ candidate batches to compute the corresponding score. However, we note that this score is a sum of concave, increasing functions, which are monotone submodular, and therefore $\overline{u}$ is a monotone submodular function itself. We thus opt to *greedily* optimize $\overline{u}$ by sequentially maximizing the pointwise marginal gain; the resulting batch provides a $(1-1/e)$-approximation for the optimal batch (Nemhauser et al., 1978; Krause and Guestrin, 2007). We briefly remark on the nature of the batches resulting from this greedy procedure, which naturally encourages batch members to be diverse in their likely labels: once a point having a high p_c is added to X, others with high $p_{c'}$ for another target will be prioritized during the next selection. This is a desideratum of a batch policy when seeking to encourage diversity, indicating the output of the algorithm is a reasonable approximation of the optimal batch. Further, when $b = 1$ (that is, at the second-to-last iteration), this procedure makes the true expected-case optimal decision – a reassuring feature.

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

Algorithm 1 Diversity-aware active search (DAS)

inputs observations $\mathcal{D}_t$, remaining budget $T - t$
returns x^*_{t+1} maximizing the score in Eq. (11)
for $x_{t+1} \in \mathcal{X} \setminus \mathcal{D}_t$ **do**
  for $y_{t+1} \leftarrow 1$ **to** C **do**
    $\alpha\left(x_{t+1} \mid y_{t+1}\right) \leftarrow \overline{u}\left(X \mid \mathcal{D}_t \cup \{(x_{t+1}, y_{t+1})\}\right)$
  end for
  $\alpha(x_{t+1}) \leftarrow \sum_c p_c(x_{t+1})\, \alpha\left(x_{t+1} \mid c\right)$
end for
$x^*_{t+1} \leftarrow \arg\max_{x_{t+1}} \alpha(x_{t+1})$

3.2 Completing the Algorithm

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

$$\alpha(x_{t+1}) = \mathbb{E}_{y_{t+1}}\left[\overline{u}\left(X\right) \mid x_{t+1}, \mathcal{D}_t\right], \tag{11}$$

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

As mentioned, with the lookahead batch construction simulating future queries, DAS accounts for not only the immediate reward but also the impact of the chosen query on future rewards. Additionally, the latter quantity naturally decreases as a function of the remaining budget b, allowing our policy to be budget-aware and dynamically balance exploitation and exploration without any tradeoff parameter. We briefly demonstrate the benefits of our approach by continuing with the example previously seen in Fig. 1, where ENS, in seeking to maximize only recovery, collected highly unbalanced data sets. Fig. 2 shows the results of the one-step policy and DAS under the same setting. Compared to ENS, one-step distributes its queries more equally, but it is our proposed policy DAS that constructs the most diverse data set.

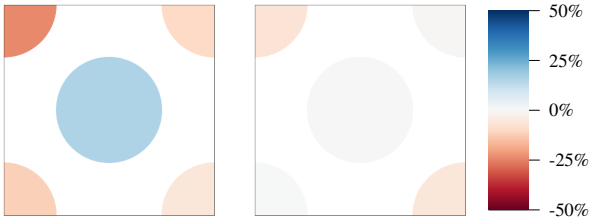
3.3 Implementation

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

We now describe the k-NN model used in our experiments, introduced by Garnett et al. (2012) in the binary setting. The idea is to use the proportion of class-c members among the observed nearest neighbors of a given point x as the posterior marginal probability $p_c\left(x \mid \mathcal{D}\right)$. Formally, denote the set of nearest neighbors of x as $\text{NN}(x)$ and the (potentially empty) subset of labeled neighbors as $\text{LNN}(x) \subseteq \text{NN}(x)$. Then, the posterior probability of x belonging to class c is

$$p_c\left(x \mid \mathcal{D}\right) = \frac{\gamma_c + \sum_{x' \in \text{LNN}(x)} \mathbb{1}\{y' = c\}}{1 + |\text{LNN}(x)|}, \tag{12}$$

where each $\gamma_c \in (0, 1)$ is a hyperparameter acting as a "pseudocount", or our prior belief about the prevalence of class c (since $p_c\left(x \mid \mathcal{D}\right) = \gamma_c$ if $\text{LNN}(x) = \varnothing$). We detail our choice for this hyperparameter in Appx. C.



(a) One-step (b) DAS (ours)

Figure 2: Discoveries by region relative to a uniform target distribution by the one-step policy and our proposal DAS, in the problem visualized in Fig. 1. One-step distributes queries more equally than the previously seen ENS; however, center points are still over-represented. DAS constructs more diverse data sets and finds more rare corner targets.

The k-NN achieves reasonable generalization error in practice (in the sparsely labeled setting we are considering here), and can be rapidly updated given a new observation, which is a valuable feature with respect to our method. Further, the k-NN only uses the similarity matrix for $\mathcal{X}$, whose calculation only needs to be done once and can be accelerated by modern similarity search libraries such as Faiss (Johnson et al., 2019). This model also allows us to employ a branch-and-bound pruning strategy that speeds up the search for the query maximizing the score α at each iteration; we detail this procedure in Appx. B.

In the event where pruning is unsuccessful, we can subsample the space (e.g., subject to a user-specified cardinality constraint) and conduct the current search within the sampled pool. This technique is extensively explored in Mirzasoleiman et al. (2015) and used by Nguyen et al. (2021).

4 Related Work

Active search (Garnett et al., 2012) is a variant of active learning (AL) (Settles, 2009) where we aim not to learn an accurate model, but to collect members of rare and valuable classes. Previous work has explored AS under a wide range of settings, such as when the goal is to hit a targeted number of positives as quickly/cheaply as possible (Warmuth et al., 2002, 2003; Jiang et al., 2019), when queries are made in batches (Jiang et al., 2018), or with multifidelity oracles (Nguyen et al., 2021). These studies all assumed there is

only one target class, and collecting a target constitutes a constant reward. Ours is the first to our knowledge to tackle multiclass AS.

Diversity as an objective has enjoyed great interest from the broader AL community. A common approach is to modify a typical AL acquisition function to encourage diversity in the resulting queries. For example, Gu et al. (2015) and Yang et al. (2015) encouraged diversity by incorporating dissimilarity terms (computed via an RBF kernel) into uncertainty sampling schemes. Brinker (2003) used the angles between the hyperplanes induced by adding new points to the training set of an SVM, and Zhdanov (2019) considered the minimum distance between any pair of labeled points. Others have employed coreset-based strategies (Sener and Savarese, 2018; Agarwal et al., 2020) to identify a set of diverse representative points. Another popular strategy is to partition a given data set into different groups (e.g., using a clustering algorithm) and inspect the groups in a round-robin manner (Madani et al., 2004; Lin et al., 2018; Ma et al., 2020; Citovsky et al., 2021). We will apply a round-robin heuristic to a number of benchmarks in our experiments.

Vanchinathan et al. (2015) was motivated by a similar problem of uncovering a diverse, valuable subset. Theirs is a regression setting in which diversity is measured in the feature space – via the logdet of the Gram matrix of the collected data. The proposed policy is myopic, maximizing the expected one-step marginal gain in a weighted sum of reward and diversity. He and Carbonell (2007) studied a related problem where the objective is to detect at least one instance of each rare target class as quickly as possible. By assuming the target classes are highly concentrated, they design a policy that optimizes the difference in local density between a given point and its nearest neighbors, which is effective at identifying targets on the boundary. Malkomes et al. (2021) considered the *constraint active search* problem, in which they seek to find a diverse set of points satisfying a set of constraints. The authors propose maximizing the expected improvement in a coverage measure given a new observation. We include these policies as AL baselines in our experiments.

Closest to the motivation of our work, a line of research (Kothawade et al., 2021, 2022) has explored a unified AL framework for querying rare, diverse subsets of a large pool using submodular information measures. Specifically, the neural network-based SIMILAR algorithm (Kothawade et al., 2021) consists of AL policies that can tackle problematic yet realistic learning scenarios such as imbalance in the training data, rare classes, out-of-distribution test data, and redundancy. While it can be used for active search, SIMILAR is not designed to specifically target discovery tasks; in Appx. E, we present results comparing the instantiation of SIMILAR for rare class detection with our method, noting a slightly lower performance from SIMILAR. Further, our diversity-aware active search framework allows the utility function to be adjusted by the user to dynamically balance between discovery and diversity, a valuable feature in many discovery tasks.

Diversity has also been explored in the related task of Bayesian optimization (Garnett, 2022). A common approach is to incorporate a determinantal point process (Kulesza and Taskar, 2012) to induce diversity in the feature space (Wang et al., 2018; Nava et al., 2021). In the multiobjective setting, many policies leverage the diversity of their collected data in the Pareto space to design queries (Gupta et al., 2018; Shu et al., 2020; Lukovic et al., 2020).

5 Experiments

We performed a series of experiments to evaluate the empirical performance of our policy DAS. As baselines, we considered related active learning/search algorithms (He and Carbonell, 2007; Vanchinathan et al., 2015; Malkomes et al., 2021) (see Sect. 4), as well as the one-step lookahead policy, which greedily maximizes the marginal utility at each iteration. Another baseline was ENS (Jiang et al., 2017), the state-of-the-art for binary AS, where we lumped all targets into a single positive class.

We also considered a family of policies that design queries in a *round-robin* (RR) manner. In each iteration t, we choose a target class c_t and seek to make a discovery for this target class. A round-robin policy then continually rotates the target class among the positive classes throughout the search, devoting an equal amount of resources to each class. The first of these policies is RR-greedy, which for a given class index c_t queries the point that maximizes the probability p_{c_t}. Another round-robin baseline is RR-UCB, which maximizes an upper confidence bound score (Auer, 2002) corresponding to c_t: $p_{c_t} + \beta\sqrt{p_{c_t}(1-p_{c_t})}$. Here β is a hyperparameter trading off exploitation (class membership probability) and uncertainty (as measured by the standard deviation of the binary indicator $[y = c_t]$). We evaluate this policy for $\beta \in \{0.1, 0.3, 1, 3, 10\}$ and report the result of the best performing value of β, denoted β^* in our results (Tab. 1). Finally, we consider RR-ENS, which applies the ENS heuristic to the subproblem of finding positives in the target class c_t. The remaining budget is equally allocated among the positive classes, and we adjust the remaining budget when constructing lookahead batches accordingly.

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

Table 1: Logarithmic utility and standard errors across 20 repeated experiments for each setting. C is the total number of unique classes in the search space. The best performance in each column is highlighted in **bold**; policies not significantly worse than the best (according to a two-sided paired t-test with a significance level of $\alpha = 0.05$) are in *blue italics*.

	Fashion-MNIST		Photoswitch	CiteSeer$^{\text{x}}$		Drug discovery		
	easy	hard		$C = 5$	$C = 10$	$C = 5$	$C = 10$	$C = 15$
He and Carbonell (2007)	3.99 (0.16)	3.95 (0.16)	12.65 (0.22)	11.30 (0.11)	24.61 (0.14)	3.49 (0.23)	7.19 (0.27)	10.35 (0.35)
Vanchinathan et al. (2015)	9.47 (0.50)	6.99 (0.51)	7.16 (0.25)	16.28 (0.06)	31.95 (0.49)	10.19 (0.70)	18.67 (0.88)	25.51 (1.40)
Malkomes et al. (2021)	11.31 (0.34)	10.93 (0.47)	6.39 (0.03)	11.28 (0.01)	22.16 (0.01)	7.41 (0.31)	13.77 (0.56)	19.60 (0.66)
ENS	10.48 (0.29)	10.91 (0.14)	5.04 (0.21)	16.57 (0.08)	32.54 (0.50)	10.29 (0.68)	13.79 (0.85)	17.00 (1.15)
RR-greedy	11.41 (0.36)	10.41 (0.44)	11.70 (0.30)	16.66 (0.14)	33.08 (0.13)	10.87 (0.82)	18.66 (1.13)	24.87 (0.95)
RR-UCB	11.51 (0.37) ($\beta^* = 1$)	11.28 (0.47) ($\beta^* = 1$)	11.70 (0.30) ($\beta^* = 3$)	16.68 (0.12) ($\beta^* = 1$)	33.22 (0.13) ($\beta^* = 3$)	11.60 (0.90) ($\beta^* = 3$)	19.27 (1.13) ($\beta^* = 3$)	26.45 (0.96) ($\beta^* = 10$)
RR-ENS	*13.97 (0.09)*	**12.78 (0.31)**	12.54 (0.11)	**17.47 (0.11)**	33.78 (0.16)	10.56 (0.72)	17.83 (0.66)	23.58 (0.88)
One-step	11.83 (0.39)	11.57 (0.47)	8.67 (0.13)	16.77 (0.13)	34.01 (0.13)	11.68 (0.92)	19.06 (0.98)	27.40 (1.11)
DAS (ours)	**14.02 (0.06)**	*12.38 (0.30)*	**13.85 (0.27)**	*17.37 (0.09)*	**34.45 (0.11)**	**13.34 (0.79)**	**23.39 (0.83)**	**31.48 (1.25)**

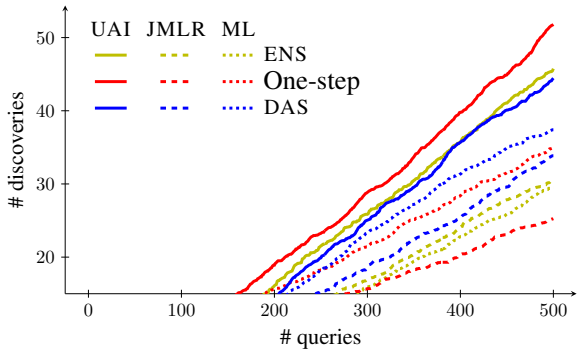


Figure 3: Average count of least-common papers found by different policies in the CiteSeer$^{\text{x}}$ $C = 10$ experiments. DAS finds more members of rare classes.

which we briefly describe below. More details on these data sets are included in Appx. C.

Product recommendation. Our first task uses the Fashion-MNIST data set (Xiao et al., 2017) of fashion item images to simulate a product recommendation setting. Here, we assume a user is looking for specific classes of fashion articles while shopping online. To simulate two different users (corresponding to two search problems), we select specific classes to be the products each user is looking for. We sub-sample these classes uniformly at random, making the positives harder to uncover; the hit rate of a random policy is roughly 2%. While the k-NN model has been found to perform well on the remaining data sets in previous studies (Garnett et al., 2012; Jiang et al., 2017; Mukadum et al., 2021), with this data, we can select targets that are either easy or difficult to identify with the k-NN, to simulate different levels of search difficulty and study the effect of the quality of the k-NN on the performance of our methods. To this end, we measure the predictive performance of the k-NN and split the two search problems into an "easy" one, where the model scores highly on precision-at-k metrics (particularly important in AS), and a "hard" one, where the k-NN scores lower. More details are included in Appx. C.

Photoswitch discovery. We also consider the task introduced by Mukadum et al. (2021) of searching for photoswitches (molecules that change their properties upon irradiation) in chemical databases that exhibit both desirable light absorbance and long half-lives. Roughly 36% of the molecules in the search space are targets. In their study, the authors partitioned the points into 29 groups by their respective substructures, thus defining a multiclass AS problem with $C = 30$ (a negative class and 29 positive classes). As this data set is smaller in size, we set the budget $T = 100$.

The CiteSeer$^{\text{x}}$ data set. We use the CiteSeer$^{\text{x}}$ citation network data (Garnett et al., 2012), which contains papers published at popular computer science conferences and journals. The label of each paper is its publication venue, and our targets are machine learning and artificial intelligence proceedings. We conduct two sets of experiments with different numbers of classes C. For $C = 5$, we select papers from NeurIPS, ICML, UAI, and JMLR as our four target classes (roughly a 14% hit rate). For $C = 10$, we further include IJCAI, AAAI, JAIR, *Artificial Intelligence,* and *Machine Learning* (ML) as targets, yielding a 31% hit rate.

Drug discovery. Finally, we experiment with a drug discovery task using a massive chemoinformatic data set. The goal is to identify chemical compounds that exhibit selective binding activity to a given protein. The data set consists of 120 such activity classes from BindingDB (Liu et al., 2007). For each class, there are a small number of compounds with significant binding activity – these are our search targets. In each experiment for a given $C \in \{5, 10, 15\}$, we select $(C - 1)$ of the 120 classes uniformly at random without replacement to form the targets. They are then combined with 100 000 "drug-like" entries in the ZINC database (Sterling and Irwin, 2015), which serve as the negatives, to make up our search space. We note that each of these active classes

Table 2: Average search utility and standard errors under various utility functions. C is the total number of unique classes in the search space. The best performance in each column is highlighted in **bold**; policies that are not significantly worse than the best (according to a two-sided paired t-test with a significance level of $\alpha = 0.05$) are in *blue italics*.

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

has a unique structure and behavior, and combining multiple classes in one AS problem makes ours a challenging task. Here, the average prevalence of a target is 0.2%; the hit rates are 1%, 2%, and 3% for $C = 5$, 10, and 15, respectively.

Discussion. We report the performance of the policies, quantified by the logarithmic utility function, in Tab. 1. Overall, DAS either is the winner or does not perform significantly worse than the best policy. This consistent performance highlights the benefits of our nonmyopic, budget-aware approach. Under Fashion-MNIST, most methods work better on the easy problem than on the hard problem, showing the importance of the predictive model in AS. That said, our method remains competitive even under the harder setting. Inspecting the optimal values for RR-UCB's tradeoff parameter β^* in the CiteSeer$^{\text{x}}$ and drug discovery experiments, we notice a natural trend: as C increases, so does the need for exploration, and larger values of β are thus selected.

To illustrate DAS is effective at constructing *diverse* observations, we show in Fig. 3 (whose y-axis is truncated for clarity) the average numbers of discoveries by the best three policies under the CiteSeer$^{\text{x}}$ $C = 10$ experiments for the three rarest classes: UAI, JMLR, and ML. Here, DAS successfully finds more targets from the rarer classes of JMLR and ML, with a better balance among these classes as well.

By design, DAS is always aware of its remaining budget during search and therefore dynamically balances exploration and exploitation. We demonstrate this with the difference in the cumulative reward of DAS vs. one-step under the drug discovery $C = 10$ setting in Fig. 4. DAS collects fewer rewards in the beginning while exploring the space. As the search progresses, the policy transitions to more exploitation and ultimately outperforms the myopic one-step.

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

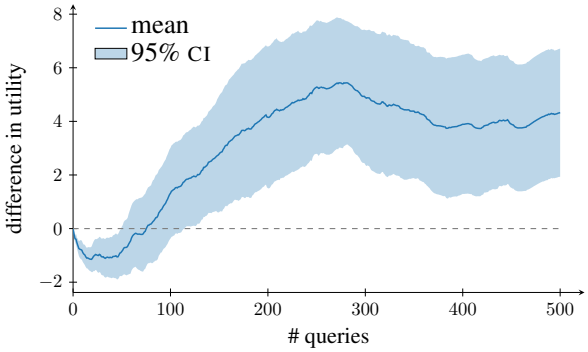


Figure 4: Difference in cumulative reward between DAS and one-step across the drug discovery $C = 10$ experiments. DAS dynamically balances exploration and exploitation.

As for robustness against utility misspecification, we look for any performance drop when DAS is evaluated under a utility different from what the policy uses during search. First, we classify the variants of DAS by the utility they use: (1) the version optimizing the logarithmic utility shown in Tab. 1 is denoted by DAS_{log}, (2) the version using the square root utility is DAS_{sqrt}, and (3) the version with the linear utility, $\text{DAS}_{\text{linear}}$, reduces to ENS. We then evaluate these policies using all three utility functions. We also include the one-step policy optimizing the correct utility in the results in Tab. 2. Overall, each DAS variant is competitive under the

correct utility, always outperforming the corresponding one-step counterpart. Crucially, both concave variants, $\text{DAS}_{\log}$ and DAS_{sqrt}, perform well "cross-utility" in each other's setting, even outperforming *the one-step policy with the correctly specified objective.* Thus there is merit in adopting DAS even when there may be uncertainty regarding the nuances of the user's "true" utility function. We hypothesize this is because the concave utilities are similar in behavior – both encourage diversity in the collected labels and are optimized by balanced data sets – and a policy effective under one utility is likely to perform well under the other. The linear utility, on the other hand, does not exhibit diminishing returns and behaves differently from the others. Utility misspecification here is thus more costly: $\text{DAS}_{\text{linear}}$ is not competitive under concave utilities, and neither are the concave variants under linear utility.

6 Conclusion

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

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

Acknowledgements

We thank the anonymous reviewers for their feedback during the review stage. This work was supported by the National Science Foundation (NSF) under award numbers OAC–1940224, IIS–1845434, and OAC–2118201.

References

Sharat Agarwal, Himanshu Arora, Saket Anand, and Chetan Arora. Contextual Diversity for Active Learning. In *European Conference on Computer Vision*, pages 137–153. Springer, 2020.

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

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Mostapha Benhenda. ChemGAN challenge for drug discovery: can AI reproduce natural chemical diversity? 2017. arXiv preprint arXiv:1708.08227 [stat.ML].

Daniel Bernoulli. *Exposition of a New Theory on the Measurement of Risk*. Econometrica, 1954.

Klaus Brinker. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of the 20th International Conference on Machine Learning*, pages 59–66, 2003.

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch Active Learning at Scale. *Advances in Neural Information Processing Systems 34*, 2021.

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.

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

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

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

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.

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

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

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

Jingrui He and Jaime Carbonell. Nearest-Neighbor-Based Active Learning for Rare Category Detection. In *Ad-*

vances in Neural Information Processing Systems 20, pages 633–640, 2007.

Shali Jiang, Gustavo Malkomes, Geoff Converse, Alyssa Shofner, Benjamin Moseley, and Roman Garnett. Efficient Nonmyopic Active Search. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1714–1723, 2017.

Shali Jiang, Gustavo Malkomes, Matthew Abbott, Benjamin Moseley, and Roman Garnett. Efficient nonmyopic batch active search. In *Advances in Neural Information Processing Systems 31*, pages 1099–1109, 2018.

Shali Jiang, Roman Garnett, and Benjamin Moseley. Cost effective active search. In *Advances in Neural Information Processing Systems 32*, pages 4880–4889, 2019.

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

Suraj Kothawade, Nathan Beck, Krishnateja Killamsetty, and Rishabh Iyer. SIMILAR: Submodular Information Measures Based Active Learning In Realistic Scenarios. *Advances in Neural Information Processing Systems 34*, pages 18685–18697, 2021.

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

Andreas Krause and Carlos Guestrin. Near-optimal Observation Selection using Submodular Functions. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, pages 1650–1654, 2007.

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

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

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.

Mina Konakovic Lukovic, Yunsheng Tian, and Wojciech Matusik. Diversity-Guided Multi-Objective Bayesian Optimization With Batch Evaluations. *Advances in Neural Information Processing Systems 33*, 33, 2020.

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

Omid Madani, Daniel J Lizotte, and Russell Greiner. Active Model Selection. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 357–365, 2004.

Gustavo Malkomes, Bolong Cheng, Eric H Lee, and Mike Mccourt. Beyond the Pareto Efficient Frontier: Constraint Active Search for Multiobjective Experimental Design. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7423–7434, 2021.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. 2018. arXiv preprint arXiv:1802.03426 [stat.ML].

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier Than Lazy Greedy. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Fatemah Mukadum, Quan Nguyen, Daniel M. Adrion, Gabriel Appleby, Rui Chen, Haley Dang, Remco Chang, Roman Garnett, and Steven A. Lopez. Efficient Discovery of Visible Light-Activated Azoarene Photoswitches with Long Half-Lives Using Active Search. *Journal of Chemical Information and Modeling*, 61(11):5524–5534, 2021.

Elvis Nava, Mojmír Mutnỳ, and Andreas Krause. Diversified Sampling for Batched Bayesian Optimization with Determinantal Point Processes. 2021. arXiv preprint arXiv:2110.11665 [cs.LG].

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.

Quan Nguyen, Arghavan Modiri, and Roman Garnett. Nonmyopic Multifidelity Acitve Search. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8109–8118, 2021.

Malcolm Pemberton and Nicholas Rau. *Mathematics for Economists: An Introductory Textbook*. Oxford University Press, 2015.

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

David Rogers and Mathew Hahn. Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Burr Settles. Active Learning Literature Survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

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

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

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

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

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.

Manfred KK Warmuth, Gunnar Rätsch, Michael Mathieson, Jun Liao, and Christian Lemmen. Active Learning in the Drug Discovery Process. In *Advances in Neural Information Processing Systems 15*, pages 1449–1456, 2002.

Peter Willett, John M Barnard, and Geoffrey M Downs. Chemical Similarity Searching. *Journal of Chemical Information and Computer Sciences*, 38(6):983–996, 1998.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017. arXiv preprint arXiv:1708.07747 [cs.LG].

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

Fedor Zhdanov. Diverse mini-batch Active Learning. 2019. arXiv preprint arXiv:1901.05954 [cs.LG].

A Hardness of Approximation

We present the proof of Theorem 2.1. This is done by constructing a class of AS problems similar to those described in Jiang et al. (2017) but with different parameterizations. Consider an AS problem whose setup is summarized in Fig. 5 and described below. The problem has $n = 16^m$ points, $C = 2^{m+1} + 1$ classes (2^{m+1} positive classes), and the search budget is $T = 2^{m+1}$, where m is a free parameter. We also have the reward function for each class f_i to be the same (arbitrary) positive, increasing, unbounded, and concave f. Each of the n points is classified into two groups: "clumps" and "isolated points."

The former consists of 4^m clumps, each of size T, and is visualized in Fig. 5(b). All points within the same clump share the same label, and exactly one clump contains positives, each of which belongs to a different positive class. In each instance of the problem class being described, this positive clump is chosen uniformly at random among the 4^m clumps. As such, the prior marginal positive probability of any of these points is $p_{\text{clump}} = 4^{-m}$.

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

- A secret set S of size $T/2 = 2^m$, visualized in Fig. 5(a), which encodes the location of the positive clump. The set S is first partitioned into $2m$ subsets $S_1, S_2, \ldots, S_{2m}$, each of size $2^m/2m$. Each subset S_i encodes one virtual bit b_i of information about the location of S, and is further split into m groups of $2^m/2m^2$ points, with each group encoding a virtual bit b_{ij} by a logical OR. The aforementioned virtual bit b_i, on the other hand, is obtained via a logical XOR: $b_i = b_{i1} \oplus b_{i1} \oplus \ldots \oplus b_{im}$.
- The remaining points, denoted as $\mathcal{R}$ and visualized in Fig. 5(c), are completely independent from each other and any other points. We have $|\mathcal{R}| = 16^m - 2(8^m) - 2^m$.

We first make the same two observations as in Jiang et al. (2017).

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

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

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

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

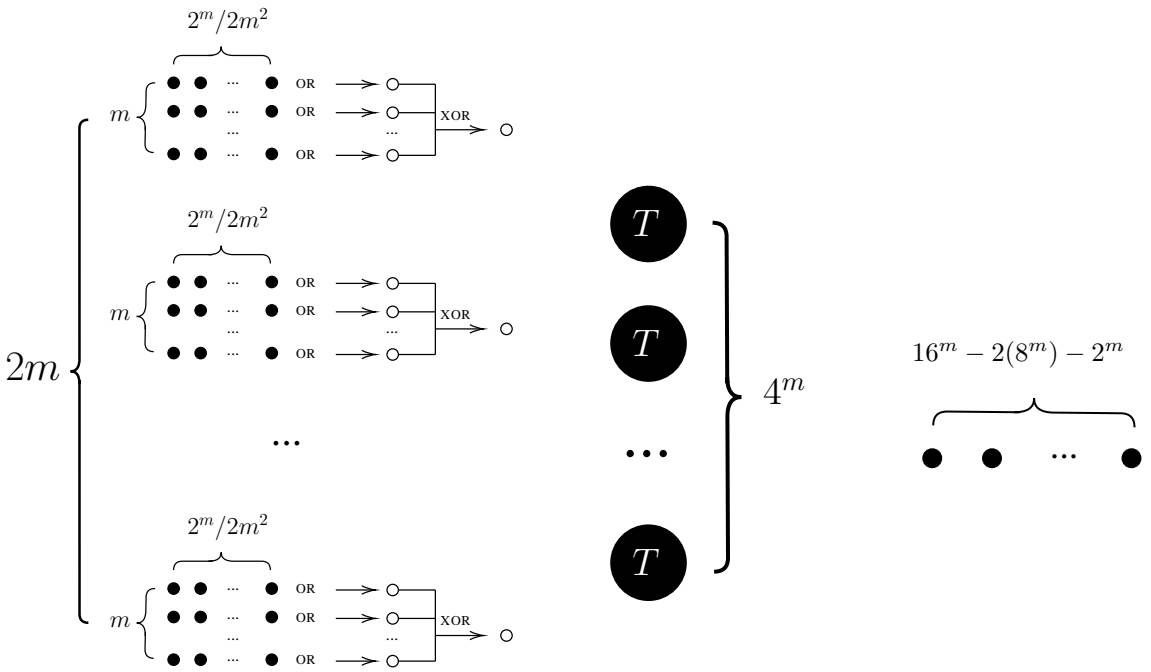
Observation A.2. *Observing any number of clump points does not change the marginal probability of any point in the secret set S.*

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

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

Now the policy spends the first half of its budget querying S, the labels of which identify the positive clump. The policy now spends the second half of the budget collecting these positive points. The resulting reward, denoted as OPT, is lower-bounded in the worst-case scenario where S does not contain any positive point:

$$\text{OPT} \geq \frac{T}{2} f(1) = 2^m f(1).$$



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

Figure 5: An instance of AS where any efficient algorithm can be arbitrarily worse than the optimal policy.

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

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

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

The RHS may further be upper-bounded by considering

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

where

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

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

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

The last inequality is due to

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

which is true by observing that for each of the m factors on each side,

$$\frac{n-2(8^m)}{i} < n-2(8^m)-i+1, \forall i = 1,\ldots,m.$$

Overall, we upper-bound the probability that $\mathcal{A}$ hits m points in S with

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

and union-bound the probability of $\mathcal{A}$ "hitting" the secret set after α inferences, denoted as p_{hit}, with

$$p_{\text{hit}} < \frac{\alpha}{\left(\frac{n-2(8^m)}{2^m\ 2^{m+1}}\right)^m}.$$

Here, $n-2(8^m) = 16^m - 2(8^m) = \Theta(16^m)$, so

$$p_{\text{hit}} < \frac{\alpha}{\Theta\left(\left(\frac{16^m}{2(4^m)}\right)^m\right)} = \frac{\alpha}{\Theta\left(4^{m^2}\right)}.$$

Hence, for any $\alpha = O(n^c) = O(16^{cm}) = O(4^{2cm})$, where c is a constant,

$$p_{\text{hit}} < O\left(\frac{4^{2cm}}{4^{m^2}}\right) = O(4^{-m^2}) = O(4^{-\log^2 n}).$$

In other words, the probability that $\mathcal{A}$ does find the secret set S decreases as a function of n. Conditioned on this event, we upper-bound its performance with $2^{m+1}\, f(1)$, assuming that every query is a hit.

On the other hand, if $\mathcal{A}$ never finds S, we further consider the following subcases: if the algorithm queries an isolated point, no marginal probability is changed; if a clump point is queried, only the marginal probabilities of the points in the same clump are updated. The expected performance in these two cases can be upper-bounded by pretending that the algorithm had a budget of size $2\,T = 2^{m+2}$, half of which is spent on querying isolated points and half on clump points.

The expected utility after T queries on isolated points is $\mathbb{E}\big[f(X)\big]$, where $X = \sum_{i=1}^{T} X_i$ and $\Pr(X_i = 1) = p_{\text{isolated}}$. We further upper-bound this expectation using Jensen's inequality:

$$\mathbb{E}\big[f(X)\big] < f\big(\mathbb{E}[X]\big) = f\big(T\, p_{\text{isolated}}\big) = f\left(2^{m+1}(1-2^{-\frac{2^m}{2m^2}})\right).$$

The expected utility after T queries on clump points is

$$\frac{T\, f(1)}{4^m} + \left(1-\frac{1}{4^m}\right)\left(\frac{(T-1)\, f(1)}{4^m-1} + \left(1-\frac{1}{4^m-1}\right)(\cdots)\right) = \frac{f(1)\sum_{i=1}^{T} i}{4^m} = \frac{f(1)\, T\,(T-1)}{2\, 4^m} = \frac{f(1)\,(2^{m+1}-1)}{2^m}.$$

Combining the two subcases, we have the expected utility in the case where $\mathcal{A}$ never hits S upper-bounded by

$$f\left(2^{m+1}(1-2^{-\frac{2^m}{2m^2}})\right) + \frac{f(2^{m+1})}{2^{m-1}}.$$

With that, the overall expected utility of $\mathcal{A}$, denoted by $E_{\mathcal{A}}$ is upper-bounded by

$$E_{\mathcal{A}} < 2^{m+1}\, f(1)\, p_{\text{hit}} + f\left(2^{m+1}(1-2^{-\frac{2^m}{2m^2}})\right) + \frac{f(2^{m+1})}{2^{m-1}}.$$

We then consider the upper bound of the approximation ratio

$$\frac{E_{\mathcal{A}}}{\text{OPT}} < 2\,p_{\text{hit}} + \frac{f\left(2^{m+1}(1-2^{-\frac{2^m}{2m^2}})\right)}{2^m\,f(1)} + \frac{(2^{m+1}-1)}{4^m}.$$

The first and third terms are arbitrarily small with increasing m. As for the second term, L'Hôpital's rule shows that $1-2^{-\frac{2^m}{2m^2}} = \Theta\left(\frac{2m^2}{2^m}\right)$, so this term scales like $\Theta\left(\frac{f(4m^2)}{2^m}\right)$, which is $O\left(\frac{4m^2}{2^m}\right)$ and also arbitrarily small with increasing m. As a result, algorithm $\mathcal{A}$ cannot approximate the optimal policy by a constant factor.

B Implementation and Pruning

As stated in Sect. 3.3, we use a k-NN model as the probabilistic classifier in our experiments, which achieves reasonable generalization error in practice and computationally efficient. Another benefit of this model is that it is possible to cheaply compute a *posterior probability upper bound* p^*, given any data set $\mathcal{D}$ and an additional observation with label y:

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

This upper bound is useful in that we may then bound the approximate expected terminal utility $\overline{u}$ conditioned on label c when computing the score in Eq. (11), i.e., $\alpha\,(x \mid y = c)$, and therefore the overall score $\alpha(x)$. With these score upper bounds in hand, we employ branch-and-bound pruning strategies used in previous work (Jiang et al., 2018; Nguyen et al., 2021). Specifically, before evaluating the score α of a given candidate x, we compare the upper bound of $\alpha(x)$ against the current best score α^* we have found. If α^* exceeds this bound, computing $\alpha(x)$ is unnecessary and we proceed to the next candidate. Otherwise, since we have access to the *conditional* score upper bounds for $\alpha\,(x \mid y = c)$, as we marginalize over each label y, we further check whether the conditional scores computed thus far, when combined with the complementary conditional upper bounds, are less than α^*. If this is the case, we terminate the current α computation "on the fly." The upper bounds p_c^* are cheap to evaluate, and these checks add a trivial overhead to the entire procedure in the pessimistic situation of no pruning – in practice, this cost is well worth it. Finally, a lazy-evaluation strategy is used, where candidates are evaluated in descending order of their score upper bounds, so that a given point that may be pruned will not be evaluated.

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

C Data Sets

We now describe the data sets used in our experiments in Sect. 5. These data sets are curated from authors of respective publications, as detailed below. No identifiable information or offensive content is included in the data.

Product recommendation. We use the Fashion-MNIST data set (Xiao et al., 2017), which 70 000 contains images of ten classes of fashion articles (e.g., t-shirts, trousers, pullovers, etc. – 10 000 instances for each class) to make a product recommendation problem for clothing items. To simulate two different users (corresponding to two search problems), we select three from the ten classes to be the positive classes:

- User 1: `trouser`, `bag`, `ankle boot`.
- User 2: `t-shirt`, `sandal`, `sneaker`.

For each user, we sub-sample these positive classes uniformly at random, making the positives harder to uncover; the hit rate that a random policy selects a positive point is roughly 2%. To build the similarity matrix used by the k-NN model, we

Table 3: Class-specific precision-at-k statistics of the k-NN, averaged across 10 randomly sampled training sets of size 499 (1% of the data set)

		precision at 1	precision at 5	precision at 10	precision at 50
easy	`trouser`	0.60	0.60	0.60	0.59
	`bag`	0.70	0.80	0.81	0.85
	`ankle boot`	0.40	0.52	0.60	0.49
hard	`t-shirt`	0.10	0.24	0.25	0.20
	`sandal`	0.50	0.60	0.56	0.49
	`sneaker`	0.70	0.68	0.68	0.66

used UMAP (McInnes et al., 2018) to obtain a five-dimensional embedding of the images and compute the top 100 nearest neighbors using Euclidean distance (similarity is calculated as $\exp\left(-d^2\right)$).

While the k-NN model has been found to perform well on the remaining data sets in previous studies (Garnett et al., 2012; Jiang et al., 2017; Mukadum et al., 2021), with this data, we can set select as targets classes that are either easy or difficult to identify with the k-NN, to simulate different levels of difficulty of search and study the effect of the quality of the k-NN on the performance of our methods. To this end, we measure the predictive performance of the k-NN and report the results in Tab. 3. We then classify the two search problems into an "easy" one, where the model scores highly on precision-at-k metrics (particularly important in AS), and a "hard" one, where the k-NN scores lower. The k-NN work as well on User 2 as on User 1, specifically on instances of `t-shirt`. This is because this class is not well-separated from the negative points. (Among the negative points, there are similar-looking classes such as pullover and shirt.)

Photoswitch discovery. We consider another AS problem introduced by Mukadum et al. (2021). The task is to search for molecular photoswitches with two specific desirable properties: high light absorbance and long half-lives. The collected data set contains 2049 molecules, 733 of which are found to be targets according to the criteria defined in the study. Further, the authors partition the points into 29 groups by their respective substructures, thus defining a multiclass AS problem with $C = 30$ (a negative class and 29 positive classes). The number of targets in a class ranges from 0 to 121. Following Mukadum et al. (2021), we used the Morgan fingerprints (Rogers and Hahn, 2010) of the molecules to compute the Tanimoto similarity coefficient (Willett et al., 1998) between each pair of molecules. These coefficients were then used to compute the nearest neighbor set of each data point.

The CiteSeer$^{\mathbf{x}}$ data set. We use the CiteSeer$^{\mathrm{x}}$ citation network data, introduced by Garnett et al. (2012). This data set contains 39 788 papers published at the 50 most popular computer science conferences and journals, and the label of each paper is its publication venue. Following Fouss et al. (2007), we compute the "graph principal component analysis" and use the first 20 components to form the feature vector of each paper. Our objective is to search for papers in machine learning and artificial intelligence proceedings. We conduct two sets of experiments with different numbers of classes C. For $C = 5$, we select papers from NeurIPS, ICML, UAI, and JMLR as our four target classes, adding up to 5575 targets (roughly 14% hit rate; an average of 3.5% per class). For $C = 10$, we additionally include papers from IJCAI, AAAI, JAIR, Artificial Intelligence, and Machine Learning as targets; this totals 12 382 targets, yielding a 31% overall hit rate and a 3.5% per class.

Drug discovery. We experiment with a drug discovery task using a massive chemoinformatic data set. The goal is to identify chemical compounds that exhibit selective binding activity given a protein. The data set consists of 120 such activity classes from BindingDB (Liu et al., 2007). For each class, there are a small number of compounds with significant binding activity – these are our search targets. We use the Morgan fingerprints (Rogers and Hahn, 2010) as the feature vectors and the Tanimoto coefficient (Willett et al., 1998) as the measure of similarity. In each experiment for a given value $C \in \{5, 10, 15\}$, we select $(C - 1)$ out of the 120 classes uniformly at random without replacement to form the target classes. They are then combined with 100 000 points sampled from the "drug-like" entries in the ZINC database (Sterling and Irwin, 2015), which serve as the negative set, to make up our search space. Features for these points are binary vectors encoding chemical properties, typically referred to as fingerprints. We use the Morgan fingerprint (Rogers and Hahn, 2010), which has shown good performance in past studies. The average prevalence of a target is 0.2%; the total hit rates are thus 1%, 2%, and 3% for $C = 5$, 10, and 15, respectively.

We finally report the values for the pseudocount γ_c of the k-NN, as described in Sect. 3.3, in Tab. 4, which roughly estimate the prevalence of each target class in each data set.

Table 4: Values for the pseudocount γ_c, as described in Sect. 3.3, used in the experiments in Sect. 5.

	Fashion-MNIST	Photoswitch	CiteSeer$^{\text{x}}$	Drug discovery
γ_c	0.01	0.005	0.05	0.001

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

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

Table 6: Quality and time of our approximation method against MC sampling, averaged across 10 random repeats of CiteSeer$^{\text{x}}$ $C = 10$ experiments. Under each setting, the approximation with the lowest error (with respect to the chosen ground truth) is highlighted **bold**, and so is the fastest method.

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

D Batch Utility Approximation Quality

We run simulations to compare the performance of the Jensen's upper bound $\overline{u}$ against Monte Carlo (MC) sampling, under the logarithmic utility function $u\left(\mathcal{D}\right) = \sum_{c>1} \log\left(1 + m_c\right)$. Using the CiteSeer$^{\text{x}}$ data set described in Appx. C, we first construct a training data set $\mathcal{D}$ by randomly selecting 50 points for each class ($|\mathcal{D}| = 50\,C$), and compute the posterior probabilities p_c with the k-NN to simulate a typical AS iteration. A random target batch X of size b is then chosen, and the considered approximation methods are applied on this batch. This entire procedure is repeated 10 times for each setting of (C, b), and the average root mean squared error (RMSE) and time taken for each method to return are reported in Tabs. 5 and 6. (MC(s) denotes MC sampling using s samples.) Note that in settings with large C or b, exact computation of Eq. (7) is prohibitively expensive, in which case MC $\left(2^{15}\right)$ is used as the ground truth. Overall, our Jensen's approximation offers a good tradeoff between accuracy and speed.

E Experimental Details & Further Results

We first detail how we set the hyperparameters of the active learning baselines used in our experiments.

- The policy by Vanchinathan et al. (2015) has two hyperparameters: λ encourages diversity (measured by the logdet of

Table 7: Square root utility and standard errors across 20 repeated experiments for each setting. C is the total number of unique classes in the search space. The best performance in each column is highlighted in **bold**; policies not significantly worse than the best (according to a two-sided paired t-test with a significance level of $\alpha = 0.05$) are in *blue italics*.

	CiteSeer$^{\text{x}}$	Drug discovery	
	$C = 10$	$C = 10$	$C = 15$
He and Carbonell (2007)	35.51 (0.22)	9.40 (0.33)	13.60 (0.45)
Vanchinathan et al. (2015)	48.61 (1.22)	32.28 (1.26)	41.10 (1.99)
Malkomes et al. (2021)	31.32 (0.01)	18.26 (0.79)	25.82 (0.87)
ENS	59.31 (0.81)	32.12 (1.17)	36.71 (1.71)
RR-GREEDY	57.37 (0.33)	30.66 (1.82)	37.51 (1.38)
RR-UCB	57.87 (0.32) ($\beta^* = 3$)	30.77 (1.47) ($\beta^* = 1$)	38.02 (1.49) ($\beta^* = 3$)
RR-ENS	60.20 (0.30)	*37.09 (1.20)*	45.05 (1.27)
One-step	60.59 (0.46)	*38.86 (1.59)*	46.60 (1.70)
DAS (ours)	**62.41 (0.24)**	**38.89 (1.76)**	**55.50 (1.41)**

Table 8: Logarithmic utility and standard errors across 20 repeated experiments on the Fashion MNIST data.

	easy	hard
DAS	14.02 (0.06)	12.38 (0.30)
SIMILAR	13.89 (0.02)	12.31 (0.58)

the Gram matrix of the collected data), and β_t encourages UCB-style exploration of the space. We run all variants of this policy with $\lambda \in \{0.25, 0.5, 0.75\}$ and $\beta_t \in \{0.1, 0.3, 1, 3, 10\}$ and report the best performance in Tabs. 1 and 7.

- The policy by He and Carbonell (2007) has a hyperparameter p, which is an estimate of the prevalence of the targets within the search space. We set this value in the same way as we set γ_c in Tab. 4.
- The policy by Malkomes et al. (2021) has a hyperparameter r, which sets the radius of the spheres that compute their coverage measure. As we have access to similarity scores (between 0 and 1) among points in each of our data sets, the spheres in this method cover points that are sufficiently similar (similarity greater than threshold $1 - r$) to a collected target. We run all variants of this policy with $r \in \{0.25, 0.5, 0.75\}$ and report the best performance in Tabs. 1 and 7.

We also include our experiment results with the square root utility function

$$u(\mathcal{D}) = \sum_{c>1} \sqrt{m_c},$$

under the CiteSeer$^{\text{x}}$ $C = 10$ and drug discovery $C \in \{10, 15\}$ settings in Tab. 7. Again, our method DAS was applied to this utility function without any algorithmic modification from the logarithmic utility. Under this utility, DAS still consistently performs the best across the investigated settings, showing that our proposed method generalizes well to this utility.

Finally, we present brief results comparing the SIMILAR algorithm (Kothawade et al., 2021) (instantiated for the task of rare class detection) with our method DAS. A neural network-based policy, SIMILAR can be readily applied to our production recommendation problem with the Fashion MNIST data, and the results are presented in Tab. 8, where we note a slight degradation in performance going from DAS to SIMILAR.

F Extension to Within-Class Diversity-Aware Active Search

In many important applications, one is concerned about not only diversity in the observed labels, but also a measure of within-class diversity, defined in the feature space. In this section, we describe how our diversity-aware active search

framework can be extended to this setting. First, we use a monotone submodular set function g (e.g., logdet) to quantify the diversity of a set of class-c positives X_c. Then, $g\left(X_c\right)$, instead of the class count m_c, is used as input to the class-specific concave utility f_c – our final utility is $\sum f_c\left(g(X_c)\right)$. This way, when a new point is added to X_c, the more different the point is from X_c, the more g, and in turn f_c, increases; this modified utility thus captures within- and cross-class diversity. Conveniently, our method still applies: $\mathbb{E}\big[f_c\left(g(X_c)\right)\big] < f_c\big(\mathbb{E}\left[g(X_c)\right]\big)$ (Jensen's inequality), and our lookahead seeks a batch maximizing $\overline{u} = \sum f_c\big(\mathbb{E}\left[g(X_c)\right]\big)$. The expectation of the submodular $g(X_c)$ is itself submodular, and thus the sum $\overline{u}$ is submodular and can be greedily optimized. In other words, our nonmyopic decision-making scheme remains viable. (In the special case of the count function $g(X_c) = |X_c|$, this reduces to what the main text presents.)

G Software

Matlab implementation of this work is released under MIT license at `https://github.com/KrisNguyen135/diverse_as`.