Discovering Valuable Items from Massive Data

Hastagiri P Vanchinathan
ETH Zurich
hastagiri@inf.ethz.ch

Andreas Marfurt
ETH Zurich
amarfurt@ethz.ch

Charles-Antoine Robelin
Amadeus IT Group SA
carobelin@amadeus.com

Donald Kossmann
ETH Zurich
donaldk@ethz.ch

Andreas Krause
ETH Zurich
krausea@ethz.ch

ABSTRACT

Suppose there is a large collection of items, each with an associated cost and an inherent utility that is revealed only once we commit to selecting it. Given a budget on the cumulative cost of the selected items, how can we pick a subset of maximal value? This task generalizes several important problems such as multi-arm bandits, active search and the knapsack problem. We present an algorithm, GP-Select, which utilizes prior knowledge about similarity between items, expressed as a kernel function. GP-Select uses Gaussian process prediction to balance exploration (estimating the unknown value of items) and exploitation (selecting items of high value). We extend GP-Select to be able to discover sets that simultaneously have high utility and are diverse. Our preference for diversity can be specified as an arbitrary monotone submodular function that quantifies the diminishing returns obtained when selecting similar items. Furthermore, we exploit the structure of the model updates to achieve an order of magnitude (up to 40X) speedup in our experiments without resorting to approximations. We provide strong guarantees on the performance of GP-Select and apply it to three real-world case studies of industrial relevance: (1) Refreshing a repository of prices in a Global Distribution System for the travel industry, (2) Identifying diverse, binding-affine peptides in a vaccine design task and (3) Maximizing clicks in a web-scale recommender system by recommending items to users.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – Data Mining; G.3 [**Probability and Statistics**]: Experimental Design

Keywords

Design of experiments, Active search, Active learning, Kernel methods, Recommender systems

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

ACM 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783360.

1. INTRODUCTION

Consider a large collection of items, each having an inherent value and an associated cost. We seek to select a subset of maximal value, subject to a constraint on the cumulative cost of the selected items. If we know the items' values and costs, this is just the classical knapsack problem - which is NP-hard, but can be near-optimally solved, e.g., using dynamic programming. But what if we do not know the values? Concretely, we consider the setting where we can choose an item, observe a noisy estimate of its value, then choose and evaluate a second item and so on, until our budget is exhausted. It is clear that in order to achieve non-trivial performance, we must be able to make predictions about the value of non-selected items given observations made so far. Hence, we will assume that we are given some information about the similarity of items (e.g., via features), whereby similar items are expected to yield similar value. As a motivating application, consider experimental design, where we may need to explore a design space, and wish to identify a set of near-optimal designs, evaluating one design at a time. In the early stages of medical drug development, for example, candidate compounds are subject to various tests and a fixed number of them are selected to the next stage to perform animal/human testing. Even the initial tests are expensive and the goal is to reduce the number of compounds on which these tests are conducted while still selecting a good set of compounds to promote to the next level. Another application is recommender systems, where for a given customer, we may seek to iteratively recommend items to read/watch, aiming to maximize the cumulative relevance of the entire set. Alternatively, we might want to pick users from our user base or a social network to promote a given item. In this setting, how should we select items to maximize total utility?

We will call this general class of problems AVID - *Adaptive Valuable Item Discovery.* To solve AVID, we need to address an exploration–exploitation dilemma, where we must select items that maximize utility (exploit) while simultaneously estimating the utility function (explore). We address these challenges by using ideas from Gaussian Process optimization and multi-armed bandits to provide a principled approach to AVID with strong theoretical guarantees. Specifically, we introduce a novel algorithm, GP-Select, for discovering high value items in a very general setting. GP-Select can be used whenever the similarity between items can be captured by a positive definite kernel function,

and the utility function has low norm in the Reproducing Kernel Hilbert Space (RKHS) associated with the kernel. The algorithm models the utility function as a sample from a Gaussian process distribution, and uses its predictive uncertainty to navigate the exploration–exploitation tradeoff via an upper confidence based sampling approach that takes item costs into account.

We also consider a natural extension of AVID, where the goal is to obtain a *diverse* set of items. This is an important requirement in many experimental design problems where, for example for reasons of robustness, we seek to identify a collection of diverse, yet high quality designs. In our drug design example, very similar compounds might cause similar side effects in the later stages of testing. Hence, we might require a certain diversity in the selected subset while still trying to maximize total value. In this work, we address the setting where our preference for diversity is quantified by a submodular function, modeling diminishing returns incurred when picking many similar items. We prove that GP-Select provides an effective tradeoff of value and diversity, establishing bounds on its regret against an omniscient algorithm with access to the unknown objective. Our results substantially expand the class of problems that can be solved with upper confidence based sampling methods – desirable for their simplicity – in a principled manner.

We evaluate GP-Select in three real-world case studies. We first demonstrate how GP-Select can be used to maintain an accurate repository of ticket prices in a Global Distribution System that serves a large number of airlines and travel agencies. Here the challenge is to selectively recompute ticket prices that likely have changed, under a budget on the number of computations allowed. Secondly, we demonstrate how GP-Select is able to determine a diverse set of candidate designs in a vaccine design application exhibiting high binding affinity to their target receptors. In these experiments, we also study the effect of inducing diversity, and non-uniform selection cost. Finally, we present results on a web-scale recommender systems dataset provided by Yahoo! where the task is to adaptively select user-item pairs that maximize interaction (clicks, likes, shares, etc.).

Our experiments highlight the efficacy of GP-Select and its applicability to a variety of problems relevant to practitioners. In particular, with our suggested application of lazy variance updates, we are able to speed up the execution by up to almost 40 times, making it usable on web-scale datasets.

2. AVID: PRELIMINARIES

We are given a set $\mathbf{V} = \{1, \ldots, n\}$ of n objects. There is a utility function $f : \mathbf{V} \to \mathbb{R}_{\geq 0}$ that assigns a non-negative value to every item in the set. Similarly, there is a function $c : \mathbf{V} \to \mathbb{R}_{>0}$, assigning a positive cost $c_v = c(v) \in [c_{min}, c_{max}]$ to each item v. Given a subset $S \subseteq \mathbf{V}$, its value $F(S) = \sum_{v \in S} f(v)$ is the sum of the values of the selected items, and its cost $C(S)$ the cumulative costs of the items. Given a budget $B > 0$, our goal is to select

$$S^*_B = \operatorname*{argmax}_{C(S) \leq B} F(S), \tag{1}$$

i.e., a subset of maximum value, with cost bounded by B.

If we knew the utility function f, then Problem (1) is the classical knapsack problem. While NP-hard, for any ε, an ε-optimal solution can be found via dynamic programming.

But what if we do not know f? In this case, we consider choosing a subset S in a sequential manner. We pick one item at a time, after which the value of the selected item is revealed (possibly perturbed by noise), and can be taken into account when selecting further items. We term this sequential problem AVID - *Adaptive Valuable Item Discovery*.

Equivalent to maximizing the cumulative value $F(S)$, we aim to minimize the *regret*, i.e., the loss in cumulative value compared to an omniscient optimal algorithm that knows f. Formally, the regret of a subset S_B of cost B is defined as: $R_B = F(S^*_B) - F(S_B)$. We seek an algorithm whose regret grows slowly (sublinearly) with the budget B, so that the average regret R_B/B goes to 0.

Diversity.

In several important applications, we not only seek items of high value, but also to optimize the diversity of the selected set. One way to achieve this goal is to add to our objective another term that prefers diverse sets. Concretely, we extend the scope of AVID by considering objective functions of the form:

$$F(S) = (1 - \lambda) \sum_{v \in S} f(v) + \lambda D(S). \tag{2}$$

Hereby, $D(S)$ is a *known* measure of the diversity of the selected subset S. Many such diversity-encouraging objectives have been considered in the literature (c.f., [18, 22, 32, 40]). We will present an algorithm that is guaranteed to choose near-optimal sets whenever the function D satisfies *submodularity*. Submodularity is a natural notion of diminishing returns, capturing the idea that adding an item helps less if more similar items were already picked [4]. We discuss examples in Section 4. $\lambda \in [0, 1]$ is a tradeoff parameter balancing the relative importance of value and diversity of the selected set. In the case where f is known, maximizing D requires maximizing a submodular function. This task is NP-hard, but can be solved near-optimally using a greedy algorithm [24]. In this paper, we address the novel setting where D is any known submodular function but f is *unknown*, and needs to be estimated.

Regularity Assumptions.

In the general case, where f can be any function, it is hopeless to compete against the optimal subset since, in the worst case, f could be adversarial and return a value of 0 for each of the items selected by the algorithm, and positive utility only for those not selected. Hence, we make some natural assumptions on f such that the problem becomes tractable. In practice, it is reasonable to assume that f varies 'smoothly' over the candidate set $\mathbf{V}$ such that similar items in $\mathbf{V}$ have similar f values. In this work, we model this by assuming that the similarity $\kappa(v, v')$ of any pair of items $v, v' \in \mathbf{V}$ is given by a positive definite kernel function [27] $\kappa : \mathbf{V} \times \mathbf{V} \to \mathbb{R}$, and that f has low "complexity" as measured by the norm in the Reproducing Kernel Hilbert Space (RKHS) associated with kernel κ. The RKHS $\mathcal{H}_\kappa(\mathbf{V})$ is a complete subspace of $L_2(\mathbf{V})$ of 'smooth' functions with an inner product $\langle \cdot, \cdot \rangle_\kappa$ s.t $\langle f, \kappa(v, .) \rangle = f(v)$ for all $f \in \mathcal{H}_\kappa(\mathbf{V})$. By choosing appropriate kernel functions, we can flexibly handle items of different types (vectors, strings, graphs etc.). We use the notation $\mathbf{K}$ to refer to the $n \times n$ kernel (Gram) matrix obtained by evaluating $\kappa(v, v')$ for all pairs of items.

Explore-Exploit Tradeoff.

Given the regularity assumptions about the unknown function f, the task can be intuitively viewed as one of trading off exploration and exploitation. That is, we can either greedily utilize our current knowledge of f by picking the next item predicted to be of high value, or we can choose to pick an item that may not have the highest expected value but most reduces the uncertainty about f across the other items. This challenge is akin to the dilemma faced in multi-arm bandit problems. An important difference in our setting, motivated by practical considerations, is that we *cannot select the same item multiple times.* As a consequence, classical algorithms for multi-armed bandits (such as UCB1 of Auer et al. [1] or GP-UCB of Srinivas et al. [30]) cannot be applied, since they require that repeated experimentation with the same "arm" is possible. Furthermore, classical bandit algorithms do not allow arms to have different costs. In fact, our setting is *strictly more general* than the bandit setting: We can allow repeated selection of a single item v by just creating multiple, identical copies $v^{(1)}, v^{(2)}, \ldots$ with identical utility (i.e., $f(v^{(1)}) = f(v^{(2)}) = \ldots$), which can be modeled using a suitably chosen kernel.

Nevertheless, we build on ideas from modern bandit algorithms that exploit smoothness assumptions on the payoff function. In particular, Srinivas et al. [30] show how the explore-exploit dilemma can be addressed in settings where, as in our case, the reward function has bounded RKHS norm for a given kernel function κ. We interpret the unknown value function f as a sample from a Gaussian Process (GP) prior [26], with prior mean 0 and covariance function κ. Consequently, we model the function as a collection of normally distributed random variables, one for each item. They are jointly distributed, such that their covariances are given by the kernel:

$$\mathrm{Cov}\big(f(v), f(v')\big) = \kappa\big(v, v'\big).$$

This joint distribution then allows us to make predictions about unobserved items via Bayesian inference in the GP model. Suppose we have already observed feedback $\mathbf{y}_t = \{y_1, \ldots, y_t\}$ for t items $S_t = \{v_1, \ldots, v_t\}$, i.e., $y_i = f(v_i) + \epsilon_i$, where ϵ_i is independent, zero-mean Gaussian noise with variance $\hat{\sigma}^2$. Then, for each remaining item v, its predictive distribution for $f(v)$ is Gaussian, with mean and variance (using noise variance $\hat{\sigma}$, according to our assumptions) given by:

$$\mu_t(v) = \mathbf{k}_t(v)^T(\mathbf{K}_t + \hat{\sigma}^2\mathbb{I})^{-1}\mathbf{y}_t, \tag{3}$$

$$\sigma_t^2(v) = \kappa(v,v) - \mathbf{k}_t(v)^T(\mathbf{K}_t + \hat{\sigma}^2\mathbb{I})\mathbf{k}_t(v), \tag{4}$$

where $\mathbf{k}_t(v) = [\kappa(v_1, v), \ldots, \kappa(v_t, v)]^T$, $\mathbf{K}_t$ is the positive semi-definite kernel matrix such that for $i, j \leq t$, $\mathbf{K}_{t,i,j} = [\kappa(v_i, v_j)]$ and $\mathbb{I}$ is the $t \times t$ identity matrix. In Section 3, we show how we can use these predictive distributions to navigate the exploration–exploitation tradeoff. Note that while we propose a Bayesian algorithm (using a GP prior, and Gaussian likelihood), we prove agnostic results about arbitrary functions f with bounded norm, and arbitrary noise bounded by $\hat{\sigma}$.

Algorithm 1 GP-SELECT

Input: Ground Set $\mathbf{V}$, kernel κ and budget B
Initialize selection set S
for $t = 1, 2, \ldots, B$ **do**
  Model Update:
    $[\mu_{t-1}(\cdot), \sigma^2_{t-1}(\cdot)] \leftarrow$ GP-Inference$(\kappa, (S, y_{\{1:t-1\}}))$
  Item Selection:
    Set $v_t \leftarrow \underset{v \in \mathbf{V}/\{v_{1:t-1}\}}{\mathrm{argmax}} \mu_{t-1}(v) + \beta_t^{1/2}\sigma_{t-1}(v)$
  $S \leftarrow S \cup \{v_t\}$
  Receive feedback $y_t = f(v_t) + \epsilon_t$
end for

3. THE UNIFORM COST CASE

We first provide the solution for the simple case of uniform costs. In this setting, if the values are known, a greedy algorithm adding items of maximal value solves Problem (1) optimally. Our key idea in the unknown value case is to mimic this greedy algorithm. Instead of greedily adding the item v with highest predicted gain $\mu_{t-1}(v)$, we trade exploration and exploitation by greedily optimizing an optimistic estimate of the item's value. Concretely, our algorithm GP-SELECT for the uniform cost case performs both a model update and selects the next item upon receiving feedback for the current selected item. The model update is performed according to Equations (3) and (4).

For our selection rule, we borrow a key concept from multi-armed bandits: upper confidence bound sampling. Concretely, we choose

$$v_t = \underset{v \in \mathbf{V} \setminus \{v_{1:t-1}\}}{\mathrm{argmax}} \mu_{t-1}(v) + \beta_t^{1/2}\sigma_{t-1}(v), \tag{5}$$

The tradeoff between exploration and exploitation is implicitly handled by the time varying parameter β_t (defined in Theorem 1) that alters the weighting of the posterior mean (favoring exploitation by selecting items with high expected value) and standard deviation (favoring exploration by selecting items that we are uncertain about). β_t is chosen such that $\mu_{t-1}(v) + \beta_t^{1/2}\sigma_{t-1}(v)$ is a high-probability upper bound on $f(v)$, explained further below.

Regret bounds.

We now present bounds on the regret R_B incurred by GP-SELECT. Crucially, they *do not* depend on the size of the ground set $|\mathbf{V}|$, but only on a quantity $C_\mathbf{K}$ that depends on the task specific kernel capturing the regularity of the utility function over the set of items. Specifically, for a kernel matrix $\mathbf{K}$, the quantity $C_\mathbf{K}$ is given by:

$$C_\mathbf{K} = \frac{1}{2}\log \mid \mathbb{I} + \hat{\sigma}^{-2}\mathbf{K} \mid . \tag{6}$$

We now present the main result about GP-SELECT in the uniform cost case.

THEOREM 1. *Let $\delta \in (0,1)$. Suppose that the function f lies in the the RKHS $\mathcal{H}_\kappa(\mathbf{V})$ corresponding to the kernel $\kappa(v, v')$ with an upper bound on the norm of f w.r.t. κ given by R (i.e., $||f||^2_\kappa \leq R$). Further suppose that the noise has zero mean conditioned on the history and is bounded by $\hat{\sigma}$ almost surely. Let $\beta_t = 2R + 300C_\mathbf{K}\log^3(t/\delta)$. Running* GP-SELECT *with a GP prior using mean zero, covariance $\kappa(v, v')$ and noise model $N(0, \hat{\sigma}^2)$, we obtain a regret bound*

of $O^*(\sqrt{B}(R\sqrt{C_{\mathbf{K}}} + C_{\mathbf{K}}))$ *w.h.p. Specifically,*

$$Pr\{R_B \leq \sqrt{C_1 B \beta_B C_{\mathbf{K}}} \ \ \forall B \geq 1\} \geq 1 - \delta$$

where $C_1 = \frac{8}{log(1+\hat{\sigma}^{-2})}$.

Due to space considerations, we omit the the proof of the Theorem but refer the reader to the longer version [35].

Interpretation of the Theorem.

Theorem 1 guarantees that under sufficiently regular f and suitable choice of β_t, the average regret compared to the best subset approaches 0 as B increases. Our regret bound depends only on the constant $C_{\mathbf{K}}$ rather than the actual size of the set $\mathbf{V}$. It is instructive to think of how the value $C_{\mathbf{K}}$ grows as the size of the ground set, $n = |\mathbf{V}|$ increases. As long as the kernel function is bounded, it can be seen that $C_{\mathbf{K}}$ is $O(n)$. For many commonly used kernel functions, however, this quantity grows strictly sublinearly in the number n of elements. For instance, for the popular RBF kernel in d dimensions (that is, $\mathbf{V} \subseteq \mathbb{R}^d$), it holds that $C_{\mathbf{K}} = C_{\mathbf{K}}(n) = O((\log n)^{d+1})$. Refer Srinivas et al. [30] for this and other analytical bounds for other kernels. In any case, a problem specific $C_{\mathbf{K}}$ can always be computed efficiently using the formula in Equation (6). Further note that as long as we use a universal kernel κ (like the commonly used Gaussian kernel), for finite item sets (as we consider here) the RKHS norm $||f||_\kappa$ is always bounded. Hence, Theorem 1 guarantees that our regret will always be bounded for such kernels, provided we choose a large enough value for R.

An important point to be made here is that the value of β_t as prescribed by Theorem 1 is chosen very conservatively for sake of the theoretical analysis. For most practical applications, β_t can be scaled down to achieve faster convergence and lower regret.

4. SELECTING DIVERSE SUBSETS

In some cases, we not only seek high cumulative value of the solution set, but also prefer *diversity*. This can be the case because we desire robustness, fairness etc. Formally, we can encode this diversity requirement into the objective function as done in (2). Hereby f is an *unknown* function that operates on individual elements, while D is a *known* set function that captures the diversity of a subset. It is natural to model diversity as a submodular function. Formally, a set function $D : 2^{\mathbf{V}} \to \mathbb{R}$ is *submodular* if for every $A \subseteq B \subseteq \mathbf{V}$ and $v \in \mathbf{V} \setminus B$, it holds that

$$\Delta_D(v \mid A) \geq \Delta_D(v \mid B), \tag{7}$$

where $\Delta_D(v \mid A) \equiv D(A \cup \{v\}) - D(A)$ is called the *marginal gain* of adding v to set A. D is called *monotone*, if, whenever $A \subseteq B$ it holds that $D(A) \leq D(B)$.

The rationale behind using submodular functions to model diversity is based on the intuition that adding a new element provides less benefit (marginal gain) as the set of similar items already selected increases. Many functions can be chosen to formalize this intuition. In our setting, a natural monotone submodular objective that captures the similarity as expressed via our kernel, is

$$D(S) = \frac{1}{2}\log\left|(\mathbb{I} + {\sigma_n}^{-2}\mathbf{K}_{S,S})\right|, \tag{8}$$

where $\sigma_n \geq 0$. We use this objective in our experiments. For this choice, the marginal gain of adding an element v to a set S is given by:

$$\Delta_D(v \mid S) = \frac{1}{2}\log(1 + {\sigma_n}^{-2}\sigma^2_{v|S}), \tag{9}$$

where $\sigma^2_{v|S}$ is the predictive variance of $f(v)$ in a GP model, where the values of elements in S have already been observed up to Gaussian noise with variance σ_n^2. Conveniently, while executing GP-Select, if $\hat{\sigma} = \sigma_n$, we already compute $\sigma^2_{v|S}$ in order to evaluate the decision rule (5). Hence, at almost no additional cost we can compute the marginal gain in diversity for any candidate item v.

In order to select items that provide value and diversity, it is natural to modify the selection rule of GP-Select in the following way:

$$v_t = \operatorname*{argmax}_{v \in \mathbf{V} \setminus \{v_{1:t-1}\}} (1-\lambda)\Big[\mu_{t-1}(v) + \beta_t^{1/2}\sigma_{t-1}(v)\Big] + \lambda\, \Delta_D(v \mid \{v_1, \ldots, v_{t-1}\}). \tag{10}$$

This decision rule greedily selects item v that maximizes a high-probability upper bound on the marginal gain $\Delta_F(v \mid \{v_1, \ldots, v_{t-1}\})$ of the *unknown* combined objective F.

Regret bound.

The regret bound in Section 3 depended on the fact that we were optimizing against f that assigned values to individual elements, $v \in \mathbf{V}$. The same bounds need not hold in the more challenging setting when trading value against diversity. In fact, even if both f and D are completely *known* for all $v \in \mathbf{V}$, it turns out that optimizing F in (2) is NP-hard for many monotone submodular functions D [8]. While finding the *optimal* set is hard, Nemhauser et al. [24] states that – for a *known* monotone submodular function – a simple greedy algorithm provides a near-optimal solution.

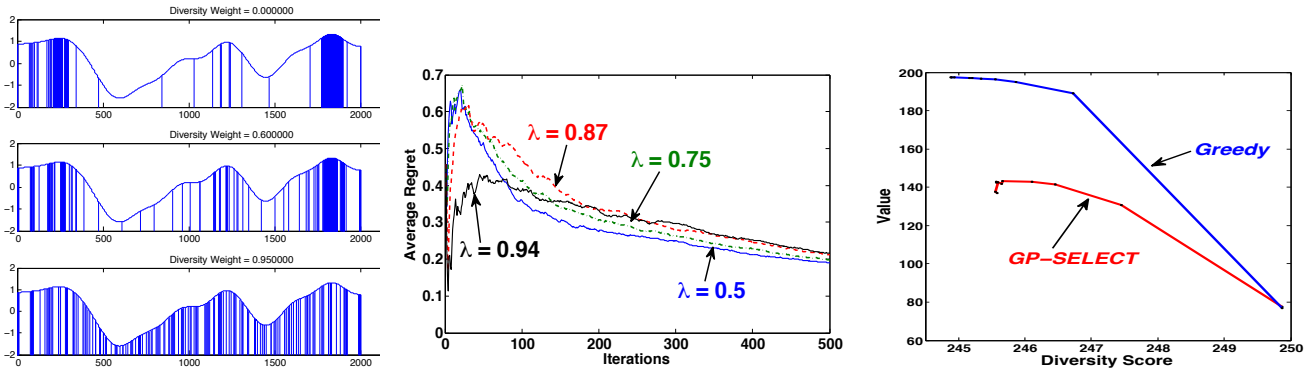
Formally, suppose $S'_0 = \emptyset$ and S'_{i+1}, the greedy extension to S'_i. That is, $S'_{i+1} = S'_i \cup \{\operatorname{argmax}_{v \in \mathbf{V} \setminus S'_i} \Delta_F(v \mid S'_i)\}$. Thus, S'_B is the set we obtain when selecting B items, always greedily maximizing the marginal gain over the items picked so far. Then it holds that $F(S'_B) \geq (1-1/e)\max_{|S| \leq B} F(S) = (1 - 1/e)F(S^*_B)$. Moreover, without further assumptions about $D(S)$ and f, no efficient algorithm will produce better solutions in general. Since we are interested in computationally efficient algorithms, we measure the regret of a solution S_B by comparing $F(S_B)$ to $F(S'_B)$, which is the bound satisfied by the greedy solution. Formally, $R_B = (1 - 1/e)F(S^*_B) - F(S_B)$.

Theorem 2. *Under the same assumptions and conditions of Theorem 1,*

$$Pr\{R_B \leq \sqrt{C_1 B \beta_B C_{\mathbf{K}}} \ \ \forall B \geq 1\} \geq 1 - \delta,$$

where $R_B = (1-1/e)F(S^*_B) - F(S_B)$ *is the regret with respect to the value guaranteed when optimizing greedily given full knowledge of* f *and* D.

The proof can be found in the longer version [35]. It rests on interpreting GP-Select as implementing an approximate version of the greedy algorithm maximizing $\Delta_F(v \mid S_t)$. In fact, Theorem 2 can be generalized to a large number of settings where the greedy algorithm is known to provide near-optimal solutions for constrained submodular maximization.



(a) *Balancing utility and diversity* (b) *Average Regret (Diversity)* (c) *Effect of Inducing Diversity*

Figure 1: (a): Illustration of sets selected for trading f against D when varying parameter λ. (b): Performance of GP-Select in selecting diverse subsets. For different values of λ, the average regret against the greedy approximate algorithm decreases. (c): Improvements in diversity can be obtained at little loss of utility.

As an illustration of the application of this modified GP-Select to diverse subset selection, refer to Figure 1(a). When $\lambda = 0$, GP-Select reverts back to Algorithm 1 and hence, picks locations only based on its expected f value. This is clear from the thick bands of points sampled near the maximum. At $\lambda = 0.6$, GP-Select balances between expected f values of the points and the marginal gain in diversity of the points picked $\Delta_D(v \mid S)$. At λ close to 1, GP-Select picks mostly by marginal gain which will be approximately uniform if the kernel used is isotropic (e.g. Gaussian kernel).

5. NON-UNIFORM COSTS

In the general case where each element $v \in \mathbf{V}$ has different costs of selection c_v, the budget B is the total cost of all items in the selected subset. We modify the selection rule in Algorithm 1 to take the estimated cost-benefit ratio into account. Most of the other steps remain the same except ensuring that we respect the budget, and the formula for computing β_t. The new selection rule for the setting without diversity is:

$$v_t = \underset{v \in \mathbf{V} \setminus S, c_v \leq B - C(S)}{\operatorname{argmax}} \frac{\mu_{t-1}(v) + \beta_t^{1/2} \sigma_{t-1}(v)}{c_v}. \qquad (11)$$

Hence, instead of maximizing an optimistic estimate of the item's value, we greedily maximize an optimistic estimate of the benefit-cost ratio. Note that this greedy rule encourages some natural opportunistic exploration: Initially, it will select items that we are very uncertain about (large σ_{t-1}), but that also have little cost. Later on, as the utility is more accurately estimated, it will also invest in more expensive items, as long as their expected value (μ_{t-1}) is high.

The idea above can be generalized to encourage diversity as well. The selection rule in (10) can be modified to maximize the ratio

$$\frac{(1-\lambda)\left[\mu_{S-1}(v) + \beta_S^{1/2} \sigma_{S-1}(v)\right] + \lambda \Delta_D(v \mid S)}{c_v}. \qquad (12)$$

Hence, in this most general setting, we greedily optimize a high-probability upper bound on the cost-benefit ratio of the marginal gain for the joint objective.

Upon these modifications, we can obtain the result presented in Theorem 3. The result holds for running GP-Select for selecting diverse items with items in the ground set having non-uniform costs of selection. Again, the proof can be found in the longer version [35].

Theorem 3. *Under the same assumptions and conditions of Theorem 1, running* GP-Select *with non-uniform costs for the items, we have that*

$$Pr\{R_B \leq \left(\max_{v \in \mathbf{V}} f(v) + c_{max}\sqrt{C_1 B \beta_B C_K}\right) \quad \forall B \geq 1\} \geq 1-\delta,$$

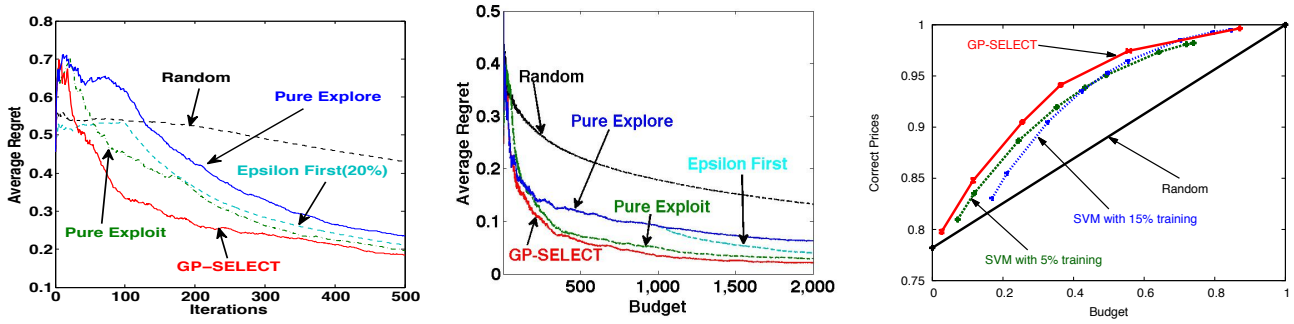
where $R_B = (1-1/e)F(S_B^*) - F(S_B)$ is the regret with respect to the value guaranteed when optimizing greedily given full knowledge of f and D.

6. EXPERIMENTAL EVALUATION

6.1 Case Study I: Airline Price Update Prediction Task

Amadeus IT group SA[1] is a Global Distribution System (GDS) for airline prices. One of the services provided by Amadeus is finding the cheapest return fare between cities X and Y on requested dates of travel. This is currently done by frequently querying all the airlines for their respective cheapest fares for each pair of cities and then aggregating the results to maintain this information. This consumes a lot of bandwidth and time. Also, computing the fare for a given request is a computationally expensive task as the cheapest fare might include multiple hops possibly operated by different airlines. Hence, a table of precomputed current best prices is maintained in order to quickly respond to fare requests by customers. Since the database is typically very large and computing fares is relatively expensive in terms of computation and network bandwidth, it is challenging to frequently recompute all fares (i.e., update the entire table). Since similar prices for similar fare requests (table entries) often change at the same time, the goal is to selectively recompute only entries that changed. This task can be naturally captured in our setting, where items correspond to

[1]http://www.amadeus.com



(a) *Average Regret* (b) *Non-uniform costs* (c) *Flight ticket price change prediction*

Figure 2: (a): While average regret decreases for all non-naive algorithms, GP-Select drops much earlier and continues to outperform the baselines in the vaccine design task. (b): Comparison of GP-Select with the baselines for the vaccine design task under non-uniform item costs. (c): GP-Select outperforms benchmarks on the fare change prediction task.

table entries selected for recomputation, and the utility of an item is 1, if the entry changed and 0, otherwise.

The data provided by Amadeus for this task was collected in December 2011. It consists of cheapest fares computed for 50,000 routes (origin-destination pairs) and for all departure dates up to 90 days into the future. For each departure date, the return date could be up to 15 days after the departure. The budget for selection corresponds to the total number of price refresh computations allowed. Our performance metric is the ratio between the total number of correct prices (i.e., correct entries in the table) and the total number of prices in the repository. Since we have the data with all the correct prices, we are able to compute the number of prices an algorithm would have missed to update (regret).

In our experiments, we pool all the data for a given route together, and sequentially process the data set, one "current date" at a time. The task is to discover items (table entries) that have changed between the current date and the next date. We thus instantiate one instance of the active discovery problem per route per day. For each instance, we select from $90 \cdot 15 = 1350$ prices to recompute. Typically only 22% of the data changed between days, hence even with a budget of 0, around 78% of the prices are correct. In order to capture similarity between items (table entries), we use the following features: *date, origin, destination, days until departure, duration of stay, current price.* We use an RBF kernel on these features and tune the bandwidth parameter using data from four routes (origin-destination pairs). We compare GP-Select against the following baselines:

1. **Random:** Naive baseline that picks points to query uniformly at random until the budget is exhausted
2. **Epsilon-First:** A Support Vector Machine (SVM) classifier is trained on a randomly sampling part of the data. Concretely, we report the values for two different settings that perform best among other options(5% and 15%) of the data. The SVM is then used to predict changes, and the predicted points are updated. When higher budgets are allowed, we use a weighted version of the SVM that penalizes false negatives stronger than false positives.

Figure 2 (c) presents the results of our experiments. In general, GP-Select performs better than the baselines. Note that all three non-naive algorithms reach similar maximum performance as the budget is increased close to 100% of the total number of items.

6.2 Case Study II: Vaccine Design Task

The second task we consider is an experimental design problem in drug design. The goal is to discover peptide sequences that bind well to major histocompatibility complex molecules (MHC). MHC molecules act as a mediator for interaction of leukocytes (white blood cells) with other leukocytes or body cells and play an important role in the immune system. In our experiments, the goal is to choose peptide sequences for vaccine design that maximizes the binding affinity to these Type I MHC molecules [25]. It is known from past experiments that similar sequences have similar binding affinity responses [12, 17, 39]. Instead of selecting only one optimal sequence, it is an important requirement to select multiple sequences as candidates and the actual determination of the best sequence is delayed until more thorough tests are completed further down the drug testing pipeline. Hence, while the task for this dataset can also be viewed as a classification task (binders vs non-binders), we are interested in the actual value of the binding affinity and want to pick a set of peptide sequences that have maximal affinity values.

The dataset [25] consists of peptide sequences of length $l = 9$ for the A_0201 task [39] which consists of 3089 peptide sequences along with their binding affinities (IC_{50}) as well features describing the peptide sequences. We normalize the binding affinities and construct a linear kernel on the peptide features. The task is then to select a subset of up to 500 sequences with maximal affinities. Since this is now inherently a regression task, we used GP regression to estimate the predictive mean of the underlying function. The following baseline algorithms were considered for comparison:

1. **Random:** Naive algorithm that picks sets of size 500 uniformly at random. We repeated this 30 times and report average total affinity values.
2. **Pure Explore:** This algorithm picks the most uncertain sequence among the remaining sequences. The GP is refitted every time an observation is made.
3. **Pure Exploit:** This algorithm always picks the next sequence as the one with the highest expected affinity as computed by GP-regression and the resulting values are used to retrain the GP. This is equivalent to the

one-step lookahead policy of [9]. It is not feasible to implement two or three step lookahead with this large dataset.

4. **Epsilon First:** This algorithm randomly explores for a few iterations and then once the GP is trained with the observed responses, behaves exactly like *Pure Exploit*. Among all the options we tried, we report results for training on the first 20% of the budget (100 sequences in this case) since this performed best. A major drawback of this algorithm is that it needs to know the budget a priori. We repeated this algorithm 30 times on the data and report the average.

The results of these experiments are presented in Figure 2 (a), which displays the average regret R_B/B. GP-SELECT clearly outperforms the baselines in the regret measure. The average regret drops much faster for GP-SELECT and continues to remain lower than all the baseline across all the iterations.

Choosing Valuable and Diverse Subsets.

Using the same vaccine design dataset, we implement the modified version of GP-SELECT presented in Section 4 to select a diverse set of peptide sequences. This requirement of diversity is quite natural for our drug testing application: Very similar sequences, while having similar affinity values, might also suffer from similar shortcomings in later stages of drug testing. We run GP-SELECT with different values of the tradeoff parameter λ, and report the results. Figure 1 (b), is the average regret R_B/B of GP-SELECT for different values of λ . The plot demonstrates that when selecting diverse subsets GP-SELECT has a similar regret performance as in the initial case when it was selecting only for value. Also, the average regret compared to the greedy optimal solution slightly increases with increase in the value of λ. Figure 1 (c) shows the inherent tradeoff between value and diversity. We use values of $\lambda = \{0, 0.5, 0.75, 0.875, 0.9375, 0.96875\}$ and plot the performance. We use the diversity function defined in Equation (2). It should be noted that this function is in log scale. From the plot it is clear that for a significant increase in the diversity score, we lose very little functional value, which suggests that robustness of the solution set can be achieved at very little cost. The *greedy* curve on this same plot shows the tradeoff that the greedy algorithm obtains *knowing* the utility function. This result serves as a reference, as no efficient algorithm can match it without actually knowing the response function over all the sequences. Note that as we put all weight on diversity, as expected, GP-SELECT's performance converges to that of the greedy algorithm.

Non-Uniform Costs.

The vaccine design task also provides a natural motivation for the non-uniform costs setting. Typically, the cost of testing depends on the actual sequence being tested. Also, field tests differ markedly in their cost of execution. For our dataset, we did not have the costs associated with testing. However, we simulated non-uniform costs for selection of the peptide sequences by sampling c_v uniformly from the range $[c_{min}, c_{max}]$. For different values of $[c_{min}, c_{max}]$, we found that GP-SELECT performed better than all the baselines considered. Note that we have used the greedy solution as the hindsight optimal one and this is known to be at most a factor of 2 away from the true optimal solution. While the performance was similar for different values of c_{min} and c_{max}, we report results of one of the settings in Figure 2 (b) where $c_{min} = 2$ and $c_{max} = 7$.

6.3 Case Study III: News Recommendation

The Yahoo! Webscope dataset R6A [2] consists of more than 45 million user visits to the *Yahoo! Today* module collected over 10 days in May 2009. The log describes the interaction (view/click) of each user with one randomly chosen article out of 271 articles. It was originally used as an unbiased evaluation benchmark for bandit algorithms [21, 34]. Each user u and each article a is described by a 6 dimensional feature vector. That is, $u \in \mathbb{R}^6$ and $a \in \mathbb{R}^6$. Thus, each possible interaction can be represented by a 36 dimensional feature vector (obtained from the vectorized outer product of user and item features) with a click (1) or no-click (0) as the outcome. Chu et al. [5] present a detailed description of the dataset, features and the collection methodology.

In our experiments, we consider an application where we seek to select a subset of articles to present to a subset of users. Hence, we sequentially pick user-item pairs aiming to maximize the number of clicks under a constraint on the number of interactions. Here, a very natural constraint is that we do not want to repeatedly show the same item to the same user. We randomly subsample 4 million user visits from the Webscope log and treat each interaction as an item with a latent reward that can be observed only when that item is picked. As baseline, we also compute the best fixed predictor of the reward given the entire log a priori. This serves as an unrealistic benchmark to compare our algorithm and other baselines against. We also compare against the other baselines used in the vaccine design task.

For GP-SELECT, we use the linear kernel to model similarities between the interactions. This is just the Kronecker product ($\otimes$) of the individual linear kernels on the users and items. We simulate the selection of 100,000 interactions. The total number of clicks in the dataset (of size 4 million) is 143,664, resulting in an average clickthrough rate (CTR) of about 0.0359.

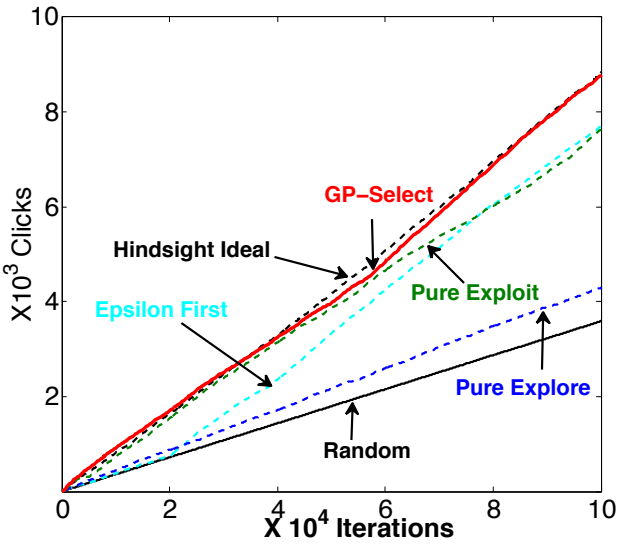
Results.

Of the 100,000 selected items, the hindsight-ideal algorithm discovers 8836 items that were eventually clicked on. In comparison, GP-SELECT discovers 8768 items beating the other baselines by at least 10%. This corresponds to a CTR of 0.0877 which is considerably higher than the average CTR in our dataset. The next best approach is the Epsilon First approach that randomly selects items for 20% of its budget and then trains a classifier to predict the reward for the remaining items. Detailed results are presented in Figure 3 (a).

Scaling to web scale datasets

The major bottleneck in using Gaussian Processes is the computation of the posterior mean and variance. There are several works that attempt to speed up GP-based algorithms [20, 37], which we can immediately benefit from. Also, our task can be inherently parallelized by distributing the computation across multiple cores/machines and a central processor collects the top UCB scores and picks the

[2] `http://webscope.sandbox.yahoo.com/`



(a) *Maximizing clicks on a web-scale recommendation task*

	Naive variance update	Lazy variance update
Avg. time for one update	5400ms (for 4m updates)	4.6ms (1 update)
Number of updates	400 Billion (Predicted)	$\sim$ 6 Billion (Actual)
Execution Time	150 hours (Predicted)	3.9 hours (Actual)

(b) *Performance Improvements*

Figure 3: Experiments on the news recommendation dataset. (a): GP-Select outperforms all the baselines by at least 10% while almost discovering as many clicks (8768) as the hindsight ideal (8863). (b): Our failsafe approach for lazy variance updates achieves almost 40X speedup.

one with the best from all the machines. The reward for the chosen item along with the item itself is communicated to the worker nodes which use the information to update the posterior mean and variances.

To obtain further improvements, we adapt the idea of lazy variance updates, originally proposed by Desautels et al. [7] for the bandit setting, and extend it with a novel failsafe variant. We note that the majority of the computation time is spent on computing the posterior variance update, which requires solving a linear system for each item. The key insight is that, for a given item v, $\sigma_t^2(v)$ is monotonically decreasing in t. We exploit this to recompute $\sigma(t)$ only for those items that could influence the selection in round t, via use of a priority queue. That is, in every round, we lazily pick the next item v_t based on the variance bound from the previous round and update the UCB score for that item. If v_t remains the selected item with the new score, we do not need to recompute the variances for the other items. We repeat this process until we find an item whose position at the head of the priority queue does not change after recomputation of the variance. However, note that if we have to recompute for many items in one round, it might be faster to update the variance for items due to the computational overhead associated with using a priority queue (and the benefits of parallelism). Thus, we include a failsafe condition whereby on crossing a machine and task dependent threshold on the number of lazy updates in one round, we switch to the full update. Thus, we eliminate the possibility that a large number of non-contiguous updates might be much slower than one full contiguous update for all the items. Using this technique, we achieve a reduction factor of almost 70 in the number of updates and an overall speedup of almost 40 in terms of computational time. The results are presented in Figure 3 (b).

7. RELATED WORK

Frequent itemset mining [11] is an important area of research in data mining. It attempts to produce subsets of items that occur together often in transactions on a database. However, it is very different in nature from AVID, the problem we address in this paper, since we do not optimize frequency, but (unknown) value.

Active learning algorithms select limited training data in order to train a classifier or regressor. Uncertainty sampling, expected model improvement, expected error reduction, variance reduction are some of the popular metrics in use in this field [28]. In (budgeted) active learning, the objective is to learn a function (regression or classification) as well as possible given a limited number of queries. In contrast, we do not seek to learn the function accurately, but only to choose items that maximize the cumulative value (e.g., the number of positive examples) of a function.

Active Search aims to discover as many members of a given class as possible [9]. Here, the authors propose single and (computationally expensive) multi-step look ahead policies. It is not clear however how their approach can be applied to regression settings, and how to select diverse sets of items. Furthermore, they do not provide any performance guarantees. Wang et al. [36] extended this approach to present a myopic greedy algorithm that scales to thousands of items. Warmuth et al. [38] proposed a similar approach based on batch-mode active learning for drug discovery. The algorithms proposed in these works are similar to our exploit-only baseline and further, work only for classification tasks.

Multi-arm bandit (MAB) problems are sequential decision tasks, where one repeatedly selects among a set of items ("arms"), and obtains noisy estimates of their values [19]. They abstract the explore – exploit dilemma. In contrast to our setting, in MAB, arms can be selected repeatedly: Choices made do *not* restrict arms available in the fu-

ture. In fact, our setting is a strict generalization of the bandit problem. Early approaches like Auer et al. [1] addressed the setting where utilities are considered independent across arms, and hence cannot generalize observations across arms. More recent approaches [3, 6, 15] address this shortcoming by exploiting assumptions on the regularity of the utility function. In particular, Srinivas et al. [30] develop a bandit algorithm, GP-UCB, with regret bounds whenever regularity is captured via a kernel function, which we build on and extend in our work. In other extensions (e.g. [13, 32]), the authors consider picking multiple arms per round. However, in these settings, subset selection is a repeated task with the same set of arms available for selection each time. Also, Kleinberg et al. [16] consider the case where only a subset of arms are available in each round. However, their results do not apply to our case where arms becomes unavailable upon being selected just once.

Stochastic Knapsacks. Budget limited explore-exploit problems have been studied in context of the stochastic knapsack problem. Hereby, the learning process is constrained by available resources. Gupta et al. [10] provide strong regret bounds for the scalar budget case. Tran-Thanh et al. [33] consider prior-free learning for the same problem. Badanidiyuru et al. [2] study the problem under multi-dimensional budget constraints. However, all approaches consider arms as independent (i.e., uncorrelated), and hence do not generalize observations across similar arms as we do.

Submodularity is a natural notion of diminishing returns of subsequent choices that arises in many applications in machine learning and other domains. A celebrated result about the performance of the greedy algorithm by Nemhauser et al. [24] allows fast yet near-optimal approximation algorithms to a number of NP-hard problems. However, these approaches assume that the underlying utility function is known, whereas here we attempt to learn it. Streeter and Golovin [31] use submodular function maximisation to solve online resource allocation tasks.

Diversity inducing rankings and selection have been studied in a variety of settings (e.g. [29]). In particular, submodular objective functions are proposed and used by Kim et al. [14], Lin and Bilmes [22], Streeter et al. [32], Yue and Guestrin [40] to model and optimize for diverse solutions. These approaches provide insights on how to quantify preference for diversity via submodularity. However, their algorithms do not apply to our setting, as they consider the setting where sets are repeatedly selected, whereas we build up a single set one element at a time.

Lazy variance updates in explore-exploit settings were proposed by Desautels et al. [7] who generalized the lazy greedy algorithm for submodular maximization [23]. We have adapted this approach to propose a failsafe lazy variance update technique that gives dramatic speedups in our experiments.

8. CONCLUSIONS

We introduced AVID - Adaptive Valuable Item Discovery, a novel problem setting capturing many important real world problems. We presented GP-Select, a theoretically well founded approach to select high-value subsets from a large pool of items. We further showed how it can be extended to select diverse subsets, by adding a submodular diversity term to the objective function, and how to handle non-uniform cost. We prove regret bounds for all these settings. We further demonstrated the effectiveness on three real world case studies of industrial relevance. To enable the application of GP-Select to web-scale problems, we proposed a failsafe lazy evaluation technique that dramatically speeds up execution of GP-Select. Empirically, we find that GP-Select allows us to obtain a fine-grained tradeoff between value and diversity of the selected items. We believe our results present an important step towards addressing complex, real-world exploration–exploitation tradeoffs.

Acknowledgments.

This research was supported in part by SNSF grant 200020_159557, ERC StG 307036 and a Microsoft Research Faculty Fellowship. The authors wish to thank Christian Widmer for providing the MHC data.

References

[1] P. Auer, N. C. Bianchi, and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, May 2002.

[2] A. Badanidiyuru, R. Kleinberg, and A. Slivkins. Bandits with knapsacks. *CoRR*, abs/1305.2545, 2013.

[3] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. Online optimization in X-armed bandits. In *In Advances in Neural Information Processing Systems (NIPS)*, 2008.

[4] G. Choquet. Theory of capacities. *Annales de l'institut Fourier*, 5:131–295, 1954.

[5] W. Chu, S.-T. Park, T. Beaupre, N. Motgi, A. Phadke, S. Chakraborty, and J. Zachariah. A case study of behavior-driven conjoint analysis on yahoo!: Front page today module. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 1097–1104, 2009.

[6] V. Dani, T. P. Hayes, and S. Kakade. The price of bandit information for online optimization. In *In Advances in Neural Information Processing Systems (NIPS)*, 2007.

[7] T. Desautels, A. Krause, and J. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research (JMLR)*, 2014.

[8] U. Feige. A threshold of ln n for approximating set cover. *Journal of ACM*, 45(4):634–652, July 1998.

[9] R. Garnett, Y. Krishnamurthy, X. Xiong, J. Schneider, and R. Mann. Bayesian optimal active search and surveying. In *Proceredings of the 29th Annual International Conference on Machine Learning (ICML)*, 2012.

[10] A. Gupta, R. Krishnaswamy, M. Molinaro, and R. Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *FOCS*, pages 827–836, 2011.

[11] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

[12] L. Jacob and J.-P. Vert. Efficient peptide-MHC-I binding prediction for alleles with few known binders. *Bioinformatics*, 24(3):358–366, Feb 2008.

[13] S. Kale, L. Reyzin, and R. E. Schapire. Non-stochastic bandit slate problems. In *In Advances in Neural Information Processing Systems (NIPS)*, pages 1054–1062, 2010.

[14] G. Kim, E. P. Xing, L. Fei-Fei, and T. Kanade. Distributed Cosegmentation via Submodular Optimization on Anisotropic Diffusion. In *13th International Conference on Computer Vision (ICCV 2011)*, 2011.

[15] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *STOC*, pages 681–690, 2008.

[16] R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Machine Learning*, 80(2-3):245–272, 2010.

[17] A. Krause and C. S. Ong. Contextual gaussian process bandit optimization. In *In Advances in Neural Information Processing Systems (NIPS)*, 2011.

[18] A. Kulesza and B. Taskar. Determinantal point processes for machine learning. *Machine Learning*, 5(2-3): 123–286, 2012.

[19] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[20] N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *NIPS*, pages 609–616, 2002.

[21] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 661–670, 2010.

[22] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 510–520, 2011.

[23] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, volume 7, pages 234–243. 1978.

[24] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions-1. *Mathematical Programming*, 1978.

[25] B. Peters, H. Bui, S. Frankild, M. Nielsen, C. Lundegaard, E. Kostem, D. Basch, K. Lamberth, M. Harndahl, W. Fleri, S. Wilson, J. Sidney, O. Lund, S. Buus, and A. Sette. A Community Resource Benchmarking Predictions of Peptide Binding to MHC-I Molecules. *PLoS Comput Biol*, 2(6), June 2006.

[26] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

[27] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

[28] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2012.

[29] A. Slivkins, F. Radlinski, and S. Gollapudi. Learning optimally diverse rankings over large document collections. In *International Conference on Machine Learning(ICML)*, pages 983–990, 2010.

[30] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, May 2012.

[31] M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems 21*, pages 1577–1584. 2008.

[32] M. Streeter, D. Golovin, and A. Krause. Online learning of assignments. In *In Advances in Neural Information Processing Systems (NIPS)*, 2009.

[33] L. Tran-Thanh, A. C. Chapman, E. M. de Cote, A. Rogers, and N. R. Jennings. Epsilon-first policies for budget-limited multi-armed bandits. In *AAAI*, 2010.

[34] H. P. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 225–232. ACM, 2014.

[35] H. P. Vanchinathan, A. Marfurt, C.-A. Robelin, D. Kossmann, and A. Krause. Discovering Valuable Items from Massive Data. *ArXiv e-prints*, June 2015. URL `http://arxiv.org/abs/1506.00935`.

[36] X. Wang, R. Garnett, and J. G. Schneider. Active search on graphs. In *KDD*, pages 731–738, 2013.

[37] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. De Freitas. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1778–1784, 2013.

[38] M. K. Warmuth, J. Liao, G. Rätsch, M. Mathieson, S. Putta, and C. Lemmen. Support vector machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, 43:667–673, 2003.

[39] C. Widmer, N. Toussaint, Y. Altun, and G. Ratsch. Inferring latent task structure for Multitask Learning by Multiple Kernel Learning. *BMC Bioinformatics*, 11 (Suppl 8):S5+, 2010.

[40] Y. Yue and C. Guestrin. Linear submodular bandits and their application to diversified retrieval. In *In Advances in Neural Information Processing Systems (NIPS)*, Granada, Spain, December 2011.