Neural Design for Genetic Perturbation Experiments

Aldo Pacchiano APACCHIANO@MICROSOFT.COM

Drausin Wulsin DRAUSIN@IMMUNAI.COM

Luis Voloch LUIS@IMMUNAI.COM

Robert Barton ROBERT.BARTON@IMMUNAI.COM

Abstract

The problem of how to genetically modify cells in order to maximize a certain cellular phenotype has taken center stage in drug development over the last few years (with, for example, genetically edited CAR-T, CAR-NK, and CAR-NKT cells entering cancer clinical trials). Exhausting the search space for all possible genetic edits (perturbations) or combinations thereof is infeasible due to cost and experimental limitations. This work provides a theoretically sound framework for iteratively exploring the space of perturbations in pooled batches in order to maximize a target phenotype under an experimental budget. Inspired by this application domain, we study the problem of batch query bandit optimization and introduce the Optimistic Arm Elimination (OAE) principle designed to find an almost optimal arm under different functional relationships between the queries (arms) and the outputs (rewards). We validate that OAE outperforms other other strategies in finding optimal actions in experiments on simulated problems, public datasets well-studied in bandit contexts, and in genetic perturbation datasets when the regression model is a deep neural network. OAE also outperforms the benchmark algorithms in 3 of 4 datasets in the GeneDisco experimental planning challenge.

Keywords: Active Learning, Optimism, Genetic Perturbation

1. Introduction

We are inspired by the problem of finding the genetic perturbations that maximize a given function of a cell (a particular biological pathway or mechanism, for example the proliferation or exhaustion of particular immune cells) while performing the least number of perturbations required. In particular, we are interested in prioritizing the set of genetic knockouts (via shRNA or CRISPR) to perform on cells that would optimize a particular scalar cellular phenotype. Since the space of possible perturbations is very large (with roughly 20K human protein-coding genes) and each knockout is expensive, we would like to order the perturbations strategically so that we find one that optimizes the particular phenotype of interest in fewer total perturbations than, say, just brute-force applying all possible knockouts.

With this objective in mind we propose a simple method for improving a cellular phenotype under a limited budget of genetic perturbation experiments. Although this work is inspired by this concrete biological problem, our results and algorithms are applicable in much more generality to the setting of experimental design with neural network models. We develop the

Optimistic Arm Elimination (OAE) algorithm for the batch query bandit problem and show that it is compatible with the use of neural network function approximation. During each time-step OAE fits a reward model on the observed responses seen so far while at the same time maximizing the reward on all the arms yet to be pulled. OAE then queries the batch of arms whose predicted reward is maximal among those not tried out yet.

We conduct a series of experiments on synthetic and public data from the UCI (3) database and show that OAE is able to find the optimal "arm" using fewer batch queries than other algorithms such as greedy and random sampling. We validate OAE on the public CMAP dataset (16), which contains tens of thousands of genetic shRNA knockout perturbations, and show that it always outperforms a baseline and almost always outperforms a simpler greedy algorithm in both convergence speed to an optimal perturbation and the associated phenotypic rewards. These results illustrate how perturbational embeddings learned from one biological context can still be quite useful in a different biological context, even when the reward functions of these two contexts are different. Finally we also benchmark our methods in the GeneDisco dataset and algorithm suite (see (9)) and show OAE to be competitive against benchmark algorithms in the task of maximizing HitRatios.

2. Related Work

The field of Bayesian optimization has long studied the problem of optimizing functions severely limited by time or cost Jones et al. (4). For example, Srinivas et al. (15) introduce the GP-UCB algorithm for optimizing unknown functions. Other approaches based on adaptive basis function regression has also been used to model the payoff function as in (14). These algorithms have also been used in the drug discovery context. Mueller et al. (10) applied Bayesian optimization to the problem of optimizing biological phenotypes. Very recently, GeneDisco was released as a benchmark suite for evaluating active learning algorithms for experiment design in drug discovery (9). Perhaps the most relevant to our setting are the many works that study batch acquisition in Bayesian active learning such as (6; 5). In this work we move beyond the typical parametric and Bayesian assumptions from these works and provide an algorithm that works in conjunction with neural network models.

3. Problem Definition

Let $f_* : \mathcal{A} \to \mathbb{R}$ be a function over the set of actions, where $\mathcal{A} \subset \mathbb{R}^d$. We assume $f_* \in \mathcal{F}$ for some (known)[1] function class $\mathcal{F}$ such that the "reward" of action $a \in \mathcal{A}$ equals $f_*(a)$. Following the typical online learning terminology we call actions $\mathcal{A}$ as arms. In this work we allow $\mathcal{A}$ to be infinite, although we only consider finite $\mathcal{A}$ in practice.

In our setting the experiment designer interacts with f_* and $\mathcal{A}$ in a sequential manner. During the $t-$th round the learner is required to query a batch of $B \in \mathbb{N}$ arms $\{a_{t,i}\}_{i=1}^B \subset \mathcal{A}$ and observe responses $\{y_{t,i} = f_*(a_{t,i}) + \xi_{t,i}\}_{i=1}^B$ where $\xi_{t,i}$ is conditionally zero mean.

We will develop a procedure able to recover an 'almost optimal' arm $a \in \mathcal{A}$ in the least number of arm pulls possible. We consider the following optimality objective,

$\tau-$**quantile optimality.** Find an arm $a_\tau \in \mathcal{A}$ in the top $\tau-$quantile of $\{f_\star(a)\}_{a\in\mathcal{A}}$.

1. As our experimental results show, precise knowledge of $\mathcal{F}$ is not required in practice.

This objective is a measure of optimality better related to practical objectives used in evaluation, such as hit ratio in the GeneDisco benchmark library (9). We show in Section 5 that our algorithms are successful at producing almost optimal points under this optimality criteria after a small number of queries. The main challenge is designing a smart choice of batch queries $\{a_{t,i}\}_{i=1}^{B}$ as this obeys the competing objectives of exploring new regions of the arm space and zooming into others that have shown promising arm values.

At time t the learner will output a candidate approximate optimal arm $\hat{a}_t$ among all the arms whose labels she has queried (henceforth referred to as $\mathcal{D}_t$) for example by considering $(\hat{a}_t, \hat{y}_t) = \arg\max_{(a,y)\in\mathcal{D}_t} y$, the point with the maximal observed reward so far. We use the statistics of these values to gauge the performance of our algorithms in the public datasets we evaluate on. Our objective is to design algorithms (possibly randomized) for which the first timestep where a $\tau-$quantile optimal point $\hat{a}_t$ been proposed is as small as possible.

4. Optimistic Arm Elimination

With these objectives in mind, we introduce the Optimistic Arm Elimination Algorithm (OAE). We call $\mathcal{U}_t$ to the subset of arms yet to be queried by our algorithm. OAE produces a batch of query points of size b from[2] $\mathcal{U}_t$. Our algorithm starts by fitting an appropriate response predictor $\widetilde{f}_t : \mathcal{U}_t \to \mathbb{R}$ based on the historical query points $\mathcal{D}_t$ and their observed responses so far. Instead of just fitting the historical responses with a square loss and produce a prediction function $\widetilde{f}_t$, we encourage the predictions of $\widetilde{f}_t$ to be optimistic on points in $\mathcal{U}_t$.

We propose two ways of achieving this. First by fitting a model (or an ensemble of models) $\widetilde{f}_t^o$ to the data in $\mathcal{D}_t$ and explicitly computing a measure of uncertainty $\tilde{u}_t : \mathcal{U}_t \to \mathbb{R}$ of its predictions on $\mathcal{U}_t$. We define the optimistic response predictor $\widetilde{f}_t(a) = \widetilde{f}_t^o(a) + \tilde{u}_t(a)$. Second, we achieve this by defining $\widetilde{f}_t$ to be the approximate solution of a constrained objective,

$$\widetilde{f}_t = \arg\max_{f\in\mathcal{F}} \mathbb{A}(f, \widetilde{\mathcal{U}}_t) \qquad \text{s.t.} \sum_{(x,y)\in\mathcal{D}_t} (f(x) - y)^2 \le \gamma. \tag{1}$$

where $\gamma \ge 0$ and $\mathbb{A}(f, \mathcal{U})$ is an acquisition objective tailored to produce an informative arm from $\mathcal{U}_t$. We consider a couple of acquisition objectives, $\mathbb{A}_{\text{avg}}(f, \mathcal{U}) = \frac{1}{|\mathcal{U}|}\sum_{a\in\mathcal{U}} f(a)$, $\mathbb{A}_{\text{hinge}}(f, \mathcal{U}) = \frac{1}{|\mathcal{U}|}\sum_{a\in\mathcal{U}} (\max(0, f(a)))^p$ and $\mathbb{A}_{\text{softmax}}(f, \mathcal{U}) = \log\left(\sum_{a\in\mathcal{U}} \exp(f(a))\right)$. An important acquisition function of theoretical interest, although hard to optimize in practice is $\mathbb{A}_{\max}(f, \mathcal{U}) = \max_{a\in\mathcal{U}} f(a)$. Our algorithm then produces a query batch B_t by solving,

$$B_t = \arg\max_{\mathcal{B}\subset\mathcal{U}_t||\mathcal{B}|=b} \sum_{a\in B} \widetilde{f}_t(a). \tag{2}$$

When the acquisition function equals $\mathbb{A}_{\max}(f, \mathcal{U})$ the OAE algorithm balances exploration and exploitation by acting greedily with respect to a model that fits the rewards of the arms in $\mathcal{D}_t$ as accurately as possible but induces large responses from the arms she has not tried.

4.1 Tractable and Implementations of OAE

We start by discussing the case when the acquisition function equals $\mathbb{A}_{\text{avg}}(f, \mathcal{U})$ and see how we may implement this objective in practice. We mainly focus on this simple tractable

2. The batch equals $\mathcal{U}_t$ when $|\mathcal{U}_t| \le B$.

Algorithm 1 Optimistic Arm Elimination (OAE)

```
Input Action set $\mathcal{A} \subset \mathbb{R}^d$, num batches $N$, batch size $B$, $\lambda_{\text{reg}}$
Initialize Unpulled arms $\mathcal{U}_1 = \mathcal{A}$. Observed points and labels dataset $\mathcal{D}$
for $t = 1, \cdots, N$ do
    Solve for $\widetilde{f}_t$ and compute $B_t = \arg\max_{\mathcal{B} \subset \mathcal{U}_t || \mathcal{B}|=b} \sum_{a \in B} \widetilde{f}_t(a)$.
    Observe batch rewards $Y_t = \{y_{t,i} \text{ for } x_{t,i} \in B_t\}$
    Update $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(B_t, Y_t)\} \in \mathcal{D}$ and $\mathcal{U}_{t+1} = \mathcal{U}_t \backslash B_t$ .
```

setting here described in our experimental evaluation. Since solving problem 1 may prove intractable, in practice -when using DNN models- we consider an approximate regularized objective whose solution will produce a surrogate solution $\widehat{f}_t$,

$$\hat{f}_t = \arg\min_{f \in \mathcal{F}} \frac{1}{|\mathcal{D}_t|} \sum_{(x,y) \in \mathcal{D}_t} (f(x) - y)^2 - \frac{\lambda_{\text{reg}}}{|\mathcal{A}_u|} \sum_{u \in \mathcal{A}_u} f(u) \tag{3}$$

and define $B_t = \arg\max_{B \subseteq \mathcal{A}_u \text{ s.t. } |B|=b} \sum_{x \in B} \hat{f}_t(x)$. In Section 5, we use the acquisition function $\mathbb{A}_{\text{avg}}(f,\mathcal{U})$. We refer to this method as MeanOpt. In Appendix A the reader may find experiments where the acquisition function is set to $\mathbb{A}_{\text{softmax}}(f,\mathcal{U})$ and $\mathbb{A}_{\text{hinge}}(f,\mathcal{U})$ with $p = 4$. We also consider a ensemble method (Ensemble) that fits M models $\{\widehat{f}_t^i\}_{i=1}^M$ to $\mathcal{D}_t$ and defines a surrogate solution $\hat{f}_t(a) = \max_i \hat{f}_t^i(a)$ for all $a \in \mathcal{U}_t$.

5. Experiments

We demonstrate the effectiveness of our OAE algorithm in several problem settings across public and synthetic datasets. We evaluate the algorithmic implementations described in Section 4.1 by setting the acquisition function to $\mathbb{A}_{\text{avg}}(f,\mathcal{U})$ and the batch selection rule as in Equation 2 and the ensemble size $M = 10$. We test OAE's performance over different values of the regularization parameter λ_{reg}, including the 'greedy' choice of $\lambda_{\text{reg}} = 0$, henceforth referred to as Greedy. We compare these algorithms to RandomBatch the baseline method that selects points B_t by selecting a uniformly random batch of size B from $\mathcal{U}_t$.

5.1 Public Supervised Learning Datasets

We test our methods on public regression (BikeSharingDay, BikeSharingHour, BlogFeedback) datasets from the UCI repository (3). To evaluate our algorithm we assume the response (regression target or binary classification label) values are noiseless. Each observation i in a dataset represents a discrete action with feature a_i and reward $f_*(a_i)$.

Figure 1 shows results for BikeSharingDay and BlogFeedback. We use a neural network with two hidden layers of size 100 and 10 trained using $5,000$ batch gradient steps (with a batch size of 10). In this case Ensemble outperforms both RandomBatch and Greedy in τ-quantile convergence time on BlogFeedback dataset. Low-λ MeanOpt also performs competitively. Thus, the optimal optimism scaling may be a property of the dataset.

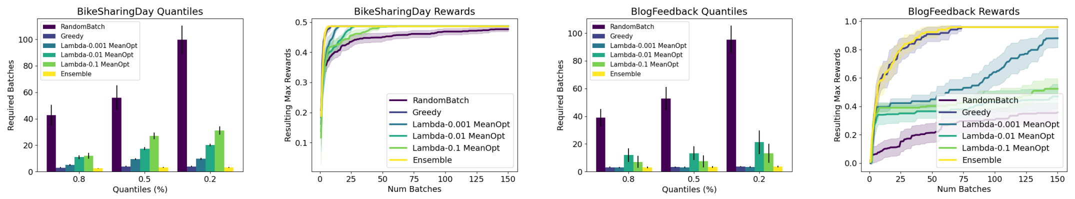


Figure 1: Algorithm τ-quantile batch convergence (**left**) and corresponding rewards over batches (right) on public BikeSharingDay and BlogFeedback datasets with original

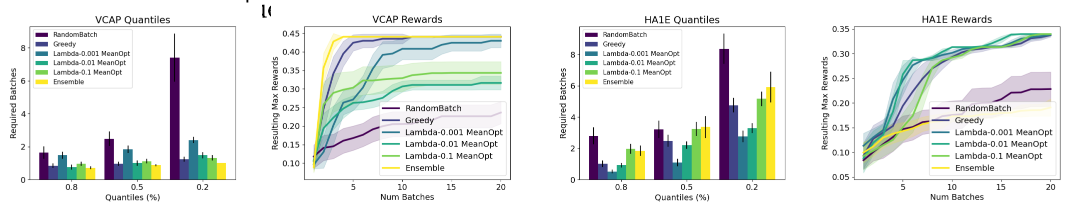


Figure 2: Algorithm τ-quantile batch convergence (**left**) and corresponding rewards over batches (right) on genetic perturbation datasets.

5.2 Transfer Learning Across Genetic Perturbation Datasets

In order to show the effectiveness of OAE in the large batch - small number of iterations regime we consider genetic perturbations from the CMAP dataset (16), which contains a 978-gene bulk expression readout from thousands of single-gene shRNA knockout perturbations[3] across a number of cell lines. We consider the setting in which we have observed the effect of knockouts in one biological context (i.e., cell line) and would like to use it to plan a series of knockout experiments in another. Related applications may have different biological contexts, from different cell types or experimental conditions. We use the level 5 CMAP observations, each of which contains of 978-gene transcriptional readout from an shRNA knockout of a particular gene in a particular cell line. In our experiments, we choose to optimize a cellular proliferation phenotype, defined as a function on the 978-gene expression space. See Appendix B for details.

We use the 4 cells lines with the most number genetic perturbations in common: VCAP (prostate cancer, $n = 14602$), HA1E (kidney epithelium, $n = 10487$), MCF7 (breast cancer, $n = 6638$), and A375 (melanoma, $n = 10033$). We first learn a 100-dimensional action (perturbation) embedding a_i for each perturbation in VCAP with an autoencoder. The autoencoder has a 100-dimension bottleneck layer and two intermediate layers of 1500 and 300 ReLU units with dropout and batch normalization and is trained using the Adam optimizer on mean squared reconstruction loss. We use these 100-dimensional perturbations embeddings as the features to train the $\widetilde{f}_t$ reward functions for each of the other cell types. According to our OAE algorithm, we train a fresh feed-forward neural network with two intermediate layers (of 100 and 10 units) for after observing the phenotypic rewards for each batch of 50 gene (knockout) perturbations.

Figure 2 shows the convergence and reward results for two of the 4 cell lines. Since the perturbation action features were learned on VCAP (though agnostic to any phenotypic reward), the optimal VCAP perturbations are found quite quickly by both the Greedy

3. The shRNA perturbations are just a subset of the 1M+ total perturbations across different perturbation classes.

and OAE relative to RandomBatch. Interestingly, OAE still outperforms RandomBatch and Greedy in the HA1E cell line. Results for the MCF7 and A375 cell lines are similar (Appendix A.3). This performance indicates that OAE is robust to a degree of difference in the biological contexts of the perturbation embedding and the phenotypic reward.

Dataset	TopUncertain	SoftUncertain	OAE
Schmidt et al. 2021 (IFNg)	0.057	0.046	**0.062**
Schmidt et al. 2021 (IL2)	0.083	0.081	**0.107**
Zhuang et al. 2019 (NK)	0.035	0.047	**0.085**
Zhu et al. 2021 (SarsCov2)	0.035	**0.049**	0.0411

Figure 3: Hit Ratio Results after 50 batches of size 16. BNN model trained with Achilles descriptors. Final Hit Ratio average of 5 runs. TopUncertain selects the 16 points with the largest uncertainty and SoftUncertain samples them using a softmax.

6. GeneDisco Experimental Planning Benchmark

We test our OAE algorithm against the GeneDisco benchmark (9), which assess the "Hit Rate" of different experimental planning algorithms over a number of pooled CRISPR experiments. We assess our performance against the other acquisition functions provided in the public implementation[4] that select batches based solely on uncertainty considerations. We use the public implementation of GeneDisco and do not change the neural network architecture provided corresponding to a Bayesian Neural Network with a hidden layer of size 32. We use the 808 dimensional Achilles treatment descriptors and built $\widetilde{f}_t$ by adding the BNN's uncertainty to the model base predictions. We test our algorithm in the Schmidt et al. 2021 (IFNg), Schmidt et al. 2021 (IL2), Zhuang et al. 2019 (NK), and Zhu et al. 2021 (SarsCov2) datasets and tested for performance using the Hit Ratio metric after collecting 50 batches of size 16. This is defined as the ratio of arm pulls lying in the top .05 quantile of genes with the largest absolute value. Our results are presented in Table 3. OAE outperforms the other algorithms by a substantial margin in 3 out of the 4 datasets that we tested.

7. Conclusion

In this work we introduce the OAE algorithm for batch bandit optimization and show lower bounds for the query complexity for linear and Lipshitz classes. Through a variety of experiments in synthetic, public and biological data we show that method can quickly search through a space of actions for the almost optimal ones. We show OAE need not be faced with a target function from a known regression class to outperform the RandomBatch and Greedy baselines. For example the perturbational embeddings learned in one biological context can predict with some accuracy the rewards in another context with a different reward function and be used in conjunction with OAE. Finally, our approach outperforms the other benchmark methods in the public GeneDisco problem in three of four datasets we tested. An exciting avenue for future work is to design adaptive algorithms that can select the best OAE implementation for the dataset at hand. We hypothesize this could be achieved by leveraging model selection techniques from any of (12; 2; 11; 8; 1).

4. `https://github.com/genedisco/genedisco-starter`

Appendix A. Additional Experimental Details

A.1 Synthetic One Dimensional Datasets

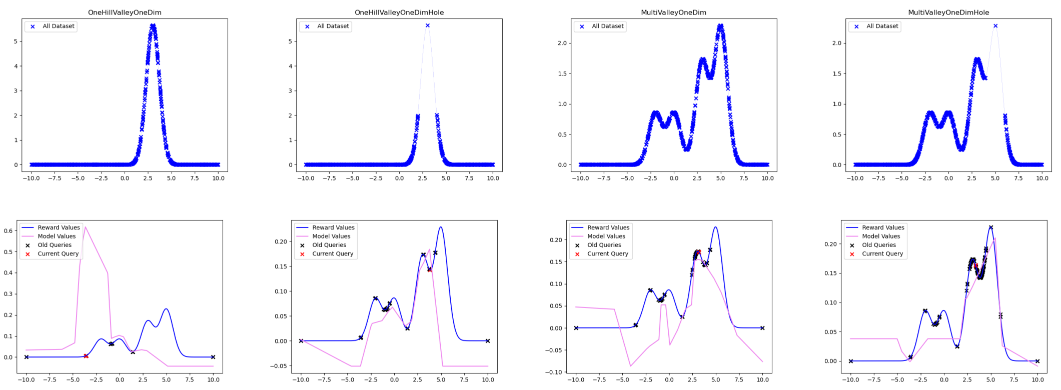


Figure 4: (top row) Synthetic one dimensional datasets (bottom) Evolution of OAE in the MultiValleyOneDimHole dataset using $\lambda_{\text{reg}} = 0.001$. *From left to right:* Iterations $5, 15, 25, 45$.

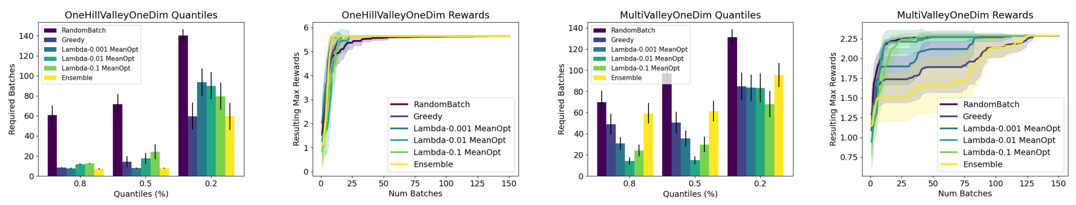


Figure 5: Algorithm τ-quantile batch convergence (**left**) and corresponding rewards over batches (right) on the OneHillValleyOneDim (easier) and MultiValleyOneDim (harder) simulated datasets show how the OAE algorithm can achieve higher reward faster than RandomBatch or Greedy.

Figure 4 shows different one dimensional synthetic datasets that we used to validate our methods. The leftmost, the OneHillValleyOneDim dataset consists of 1000 arms uniformly sampled from the interval $[-10, 10]$. The responses y are unimodal. The learner's goal is to find the arm with $x-$coordinate value equals to 3 as it is the one achieving the largest response. We use the dataset OneHillValleyOneDimHole to test for OAE's ability to find the maximum when the surrounding points are not present in the dataset.

The remaining two datasets MultiValleyOneDim and MultiValleyOneDimHole are built with the problem of multimodal optimization in mind. Each of these datasets have 4 local maxima. We use MultiValleyOneDim to test the OAE's ability to avoid getting stuck in local optima. The second dataset mimics the construction of the OneHillValleyOneDimHole dataset and on top of testing the algorithm's ability to escape local optima, it also is meant to test what happens when the global optimum's neighborhood isn't present in the dataset. Since one of the algorithms we test is the greedy algorithm (corresponding to $\lambda = 0$), the 'Hole' datasets are meant to present a challenging situation for this class of algorithms.

We use the same reward neural network architecture across all experiments in this work: 2 hidden layers of 100 and 10 units with ReLU activation functions trained for 5000 steps via the Adam optimizer using batches of size 10. We use batch size $B = 3$ and repeat each experiment a total of 25 times, reporting average results with standard error bars at each time step. We compare OAE with a simple RandomBatch strategy that selects a random batch of size B from the dataset points that have not been queried yet and a Greedy algorithm corresponding to OAE with $\lambda_{\text{reg}} = 0$.

Figure 5 shows representative results for the OAE algorithm with ($\lambda_{\text{reg}} = 0.001, 0.01, 0.1$), Ensemble, Greedy, and RandomBatch algorithms evaluated for two of the simulated datasets. (See Figure 6 for more results.) In all of the experiments we have conducted on these synthetic one dimensional datasets setting λ_{reg} optimistically (i.e. $\lambda_{\text{reg}} > 0$) outperforms RandomBatch and $\lambda_{\text{reg}} > 0$ substantially outperform Greedy in the MultiValley dataset. Although the RandomBatch method is fast at the start of optimization in finding points with large reward values it takes longer than OAE and Greedy in converging to the exact optimum in both one dimensional unimodal datasets.

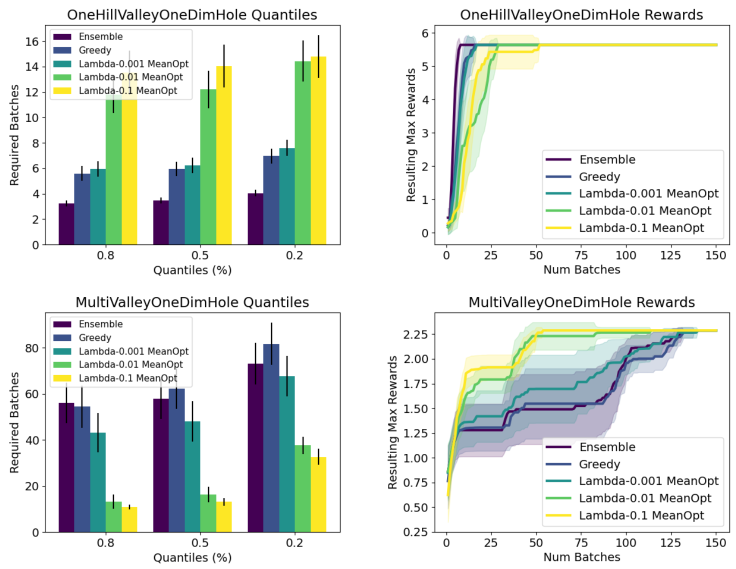


Figure 6: Algorithm τ-quantile batch convergence (**left**) and corresponding rewards over batches (right) on the OneHillValleyOneDimHole (easier) and MultiValleyOneDimHole (harder) simulated datasets. The ensemble method outperforms all other approaches on the easy dataset while the $\lambda-$optimism approaches worked best in the multi valley dataset. In all cases optimistic approaches beat both RandomBatch and Greedy.

A.2 Public Supervised Learning Datasets

We test our methods on public binary classification (Adult, Bank) and regression (BikeSharingDay, BikeSharingHour) datasets from the UCI repository (3).

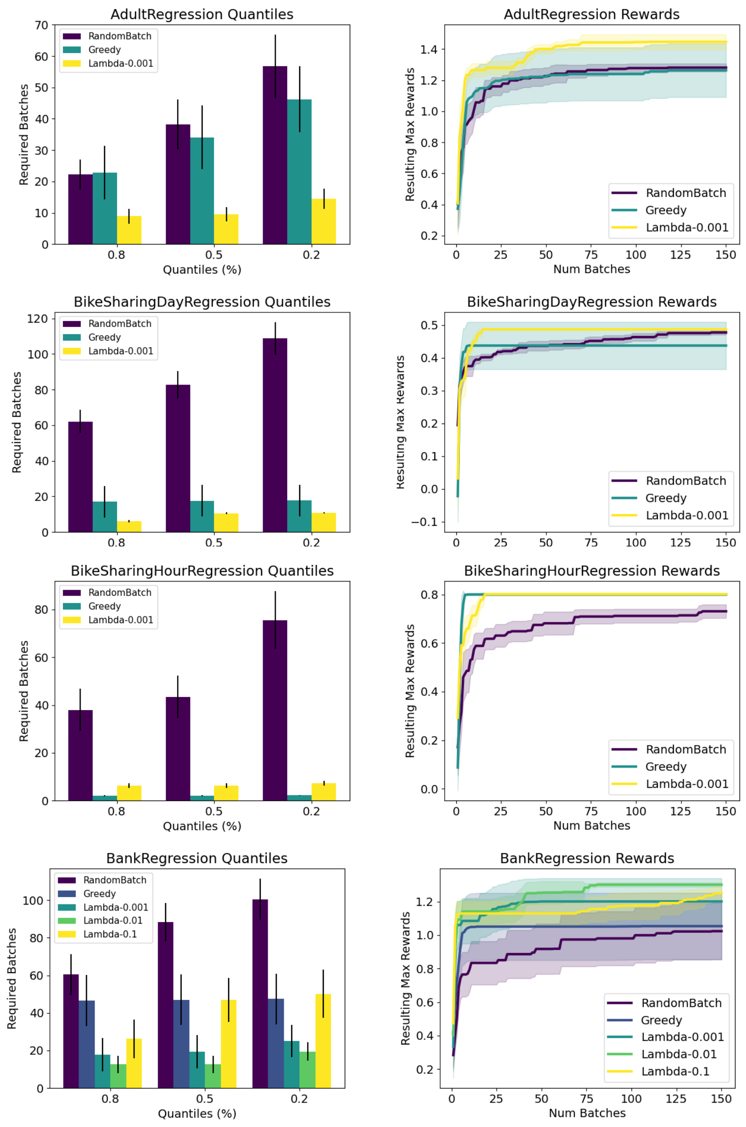


Figure 7: Algorithm τ-quantile batch convergence (**left**) and corresponding rewards over batches (**right**) on the Adult, BikeSharingDay, BikeSharingHour and Bank datasets with regression-fitted response values. Optimistic approaches outperform RandomBatch.

We first use all 4 public datasets to test OAE in the setting when the true function class $\mathcal{F}$ is known (in this case, a neural network) is known contain the function OAE learns over the course of the batches. We train a neural network under a simple mean squared error regression fit to the binary responses (for the binary classification datasets) or real-valued responses (for the regression datasets). This regression neural network consists of a neural network with two hidden layers of dimension 100 and 10 trained on the provided datasets using $5,000$ batch gradient steps (with a batch size of 10). For each dataset, we use the real-valued response predicted by our regressor as the reward for the corresponding action.

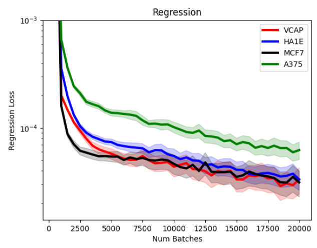


Figure 8: Regression mean squared error loss for models that predict cell-line specific phenotypic rewards from VCAP-derived perturbational features.

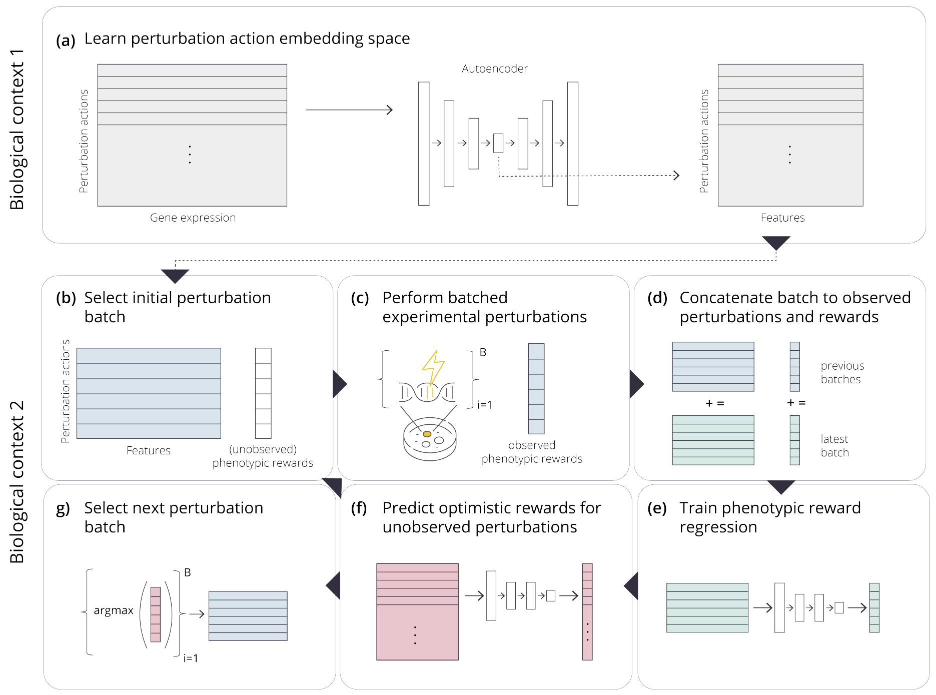
This ensures the dataset's true reward response model has the same architecture as the reward model used by OAE.

We use the same experimental parameters and comparison algorithms as in the synthetic dataset experiments. Figure 7 shows the results on the binary classification Adult, Bank and regression BikeSharingHour, BikeSharingDay datasets using these fitted responses. We observe the OAE algorithm handily outperforms RandomBatch on all datasets and Greedy on all except BikeSharingHour.

A.3 Transfer Learning Across Genetic Perturbation Datasets

Figure 8 shows the mean squared error loss of models trained to predict the cell-line specific phenotypic reward from the 100-dimensional VCAP-derived perturbational features. These models are trained on successive batches of perturbations sampled via RandomBatch. Not surprisingly, the loss for the VCAP reward is one of the lowest, but that of two other cell lines (HA1E and MCF7) are also quite similar, showing that our neural net regressor function class is flexible to learn the reward function in one context from the perturbational embedding in another.

In figure 10 we present the OAE performance in the A375 and MCF7 datasets and in figure 9 we have a diagramatic representation of the whole transfer learning pipeline we study in this section.



Biological context 1
(a) Learn perturbation action embedding space
Perturbation actions
Gene expression
Autoencoder
Perturbation actions
Features
Biological context 2
(b) Select initial perturbation batch
Perturbation actions
Features
(unobserved) phenotypic rewards
(c) Perform batched experimental perturbations
B
i=1
observed phenotypic rewards
(d) Concatenate batch to observed perturbations and rewards
previous batches
+ =
+ =
latest batch
g) Select next perturbation batch
argmax
B
i=1
(f) Predict optimistic rewards for unobserved perturbations
(e) Train phenotypic reward regression

Figure 9: Neural design for genetic perturbation experiments. (a) Learn a perturbation action embedding space by training an autoencoder on the gene expression resulting from a large set of observed genetic perturbations in a particular biological context (e.g., shRNA gene knockouts for a particular cell line in CMAP). (b) Select an initial batch of B perturbation actions to perform in parallel within a related (but different) biological context. Selection can be random (uniform) or influenced by prior information about the relationship between genes and the phenotype to be optimized. (c) Perform the current batch of experimental perturbations *in vitro* and observed their corresponding phenotypic rewards. (d) Concatenate the latest batch's features and observed rewards to those of previous batches to update the perturbation reward training set. (e) Train a new perturbation reward regression (with some degree of pre-defined optimism) network on the observed perturbation rewards. (f) Use this regressor to predict the optimistic rewards for the currently unobserved perturbations. (g) Select the next batch from these unobserved perturbations with the highest optimistic reward.

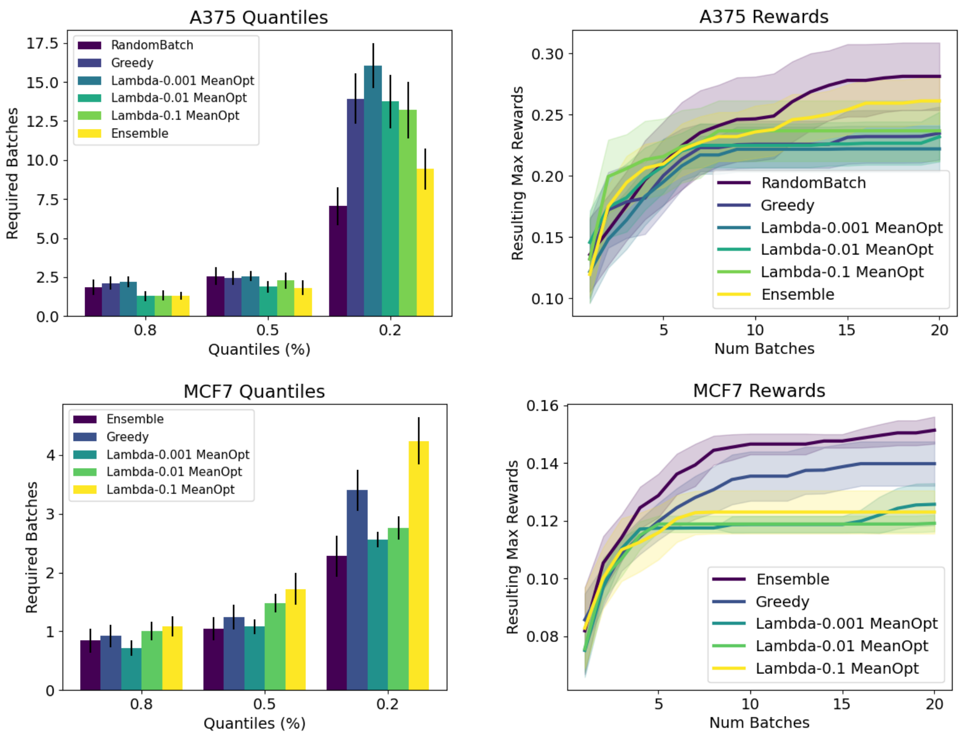


Figure 10: CMAP data. Batch size 50. Additional experiments.

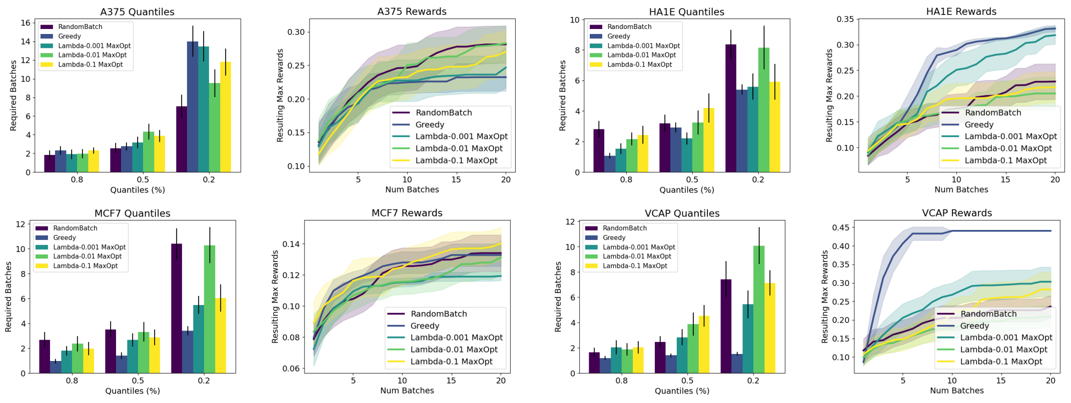


Figure 11: Softmax Optimism. CMAP data. Batch size 50.

A.4 Softmax Optimism

A.5 Hinge Optimism

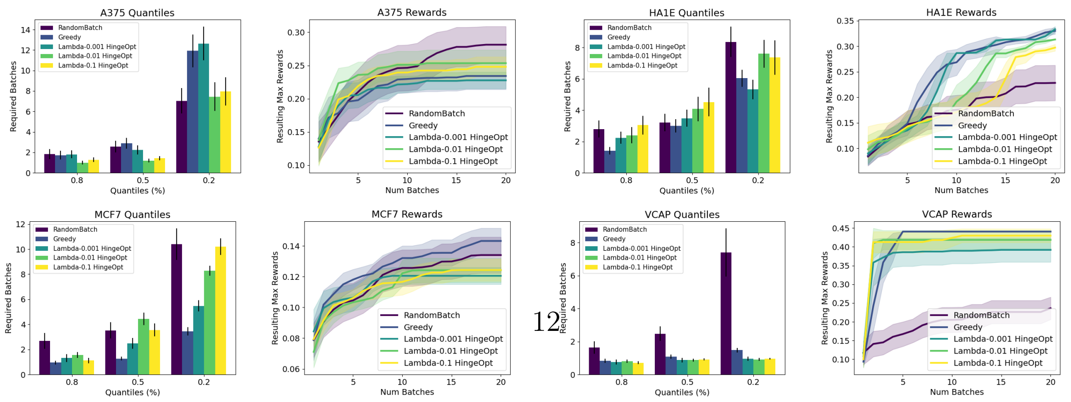


Figure 12: Hinge Optimism. CMAP data. Batch size 50.

Appendix B. Cellular proliferation phenotype

Let $\mathcal{G}$ be the list of genes present in CMAP also associated with proliferation phenotype according to the Seurat cell cycle signature, and let $x_{i,g}$ represent the level 5 gene expression of perturbation a_i for gene $g \in \mathcal{G}$. We define the proliferation reward for perturbation a_i as the average expression of the genes in $\mathcal{G}$,

$$f_*^{\text{prolif}}(a_i) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} x_{i,g}$$

For convenience, $\mathcal{G} = \{\text{AURKB}, \text{BIRC5}, \text{CCNB2}, \text{CCNE2}, \text{CDC20}, \text{CDC45},$
$\text{CDK1}, \text{CENPE}, \text{GMNN}, \text{KIF2C},$
$\text{LBR}, \text{NCAPD2}, \text{NUSAP1}, \text{PCNA}, \text{PSRC1},$
$\text{RFC2}, \text{RPA2}, \text{SMC4}, \text{STMN1}, \text{TOP2A},$
$\text{UBE2C}, \text{UBR7}, \text{USP1}\}$.

Appendix C. Diversity seeking versions of OAE

In the case $B > 1$, the explore / exploit trade-off is not the sole consideration in selecting the arms that make up B_t. In this case, we should also be concerned about selecting sufficiently diverse points within the batch B_t to cover the space. Since reward feedback is collected on the totality of the batch points at once, it may prove wasteful to query points that are very close by within the same batch. Nonetheless we did not see substantial gains from adding a diversity seeking term to our algorithms. The experimental details are described below. We conjecture this is because in these tasks we tested these approaches where the batch size is large, there may be enough inherent diversity in the initial batches that explicitly encouraging diversity may harm exploitation in subsequent steps.

C.1 Diversity via Determinants

Inspired by diversity-seeking methods in the Determinantal Point Processes (DPPs) literature (7), we introduce the $\mathrm{OAE} - \mathrm{DvD}$ algorithm. DPPs can be used to produces diverse subsets by sampling proportionally to the determinant of the kernel matrix of points within the subset. From a geometric perspective, the determinant of the kernel matrix represents the volume of a parallelepiped spanned by feature maps corresponding to the kernel choice. We seek to maximize this volume, effectively "filling" the feature space. Using a determinant score to induce diversity has been proposed as a strategy in other domains, most notably in the form of the Diversity via Determinants (DvD) algorithm from (13) for Population Based Reinforcement Learning. It is from this work that we take inspiration to name our algorithm $\mathrm{OAE} - \mathrm{DvD}$. The method works as follows. At time t, the learner will build a regression estimator $\widetilde{f}_t$ using the arms and responses contained in the data collected so far $\mathcal{D}_t$.

We assume access to a kernel function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Let $c_t = |\mathcal{U}_t|$.

We build a kernel matrix $\mathbf{K}_t \in \mathbb{R}^{c_t \times c_t}$ defined as,

$$\mathbf{K}_t[i,j] = \mathcal{K}(a_i, a_j), \quad \forall i,j \in \mathcal{U}_t.$$

For any subset $B \subseteq \mathcal{U}_t$ we define a diversity-aware score as,

$$\mathrm{Div}(B, f) = \mathbb{A}(f, B) + \lambda_{\mathrm{div}} \mathrm{Det}(\mathbf{K}_t[B, B])$$

Where $\mathbf{K}_t[B, B]$ corresponds to the submatrix of $\mathbf{K}_t$ with columns (and rows) indexed by B and λ_{div} is a diversity regularizer. In our experiments we use $\mathbb{A}_{\mathrm{avg}}$ as the batch score. Since this optimization problem may prove to be extremely hard to solve, we design a greedy maximization algorithm to produce a surrogate solution. We build the batch B_t greedily. The first point $a_t^{(1)}$ in the batch is selected to be the point in $\mathcal{U}_t$ that maximizes the response $\widetilde{f}_t$. For all $i \geq 2$ the point $a_t^{(i)}$ in $\mathcal{U}_t$ is selected from $\mathcal{U}_t \backslash \{a_t^{(j)}\}_{j=1}^{i-1}$ such that,

$$a_t^{(i)} = \max_{a \in \mathcal{U}_t \backslash \{a_t^{(j)}\}_{j=1}^{i-1}} \mathrm{Div}(\widetilde{f}_t, \{a\} \cup \{a_t^{(j)}\}_{j=1}^{i-1})$$

In our experiments we set $\lambda_{\mathrm{div}} = 1$ and set $\widetilde{f}_t$ to be the result of solving problem 3 for different values of λ_{reg}. We used a gaussian Kernel in our experiments. The remaining experimental details such as the neural network architecture and the number of experiment repetitions is the same as in the main.

Algorithm 2 Optimistic Arm Elimination - DvD (OAE – DvD)

```
**Input** Action set $\mathcal{A} \subset \mathbb{R}^d$, num batches $N$, batch size $B$, $\lambda_{\mathrm{reg}}$
**Initialize** Unpulled arms $\mathcal{U}_1 = \mathcal{A}$. Observed points and labels dataset $\mathcal{D}$
**for** $t = 1, \cdots, N$ **do**
    **if** $t = 1$ **then**
        Sample uniformly a size $B$ batch $B_t \sim \mathcal{U}_1$.
    **else**
        Solve for $\widetilde{f}_t$ via Equation 3.
        Compute $B_t$ using the Greedy procedure described above.
    Observe batch rewards $Y_t = \{f_*(x) \text{ for } x \in B_t\}$
    Update $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(B_t, Y_t)\} \in \mathcal{D}$.
    Update $\mathcal{U}_{t+1} = \mathcal{U}_t \backslash B_t$ .
```

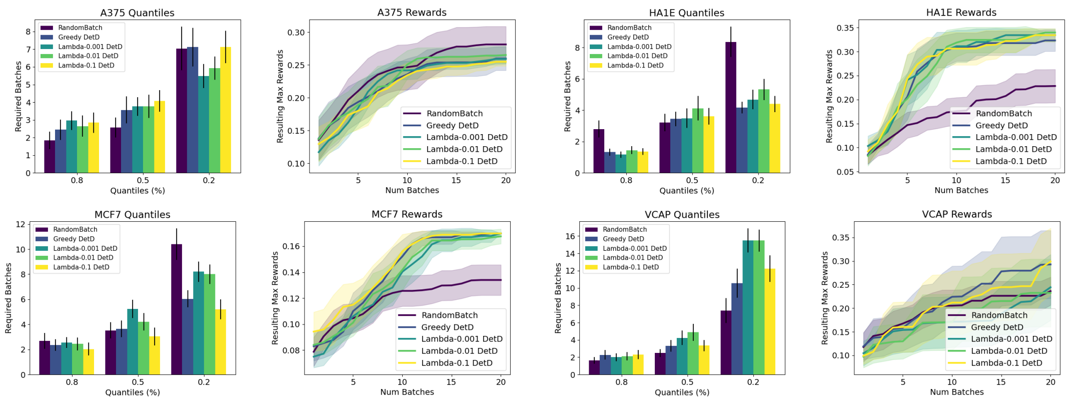


Figure 13: Determinants Diversity Guided Optimism. CMAP data. Batch size 50.

C.2 Sequential batch selection rules

In this section we introduce $\mathrm{OAE}-\mathrm{Seq}$ a more general form of the OAE algorithm. We propose an algorithm that produces a query batch by solving a sequence of B optimization problems. The first one of these makes use of $\mathcal{D}_t$, the set of arms pulled so far as well as $\mathcal{U}_t$ (the set of arms yet to be pulled) to produce a function $\widetilde{f}_{t,1}$ that is used to determine the initial arm in the batch via the greedy choice $a_{t,1} = \arg\max_{a\in\mathcal{U}_t} \widetilde{f}_{t,1}(a)$. Having chosen this arm, in the case when $B > 1$, a virtual reward $\widetilde{y}_{t,1}$ (possibly different from $\widehat{y}_t$) is assigned to the query arm $a_{t,1}$, and datasets $\widetilde{\mathcal{D}}_{t,1} = \mathcal{D}_t \cup \{(a_{t,1}, \widetilde{y}_{t,1})\}$ and $\widetilde{\mathcal{U}}_{t,1} = \mathcal{U}_t \backslash \{a_{t,1}\}$ are defined. The same optimization procedure that produced $\widetilde{f}_{t,1}$ is used to output $\widetilde{f}_{t,2}$ now with $\widetilde{\mathcal{D}}_{t,1}$ and $\widetilde{\mathcal{U}}_{t,1}$ as inputs. Arm $a_{t,2}$ is defined as the greedy choice $a_{t,2} = \arg\max_{a\in\widetilde{\mathcal{U}}_{t,1}} \widetilde{f}_{t,2}(a)$. The remaining batch elements (if any) are determined by successive repetition of this process. The trace of this procedure leaves behind a sequence of functions and datasets $\{(\widetilde{f}_{t,i}, \widetilde{\mathcal{U}}_{t,i})\}_{i=1}^{B}$ such that $a_{t,i} = \arg\max_{\widetilde{\mathcal{U}}_{t,i}} \widetilde{f}_{t,i}(a)$. The prediction function $\widetilde{f}_t$ may encourage optimism (or not).

Algorithm 3 Optimistic Arm Elimination - Batch Sequential ($\mathrm{OAE}-\mathrm{Seq}$)

```
Input Action set $\mathcal{A} \subset \mathbb{R}^d$, num batches $N$, batch size $B$, $\lambda_{\mathrm{reg}}$
Initialize Unpulled arms $\mathcal{U}_1 = \mathcal{A}$. Observed points and labels dataset $\mathcal{D}$
for $t = 1, \cdots, N$ do
    if $t = 1$ then
        Sample uniformly a size $B$ batch $B_t \sim \mathcal{U}_1$.
    else
        Solve for $\{(\widetilde{f}_{t,i}, \widetilde{\mathcal{U}}_{t,i})\}_{i=1}^{b}$
        Compute $B_t = \{a_{t,i} = \arg\max_{a\in\widetilde{U}_{t,i}} \widetilde{f}_{t,i}(a)\}_{i=1}^{b}$.
    Observe batch rewards $Y_t = \{f_*(x)$ for $x \in B_t\}$
    Update $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(B_t, Y_t)\} \in \mathcal{D}$.
    Update $\mathcal{U}_{t+1} = \mathcal{U}_t \backslash B_t$ .
```

If instead the querying procedure over arms $\{a_{t,1}\}_{i=1}^{b}$ took place sequentially, the discovery of reward values $\{y_{t,j}\}_{j\le i-1}$ can inform the learner what to try out next. In particular, if an early arm of the batch resulted in a small reward value, the learner would be imprudent to waste lots of query arms in the vicinity of that small value arm. For this reason, we introduce the use of fictitious rewards $\widetilde{y}_{t,i}$ when fitting the model $\widetilde{f}_{t,i}$ to ensure the empirical maximizer of the model $\widetilde{f}_{t,i}$ over arms $\widetilde{\mathcal{U}}_{t,i}$ takes into account the possibility that previous arm values $\{a_{t,j}\}_{j\le i-1}$ could have an underwhelming reward. Optimizing with these objectives in mind ensures OAE not only successfully trades off the need to explore new arms and exploit arms in high reward regions, but also that the batch B_t achieves diverse coverage of the arm space.

To determine the value of the virtual rewards $\widetilde{y}_{t,i}$, we consider a variety of options. We start by discussing the case when the fake reward $\widetilde{y}_{t,i} = \widetilde{f}_t(a_{t,i})$ and the acquisition function equals $\mathbb{A}_{\mathrm{avg}}(f, \mathcal{U})$.

This is perhaps the simplest as it can be shown that in this case where $\gamma = 0$ this implies $\widetilde{f}_{t,i} = \widetilde{f}_t$ independent of $i \in [B]$. In this case producing B_t can be done by solving for $\widetilde{f}_t$,

$$\widetilde{f}_t = \arg\max_{f \in \mathcal{F}} \sum_{a \in \mathcal{U}_t} f(a) \qquad \text{s.t.} \quad \sum_{(x,y) \in \widetilde{\mathcal{D}}_{t,i}} (f(x) - y)^2 = 0. \tag{4}$$

And defining the batch B_t as

$$B_t = \arg\max_{b \subset \mathcal{U}_t \text{ s.t. } |b| = B} \sum_{a \in b} \widetilde{f}_t(a).$$

The equivalence between these two batch selection rules follows by noting the equality constraint from 4 ensures each of the intermediate problems defining the sequence $\{(\widetilde{f}_{t,i}, \widetilde{U}_{t,i})\}_{i=1}^{B}$ satisfies $\widetilde{f}_{t,i} = \widetilde{f}_t$.

Algorithm 3 allowing for more general batch selection rules that may yield intermediate arm selection functions $\{\widetilde{f}_{t,i}\}$. In our experiments we compute $\widetilde{f}_{t,i}$ as an average of $\widetilde{f}_{t,i}^{\text{optimistic}}$ and $\widetilde{f}_{t,i}^{\text{pessimistic}}$. The function $\widetilde{f}_{t,i}^{\text{optimistic}}$ is produced by solving Equation 3 over the augmented batch $\widetilde{\mathcal{U}}_{t,i}$ with a $\lambda_{\text{reg}} > 0$ while $\widetilde{f}_{t,i}^{\text{pessimistic}}$ is produced by solving Equation 3 over the augmented batch $\widetilde{\mathcal{U}}_{t,i}$ with a $\lambda_{\text{reg}} < 0$. In each experiment we use the same value of λ_{reg} and it is the λ value appearing in the plot labels. The function producing the fictitious rewards is the average of the pessimistic and optimistic predictors $\widetilde{f}_{t,i} = \frac{\widetilde{f}_{t,i}^{\text{optimistic}} + \widetilde{f}_{t,i}^{\text{pessimistic}}}{2}$. The remaining experimental details such as the neural network architecture and the number of experiment repetitions is the same as in the main.

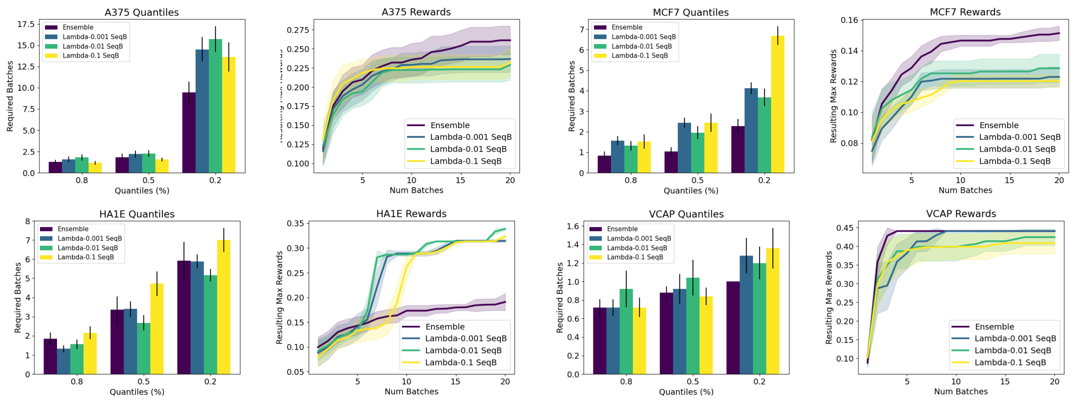


Figure 14: Sequential in Batch Optimism. CMAP data. Batch size 50.

Appendix D. Quantifying the Query Complexity of $\mathcal{F}$

Let $\epsilon \geq 0$ and define $t_{\text{opt}}^{\epsilon}(\mathcal{A}, f)$ to be the first time-step when an ϵ−optimal point $\hat{a}_t$ is proposed by a learner (possibly randomized) when interacting with arm set $\mathcal{A} \subset \mathbb{R}^d$ and the pseudo rewards are noiseless evaluations $\{f(a)\}_{a \in \mathcal{A}}$ with $f \in \mathcal{F}$. We define the query complexity of the $\mathcal{A}, \mathcal{F}$ pair as,

$$\mathbb{T}^{\epsilon}(\mathcal{A}, \mathcal{F}) = \min_{\text{Alg}} \max_{f \in \mathcal{F}} \mathbb{E}\left[t_{\text{opt}}^{\epsilon}(\mathcal{A}, f)\right]$$

Where the minimum iterates over all possible learning algorithms. We can characterize the upper bound of the problem complexity for several simple problem classes,

Lemma 1 *When $\mathcal{A} = \{\|x\| \leq 1 \text{ for } x \in \mathbb{R}^d\}$ and*

1. *If $\mathcal{F}$ is the class of linear functions defined by vectors in the unit ball $\mathcal{F} = \{x \to \theta^\top x : \|\theta\| \leq 1 \text{ for } \theta \in \mathbb{R}^d\}$ then $\mathbb{T}^\epsilon(\mathcal{A}, \mathcal{F}) \geq d$ when $\epsilon < \frac{1}{d}$ and $\mathbb{T}^\epsilon(\mathcal{A}, \mathcal{F}) \geq \lceil d - \epsilon d \rceil$ otherwise.*

2. *If $\mathcal{F}$ is the class of 1-Lipschitz functions functions then $\mathbb{T}^\epsilon(\mathcal{A}, \mathcal{F}) \geq \left(\frac{1}{4\epsilon}\right)^d$.*

Proof As a consequence of Yao's principle, we can restrict ourselves to deterministic algorithms. Indeed,

$$\min_{\text{Alg}} \max_{f \in \mathcal{F}} \mathbb{E}\left[t^\epsilon_{\text{opt}}(\mathcal{A}, f)\right] = \max_{\mathcal{D}_\mathcal{F}} \min_{\text{DetAlg}} \mathbb{E}_{f \sim \mathcal{D}_\mathcal{F}}\left[t^\epsilon_{\text{opt}}(\mathcal{A}, f)\right]$$

Thus, to prove the lower bound we are after it is enough to exhibit a distribution $\mathcal{D}_\mathcal{F}$ over instances $f \in \mathcal{F}$ and show a lower bound for the expected $t^\epsilon_{\text{opt}}(\mathcal{A}, f)$ where the expectation is taken using $\mathcal{D}_\mathcal{F}$.

With the objective of proving item 1 let $\mathcal{D}_\mathcal{F}$ be the uniform distribution over the sphere $\mathbf{Unif}_d(1)$. By symmetry it is easy to see that

$$\mathbb{E}_{\theta \sim \mathbf{Unif}_d(r)}\left[\theta_i\right] = 0, \quad \forall i \in d \text{ and } \forall r \geq 0.$$

Thus,

$$\text{Var}_{\theta \sim \mathbf{Unif}_d(r)}\left(\theta_i\right) = \mathbb{E}_{\theta \sim \mathbf{Unif}_d(r)}[\theta_i^2] \overset{(i)}{=} \frac{r^2}{d}. \tag{5}$$

Equality (i) follows because

$$\sum_{i=1}^d \mathbb{E}_{\theta \sim \mathbf{Unif}_d(r)}[\theta_i^2] = \mathbb{E}_{\theta \sim \mathbf{Unif}_d(r)}[\|\theta\|^2] = r^2$$

and because by symmetry for all $i, j \in [d]$ the second moments agree,

$$\mathbb{E}_{\theta \sim \mathbf{Unif}_d(r)}[\theta_i^2] = \mathbb{E}_{\theta \sim \mathbf{Unif}_d(r)}[\theta_j^2]$$

Finally,

$$\mathbb{E}_{\theta \sim \mathbf{Unif}_d(r)}[|\theta_i|] \leq \sqrt{\mathbb{E}_{\theta \sim \mathbf{Unif}_d(r)}[\theta_i^2]} = \frac{r}{\sqrt{d}}. \tag{6}$$

Let DetAlg be the optimal deterministic algorithm for $\mathcal{D}_\mathcal{F}$ and a_1 be its first action. Since $\mathcal{D}_\mathcal{F}$ is the unofrm distribution over the sphere, inequality 6 expected scale of the reward reward experienced is upper bounded by $\frac{1}{\sqrt{d}}$, and furthermore, since $\|a_1\| = 1$, the expected second moment of the reward experienced (where expectations are taken over $\mathcal{D}_\mathcal{F}$) equals $\frac{1}{d}$.

We now employ a conditional argument, if DetAlg has played a_1 and observed a reward r_1, We assume that up to time m algorithm DetAlg has played actions $a_1, \cdots, a_m$ and received rewards $r_1, \cdots, r_m$.

Given these outcomes, DetAlg can recover the component of θ lying in $\mathbf{span}(a_1, \cdots, a_m)$. Let a_{m+1} be DetAlg's action at time $m+1$. By assumption this is a deterministic function of $a_1, \cdots, a_m$ and $r_1, \cdots, r_m$. Since θ is drawn from $\mathbf{Unif}_d(1)$, the expected squared dot product between the component of $a_{m+1}^{\perp} = \mathbf{Proj}(a_{m+1}, \mathbf{span}(a_1, \cdots, a_m)^{\perp})$ satisfies,

$$\begin{aligned}\mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}} | \{a_1^\top \theta = r_i\}_{i=1}^m} \left[\left(\theta^\top a_{m+1}^{\perp} \right)^2 \right] &= \frac{1 - \|\theta_m^0\|^2}{d-m} \left(1 - \|a_{m+1}^0\|^2 \right) \\ &= \frac{1 - \|\theta_m^0\|^2}{d-m}. \end{aligned} \tag{7}$$

where $\theta_m^0 = \mathbf{Proj}(\theta, \mathbf{span}(a_1, \cdots, a_m))$. The last inequality follows because the conditional distribution of $\mathbf{Proj}(\theta, \mathbf{span}(a_1, \cdots, a_m)^{\perp})$ given $a_1, \cdots, a_m$ and $r_1, \cdots, r_m$ is a uniform distribution over the $d-m$ dimensional sphere of radius $\sqrt{1 - \|\theta_m^0\|^2}$, the scale of $a_{m+1}^{\perp}$ is $\sqrt{1 - \|a_{m+1}^0\|^2}$ and we have assumed the $\|a_{m+1}^0\| = 0$. Thus, the agreement of $a_{m+1}^{\perp}$ with $\mathbf{Proj}(\theta, \mathbf{span}(a_1, \cdots, a_m)^{\perp})$ satisfies Equation 5.

We consider the expected square norm of the recovered θ up to time m. This is the random variable $\|\theta_m^0\|^2 = \sum_{t=1}^m \left(\theta^\top a_t^{\perp} \right)^2$ where $a_t^{\perp} = \mathbf{Proj}(a_t, \mathbf{span}(a_1, \cdots, a_{t-1})^{\perp})$. Thus,

$$\begin{aligned}\mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}}} \left[\|\theta_m^0\|^2 \right] &= \mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}}} \left[\sum_{t=1}^m \left(\theta^\top a_t^{\perp} \right)^2 \right] \\ &= \mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}}} \left[\sum_{t=1}^m \mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}} | \{a_1^\top \theta = r_i\}_{i=1}^{t-1}} \left[\left(\theta^\top a_t^{\perp} \right)^2 \right] \right] \\ &\overset{(i)}{=} \mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}}} \left[\sum_{t=1}^m \frac{1 - \|\theta_{t-1}^0\|^2}{d-m} \right] \end{aligned}$$

Equality (i) holds because of 7. Recall that by Equation 5,

$$\mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}}} \left[\|\theta_1^0\|^2 \right] = \frac{1}{d}.$$

Thus by the above equalities,

$$\mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}}} \left[\|\theta_2^0\|^2 \right] = \frac{1}{d} + \frac{1 - \frac{1}{d}}{d-1} = \frac{2}{d}.$$

Unrolling these equalities further we conclude that

$$\mathbb{E}_{\theta \sim \mathcal{D}_{\mathcal{F}}} \left[\|\theta_m^0\|^2 \right] = \frac{m}{d}.$$

This implies the expected square agreement between the learner's virtual guess $\hat{a}_t$ is upper bounded by $\frac{m}{d}$. Thus, when $\epsilon < \frac{1}{d}$, the expected number of queries required is at least d. When $\epsilon > d$, the expected number of queries instead satisfies a lower bound of $\lceil d - \epsilon d \rceil$.

We now show shift our attention to Lipschitz functions. First we introduce the following simple construction of a $1-$Lipschitz function over a small ball of radius ϵ. We use this construction throughout our proof. Let $\mathbf{x} \in \mathbb{R}^d$ be an arbitrary vector, define $B(\mathbf{x}, \epsilon)$ as the ball centered around $\mathbf{x}$ of radius 2ϵ under the $\| \cdot \|_2$ norm and $S(\mathbf{x}, 2\epsilon)$ as the sphere (the surface of $B(\mathbf{x}, 2\epsilon)$) centered around $\mathbf{x}$

Define the function $f^\epsilon_{\mathbf{x}} : \mathbb{R}^d \to \mathbb{R}$ as,

$$f^\epsilon_{\mathbf{x}}(\mathbf{z}) = \begin{cases} \min_{\mathbf{z}' \in S(\mathbf{x}, 2\epsilon)} \|\mathbf{z} - \mathbf{z}'\|_2 & \text{if } \mathbf{z} \in B(\mathbf{x}, 2\epsilon) \\ 0 & \text{o.w.} \end{cases}$$

It is easy to see that $f^\epsilon_{\mathbf{x}}$ is $1-$Lipschitz. We consider three different cases,

1. If $\mathbf{z}_1, \mathbf{z}_2 \in B(\mathbf{x}, 2\epsilon)^c$ then $|f^\epsilon_{\mathbf{x}}(\mathbf{z}_1) - f^\epsilon_{\mathbf{x}}(\mathbf{z}_2)| = 0 \leq \|\mathbf{z}_1 - \mathbf{z}_2\|$. The result follows.
2. If $\mathbf{z}_1 \in B(\mathbf{x}, 2\epsilon)$ but $\mathbf{z}_2 \in B(\mathbf{x}, 2\epsilon)^c$. Let z_3 be the intersection point in the line going from $\mathbf{z}_1$ to $\mathbf{z}_2$ lying on $S(\mathbf{x}, 2\epsilon)$. Then $|f^\epsilon_{\mathbf{x}}(\mathbf{z}_1) - f^\epsilon_{\mathbf{x}}(\mathbf{z}_2)| = \min_{\mathbf{z}' \in S(\mathbf{x}, 2\epsilon)} \|\mathbf{z}_1 - \mathbf{z}'\|_2 \leq \|\mathbf{z}_1 - \mathbf{z}_3\|_2 \leq \|\mathbf{z}_1 - \mathbf{z}_2\|_2$.
3. If $\mathbf{z}_1, \mathbf{z}_2 \in B(\mathbf{x}, 2\epsilon)$. It is easy to see that $|f^\epsilon_{\mathbf{x}}(\mathbf{z}_1) - f^\epsilon_{\mathbf{x}}(\mathbf{z}_2)| = |\|\mathbf{z}_1 - \mathbf{x}\|_2 - \|\mathbf{z}_2 - \mathbf{x}\|_2|$. And therefore by the triangle inequality applied to $\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2$, that $|\|\mathbf{z}_1 - \mathbf{x}\|_2 - \|\mathbf{z}_2 - \mathbf{x}\|_2| \leq \|\mathbf{z}_1 - \mathbf{z}_2\|_2$. The result follows.

Let $\mathcal{N}(B(\mathbf{0}, 1), 2\epsilon)$ be a $2\epsilon-$packing of the unit ball. For simplicity we'll use the notation $N_{2\epsilon} = |\mathcal{N}(B(\mathbf{0}, 1), 2\epsilon)|$. Define the set of functions $\mathcal{F}^\epsilon = \{f^\epsilon_{\mathbf{x}} \text{ for all } \mathbf{x} \in \mathcal{N}(B(\mathbf{0}, 1), 2\epsilon)\}$ and define $\mathcal{D}_{\mathcal{F}}$ as the uniform distribution over $\mathcal{F}^\epsilon \subset \mathcal{F}$. Similar to the case when $\mathcal{F}$ is the set of linear functions, we make use of Yao's principle. Let DetAlg be an optimal deterministic algorithm for $\mathcal{D}_{\mathcal{F}}$.

Let a_i be DetAlg's $i-$th query point and r_i be the $i-$th reward it receives. If the ground truth was $f^\epsilon_{\mathbf{x}}$ and the algorithm does not sample a query point from inside $B(\mathbf{x}, 2\epsilon)$, it will receive a reward of 0 and thus would not have found an $\epsilon-$optimal point. Thus $t_{\text{opt}}(f^\epsilon_{\mathbf{x}}) \geq$ first time to pull an arm in $B(\mathbf{x}, 1)$. As a consequence of this fact,

$$\mathbb{E}_{f^\epsilon_{\mathbf{x}} \sim \mathcal{D}_{\mathcal{F}}} [\mathbf{1}(a_1 \in B(x, 2\epsilon))] \leq \frac{1}{N_{2\epsilon}}.$$

Hence $\mathbb{E}_{f^\epsilon_{\mathbf{x}} \sim \mathcal{D}_{\mathcal{F}}} [\mathbf{1}(a_1 \notin B(x, 2\epsilon))] \geq \frac{N_{2\epsilon} - 1}{N_{2\epsilon}}$. Therefore,

$$\begin{aligned} \mathbb{E}\left[t^\epsilon_{\text{opt}}\right] &\geq \sum_{\ell=1}^{N_{2\epsilon}} \frac{N_{2\epsilon} - \ell}{N_{2\epsilon} - \ell + 1} \\ &\geq \sum_{\ell=1}^{N_{2\epsilon}/2} \frac{N_{2\epsilon} - \ell}{N_{2\epsilon} - \ell + 1} \\ &\geq \frac{1}{4} N_{2\epsilon}. \end{aligned}$$

Since $N_{2\epsilon} \overset{(i)}{\geq} \text{Covering}(B(\mathbf{0}, 1), 4\epsilon) \overset{(ii)}{\geq} \left(\frac{1}{4\epsilon}\right)^d$ where inequality (i) is a consequence of Lemma 5.5 and inequality (ii) from Lema 5.7 in (17).

■

The results of Lemma 1 hold regardless of the batch size B. It is thus impossible to design an algorithm that can single out an ϵ-optimal arm in less than $\mathbb{T}^{\epsilon}(\mathcal{A}, \mathcal{F})$ queries for all problems defined by the pair $\mathcal{A}, \mathcal{F}$ simultaneously.

Translating to Quantile Optimality . The results of Lemma 1 can be interpreted in the langauge of quantile optimality by imposing a uniform measure over the sphere. In this case $\epsilon-$optimality is equivalent (approximately) to a $1 - 2^{-d(1-\epsilon)^2}$ quantile.

Noisy Labels and Repeated Queries . The OAE principle can be used in the presence of noisy labels and when it makes sense to repeatedly query arms. In this case, many options are available such as repeated querying of any selected arm (for noise reduction) or setting $\mathcal{U}_t$ to the set of all arms, avoiding the elimination of queries in the batch B_t.

References

[1] Alekh Agarwal, Haipeng Luo, Behnam Neyshabur, and Robert E Schapire. Corralling a band of bandit algorithms. In *Conference on Learning Theory*, pages 12–38. PMLR, 2017.

[2] Ashok Cutkosky, Christoph Dann, Abhimanyu Das, Claudio Gentile, Aldo Pacchiano, and Manish Purohit. Dynamic balancing for model selection in bandits and rl. In *International Conference on Machine Learning*, pages 2276–2285. PMLR, 2021.

[3] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[4] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[5] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. *Advances in Neural Information Processing Systems*, 29:4206–4214, 2016.

[6] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32:7026–7037, 2019.

[7] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012.

[8] Jonathan Lee, Aldo Pacchiano, Vidya Muthukumar, Weihao Kong, and Emma Brunskill. Online model selection for reinforcement learning with function approximation. In *International Conference on Artificial Intelligence and Statistics*, pages 3340–3348. PMLR, 2021.

[9] Arash Mehrjou, Ashkan Soleymani, Andrew Jesson, Pascal Notin, Yarin Gal, Stefan Bauer, and Patrick Schwab. Genedisco: A benchmark for experimental design in drug discovery. *arXiv preprint arXiv:2110.11875*, 2021.

[10] Jonas Mueller, David Reshef, George Du, and Tommi Jaakkola. Learning optimal interventions. In *Artificial Intelligence and Statistics*, pages 1039–1047. PMLR, 2017.

[11] Aldo Pacchiano, Christoph Dann, Claudio Gentile, and Peter Bartlett. Regret bound balancing and elimination for model selection in bandits and rl. *arXiv preprint arXiv:2012.13045*, 2020.

[12] Aldo Pacchiano, My Phan, Yasin Abbasi Yadkori, Anup Rao, Julian Zimmert, Tor Lattimore, and Csaba Szepesvari. Model selection in contextual stochastic bandit problems. *Advances in Neural Information Processing Systems*, 33:10328–10337, 2020.

[13] Jack Parker-Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18050–18062, 2020.

[14] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

[15] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[16] Aravind Subramanian, Rajiv Narayan, Steven M Corsello, David D Peck, Ted E Natoli, Xiaodong Lu, Joshua Gould, John F Davis, Andrew A Tubelli, Jacob K Asiedu, David L Lahr, Jodi E Hirschman, Zihan Liu, Melanie Donahue, Bina Julian, Mariya Khan, David Wadden, Ian C Smith, Daniel Lam, Arthur Liberzon, Courtney Toder, Mukta Bagul, Marek Orzechowski, Oana M Enache, Federica Piccioni, Sarah A Johnson, Nicholas J Lyons, Alice H Berger, Alykhan F Shamji, Angela N Brooks, Anita Vrcic, Corey Flynn, Jacqueline Rosains, David Y Takeda, Roger Hu, Desiree Davison, Justin Lamb, Kristin Ardlie, Larson Hogstrom, Peyton Greenside, Nathanael S Gray, Paul A Clemons, Serena Silver, Xiaoyun Wu, Wen-Ning Zhao, Willis Read-Button, Xiaohua Wu, Stephen J Haggarty, Lucienne V Ronco, Jesse S Boehm, Stuart L Schreiber, John G Doench, Joshua A Bittker, David E Root, Bang Wong, and Todd R Golub. A next generation connectivity map: L1000 platform and the first 1,000,000 profiles. *Cell*, 171 (6):1437–1452.e17, November 2017.

[17] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.