Batch Active Learning of Reward Functions from Human Preferences

ERDEM BIYIK, Thomas Lord Department of Computer Science, University of Southern California, USA
NIMA ANARI, Department of Computer Science, Stanford University, USA
DORSA SADIGH, Department of Computer Science, Stanford University, USA

Data generation and labeling are often expensive in robot learning. Preference-based learning is a concept that enables reliable labeling by querying users with preference questions. Active querying methods are commonly employed in preference-based learning to generate more informative data at the expense of parallelization and computation time. In this paper, we develop a set of novel algorithms, *batch active preference-based learning* methods, that enable efficient learning of reward functions using as few data samples as possible while still having short query generation times and also retaining parallelizability. We introduce a method based on determinantal point processes (DPP) for active batch generation and several heuristic-based alternatives. Finally, we present our experimental results for a variety of robotics tasks in simulation. Our results suggest that our batch active learning algorithm requires only a few queries that are computed in a short amount of time. We showcase one of our algorithms in a study to learn human users' preferences.

CCS Concepts: • **Computing methodologies** → **Batch learning**; **Active learning settings**; *Markov decision processes*; **Inverse reinforcement learning**.

Additional Key Words and Phrases: reward learning, active learning, preference-based learning, human-robot interaction, robot learning, inverse reinforcement learning

ACM Reference Format:
Erdem Bıyık, Nima Anari, and Dorsa Sadigh. 2024. Batch Active Learning of Reward Functions from Human Preferences. 1, 1 (February 2024), 27 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Machine learning algorithms have been quite successful in the past decade. A significant part of this success can be associated to the availability of large amount of labeled data. However, collecting and labeling data can be costly and time-consuming in many fields such as speech recognition [75], recommendation systems [21], dialog control [72], image recognition [70], as well as in robotics [4, 45, 65, 68]. Lack of labeled data is a common problem in many of these machine learning applications; however, it is particularly difficult in robot learning. Humans cannot reliably assign a *success value* (reward) to a given robot trajectory, i.e., it is not obvious what labels need to be assigned to a particular robot behavior. When assigning labels is difficult, we often fall back to collecting expert demonstrations from humans to learn the desired behavior; however, this is also not easy in robotics applications as human experts often have a difficult

Authors' addresses: Erdem Bıyık, biyik@usc.edu, Thomas Lord Department of Computer Science, University of Southern California, Los Angeles, California, USA; Nima Anari, anari@cs.stanford.edu, Department of Computer Science, Stanford University, Stanford, California, USA; Dorsa Sadigh, dorsa@cs.stanford.edu, Department of Computer Science, Stanford University, Stanford, California, USA.

Manuscript submitted to ACM

time demonstrating optimal trajectories on a robot with high degrees of freedom [2, 60], when there is uncertainty in the system or environment [53], or under their cognitive biases about how a robot should operate [11].

To address the problem of assigning meaningful labels to single trajectories or the challenge of providing desirable and optimal trajectories, we instead focus on using *preference-based learning* methods that query the users for their preferences in the form of comparison questions between multiple trajectories synthesized by the robot itself [27, 30, 35, 42, 54, 58, 66]. These methods enable us to learn a regression model by only using the preferences of the users as opposed to relying on expert demonstrations.

Furthermore to address the lack of data in these applications, we leverage *active* preference-based learning techniques, where we query the human with the most informative pairwise comparison question to recover their preferences of how a robot should act [68, 74, 82]. These preferences are often modeled using a reward function. However, active preference-based learning of reward functions can in practice be extremely time-inefficient, as these methods often require modeling a belief over a continuous reward function space and sampling from this belief. In addition, the states and actions in every trajectory that is shown to the human naturally are drawn from a continuous space, which often amplifies the time-inefficiency of these methods.

We thus propose using methods that generate a *batch* of comparison queries optimized at the same time as opposed to generating queries one after the other. These batch methods not only improve time-efficiency, but also have other computational benefits. For example, they can help when fitting the learning model is expensive, e.g., as in Gaussian processes [14, 15, 57], as the model should be retrained only after all queries in the batch are responded, rather than after every single query. In addition, these methods are parallelizable, which is a desirable feature when the robot is learning from multiple humans who share the same (or similar) preferences about how the task should be done.

While larger batches amplify these advantages, they can hurt data-efficiency, because new queries become less optimized with respect to the queries made earlier (and so the learned model so far). Hence, there is a direct tradeoff between *the required number of queries* and the *time it takes to generate each query*. Besides, it is challenging to decide how an informative batch must be generated. While a batch of random queries hurts data-efficiency, finding the optimal batch is computationally intractable because it requires an exhaustive search over all possible human responses to the queries in the batch.

Ideally, we would like to develop an algorithm that requires only a few number of comparison queries while generating each query efficiently. In this work, we propose a new set of algorithms — *batch active preference-based learning* methods — that balance this tradeoff between the number of queries it requires to learn human preferences and the time it spends on generation of each comparison query.

To this end, we actively generate each batch based on the data collected so far. Therefore, in our framework, we select and query k pairs of trajectories, to be compared by the user or users, at once. Since k queries are generated at once, our framework is parallelizable for data collection as opposed to standard active learning methods that require data to come sequentially.

What makes batch active learning more difficult than standard active learning problems is that we cannot select the queries by simply maximizing their individual informativeness. Since a batch of queries is selected all at once, they must be selected without any information about the user responses to the queries within that batch. The batch active learning methods should then try to maximize the diversity between the queries in order to avoid selecting very similar queries in a single batch [26, 86]. Therefore, a good batch active learning method must produce batches that consist of both dissimilar and informative queries. This is visualized in Figure 1.

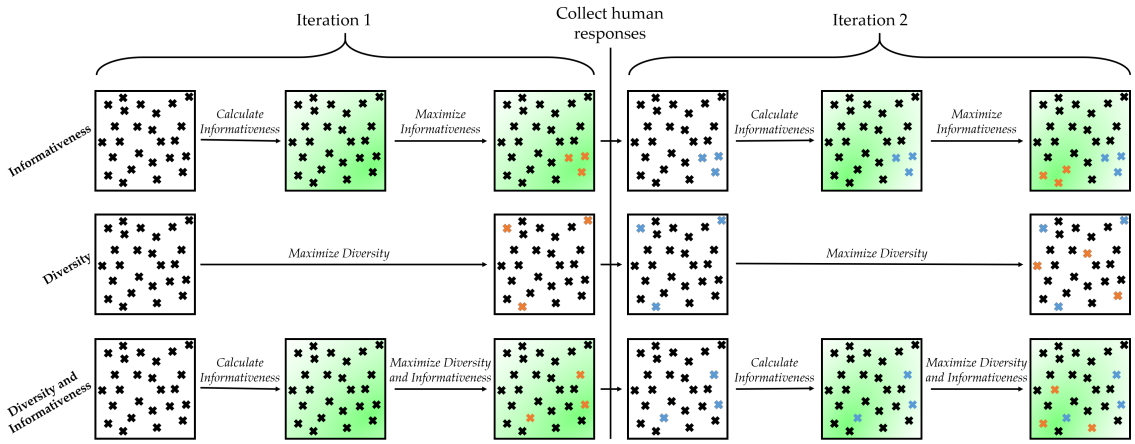


Fig. 1. Batches should be both diverse and informative in batch active preference-based learning. Here, a hypothetical batch selection problem is visualized. Each cross represents a query. Similar queries are close to each other. Orange shows the queries selected in that iteration, and blue shows the queries for which the human responses have already been collected in the previous iterations. Green color represents informativeness: darker regions correspond to the queries with high informativeness based on the information collected until that iteration. **(Top)** Maximizing only informativeness generates batches that include very similar queries which, when queried together, carry redundant information. **(Middle)** Maximizing only diversity does not take informativeness into account at all, and so is wasteful as it selects some queries that are not informative. **(Bottom)** A good batch active learning algorithm should both select informative queries and avoid redundancy.

To this end, we summarize our contributions as[1]:

(1) Developing a batch active learning algorithm based on determinantal point processes (DPP) that leads to the highest performance by balancing the tradeoff between the informativeness and diversity of queries.
(2) Designing a set of heuristic-based approximation algorithms for efficient batch active learning to learn about human preferences from comparison queries.
(3) Experimenting and comparing approximation methods for batch active learning in complex preference based learning tasks.
(4) Showcasing our framework in predicting human users' preferences in simulated autonomous driving and robotics tasks.

For the rest of the paper, we will start with going over the related works in the literature and formalizing the problem. We will then present how standard active learning methods select queries. After introducing the general batch-mode active learning idea, we propose our methods for batch selection. First, we propose heuristic-based batch generation methods that avoid hyperparameter tuning. Next, we propose our primary method based on determinantal point processes. After proposing these different approaches, we present our experiments with both simulated and real users. We conclude the paper with a discussion of limitations and future work.

[1]Note that parts of this work have been published at Conference on Robot Learning [18] and as a preprint in [20]. The DPP-based algorithm and all simulation experiment results are new compared to the conference paper.

2 RELATED WORK

Inverse Reinforcement Learning. There has been a lot of work on learning a model of the human preferences about the robots' trajectories through inverse reinforcement learning (IRL) [55, 67, 69, 89]. In these works, a reward function is learned directly from a human demonstrating how to operate a robot. However, learning a reward function from human demonstrations can be problematic for a few reasons. First, providing demonstrations for robots with higher degrees of freedom can be quite challenging even for human experts [2, 60]. Furthermore, human preferences tend to differ from their demonstrations [11]. Therefore, prior work has extensively studied learning reward functions using other sources of information, e.g., human corrections [9], assessments [71], or rankings [25, 62]. In this work, we learn the reward functions using *preference queries* where the human is asked to simply compare two generated trajectories instead of demonstrating a trajectory.

Batch Active Learning. The problem of actively generating a batch of data is well-studied in other machine learning problems such as classification [36, 78, 85], where decision boundaries may inform the active learning algorithms. While this may simplify the problem, it is not applicable in our setting, where we attempt to actively learn a reward function for dynamical systems using preference queries as opposed to data point – label pairs where the labels are directly and persistently associated with the corresponding data points.

Determinantal Point Processes. While existing batch active learning methods are not readily applicable in our problem, we have the same challenge of generating both informative and diverse batches. For this, determinantal point processes (DPP) are a natural fit. DPPs are a mathematical tool that is often used for generating diverse batches from a set of items [52] and are used to generate batches in other machine learning applications, such as for improving the convergence of stochastic gradient descent [87, 88]. Here, we propose using DPPs to generate not only diverse but also informative batches in active preference-based reward learning.

Active Preference-based Learning. Several works leveraged active preference-based techniques to synthesize pairwise comparison queries for the goal of efficiently learning humans' preferences [3, 16, 37, 38, 72, 83]. However, there is a tradeoff between the time spent to generate a query at every time step and the number of queries required until converging to the human's preference reward function. Although actively synthesizing queries can reduce the total number of queries, generating each query can be quite time-consuming, which can make the approach impractical by creating a slow interaction with humans.

Most related to our work, Sadigh et al. [68] and Palan et al. [65] have focused on learning reward functions by actively synthesizing comparison queries directly from the continuous space of states and actions, which caused these methods to be extremely slow. In fact, the participants of the user study by Palan et al. [65] raised concerns about the speed of the active preference-based reward learning framework.

Other methods have adopted the idea that the comparison queries can be actively selected from a pre-generated set of trajectories to reduce computational burden [10]. While this improved the time-efficiency, the resulting method was still slow due to the fact that each and every query requires solving an optimization problem and the model has to be retrained after every human response. Moreover, none of these methods were parallelizable, i.e., they require the users to respond to the queries *sequentially*, preventing parallel data collection. This negatively affects the use of these algorithms in the settings where multiple humans with shared preferences could provide responses to preference queries.

To this end, we propose a set of time-efficient *batch* active learning methods, that balance between minimizing the number of queries and being time-efficient in its interaction with the human expert. Batch active learning has two main

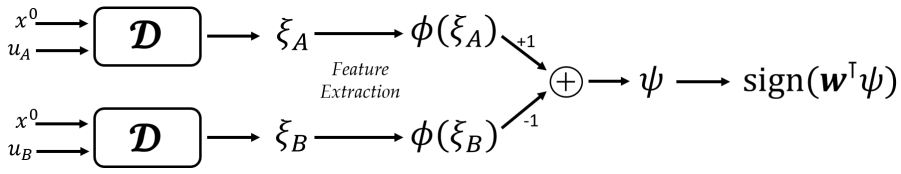


Fig. 2. The schematic of the preferences based-learning problem starting from two sample inputs $(x^0, \mathbf{u}_A)$ and $(x^0, \mathbf{u}_B)$.

benefits: i) generating a batch of queries can create a more *time-efficient* interaction with the human, ii) the procedure can be *parallelized* among multiple humans.

3 PROBLEM STATEMENT

Modeling Choices. We start by modeling human preferences about how a robot should act. We model these preferences over the actions of a robot in a fully observable deterministic[2] dynamical system $\mathcal{D}$. Let $f_{\mathcal{D}}$ denote the dynamics of the robot:

$$x^{t+1} = f_{\mathcal{D}}(x^t, u^t) \tag{1}$$

Here u^t denotes the action of the robot at time step t. The state x^t evolves through the dynamics and the actions.

A finite trajectory $\xi \in \Xi$ is a sequence of state-action pairs $\left((x^0, u^0) \dots \left(x^T, u^T\right)\right)$ over a finite horizon $t = 0, 1, \dots, T$. Here Ξ is the set of feasible trajectories, i.e., trajectories that satisfy the dynamics of the system.

Preference Reward Function. We model human preferences through a preference reward function $R_H : \Xi \to \mathbb{R}$ that maps a feasible trajectory to a real number corresponding to a score for preference of the input trajectory. We assume the reward function is a linear combination of a given set of features over trajectories $\phi(\xi) \in \mathbb{R}^d$, where:

$$R_H(\xi) \coloneqq \boldsymbol{w}^\top \phi(\xi) \,. \tag{2}$$

The goal of *preference-based learning* is to learn $R_H(\xi)$, or equivalently the weights $\boldsymbol{w} \in \mathbb{R}^d$ through preference queries from a human expert. A preference query is a question in the form of "do you prefer trajectory ξ_A or ξ_B?". For any two trajectories ξ_A and ξ_B, the human expert prefers ξ_A over ξ_B if and only if $R_H(\xi_A) > R_H(\xi_B)$. From this preference encoded as a strict inequality, we can conclude $\boldsymbol{w}^\top \phi(\xi_A) > \boldsymbol{w}^\top \phi(\xi_B)$ or equivalently:

$$\boldsymbol{w}^\top (\phi(\xi_A) - \phi(\xi_B)) > 0 \,. \tag{3}$$

We use ψ to refer to this difference: $\psi(\xi_A, \xi_B) \coloneqq \phi(\xi_A) - \phi(\xi_B)$. Therefore, ψ sufficiently characterizes the information we get from a query. Similarly, the sign of $\boldsymbol{w}^\top \psi$ is sufficient to reveal the preference of the human expert for every trajectory pair ξ_A and ξ_B. We thus let $I = \text{sign}(\boldsymbol{w}^\top \psi)$ denote the human's input (answer) to a query ψ. Figure 2 summarizes the flow that leads to the human's preference I.

[2]We study deterministic dynamical systems to be able to compare with prior work [68] without any major modifications. However, our methods can be easily extended to stochastic systems as long as the batch generation is performed over a predefined set of trajectories as we will elaborate in the text.

In addition, the input from the human can be noisy due to the uncertainty of their preferences. A common noise model assumes human's preferences are probabilistic and can be modeled using a softmax function [30, 44, 68]:

$$P(I \mid \boldsymbol{w}) = \begin{cases} \frac{\exp(R_H(\xi_A))}{\exp(R_H(\xi_A))+\exp(R_H(\xi_B))} & I = +1 \\ \frac{\exp(R_H(\xi_B))}{\exp(R_H(\xi_A))+\exp(R_H(\xi_B))} & I = -1 \end{cases}$$
$$= \frac{1}{1+\exp(-I\boldsymbol{w}^\top \psi)}, \tag{4}$$

where $I = \text{sign}(\boldsymbol{w}^\top \psi)$ represents the preference of the human on the query ψ. This model led to successful inference of reward functions not only when preference data are provided by humans but also by vision-language models [77].

Approach Overview. Our goal is to learn the human's reward function parameters $\boldsymbol{w}$ in a both data-efficient and time-efficient way. To this end, we develop batch-active preference-based reward learning methods, that actively generate a batch of preference queries based on the previous queries and the human's responses to them.

In the next section, we first start with an overview of actively synthesizing queries where queries are optimized one by one for their informativeness. We then proceed with batch-active methods where we need to optimize both for informativeness (as in the non-batch setting) and diversity to avoid having similar (and so redundant) queries within a batch.

4 ACTIVELY SYNTHESIZING PAIRWISE QUERIES

In active preference-based learning, the goal is to synthesize or search for the next pairwise comparison query to ask a human expert in order to maximize the information received. While optimal querying is NP-hard [1], there exist techniques that pose the problem as a sequential optimization for which greedy solutions that work well in practice exist, e.g., volume removal [68] and mutual information maximization [17] methods. Since Biyik et al. [17] identified failure cases of the former method and showed that maximizing mutual information yields consistently better results, we adopt this approach in our paper. We now describe this active preference-based reward learning approach.

The goal is to search for the human's preference reward function $R_H(\xi) = \boldsymbol{w}^\top \phi(\xi)$ by actively querying the human. We let $p(\boldsymbol{w})$ be the belief distribution of the unknown weight vector $\boldsymbol{w}$. Since $\boldsymbol{w}$ and $c\boldsymbol{w}$ yield to the same true preferences for a positive constant c, we constrain the prior of the belief such that $\|\boldsymbol{w}\|_2 \leq 1$. Every query provides a human input I, which then enables us to perform a Bayesian update on this distribution:

$$p(\boldsymbol{w} \mid I) \propto p(I \mid \boldsymbol{w}) p(\boldsymbol{w}) \,, \tag{5}$$

using the human preference model given in Eqn. (4). Since we do not know the shape of $p(\boldsymbol{w})$, we sample M values from $p(\boldsymbol{w})$ using an adaptive Metropolis algorithm [41]. In order to speed up this sampling process, we approximate $p(I \mid \boldsymbol{w})$ with a log-concave function whose mode always evaluates to one:

$$p(I \mid \boldsymbol{w}) = \min(1, \exp(I\boldsymbol{w}^\top \psi)) \,. \tag{6}$$

Based on [17], generating the next most informative query can be formulated as maximizing the mutual information between the human input I and reward function parameters $\boldsymbol{w}$ at every iteration. Put another way, we want to find the query that maximizes the difference between the prior entropy over $p(\boldsymbol{w})$ and the posterior entropy. We note that every query, i.e., a pair of trajectories (ξ_A, ξ_B) is parameterized by the initial state of the system x^0, and the two sequences of actions $\mathbf{u}_A$ and $\mathbf{u}_B$ corresponding to ξ_A and ξ_B, respectively, because the dynamics are deterministic with respect to Eqn. (1). Here, we assume the initial state of the system is the same between the two trajectories of a query (but not

necessarily between queries) to make sure the trajectories are comparable to each other. The query selection problem is then:

$$\max_{x^0,\mathbf{u}_A,\mathbf{u}_B} \; H(\boldsymbol{w}) - \mathbb{E}_I \left[H(\boldsymbol{w} \mid I) \right] \tag{7}$$

with appropriate feasibility constraints to make sure the initial system state x^0 and the action sequences, $\mathbf{u}_A$ and $\mathbf{u}_B$, are feasible. Here, H denotes information entropy [33]:

$$H(\boldsymbol{w}) = -\mathbb{E}_{\boldsymbol{w}} \left[\log(p(\boldsymbol{w})) \right] \, , \tag{8}$$

and we are trying to maximize the difference between the prior and the posterior entropies (mutual information) under the expected human input I.

Following the derivation for mutual information maximization in [17], we equivalently write the objective as:

$$\max_{x^0,\mathbf{u}_A,\mathbf{u}_B} \sum_{I \in \{-1,+1\}} \mathbb{E}_{\boldsymbol{w}} \left[p(I \mid \boldsymbol{w}) \log_2 \left(\frac{p(I \mid \boldsymbol{w})}{\mathbb{E}_{\tilde{\boldsymbol{w}}} \left[p(I \mid \tilde{\boldsymbol{w}}) \right]} \right) \right] \, , \tag{9}$$

which we can approximately compute using the $\boldsymbol{w}$ samples for the expectation terms. This query selection approach is similar to the expected value of information of the query [50, 63] and the optimization can be solved using a Quasi-Newton method [7].

Actively generating queries significantly improves data-efficiency. However, it is not a time-efficient solution, since each and every query requires solving the optimization in (9) and running the adaptive Metropolis algorithm [41] for sampling. Performing these operations for every single query might be quite slow and not very practical while interacting with a human expert in real-time. The human has to wait for the solution of optimization before being able to respond to the next query. Besides, all queries have to come sequentially, as each of them uses the information from all previous ones. This prevents collecting data in parallel where multiple people are available to provide preferences.

4.1 Batch Active Learning

Our insight is that we can in fact balance between the number of queries required for convergence to R_H and the time required to generate each query. We construct this balance by introducing a *batch active learning* approach, where k queries are simultaneously generated at a time based on the current estimate of $\boldsymbol{w}$. The batch approach can significantly reduce the total time required for the satisfactory estimation of $\boldsymbol{w}$ at the expense of increasing the number of queries needed for convergence to true R_H.

To obtain a batch of queries that are informative, we need to find queries that have high mutual information values as computed by the objective of Eqn. (9). However, small perturbations of the inputs could lead to very minor changes in this objective value, and so continuous optimization of this objective can result in generating same or very similar queries within a batch. Besides, we want to increase the diversity of queries in a batch. We thus fall back to a discretization method. We discretize the space of queries by randomly sampling K pairs of trajectories from the input space of $\xi = (x^0, \mathbf{u})$. While increasing K may lead to more accurate optimization results, the computation time also increases linearly with K.

The batch active learning problem is then an optimization that attempts to find the k queries out of K that will maximize the mutual information between the human's responses and the reward function parameters $\boldsymbol{w}$. Formally, we

need to solve

$$\max_{\xi_{1_A},\xi_{1_B},\dots,\xi_{k_A},\xi_{k_B}\in\mathcal{K}} H(\boldsymbol{w}) - \mathbb{E}_{I_1,I_2,\dots,I_k}\left[H(\boldsymbol{w} \mid I_1, I_2, \dots, I_k)\right] \tag{10}$$

Although we can make a similar derivation for the objective as in (9), this is a combinatorial optimization problem that is often computationally hard (see [34] and [29] for the proofs with similar objectives). Even though we first reduce the query set into a smaller set $\mathcal{X}$ of size N by picking the queries which will individually maximize the mutual information (see Algorithm 1 for the pseudo-code of this procedure), the solution still requires an exhaustive search which is intractable in practice as the search space is exponentially large [40]. For example, solving this combinatorial optimization with $k = 10$ and $N = 200$ would require evaluating the objective $C(200, 10) \cong 2.25 \times 10^{16}$ times.

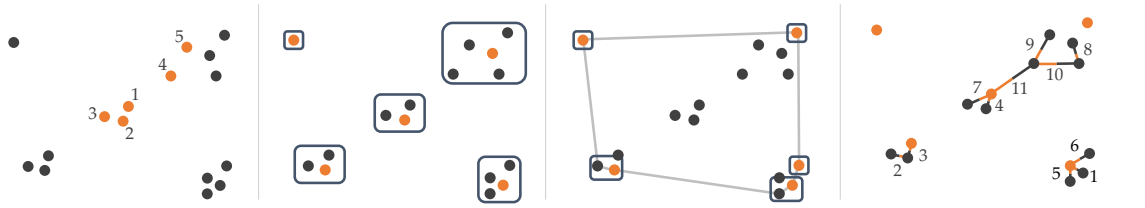
Algorithm 1 ReduceDataset($\boldsymbol{w}, \mathcal{K}, N$)

Input: $\boldsymbol{w}_1, \dots, \boldsymbol{w}_M$ ▷ Sampled $\boldsymbol{w}$ estimates
Input: $\mathcal{K} \coloneqq \left((\xi_{1_A}, \xi_{1_B}), \dots, (\xi_{K_A}, \xi_{K_B})\right)$ ▷ Dataset
1: **for** $i = 1, \dots, K$ **do**
2: $\quad\psi_i \leftarrow \psi(\xi_{i_A}, \xi_{i_B})$
3: $\quad q_i \leftarrow H(\boldsymbol{w}) - \mathbb{E}_I\left[H(\boldsymbol{w} \mid I)\right]$
4: **end for**
5: $\mathcal{X} \leftarrow \psi_i$'s with N highest q_i values ▷ Reduction
6: $\mathbf{q} \leftarrow q_i$ values corresponding to $\mathcal{X}$
7: **return** $\mathcal{X}, \mathbf{q}$

Algorithm 2 Batch Active Preference-based Learning

1: Generate query dataset $\mathcal{K} \coloneqq (\xi_{i_A}, \xi_{i_B})_{i=1}^{K}$ w.r.t. (1)
2: **for** $m = 1, 2, \dots$ **do**
3: $\quad$Get M samples $\boldsymbol{w} \sim p(\boldsymbol{w})$
4: $\quad\mathcal{X}, \mathbf{q} \leftarrow$ ReduceDataset$(\boldsymbol{w}, \mathcal{K}, N)$
5: $\quad A \leftarrow$ a batch of size k using $\boldsymbol{w}$ from $\mathcal{X}$
6: $\quad$Get the human response for each query in A
7: $\quad$Update $p(\boldsymbol{w})$ according to (5)
8: **end for**
9: **return** $\mathbb{E}[\boldsymbol{w}]$

In the subsequent sections, we present our batch generation algorithms that attempt to find approximately optimal batches without solving the combinatorial optimization and instead by using the individual mutual information values. We start with the most time-efficient heuristics with increasing complexity, and eventually present our premier method, DPP-based batch active learning, which leads to the best learning performance as we will present in our experiments. Algorithm 2 gives an overview of the overall batch active preference-based learning approach: line 1 discretizes the space of queries, line 3 samples a set of $\boldsymbol{w}$ from the belief distribution $p(\boldsymbol{w})$, and line 4 performs optional dataset reduction to work with the reduced set $\mathcal{X}$ for the current iteration instead of the full set $\mathcal{K}$. Line 5 produces a batch of queries, for which we present several methods in the subsequent sections. After the human responses are collected for the queries in the batch in line 6, a Bayesian update is performed to obtain the posterior belief distribution in line 7.



(a) *Greedy Selection.* (b) *Medoids Selection.* (c) *Boundary Medoids Selection.* (d) *Successive Elimination.*

Fig. 3. Visualizations of the batch generation process of the proposed time-efficient batch active learning algorithms. In each visual, a simple 2D space with 16 different ψ values that correspond to the reduced set $\mathcal{X}$ is shown. The goal is to select a batch of $k = 5$ that will near-optimally maximize the joint information gain. The selected queries are shown in orange. (a) Greedy Selection. (b) Medoids Selection. The points are selected based on the k-medoids clustering algorithm. (c) Boundary Medoids Selection. The clusters are chosen over the boundary of the convex hull of all samples. (d) Successive Elimination. One point is selected and another is eliminated based on pairwise comparisons of mutual information.

5 TIME-EFFICIENT BATCH ACTIVE LEARNING METHODS

We now describe a set of alternative methods in increasing order of complexity to provide approximations to the batch active learning problem. Figure 3 visualizes each approach in this section for a small set of queries.

5.1 Greedy Selection

The simplest method to approximate the optimal batch generation is using a greedy strategy. In the greedy selection approach, we conveniently assume the k different queries in a batch are independent from each other. Of course this is not a valid assumption, but the independence assumption allows us to choose the k-many maximizers of the objective of Eqn. (9) among the K discrete queries.

While this method can easily be employed; it is suboptimal as similar or redundant queries can be selected together in the same batch because these similar queries are likely to lead to high mutual information values. For instance, as shown in Figure 3 (a), the 5 orange queries chosen are all going to be very close to the center where mutual information values are high.

5.2 Medoid Selection

To avoid the redundancy in the batch created by the greedy selection, we need to increase the dissimilarity between the selected queries. We introduce an approach, *Medoid Selection*, that leverages clustering as a similarity measure between the samples. In this approach, with the goal of picking the most dissimilar queries, we cluster ψ-vectors associated with the elements of the reduced set $\mathcal{X}$, whose elements are already invidual maximizers of mutual information, into k clusters using standard Euclidean distance. We then restrict ourselves to only selecting one element from each cluster, which prevents us from selecting very similar trajectories.

One can think of using the well-known k-means algorithm [59] for clustering and then selecting the centroid of each cluster. However, these centroids are not necessarily from the reduced set, so they can have lower mutual information values. More importantly, they might be infeasible, i.e., there might not be a pair of trajectories that produce the ψ vectors corresponding to the centroids.

Instead, we use the k-medoids algorithm [12, 47] which again clusters the queries into k sets. The main difference between k-means and k-medoids is that k-medoids enables us to select medoids as opposed to the centroids, which are queries *in the set* $\mathcal{X}$ that minimize the average distance to the other queries in the same cluster. While k-medoids is known to be a slower algorithm than k-means [76], efficient approximate algorithms exist [8]. Figure 3 (b) shows the medoids selection approach, where 5 orange queries are selected from the 5 clusters.

5.3 Boundary Medoid Selection

We note that picking the medoid of each cluster is not the best option for increasing dissimilarity —instead, we can further exploit clustering to select queries more effectively. In the *Boundary Medoid Selection* method, we propose restricting the selection to be only from the boundary of the convex hull of the reduced set $\mathcal{X}$. If feasible, this selection criteria can separate out the selected queries from each other on average. We note that when d, the dimension of ψ, is large enough compared to k, most of the clusters will have queries on the boundary. We thus propose the following modifications to the medoid selection algorithm. The first step is to only select the queries that are on the boundary of the convex hull of the reduced set $\mathcal{X}$. We then apply k-medoids with k clusters over the queries on the boundary and finally only accept the cluster medoids as the selected batch. As shown in Figure 3 (c), we first find $k = 5$ clusters over the points on the boundary of the convex hull of $\mathcal{X}$. We note that the number of queries on the boundary of convex hull of $\mathcal{X}$ can be larger than the number of queries needed in a batch, e.g., there are 7 points on the boundary; however, we only select the medoids of the 5 clusters created over these boundary queries shown in orange.

5.4 Successive Elimination

One of the main objectives of batch generation for active learning as described in the previous methods is to select k queries that will maximize the average distance among them out of the N queries in the reduced set $\mathcal{X}$. This problem is also referred to as *max-sum diversification* in literature, which is known to be NP-hard [22, 39]. However, there exists a set of algorithms that provide approximate solutions [28].

What makes our batch generation problem special and different from standard max-sum diversification is that we can compute the mutual information for each query. Mutual information is a metric that models how much we want a query to be in the final batch. Thus, we propose a novel method that leverages the mutual information values to successively eliminate queries for the goal of obtaining a satisfactory diversified set. We refer to this algorithm as *Successive Elimination*. At every iteration of the algorithm, we select two closest queries (in terms of Euclidean distance of their ψ vectors) in the reduced set $\mathcal{X}$, and remove the one with lower mutual information value. We repeat this procedure until k points are left in the set, resulting in the k queries in our final batch, which efficiently increases the diversity among queries.

A pseudo-code of this method is given in Algorithm 3. Figure 3 (d) shows the successive pairwise comparisons between two queries based on their corresponding mutual information. In every pairwise comparison, we eliminate one of the queries, shown with black edge, keeping the query connected with the orange edge. The numbers show the order of comparisons made before finding $k = 5$ queries shown in orange.

So far, we have presented four methods for batch selection in active preference-based reward learning: greedy, medoids, boundary medoids and successive elimination. While these methods are computationally efficient and easy-to-implement, they rely on rough heuristics. Determinantal point processes (DPP), on the other hand, provide a tractable mathematical procedure that we can elegantly adopt for batch selection. In the next section, we present our premier method based on DPPs.

Algorithm 3 Successive Elimination

```
X, q ← ReduceDataset(w, K, N)
A ← X                                          ▷ Initialize the batch
while |A| > k do
    (ψ_i, ψ_j) ← arg min_{ψ_i,ψ_j ∈A} ||ψ_i − ψ_j||_2
    if q_i < q_j then
        Remove ψ_i from A
    else
        Remove ψ_j from A
    end if
end while
return A
```

6 DPP-BASED BATCH ACTIVE LEARNING

Determinantal point processes (DPP) are a class of distributions that promote diversity. They are a natural fit for our problem as they can be tuned to balance the tradeoff between diversity and how desirable each item is. In our approach, we regard the set of queries as the item set of DPPs. We first start with presenting the necessary background on DPPs.

6.1 Background

A point process is a probability measure on a ground set $\mathcal{X}$ over finite subsets of $\mathcal{X}$. In our batch active preference-based learning framework, $\mathcal{X}$ is a set of queries. We let $|\mathcal{X}| = K$.

An L-ensemble defines a DPP through a real, symmetric and positive semidefinite (PSD) K-by-K kernel matrix L [23]. Then, sampling a subset $X = A \subseteq \mathcal{X}$ has the probability

$$P(X = A) \propto \det L_A \tag{11}$$

where L_A is an $|A|$-by-$|A|$ matrix that consists of the rows and columns of L that correspond to the queries in A. For instance, if $A = \{i, j\}$, i.e. A is a set consisting of i^{th} and j^{th} queries in $\mathcal{X}$, then

$$P(X = A) \propto L_{ii}L_{jj} - L_{ij}L_{ji}.$$

We can consider $L_{ij} = L_{ji}$ as a similarity measure between the queries i and j in the set. The nonnegativeness of the second term in the above expression shows an example of *repulsiveness* property of DPPs. This property makes DPPs the ubiquitous tractable point process to model negative correlations, and useful for generating diverse batches.

As $\det L_A$ can be positive for various A with different cardinalities, we do not know $|A|$ in advance. There is an extension of DPPs referred to as k-DPP where it is guaranteed that $|A| = k$, and Eqn. (11) remains valid [51]. In this work, we employ k-DPPs and refer to them as DPPs for the rest of the paper for brevity.

Now, we explain what parameters we can have in an L-ensemble DPP. We note that

$$P(X = A) \propto \det L_A = \text{Vol}(\{L_i\}_{i \in A}),$$

so the probability is proportional to the square of the associated volume.[3] In fact, by using a generalized version of DPPs, we can approximately achieve [6, 61]:

$$P(X = A) \propto \text{Vol}^{\alpha}(\{L_i\}_{i \in A}), \tag{12}$$

[3]Volume here refers to the volume of the parallelepiped spanned by the columns of L.

for $\alpha \geq 0$. One can note that higher α enforces more diversity, because the probability of more diverse sets (larger volumes) will be boosted against the less diverse sets. We visualize this in Figure 4.

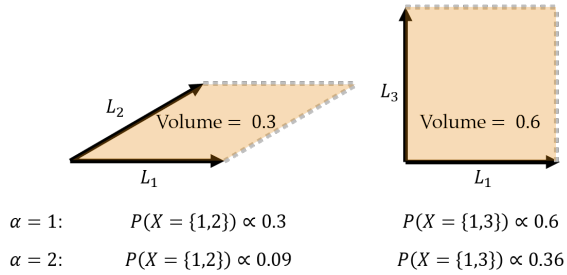


Fig. 4. The effect of α is visualized. The columns of the matrix L have the same length here; however $\{1, 3\}$ is a more diverse set than $\{1, 2\}$. When $\alpha = 1$, $\{1, 3\}$ is two times more likely to be sampled from the DPP distribution than $\{1, 2\}$. When we increase α to 2, this ratio increases to 4, since more diverse sets are boosted against the less diverse sets.

What remains is to construct the kernel matrix L. For this, we first define a matrix $S \in \mathbb{R}^{K \times K}$ whose entries measure the similarity between the queries. In our problem, each query has a feature difference vector ψ, and close ψ's correspond to similar queries in terms of the information they provide. Therefore, we let

$$S_{ij} = \exp\left(-\frac{\|\psi_i - \psi_j\|_2^2}{2\sigma^2}\right), \tag{13}$$

where σ is a hyperparameter. We then define the matrix L as

$$L_{ij} = q_i^{\gamma/\alpha} S_{ij} q_j^{\gamma/\alpha}, \tag{14}$$

which is guaranteed to be PSD by the construction of S. Here, γ is another hyperparameter and $q_i \in \mathbb{R}_{\geq 0}$ is the *score* of i^{th} query that represents how much we want that query in our batch. We use these scores to weight the queries based on their mutual information values, as computed by the objective of Eqn. (9). By increasing γ for fixed α, we give more importance to the scores than diversity. This enables us set the tradeoff between informativeness and diversity.

Relating the Mode of a DPP with High Diversity and Informativeness. With proper tuning of α and γ, the batches that are both diverse and informative will have higher probabilities of being sampled. This motivates us to find the mode of the distribution, i.e., $\arg\max_A P(X = A)$, which will guarantee informativeness and diversity. Another advantage of using the mode, instead of a random sample from the distribution, is the fact that it is significantly faster to approximate, even compared to the approximate sampling methods [5, 6, 56, 61].

6.2 Approximating the Mode of a DPP

Finding the mode of a DPP exactly is NP-hard [48]. It is hard to even approximate it better than a factor of 2^{ck} for some $c > 0$, under a cardinality constraint of size k [32]. Here, we discuss a greedy optimization algorithm to approximate the mode of a DPP.

In this approach, queries are greedily added to the batch. More formally, to approximate

$$\arg\max_A P(X = A) = \arg\max_A \text{Vol}^\alpha(\{L_i\}_{i \in A}),$$

we greedily add queries to A. Let $A^{(l)}$ denote the set of selected queries at iteration l of batch generation. We have

$$A^{(l+1)} = A^{(l)} \cup \{\arg\max_j \text{Vol}^\alpha(\{L_i\}_{i \in A^{(l)} \cup \{j\}})\},$$

which we repeat until we obtain k queries in A. Çivril and Magdon-Ismail [31] showed that the greedy algorithm always finds a $k^{O(k)}$-approximation to the mode.

An important advantage of greedily approximating the mode is that the hyperparameter α becomes irrelevant, as it is just an exponent in the objective in every iteration of batch generation, unless trivially $\alpha = 0$. This reduces the burden of hyperparameter tuning.

While we use this greedy approach in our experiments for DPP-based batch generation, we also present a novel tractable algorithm, *maximum coordinate rounding*, for approximating the mode of a DPP with better approximation ratio in Appendix A. This algorithm is based on a convex relaxation of the optimization problem for finding the mode, which we solve using stochastic mirror descent. With the convex relaxation, we first perform the optimization as if we can take proportions of each query and then recursively select the queries by rounding. This algorithm achieves an e^k-approximation of the mode. The reason why we resort to the greedy approach in our experiments is because the maximum coordinate rounding algorithm, despite having polynomial time complexity, has much higher computational cost with large batch sizes in practice.

6.3 Overall Algorithm

Having presented the background in DPPs and the method to approximately find the DPP-mode, which corresponds to our diverse and informative batch, we are now ready to present our overall DPP-based batch active preference-based learning algorithm.

In this algorithm, we approximately compute the mode of the DPP distribution over the reduced set $\mathcal{X}$ as our batch. Algorithm 4 presents the DPP-based method. The first for-loop (lines 2 through 7) constructs the DPP kernel, and the second part (lines 8 through 11) generates the batch by greedily approximating the mode of the constructed DPP. In our experiments, we set $\gamma = 1$ and σ to be the expected distance between two nearest points (in terms of Euclidean distance) when k points are selected uniformly at random in the space $[0, 1]^d$ where d is the number of features, i.e., $d = \dim(\phi(\xi))$. This heuristic ensures a good kernel in line 4 of the algorithm as it is proportional to the Euclidean distance between queries.

We make the code for all of our batch active learning methods available at https://bit.ly/381brBK. We also integrated them into APReL [19], our comprehensive Python library for active preference based reward learning algorithms, which now enables experimenting batch active learning methods with various objectives, e.g., volume removal [68], max-regret [82], etc. This is available at https://github.com/Stanford-ILIAD/APReL.

7 SIMULATIONS AND EXPERIMENTS

Experimental Setup. We have performed several simulations and experiments to compare the methods we propose and to demonstrate their performance. Unless otherwise stated, we set batch size $k = 10$, reduced query set size $N = 200$ and number of $\boldsymbol{w}$ samples $M = 1000$ in all experiments.

Alignment Metric. For our simulations, we generate synthetic random $\boldsymbol{w}_{\text{true}}$ vectors as our true preference vector. We have used the following alignment metric [68] in order to compare non-batch active, batch active and random query

Algorithm 4 DPP-based Batch Generation

Require: DPP hyperparameters σ, γ
$\mathcal{X}, \mathbf{q} \leftarrow \textsc{ReduceDataset}(\boldsymbol{w}, \mathcal{K}, N)$
for ψ_i in $\mathcal{X}$ **do**
  for ψ_j in $\mathcal{X}$ **do**
    $S_{ij} \leftarrow \exp\left(-\frac{\|\psi_i - \psi_j\|_2^2}{2\sigma^2}\right)$
    $L_{ij} \leftarrow q_i^\gamma S_{ij} q_j^\gamma$
  end for
end for
$A \leftarrow \emptyset$ ▷ Initialize the batch
for $i = 1, \ldots, k$ **do**
  $A \leftarrow A \cup \{\arg\max_j \det L_{A\cup\{j\}}\}$
end for
return A

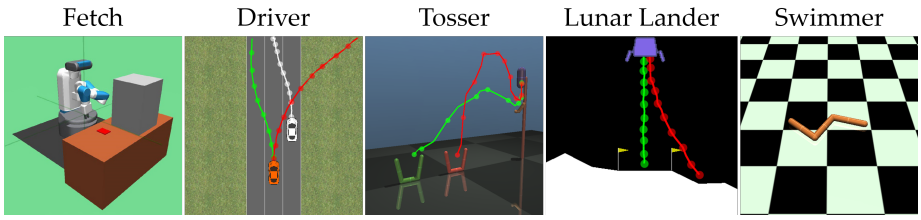


Fig. 5. Simulation view of each environment. (a) Fetch, (b) Driver, (c) Tosser, (d) Lunar Lander, (e) Swimmer.

selection methods, where all queries are selected randomly over all feasible trajectories.

$$m = \frac{\boldsymbol{w}_{\text{true}}^\top \hat{\boldsymbol{w}}}{\|\boldsymbol{w}_{\text{true}}\|_2 \|\hat{\boldsymbol{w}}\|_2}. \tag{15}$$

where $\hat{\boldsymbol{w}}$ is $\mathbb{E}[\boldsymbol{w}]$ based on the estimate of the learned distribution of $\boldsymbol{w}$. We note that this alignment metric can be used to test convergence, because the value of m being close to 1 means the estimate of $\boldsymbol{w}$ is very close to (aligned with) the true weight vector. In our experiments, we compare the methods using m and the number of queries generated.

Loglikelihood Metric. Recent work has identified drawbacks of the alignment metric alogn with a discussion of other possible metrics [80]. Since our focus in this paper is reward learning, not reward optimization [15], measuring the reward (or regret) of a policy that is optimized via the learned method is not a suitable metric. Instead, we use the loglikelihood metric which measures the loglikelihood of a held-out preference dataset [14, 15, 81].

7.1 Tasks

We perform experiments in different simulation environments that are summarized in Table 1 with a list of the variables associated with every environment, where d is the number of features, and T is the horizon, i.e., the number of time steps. Note that these are all relatively short-horizon environments. This is because the optimization of queries in the non-batch active method, as it was described in [68], is over a $2 \times (T \dim(u^t)) + \dim(x^0)$ dimensional space, where the factor 2 is because we generate 2 trajectories with the same initial state for each query. To keep the query synthesis tractable, we therefore modified the original environments to have relatively short horizons. Using a pregenerated

Table 1. Environment Properties

Task Name	$\dim(u^t)$	T	d
Fetch	7	19	4
Driver	2	5	4
Tosser	2	2	4
Lunar Lander*	2	5	6
Swimmer	2	12	3

* Continuous version has been used.

dataset of queries as we do in the batch-active methods, however, eliminates this issue. Figure 5 visualizes each of the experiment environments with some sample trajectories.

Fetch. Following [65], we use the simulator for Fetch mobile manipulator robot [84], visualized in Figure 5 (a). For $\phi(\xi)$, we use features that correspond to average and final distances to the target object (red block), average distance to the table (brown block), and average distance to the obstacle (gray block).

Driver. We use the 2D driving simulator [69], shown in Figure 5 (b). We use features corresponding to distance to the closest lane, speed, heading angle, and distance to the other vehicle in the scenario. Two sample trajectories are shown in red and green in Figure 5 (b). In addition, the white line shows the fixed trajectory of the other vehicle on the road.

Tosser. We use MuJoCo's Tosser [73] where a robot tosses a capsule-shaped object. The features we use are maximum horizontal range, maximum altitude, the sum of angular displacements at each timestep and final distance to closest basket of the object. The two red and green trajectories in Figure 5 (c) correspond to synthesized queries showing different preferences for what basket to toss the object to.

Lunar Lander. We use OpenAI Gym's Lunar Lander [24] where a spacecraft is controlled. We use features corresponding to final heading angle, final distance to landing pad, total rotation, path length, final vertical speed, and flight duration. Two sample trajectories are shown in red and green in Figure 5 (d).

Swimmer. We use OpenAI Gym's Swimmer [24]. We use features corresponding to horizontal displacement, vertical displacement, and total distance traveled. The environment is shown in Figure 5 (e).

7.2 Comparison of Batch-Active Learning Methods

We first quantitatively compare the batch-active methods we proposed with each other, as well as the combinatorial optimization problem posed in (10) for which we use simulated annealing [13, 49] as an approximate solution method. While simulated annealing could be run longer and longer to achieve better results, this would defeat the purpose of batch-mode active learning. Therefore, we limit its running time to match with the slowest algorithm among our proposed batch active methods. For each environment, we create a dataset of $K = 500{,}000$ queries that consist of trajectories that are generated by taking random actions from a fixed initial state. Note that exploration within the environment is not an important issue here, because it is possible to learn the true reward function by only comparing suboptimal trajectories as long as the queries cover the feature space well, i.e., we do not need to ensure there are successful trajectories in the query dataset.

Independently for each environment, we randomly generated 100 different reward functions ($\boldsymbol{w}_{\text{true}}$ vectors) for tests of all methods. We then simulated noiseless users, who always reveal their true preferences in order to eliminate the effect of noise in the results. However, the learning methods still adopted the noisy user model we presented in Eqn. (6).

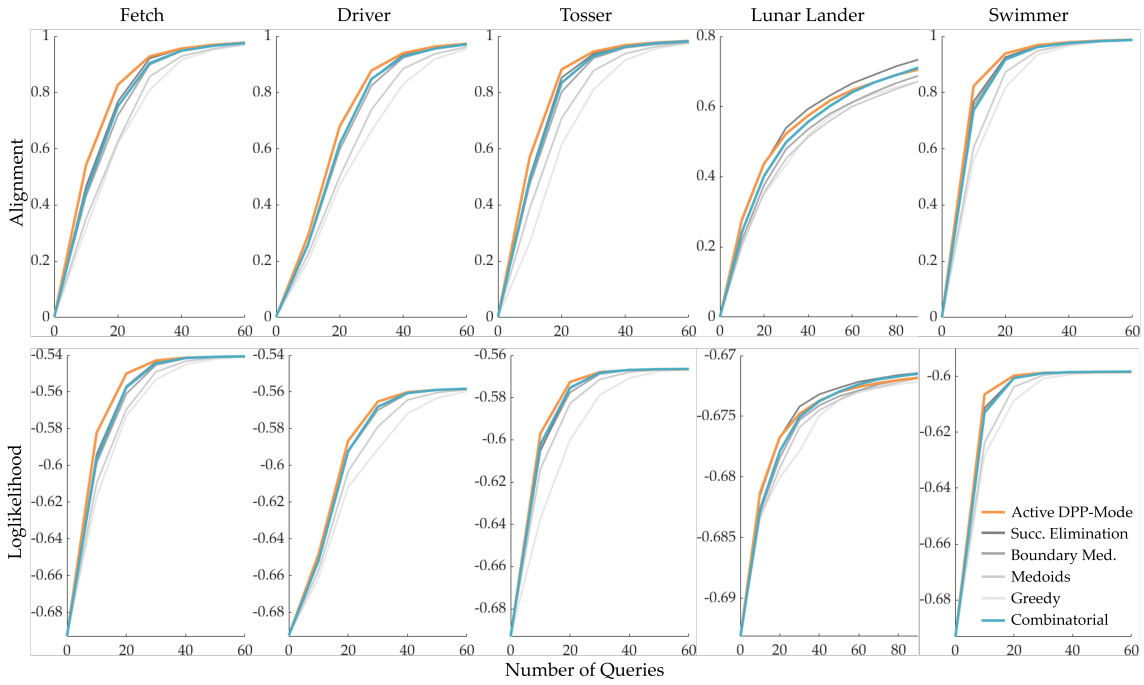


Fig. 6. Batch-active learning methods are compared.

For each simulated reward function during our tests, we ran 6 batch generations with each method, summing up to 60 pairwise comparison queries. For Lunar Lander where learning is more dificult due to larger feature dimensionality, we ran 9 batches (90 queries). We demonstrate the results in Figure 6. Since the reward functions are paired between the methods, we use Wilcoxon signed-rank tests [79] over the *area under the curve* for the alignment metric as it was done in previous work [62]. Our results suggest that the DPP-based method significantly outperforms all other methods, including the combinatorial optimization via simulated annealing, in all environments ($p < 0.005$) except Lunar Lander where successive elimination is the best-performing method.

Among the heuristic-based batch active learning methods we proposed, successive elimination method significantly outperforms the others ($p < 0.005$ in all except Swimmer where it is only $p < 0.05$ against the boundary medoids method). It also outperforms the combinatorial optimization in all environments ($p < 0.005$) except Driver where both methods perform comparably.

Combinatorial optimization and boundary medoids both significantly outperform medoids and greedy methods in all environments ($p < 0.005$). While boundary medoids method significantly outperforms combinatorial optimization in Swimmer ($p < 0.05$), combinatorial optimization is significantly better in the other environments ($p < 0.05$ in Fetch and $p < 0.005$ in others). Finally, medoids method significantly outperforms the greedy method in all environments ($p < 0.005$) except Lunar Lander where they perform comparably.

Overall, these results show us the ranking of batch active learning methods from the best to the worst are as follows:

(1) Active DPP-Mode
(2) Successive Elimination

Table 2. Average Query Generation Times (seconds)

Environment	Non-Batch	Batch Active Learning					
		Combinatorial	*Greedy*	*Medoids*	*Boundary Med.*	*Succ. Elimination*	*DPP*
Fetch	N/A	3.70	3.34	3.33	3.36	3.67	3.36
Driver	36.42	3.28	2.85	2.87	2.87	3.23	2.89
Tosser	46.21	3.31	2.88	2.89	2.90	3.21	2.89
LunarLander	69.42	3.33	3.03	3.02	3.03	3.36	3.04
Swimmer	146.32	3.29	2.87	2.89	2.89	3.22	2.89

(3) Combinatorial Optimization via Simulated Annealing
(4) Boundary Medoids
(5) Medoids
(6) Greedy

7.3 Comparison to Non-Batch Active Learning

We next investigated the average time it required to generate one query. For this, we again took a dataset of $K = 500{,}000$ queries. We recorded the batch generation times, and divided it by $k = 10$ to get the time per query. To show the advantage of batch-active learning methods, we also ran the same analysis on the non-batch active learning approach. We had to exclude the non-batch active method in the *Fetch* environment, as it was not able to synthesize queries in reasonable amounts of time due to the large action space. The results are shown in Table 2. It can be seen that batch active learning methods lead to a great decrease in query generation times compared to the non-batch method.

As we observed that our DPP-based method generates highly informative queries in a time-efficient way, we now compare its performance to the non-batch active learning approaches. Specifically, we assessed its performance against non-batch active learning and random query selection where queries are selected uniformly at random from $\mathcal{K}$.

We show the results in Figures 7 and 8. Figure 7 shows the convergence to the true weights $\boldsymbol{w}_{\text{true}}$ in terms of alignment and loglikelihood metrics as the number of queries increases. It is interesting to note that non-batch active learning performs suboptimally in *LunarLander* and *Tosser*. This is because the continuous query synthesis by optimizing over the initial state and the sequences of actions fails in these environments. Our DPP-based method, on the other hand, sidesteps this problem due to discrete optimization over the query set.

Naturally, non-batch active method would perform better than the DPP-based method if it also selected queries over the discrete set. However, this would require computing mutual information for every query in the set in each and every iteration, which is too slow. Figure 8 evaluates the computation time required for querying among our DPP-based method, non-batch active, and random querying by plotting the learning curves during the first 4 minutes of learning averaged over 100 seeds. It is clearly visible that batch active learning makes the process much faster than the non-batch active method and random querying. Therefore, batch active learning is preferable over other methods as it balances the tradeoff between the number of queries required and the time it takes to compute the queries. This tradeoff can be seen in Figure 9 where we simulated *Driver* with varying k values. For these simulations, we set $N = 20k$ in accordance with other experiments.

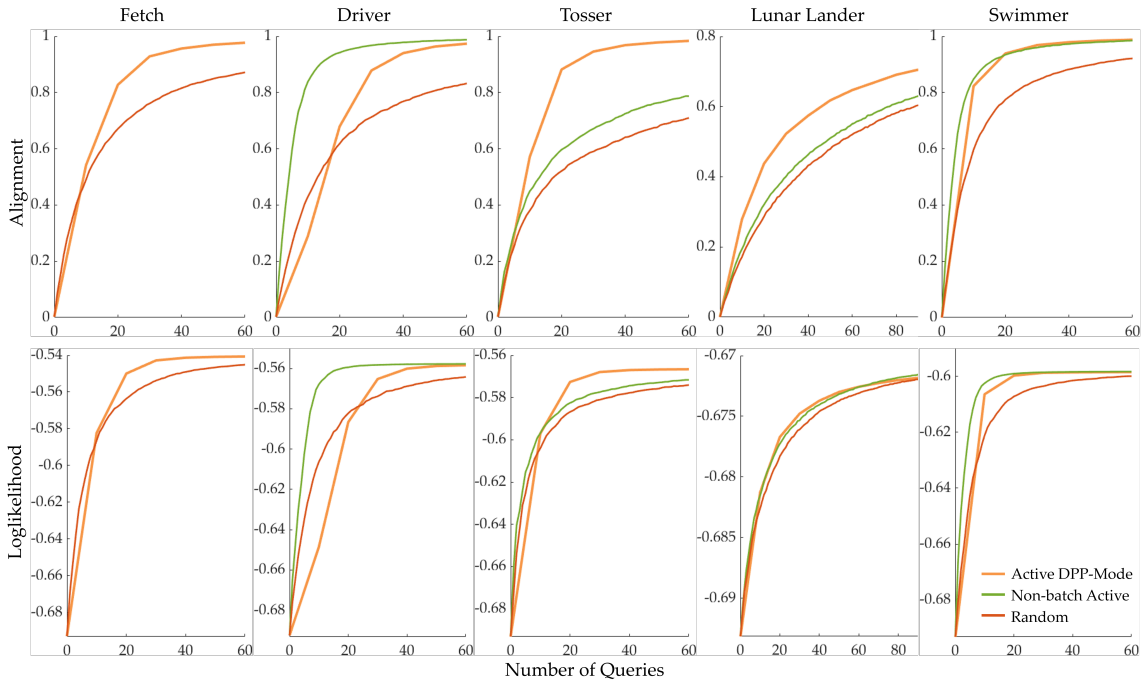


Fig. 7. The performance the algorithms is shown. The non-batch active method performs poorly on Lunar Lander and Tosser.

7.4 User Preferences

In addition to our simulation results using synthetic $\boldsymbol{w}_{\text{true}}$ vectors, we perform a user study to learn humans' preferences for the *Driver* and *Tosser* environments. This experiment is mainly designed to show the ability of our framework to learn humans' preferences when human responses may be noisy.

Setup. We recruited 10 users (ages 18–53, 4 female, 6 male) each of whom responded to 150 queries generated by successive elimination algorithm under the volume removal acquisition function [68] for each environment (*Driver* or *Tosser*) with batch size $k = 10$. None of the users had prior experience with the environments. Users were instructed their goal is to achieve safe and efficient driving in the *Driver* environment. On the other hand, they were told they are free to choose any of the two baskets in the *Tosser* environment as the target (see Figure 5 (c)). The goal of this is to demonstrate our batch-active preference based learning algorithms can learn different reward functions that correspond to different behaviors in the environment.

Driver Preferences. Using successive elimination, we are able to learn humans' driving preferences. Our results show that the preferences of users are very close to each other as this task mainly models natural driving behavior. This is consistent with results shown by [68], where non-batch techniques are used. We noticed a few differences between the driving behavior as shown in Figure 10. This figure shows the distribution of the weights for the four features after 150 queries. Two of the users (plot on the right) seem to have slightly different preferences about collision avoidance, which can correspond to more aggressive driving behavior. We observed that 70 queries were enough for qualitatively converging to safe and sensible driving in the defined scenario. The optimized driving with different number of queries can be watched on https://youtu.be/MaswyWRep5g.

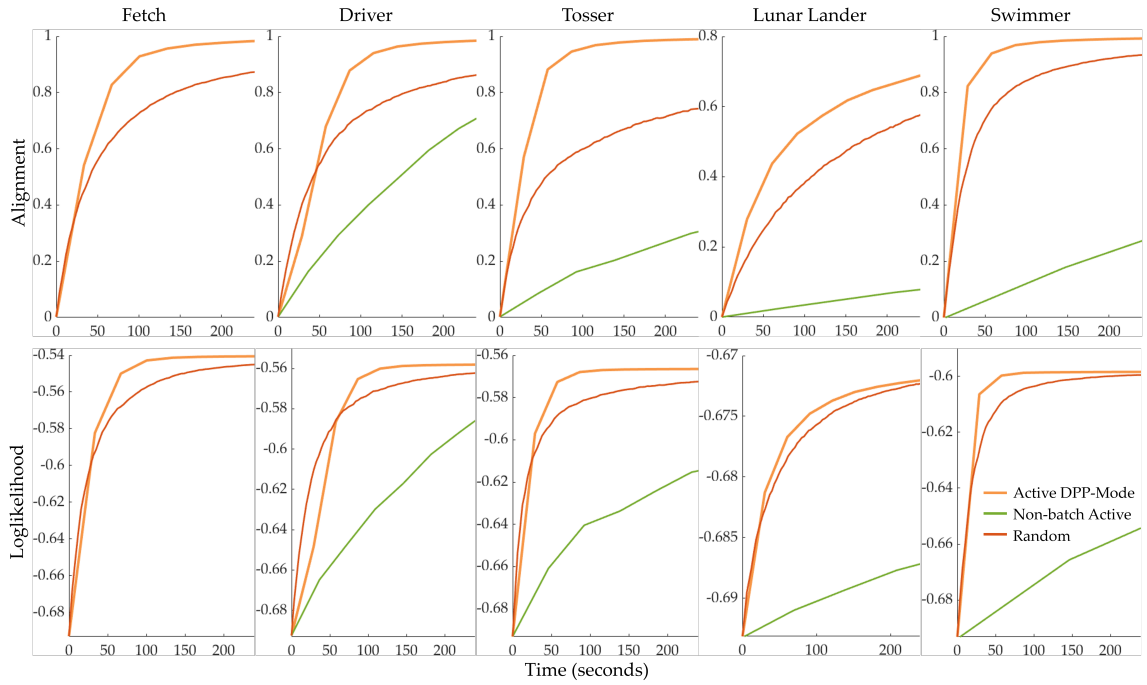


Fig. 8. Alignment and loglikelihood as a function of time are plotted for each environment. Non-batch active learning method is slow due to the optimization and adaptive metropolis algorithm involved in each iteration, whereas random querying performs poorly due to redundant queries. The DPP-based method clearly outperforms both of them.

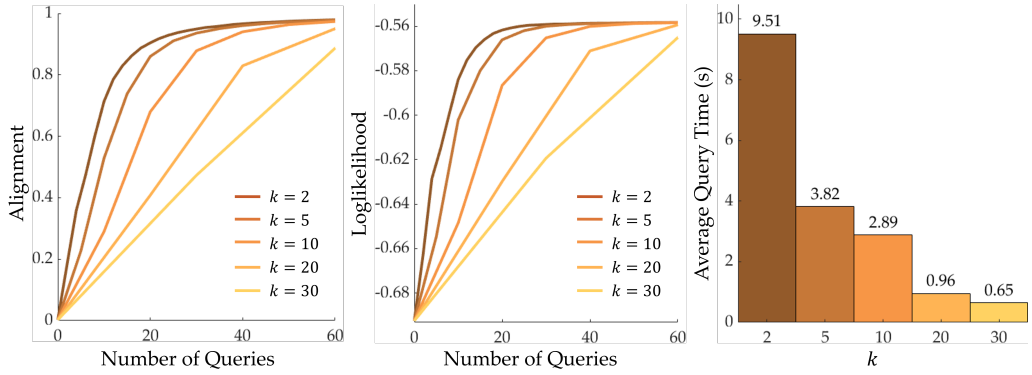


Fig. 9. The performance of our DPP-based algorithm with varying k values was averaged over 100 different runs with *Driver* where $\boldsymbol{w}_{\text{true}}$ is uniformly randomly generated and $N = 20k$. (a) Alignment, (b) loglikelihood, and (c) average query times.

Tosser Preferences. Similarly, we use successive elimination to learn humans' preferences on the tosser task. Figure 11 shows we learn interesting tossing preferences, varying between the users based on their target. For demonstration purposes, we optimize the control inputs with respect to the preferences of two of the users, one of whom prefers the green basket while the other prefers the red one. The evolution of the learning can be watched on

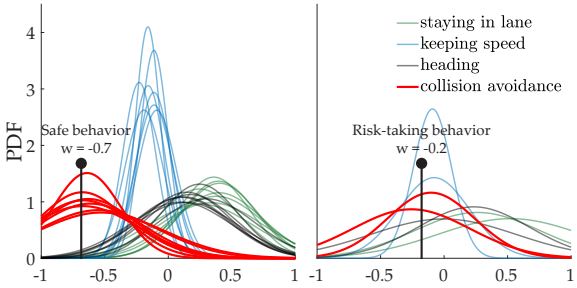


Fig. 10. User preferences on *Driver* task are grouped into two sets. While the first set shows the preferences conforming with the natural driving behavior, the second set is comprised of data from two users one of whom preferred collisions with the other car over leaving the road and the other regarded some collisions as near-misses and thought they can be acceptable in order to keep speed. It can be seen that the uncertainty in their learned preferences is higher.

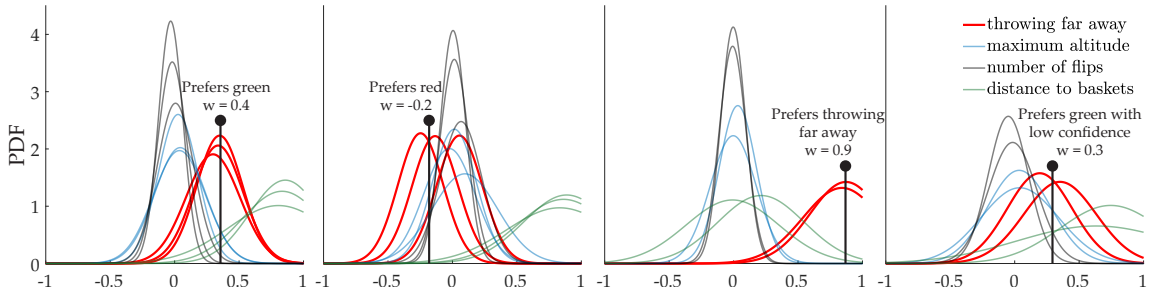


Fig. 11. User preferences on *Tosser* task are grouped into four sets. The first set shows the preferences of people who aimed at throwing the ball into the green basket (the distant one) but accepted throwing into the other basket is better than not throwing into any baskets. The second set is comprised of data from three users who preferred the red basket (the closer one). In the third group, the users preferred the green basket over the red one, but also accepted throwing far away is better than throwing into the red basket, because it is an attempt for the green basket. Lastly, the fourth group is similar to the first group; however the confidence over preferences is much less, because the users were not sure about how to compare the cases where the ball was dropped between the baskets in one of the trajectories.

https://youtu.be/cQ7vvUg9rU4. We note that 100 queries were enough to qualitatively see reasonable convergence in this task.

8 DISCUSSION

Summary. In this work, we proposed several end-to-end methods to efficiently learn humans' preferences on dynamical systems. Compared to the previous studies, our method requires only a small number of queries which are generated in a reasonable amount of time. We demonstrated the performance of our algorithms in simulation.

Limitations. In our experiments, we sampled from the trajectory space in advance for batch active learning methods to generate the dataset of queries $\mathcal{K}$, while we still employed the optimization formulation for the non-batch active version as it was originally proposed by [68]. It can be argued that this creates a bias on the computation times. However, there are two points that make batch active techniques computationally more efficient than the non-batch version. First, they require computing the mutual information values only once in every k queries, whereas the non-batch method requires it for every query. Second, even if we used a pre-generated query dataset for the non-batch active

learning method, it would be still inefficient due to adaptive Metropolis algorithm. Furthermore, it can be inferred from Figure 9 that non-batch active learning with sampling the control space would take a significantly longer running time compared to batch active versions. We also note that query space discretization could reduce the performance of non-batch active learning.

Future directions. In this study, we used a fixed batch-size. However, we know that the first queries are more informative than the following queries. Therefore, one could start with smaller batch sizes and increase over time. This would both make the first queries more informative and the following queries computationally faster. Hence, further research is warranted to optimize varying batch sizes.

Lastly, we used handcrafted feature transformations, ϕ's, in this study. In the future we plan to learn those transformations, as in [46], along with the preferences by developing batch techniques that use as few preference queries as possible generated in a short amount of time.

ACKNOWLEDGMENTS

The authors would like to thank Kenneth Wang for fruitful discussions about the DPP-based batch active learning method; and acknowledge ONR, AFOSR YIP, DARPA YFA, NSF awards #2218760 and #2125511, and Ford.

REFERENCES

[1] Nir Ailon. 2012. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research* 13, Jan (2012), 137–164.

[2] Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L Thomaz. 2012. Keyframe-based learning from demonstration. *International Journal of Social Robotics* 4, 4 (2012), 343–355.

[3] Riad Akrour, Marc Schoenauer, and Michele Sebag. 2011. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 12–27.

[4] Riad Akrour, Marc Schoenauer, and Michèle Sebag. 2012. April: Active preference learning-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 116–131.

[5] Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. 2016. Monte Carlo Markov chain algorithms for sampling strongly Rayleigh distributions and determinantal point processes. In *Conference on Learning Theory*. 103–115.

[6] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. 2019. Log-concave polynomials II: high-dimensional walks and an FPRAS for counting bases of a matroid. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 1–12.

[7] Galen Andrew and Jianfeng Gao. 2007. Scalable training of L 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*. ACM, 33–40.

[8] Vivek Bagaria, Govinda M Kamath, Vasilis Ntranos, Martin J Zhang, and David Tse. 2017. Medoids in almost linear time via multi-armed bandits. *arXiv preprint arXiv:1711.00817* (2017).

[9] Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. 2018. Learning from physical human corrections, one feature at a time. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. 141–149.

[10] Chandrayee Basu, Erdem Biyik, Zhixun He, Mukesh Singhal, and Dorsa Sadigh. 2019. Active Learning of Reward Dynamics from Hierarchical Queries. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. https://doi.org/10.1109/IROS40897.2019.8968522

[11] Chandrayee Basu, Qian Yang, David Hungerman, Mukesh Singhal, and Anca D Dragan. 2017. Do you want your autonomous car to drive like you?. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. ACM, 417–425.

[12] Christian Bauckhage. 2015. Numpy/scipy Recipes for Data Science: k-Medoids Clustering. *researchgate.net, Feb* (2015).

[13] Dimitris Bertsimas and John Tsitsiklis. 1993. Simulated annealing. *Statistical science* 8, 1 (1993), 10–15.

[14] Erdem Biyik, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. 2020. Active Preference-Based Gaussian Process Regression for Reward Learning. In *Proceedings of Robotics: Science and Systems (RSS)*.

[15] Erdem Biyik, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. 2023. Active Preference-Based Gaussian Process Regression for Reward Learning and Optimization. *The International Journal of Robotics Research (IJRR)* (2023).

[16] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. 2022. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research* 41, 1 (2022), 45–67.

[17] Erdem Biyik, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. 2019. Asking Easy Questions: A User-Friendly Approach to Active Reward Learning. In *Proceedings of the 3rd Conference on Robot Learning (CoRL)*.

[18] Erdem Biyik and Dorsa Sadigh. 2018. Batch Active Preference-Based Learning of Reward Functions. In *Proceedings of the 2nd Conference on Robot Learning (CoRL) (Proceedings of Machine Learning Research, Vol. 87)*. PMLR, 519–528.

[19] Erdem Bıyık, Aditi Talati, and Dorsa Sadigh. 2022. APReL: A Library for Active Preference-based Reward Learning Algorithms. In *Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction*. 613–617.

[20] Erdem Biyik, Kenneth Wang, Nima Anari, and Dorsa Sadigh. 2019. Batch Active Learning Using Determinantal Point Processes. *arXiv preprint arXiv:1906.07975* (2019).

[21] Erdem Biyik, Fan Yao, Yinlam Chow, Alex Haig, Chih-wei Hsu, Mohammad Ghavamzadeh, and Craig Boutilier. 2023. Preference Elicitation with Soft Attributes in Interactive Recommendation. *arXiv preprint arXiv:2311.02085* (2023).

[22] Allan Borodin, Hyun Chul Lee, and Yuli Ye. 2012. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*. ACM, 155–166.

[23] Alexei Borodin and Eric M Rains. 2005. Eynard–Mehta theorem, Schur process, and their Pfaffian analogs. *Journal of statistical physics* 121, 3-4 (2005), 291–317.

[24] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[25] Daniel S Brown, Wonjoon Goo, and Scott Niekum. 2020. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*. PMLR, 330–359.

[26] Thiago NC Cardoso, Rodrigo M Silva, Sérgio Canuto, Mirella M Moro, and Marcos A Gonçalves. 2017. Ranked batch-mode active learning. *Information Sciences* 379 (2017), 313–337.

[27] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. 2023. Open problems and fundamental limitations of reinforcement learning from human feedback. *Transactions on Machine Learning Research* (2023).

[28] Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. 2017. Local search for max-sum diversification. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 130–142.

[29] Yuxin Chen and Andreas Krause. 2013. Near-optimal Batch Mode Active Learning and Adaptive Submodular Optimization. *ICML (1)* 28 (2013), 160–168.

[30] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*. 4302–4310.

[31] Ali Çivril and Malik Magdon-Ismail. 2009. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science* 410, 47 (2009), 4801.

[32] Ali Civril and Malik Magdon-Ismail. 2013. Exponential inapproximability of selecting a maximum volume sub-matrix. *Algorithmica* 65, 1 (2013), 159–176.

[33] Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.

[34] Nguyen Viet Cuong, Wee Sun Lee, Nan Ye, Kian Ming A Chai, and Hai Leong Chieu. 2013. Active learning for probabilistic hypotheses using the maximum Gibbs error criterion. In *Advances in Neural Information Processing Systems*. 1457–1465.

[35] Marco De Gemmis, Leo Iaquinta, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. 2009. Preference learning in recommender systems. *Preference Learning* 41 (2009).

[36] Ehsan Elhamifar, Guillermo Sapiro, and S Shankar Sastry. 2016. Dissimilarity-based sparse subset selection. *IEEE transactions on pattern analysis and machine intelligence* 38, 11 (2016), 2182–2197.

[37] Evan Ellis, Gaurav R. Ghosal, Stuart J. Russell, Anca Dragan, and Erdem Biyik. 2024. A Generalized Acquisition Function for Preference-based Reward Learning. In *International Conference on Robotics and Automation (ICRA)*.

[38] Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. 2012. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning* 89, 1-2 (2012), 123–156.

[39] Sreenivas Gollapudi and Aneesh Sharma. 2009. An axiomatic approach for result diversification. In *Proceedings of the 18th international conference on World wide web*. ACM, 381–390.

[40] Yuhong Guo and Dale Schuurmans. 2008. Discriminative batch mode active learning. In *Advances in neural information processing systems*. 593–600.

[41] Heikki Haario, Eero Saksman, Johanna Tamminen, et al. 2001. An adaptive Metropolis algorithm. *Bernoulli* 7, 2 (2001), 223–242.

[42] Joey Hejna, Rafael Rafailov, Harshit Sikchi, Chelsea Finn, Scott Niekum, W. Bradley Knox, and Dorsa Sadigh. 2024. Contrastive Preference Learning: Learning From Human Feedback without RL. In *International Conference on Learning Representations (ICLR)*.

[43] Jonathan Hermon and Justin Salez. 2019. Modified log-Sobolev inequalities for strong-Rayleigh measures. *arXiv preprint arXiv:1902.02775* (2019).

[44] Rachel Holladay, Shervin Javdani, Anca Dragan, and Siddhartha Srinivasa. 2016. Active comparison based learning incorporating user uncertainty and noise. In *RSS Workshop on Model Learning for Human-Robot Communication*.

[45] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. 2015. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research* 34, 10 (2015), 1296–1313.

[46] Sydney M Katz, Amir Maleki, Erdem Bıyık, and Mykel J Kochenderfer. 2021. Preference-based learning of reward function features. *arXiv preprint arXiv:2103.02727* (2021).

[47] Leonard Kaufman and Peter Rousseeuw. 1987. *Clustering by means of medoids*. North-Holland.

[48] Chun-Wa Ko, Jon Lee, and Maurice Queyranne. 1995. An exact algorithm for maximum entropy sampling. *Operations Research* 43, 4 (1995), 684–691.
[49] Mykel J Kochenderfer and Tim A Wheeler. 2019. *Algorithms for optimization.* Mit Press.
[50] David Krueger, Jan Leike, Owain Evans, and John Salvatier. 2016. Active reinforcement learning: Observing rewards at a cost. In *Future of Interactive Learning Machines, NIPS Workshop.*
[51] Alex Kulesza and Ben Taskar. 2011. k-DPPs: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11).* 1193–1200.
[52] Alex Kulesza and Ben Taskar. 2012. *Determinantal Point Processes for Machine Learning.* Now Publishers Inc., Hanover, MA, USA.
[53] Minae Kwon, Erdem Biyik, Aditi Talati, Karan Bhasin, Dylan P. Losey, and Dorsa Sadigh. 2020. When Humans Aren't Optimal: Robots that Collaborate with Risk-Aware Humans. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI).* https://doi.org/10.1145/3319502.3374832
[54] Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. 2021. B-Pref: Benchmarking Preference-Based Reinforcement Learning. In *Neural Information Processing Systems (NeurIPS).*
[55] Sergey Levine and Vladlen Koltun. 2012. Continuous inverse optimal control with locally optimal examples. In *Proceedings of the 29th International Conference on Machine Learning.* 475–482.
[56] Chengtao Li, Suvrit Sra, and Stefanie Jegelka. 2016. Fast mixing markov chains for strongly Rayleigh measures, DPPs, and constrained sampling. In *Advances in Neural Information Processing Systems.* 4188–4196.
[57] Kejun Li, Maegan Tucker, Erdem Biyik, Ellen Novoseller, Joel W. Burdick, Yanan Sui, Dorsa Sadigh, Yisong Yue, and Aaron D. Ames. 2020. ROIAL: Region of Interest Active Learning for Characterizing Exoskeleton Gait Preference Landscapes. *arXiv preprint arXiv:2011.04812* (2020).
[58] Yi Liu, Gaurav Datta, Ellen Novoseller, and Daniel S Brown. 2023. Efficient preference-based reinforcement learning using learned dynamics models. *arXiv preprint arXiv:2301.04741* (2023).
[59] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.
[60] Dylan P. Losey, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, and Dorsa Sadigh. 2020. Controlling Assistive Robots with Learned Latent Actions. In *International Conference on Robotics and Automation (ICRA).* https://doi.org/10.1109/ICRA40945.2020.9197197
[61] Zelda E Mariet, Suvrit Sra, and Stefanie Jegelka. 2018. Exponentiated Strongly Rayleigh Distributions. In *Advances in Neural Information Processing Systems.* 4464–4474.
[62] Vivek Myers, Erdem Biyik, Nima Anari, and Dorsa Sadigh. 2022. Learning multimodal rewards from rankings. In *Conference on Robot Learning.* PMLR, 342–352.
[63] Vivek Myers, Erdem Biyik, and Dorsa Sadigh. 2023. Active Reward Learning from Online Preferences. In *International Conference on Robotics and Automation (ICRA).* https://doi.org/10.1109/ICRA48891.2023.10160439
[64] Aleksandar Nikolov. 2015. Randomized rounding for the largest simplex problem. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing.* ACM, 861–870.
[65] Malayandi Palan, Nicholas C. Landolfi, Gleb Shevchuk, and Dorsa Sadigh. 2019. Learning Reward Functions by Integrating Human Demonstrations and Preferences. In *Proceedings of Robotics: Science and Systems (RSS).* https://doi.org/10.15607/rss.2019.xv.023
[66] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290* (2023).
[67] Dorsa Sadigh. 2017. *Safe and Interactive Autonomy: Control, Learning, and Verification.* Ph.D. Dissertation. EECS Department, University of California, Berkeley.
[68] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. 2017. Active Preference-Based Learning of Reward Functions. In *Proceedings of Robotics: Science and Systems (RSS).*
[69] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. 2016. Planning for Autonomous Cars that Leverage Effects on Human Actions.. In *Robotics: Science and Systems.*
[70] Ozan Sener and Silvio Savarese. 2017. A geometric approach to active learning for convolutional neural networks. *arXiv preprint arXiv:1708.00489* (2017).
[71] Ankit Shah, Samir Wadhwania, and Julie Shah. 2020. Interactive robot training for non-markov tasks. *arXiv preprint arXiv:2003.02232* (2020).
[72] Hiroaki Sugiyama, Toyomi Meguro, and Yasuhiro Minami. 2012. Preference-learning based inverse reinforcement learning for dialog control. In *Thirteenth Annual Conference of the International Speech Communication Association.*
[73] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.* IEEE, 5026–5033.
[74] Maegan Tucker, Ellen Novoseller, Claudia Kann, Yanan Sui, Yisong Yue, Joel W Burdick, and Aaron D Ames. 2020. Preference-based learning for exoskeleton gait optimization. In *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2351–2357.
[75] Balakrishnan Varadarajan, Dong Yu, Li Deng, and Alex Acero. 2009. Maximizing global entropy reduction for active learning in speech recognition. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on.* IEEE, 4721–4724.
[76] T Velmurugan and T Santhanam. 2010. Computational complexity between K-means and K-medoids clustering algorithms for normal and uniform distributions of data points. *Journal of computer science* 6, 3 (2010), 363.
[77] Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. 2024. RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback. *arXiv preprint arXiv:2402.03681* (2024).

[78] Kai Wei, Rishabh Iyer, and Jeff Bilmes. 2015. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*. 1954–1963.

[79] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* 1, 6 (1945), 80–83.

[80] Nils Wilde and Javier Alonso-Mora. 2022. Do we use the Right Measure? Challenges in Evaluating Reward Learning Algorithms. In *6th Annual Conference on Robot Learning*. https://openreview.net/forum?id=1vV0JRA2HY0

[81] Nils Wilde and Erdem Biyik. 2021. Learning Reward Functions from Scale Feedback. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*.

[82] Nils Wilde, Dana Kulic, and Stephen L Smith. 2020. Active Preference Learning using Maximum Regret. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[83] Aaron Wilson, Alan Fern, and Prasad Tadepalli. 2012. A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems*. 1133–1141.

[84] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. 2016. Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots*.

[85] Yazhou Yang and Marco Loog. 2019. Single shot active learning using pseudo annotators. *Pattern Recognition* 89 (2019), 22–31.

[86] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision* 113, 2 (2015), 113–127.

[87] Cheng Zhang, Hedvig Kjellström, and Stephan Mandt. 2017. Determinantal point processes for mini-batch diversification. In *33rd Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press Corvallis.

[88] Cheng Zhang, Cengiz Öztireli, Stephan Mandt, and Giampiero Salvi. 2019. Active mini-batch sampling using repulsive point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5741–5748.

[89] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning.. In *Aaai*, Vol. 8. Chicago, IL, USA, 1433–1438.

A MAXIMUM COORDINATE ROUNDING

We first present an important point about DPPs that is needed for our maximum coordinate rounding algorithm to approximate the mode of a DPP distribution: conditioning a DPP distribution still results in a DPP. That is, $P(X = A \cup B \mid B \subseteq X)$ is distributed according to a DPP with a transformed kernel:

$$L' = \left(\left[(L + I_{\bar{B}})^{-1} \right]_{\bar{B}} \right)^{-1} - I$$

where $\bar{B} = \mathcal{X} \setminus B$, I is the identity matrix, and $I_{\bar{B}}$ is the projection matrix with all zeros except at the diagonal entries (i, i) for $\forall i \in \bar{B}$ where the entry is 1.

The greedy approach for approximating the mode of a DPP distribution does not provide the state-of-the-art approximation guarantee. [64] showed that one can find an e^k-approximation to the mode by using a convex relaxation. We present the algorithm of [64] stated in an equivalent form: Formally, consider the generating polynomial associated to the DPP:

$$g(v_1, \ldots, v_K) = \sum_{A:|A|=k} \det(L_A) \prod_{i \in A} v_i \ .$$

Finding the mode is equivalent to maximizing $g(v_1, \ldots, v_K)$ over *nonnegative integers* $v_1, \ldots, v_K$ satisfying the constraint $v_1 + \cdots + v_K = k$. We get a relaxation by replacing integers with nonnegative reals, and using the insight that $\log(g)$ is a concave function which can be maximized efficiently:

$$\max \{ \log g(v_1, \ldots, v_K) \mid v_1 + \cdots + v_K = k \} \ .$$

If $v_1^*, \ldots, v_K^*$ is the maximizer, one can then choose a set A of size k with $P(A) \propto \prod_{i \in A} v_i^*$. Then $\mathbb{E}[\det L_A]$ will be an e^k-approximation to the mode. Although this approximation holds in expectation, the probability that the sampled A is an e^k-approximation can be exponentially small. To resolve this, [64] resorted to the method of conditional expectations, each time deciding whether to include an element in the set A or not.

The main drawback of this method is its computational cost. In particular, the running time of the methods that compute g scale as a super-linear polynomial in K, which is problematic for the typical use cases where K is large. Computing g and ∇g is needed for solving the relaxation as well as running the method of conditional expectations.

We instead propose a new algorithm that avoids the method of conditional expectations. We also propose a heuristic method to find the maximizers $v_1^*, \ldots, v_K^*$ by stochastic mirror descent, where each stochastic gradient computation requires sampling from a DPP (see Appendix B). Approximate sampling from DPPs can be done in time $O(K \cdot k^2 \log k)$, scaling linearly with K [43]. Our algorithm is:

(1) Find the nonnegative real maximizers $v_1^*, \ldots, v_K^*$ of $\log g(v_1, \ldots, v_K)$ subject to $v_1 + \cdots + v_K = k$.
(2) Let v_i^* be the maximum among $v_1^*, \ldots, v_K^*$. Put i in A, and recursively find $k - 1$ extra elements to put in A, working with the conditioned DPP.

Theorem 1. *The above algorithm finds an e^k-approximation of the mode.*

Proof. We prove this by induction on k. We simply prove that each time we select an element and put it in A, we only lose a factor of at most e. Note that the first-order optimality condition of $v_1^*, \ldots, v_K^*$ means that

$$\nabla \log g(v_1^*, \ldots, v_K^*) = c\mathbf{1} - \sum_{j: v_j^* = 0} c_j e_j \ ,$$

for some c and collection of $c_j \geq 0$. Here $\mathbf{1}$ is the all-ones vector and $e_1, \ldots, e_K$ are the standard basis vectors. By complementary slackness, we have $c_j v_j^* = 0$ for all j. Since $v_i^* > 0$, it must be that $c_i = 0$, and it follows that $c = \|\nabla \log g(v_1^*, \ldots, v_K^*)\|_\infty = \partial_i \log g(v_1^*, \ldots, v_K^*)$. Note that g is a k-homogeneous polynomial and it follows that $\langle \nabla g(v), v \rangle = k g(v)$. Applying the inequality $\langle \nabla g, v \rangle \leq \|\nabla g\|_\infty \cdot \|v\|_1$, we get

$$k g(v_1^*, \ldots, v_K^*) \leq \|\nabla g(v_1^*, \ldots, v_K^*)\|_\infty \cdot \|v^*\|_1 ,$$

Noting that $\|v^*\|_1 = k$ and $\|\nabla g(v_1^*, \ldots, v_K^*)\|_\infty = \partial_i g(v_1^*, \ldots, v_K^*)$, we get

$$\partial_i g(v_1^*, \ldots, v_K^*) \geq g(v_1^*, \ldots, v_K^*) .$$

But note that $\partial_i g$ is exactly the generating polynomial for the conditioned DPP (where we condition on $i \in A$). So it is enough to show that $\max \partial_i g(u_1, \ldots, u_K)$ over $u_1 + \cdots + u_K = k - 1$ is at least $1/e$ times the above amount. To do this we simply let $u_j^* = (k-1)v_j^*/(k - v_i^*)$ for $j \neq i$ and we set $u_i^* = 0$. Since $\partial_i g$ is $(k-1)$-homogeneous we get

$$\partial_i g(u_1^*, \ldots, u_K^*) = \left(\frac{k-1}{k - v_j}\right)^{k-1} \partial_i g(v_1^*, \ldots, v_K^*) \geq \left(\frac{k-1}{k}\right)^{k-1} g(v_1^*, \ldots, v_K^*) .$$

We conclude by noting that $((k-1)/k)^{k-1} \geq 1/e$. □

B STOCHASTIC MIRROR DESCENT ALGORITHM

In this section we propose a stochastic mirror descent algorithm to optimize the following convex program over nonnegative reals

$$\max \{\log g(v_1, \ldots, v_N) \mid v_1 + \cdots + v_N = k\} ,$$

where g is the generating polynomial associated to a k-DPP, i.e.,

$$g(v_1, \ldots, v_K) = \sum_{A : |A| = k} \det(L_A) \prod_{i \in A} v_i .$$

Our proposed algorithm is repetitions of the following iteration:

(1) Sample a set A with $P(A) \propto \prod_{i \in A} v_i \det(L_A)$.
(2) Let $u \leftarrow v + \eta \mathbf{1}_A$, where $\mathbf{1}_A$ is the indicator of A.
(3) Let $v \leftarrow ku/(\sum_i u_i)$.

Note that the sampling in step 1 can be done by Markov chain Monte Carlo (MCMC) methods, since we are sampling A according to a DPP. Careful implementations of the latest MCMC methods (e.g. [43]) run in time $O(K \cdot k^2 \log k)$ time. The parameter η is the step size and can be adjusted.

Now we provide the intuition behind this iterative procedure. First, let us compute $\nabla \log g$. We have

$$\frac{\partial_i g(v_1, \ldots, v_K)}{g(v_1, \ldots, v_K)} = \frac{1}{v_i} \frac{\sum_{A : i \in A} \det(L_A) \prod_{j \in A} v_j}{\sum_A \det(L_A) \prod_{j \in A} v_j},$$

but this is equal to $P(i \in A)/v_i$. Therefore $\nabla \log g = \mathrm{diag}(v)^{-1} p$, where p is the vector of marginal probabilities, i.e. $p_i = P(i \in A)$. Note however that $\mathbb{E}[\mathbf{1}_A] = p$. So this suggests that we can use $\mathrm{diag}(v)^{-1}\mathbf{1}_A$ as a stochastic gradient.

Numerically we found $\mathrm{diag}(v)^{-1}\mathbf{1}_A$ to be unstable. This is not surprising as v can have small entries, resulting in a blow up of this vector. Instead we use a stochastic mirror descent algorithm, where we choose a convex function h and modify our stochastic gradient vector by multiplying $(\nabla^2 h)^{-1}$ on the left.

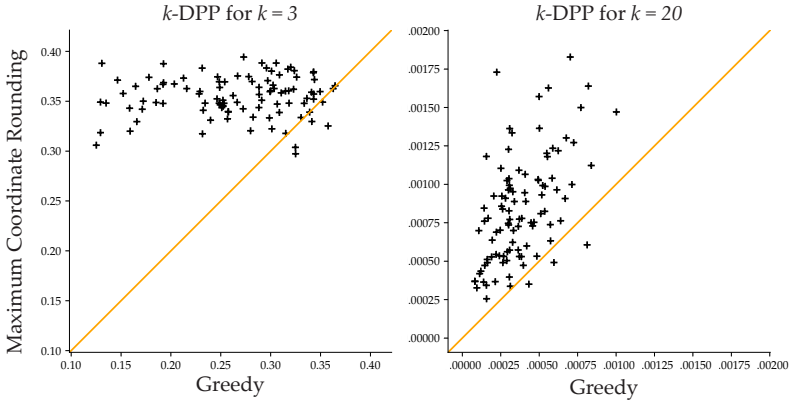


Fig. 12. Comparison of the greedy algorithm and the maximum coordinate rounding algorithm. In 93% of the $k=3$ cases, and in 97% of the $k=20$ cases, our method returns a better or equal solution.

We found the choice of $h(v_1, \ldots, v_K) = \sum_i v_i \log v_i$ to be reasonable. Accordingly, we have $\nabla^2 h = \operatorname{diag}(v)^{-1}$, and therefore

$$(\nabla^2 h)^{-1} \operatorname{diag}(v)^{-1} \mathbf{1}_A = \mathbf{1}_A.$$

Finally, note that step 3 of our algorithm is simply a projection back to the feasible set of our constraints (according to the Bregman divergence imposed by h).

C CHOICE OF STOCHASTIC GRADIENT VECTOR

Note that the vector $\mathbf{1}_A$ in step 2 of the algorithm can be replaced by any other random vector y, as long as the expectation is preserved. One can extract such vectors y from implementations of MCMC methods [6, 43]. The MCMC methods that aim to sample a set A with probability proportional to $\prod_{i \in A} v_i \det(L_A)$ work as follows: starting with a set A, one drops an element $i \in A$ chosen uniformly at random, and adds an element j back with probability proportional to $\det L_{A-i+j}$, in order to complete one step of the Markov chain. We can implement the same Markov chain, and let y_j be k times the probability of transitioning from $A - i$ to $A - i + j$ in this chain. It is easy to see that if the chain has mixed and A is sampled from the stationary distribution

$$\mathbb{E}[y] = \mathbb{E}[\mathbf{1}_A] \,.$$

We found this choice of y to have less variance than $\mathbf{1}_A$ in practice.

D EMPIRICAL COMPARISON OF MAXIMUM COORDINATE ROUNDING WITH GREEDY APPROACH

Here we provide an empirical comparison between the performance of the greedy approach to approximating the mode of a DPP distribution versus our maximum coordinate rounding algorithm.

We used two sets of experiments where $q_i = 1$ for all i in the query set. In the first, we generated 200 random queries inside $[0, 1]^2$ (meaning their feature difference vectors, ψ's, are inside $[0, 1]^2$), set $\sigma = 1$ and attempted to find the mode of the k-DPP for $k = 3$. In the second, we generated 200 random queries inside $[0, 1]^2$, set $\sigma = 0.2$ and attempted to find the mode of the k-DPP for $k = 20$. We ran each experiment 100 times (each time generating a new set of random queries).

The results can be seen in Fig. 12. We plotted $\det(L_A)$ vs. $\det(L_B)$, where A is the set returned by the greedy approach, and B is the set returned by our maximum coordinate rounding algorithm.