

Batch Active Preference-Based Learning of Reward Functions

Erdem Biyik

Electrical Engineering
Stanford University
ebiyik@stanford.edu

Dorsa Sadigh

Computer Science & Electrical Engineering
Stanford University
dorsa@cs.stanford.edu

Abstract: Data generation and labeling are usually an expensive part of learning for robotics. While active learning methods are commonly used to tackle the former problem, preference-based learning is a concept that attempts to solve the latter by querying users with preference questions. In this paper, we will develop a new algorithm, *batch active preference-based learning*, that enables efficient learning of reward functions using as few data samples as possible while still having short query generation times. We introduce several approximations to the batch active learning problem, and provide theoretical guarantees for the convergence of our algorithms. 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 then showcase our algorithm in a study to learn human users’ preferences.

Keywords: batch active, pool based active, active learning, preference learning

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 amounts of labeled data. However, collecting and labeling data can be costly and time-consuming in many fields such as speech recognition [1], dialog control [2], text classification [3], image recognition [4], influence maximization in social networks [5], as well as in robotics [6, 7, 8]. In addition to lack of labeled data, *robot learning* has a few other challenges that makes it particularly difficult. First, humans cannot (and do not) reliably assign a *success value* (reward) to a given robot action. Furthermore, we cannot simply fall back on collecting demonstrations from humans to learn the desired behavior of a robot since human experts usually provide suboptimal demonstrations or have difficulty operating a robot with more than a few degrees of freedom [9, 10]. Instead, we use preference-based learning methods that enable us to learn a regression model by using the preferences of users [11] as opposed to expert demonstrations.

To address the lack of data in robotics applications, we leverage *active preference-based learning* techniques, where we learn from the most informative data to recover humans’ preferences of how a robot should act. However, this can be challenging due to the time-inefficiency of most of the active-preference based learning methods. The states and actions in every trajectory that is shown to the human naturally are drawn from a continuous space. Previous work has focused on actively synthesizing comparison queries directly from the continuous space [6], but these active methods can be quite inefficient. Similarly, using the variance of reward estimates to select queries has been explored, but the use of deep reinforcement learning can increase the number of queries required [12]. Ideally, we would like to develop an algorithm that requires only a few number of queries while generating each query efficiently.

Our insight is that there is a direct tradeoff between *the required number of queries* and the *time it takes to generate each query*.

Leveraging this insight, we propose a new algorithm—*batch active preference-based learning*—that balances between the number of queries it requires to learn humans’ preferences and the time it spends on generation of each comparison query. We will actively generate each batch based on the labeled data collected so far. Therefore, in our framework, we synthesize and query b pairs of samples, to be compared by the user, at once. In addition, if we are not interested in personalized data collection, the batch query process can be parallelized leading to more efficient results. Our work differs from the existing batch active learning studies as it involves actively learning a reward function for dynamical systems. Moreover, as we have a continuous set for control inputs and do not have a prior likelihood information of those inputs, we cannot use the *representativeness* measure [13, 14, 15], which can

significantly simplify the problem by reducing it to a submodular optimization. We summarize our contributions as:

1. Designing a set of approximation algorithms for efficient batch active learning to learn about humans’ preferences from comparison queries.
2. Formalizing the tradeoff between query generation time and the number of queries, and providing convergence guarantees.
3. Experimenting and comparing approximation methods for batch active learning in complex preference based learning tasks.
4. Showcasing our algorithm in predicting human users’ preferences in autonomous driving and tossing a ball towards a target.

2 Problem Statement

Modeling Choices. We start by modeling human preferences about how a robot should act in interaction with other agents. We model these preferences over the actions of an agent in a fully observable dynamical system \mathcal{D} . Let $f_{\mathcal{D}}$ denote the dynamics of the system that includes one or multiple robots. Then, $x^{t+1} = f_{\mathcal{D}}(x^t, u_H^t, u_R^t)$, where u_H^t denotes the actions taken by the human, and u_R^t corresponds to the actions of other robots present in the environment. The state of the system x^t evolves through the dynamics and the actions.

A finite trajectory $\xi \in \Xi$ is a sequence of continuous state and action pairs $(x^0, u_H^0, u_R^0) \cdots (x^T, u_H^T, u_R^T)$ over a finite horizon time $t = 0, 1, \dots, T$. 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 $R_H : \Xi \mapsto \mathbb{R}$ 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 set of features over trajectories $\phi(\xi)$, where $R_H(\xi) = \mathbf{w}^\top \phi(\xi)$. The goal of *preference-based learning* is to learn $R_H(\xi)$, or equivalently \mathbf{w} through preference queries from a human expert. For any ξ_A and ξ_B , $R_H(\xi_A) > R_H(\xi_B)$ if and only if the expert prefers ξ_A over ξ_B . From this preference encoded as a strict inequality, we can equivalently conclude $\mathbf{w}^\top (\phi(\xi_A) - \phi(\xi_B)) > 0$. We use ψ to refer to this difference: $\psi(\xi_A, \xi_B) = \phi(\xi_A) - \phi(\xi_B)$. Therefore, the sign of $\mathbf{w}^\top \psi$ is sufficient to reveal the preference of the expert for every ξ_A and ξ_B . We thus let $I = \text{sign}(\mathbf{w}^\top \psi)$ denote human’s input to a query: “Do you prefer ξ_A over ξ_B ?”. Figure 1 summarizes the flow that leads to the preference I .

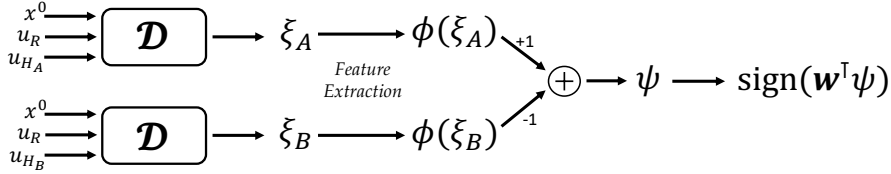


Figure 1: The schematic of the preferences based-learning problem starting from two sample inputs (x^0, u_{H_A}, u_R) and (x^0, u_{H_B}, u_R)

In addition, the input from the human can be noisy due to the uncertainty of her preferences [6, 12, 16]. A common noise model assumes human’s preferences are probabilistic and can be modeled using a softmax function:

$$P(I_i | \mathbf{w}) = \frac{1}{1 + \exp(-I_i \mathbf{w}^\top \psi)} \quad (1)$$

where $I_i = \text{sign}(\mathbf{w}^\top \psi_i)$ represents human’s preference on the i^{th} query with trajectories ξ_A and ξ_B .

Approach Overview. In many robotics tasks, we are interested in learning a model of the humans’ preferences about the robots’ trajectories. This model can be learned through inverse reinforcement learning (IRL), where a reward function R_H is learned directly from the human demonstrating how to operate a robot [17, 18, 19, 20]. However, learning a reward function from humans’ preferences as opposed to demonstrations can be more favorable for a few reasons. First, providing demonstrations for robots with higher degrees of freedom can be quite challenging even for human experts [9]. Furthermore, humans’ preferences tend to defer from their demonstrations [21].

We plan to leverage *active* preference-based techniques to synthesize pairwise queries over the continuous space of trajectories for the goal of efficiently learning humans’ preferences [6, 22, 23, 2, 24]. However, there is a tradeoff between the time spent to generate a query 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.

Instead, we propose a time-efficient method, *batch active learning*, that balances between minimizing the number of queries and being time-efficient in its interaction with the human expert. Batch active learning has two main benefits: i) Creating a batch of queries can create a more *time-efficient* interaction with the human. ii) The procedure can be *parallelized* when we look for the preferences of a population of humans.

3 Time-Efficient Active Learning for Synthesizing Queries

Actively Synthesizing Pairwise Queries. In active preference-based learning, the goal is to synthesize the next pairwise query to ask a human expert to maximize the information received. While optimal querying is NP-hard [25], there exist techniques that pose the problem as a submodular optimization, where suboptimal solutions that work well in practice exist. We follow the work in [6], where we model active preference-based learning as a *maximum volume removal* problem.

The goal is to search for the human’s preference reward function $R_H = \mathbf{w}^\top \phi(\xi)$ by actively querying the human. We let $p(\mathbf{w})$ be the distribution of the unknown weight vector \mathbf{w} . Since \mathbf{w} and $c\mathbf{w}$ yield to the same preferences for a positive constant c , we constrain the prior such that $\|\mathbf{w}\|_2 \leq 1$. Every query provides a human input I_i , which then enables us to perform a Bayesian update of this distribution as $p(\mathbf{w}|I_i) \propto p(I_i|\mathbf{w})p(\mathbf{w})$. Since we do not know the shape of $p(\mathbf{w})$ and cannot differentiate through it, we sample M values from $p(\mathbf{w})$ using an adaptive Metropolis algorithm [26]. In order to speed up this sampling process, we approximate $p(I_i|\mathbf{w})$ as $\min(1, \exp(I_i \mathbf{w}^\top \psi))$. Generating the next most informative query can be formulated as maximizing the minimum volume removed from the distribution of \mathbf{w} at every step. We note that every query, i.e., a pair of trajectories (ξ_A, ξ_B) is parameterized by the initial state x^0 , a set of actions for all the other agents u_R , and the two sequence of actions u_{H_A} and u_{H_B} corresponding to ξ_A and ξ_B respectively. The query selection problem in the i^{th} iteration can then be formulated as:

$$\max_{x^0, u_R, u_{H_A}, u_{H_B}} \min\{\mathbb{E}[1 - p(I_i|\mathbf{w})], \mathbb{E}[1 - p(-I_i|\mathbf{w})]\} \quad (2)$$

with an appropriate feasibility constraint. Here the inner optimization (minimum between two volumes for the two choices of user input) provides robustness against the user’s preference on the query, the outer optimization ensures the maximum volume removal, where *volume* refers to the unnormalized distribution $p(\mathbf{w})$. This sample selection approach is based on the expected value of information of the query [27] and the optimization can be solved using a Quasi-Newton method [28].

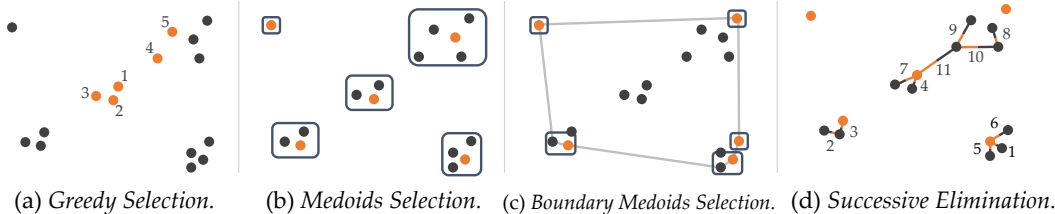
Batch Active Learning. Actively generating a new query requires solving the optimization in equation (2) and running the adaptive Metropolis algorithm for sampling. Performing these operations for every single query synthesis can be quite slow and not very practical while interacting with a human expert in real-time. The human has to wait for the solution of optimization before being able to respond to the next query. 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 b queries are simultaneously synthesized at a time. The batch approach can significantly reduce the total time required for the satisfactory estimation of \mathbf{w} at the expense of increasing the number of queries.

Since small perturbations of the inputs could lead to only minor changes in the objective of equation (2), continuous optimization of this objective can result in generating same or sufficiently similar queries within a batch. We thus fall back to a discretization method. We discretize the space of trajectories by sampling K pairs of trajectories from the input space of $\xi = (x^0, u_H, u_R)$. While increasing K yields more accurate optimization results, computation time increases linearly with K .

A similar viewpoint to optimization in (2) is to use the notion of information entropy. As in uncertainty sampling, a similar interpretation of equation (2) is to find a set of feasible queries that maximize the conditional entropy $\mathcal{H}(I_i|\mathbf{w})$. Following this conditional entropy framework, we formalize the batch active learning problem as the solution of the following optimization:

$$\max_{\xi_{ib+1_A}, \xi_{ib+1_B}, \dots, \xi_{(i+1)b_A}, \xi_{(i+1)b_B}} \mathcal{H}(I_{ib+1}, I_{ib+2}, \dots, I_{(i+1)b}|\mathbf{w}) \quad (3)$$

for the $(i + 1)^{\text{th}}$ batch with the appropriate feasibility constraint. This problem is known to be computationally hard [3, 5]—it requires an exhaustive search which is intractable in practice, since the search space is exponentially large [29].



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

Figure 2: Visualizations of the selection process of batch active learning. A simple 2D space with 16 different ψ values that correspond to inputs individually maximizing the conditional entropy. The goal is to select a batch of $b=5$ that will near-optimally maximize the joint conditional entropy. The selected samples are shown in orange. (a) Greedy Selection, (b) Medoids Selection, (c) Boundary Medoids Selection, (d) Successive Elimination.

3.1 Algorithms for Time-Efficient Batch Active Learning

We now describe a set of methods in increasing order of complexity to provide an approximation to the batch active learning problem. Figure 2 visualizes each approach for a small set of samples.

Greedy Selection. The simplest method to approximate the batch learning problem in equation (3) is using a greedy strategy. In the greedy selection approach, we conveniently assume the b different human inputs are independent of each other. Of course this is not a valid assumption, but the independence assumption creates the following approximation, where we need to choose the b -many maximizers of equation (2) among the K samples:

$$\max_{\xi_{ib+1_A}, \xi_{ib+1_B}} \mathcal{H}(I_{ib+1}|\mathbf{w}) + \dots + \max_{\xi_{(i+1)b_A}, \xi_{(i+1)b_B}} \mathcal{H}(I_{(i+1)b}|\mathbf{w}) \quad (4)$$

with an additional set of constraints that specify the trajectory sets (ξ_A, ξ_B) are different among queries. While this method can easily be employed, it is suboptimal as redundant samples can be selected together in the same batch, since these similar queries are likely to lead to high entropy values. For instance, as shown in Fig. 2 (a) the 5 orange samples chosen are all going to be close to the center where there is high conditional entropy.

Medoid Selection. To avoid the redundancy in the samples created by the greedy selection, we need to increase the dissimilarity between the selected batch samples. Our insight is to define a new approach, *Medoid Selection*, that leverages clustering as a similarity measure between the samples. In this method, we let \mathcal{G}_B be the set of ψ -vectors that correspond to B samples selected using the greedy selection strategy, where $B > b$. With the goal of picking the most dissimilar samples, we cluster \mathcal{G}_B into b 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 [30] for clustering and then selecting the centroid of each cluster. However, these centroids are not necessarily from the set of greedily selected samples, so they can have lower expected information.

Instead, we use the K-medoids algorithm [31, 32] which again clusters the samples into b 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 points *in the set* \mathcal{G}_B that minimize the average distance to the other points in the same cluster. Fig. 2 (b) shows the medoid selection approach, where 5 orange points are selected from 5 clusters.

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 samples more effectively. In the *Boundary Medoid Selection* method, we propose restricting the selection to be only from the boundary of the convex hull of \mathcal{G}_B . This selection criteria can separate out the sample points from each other on average. We note that when $\dim(\psi)$ is large enough, most of the clusters will have points on the boundary. We thus propose the following modifications to the medoid selection algorithm. The first step is to only select the points that are on the boundary of the convex hull of \mathcal{G}_B . We then apply K-medoids with b clusters over the points on the boundary and finally only accept the cluster medoids as the selected samples. As shown in Fig. 2 (c), we first find $b=5$ clusters over the points on the boundary of the convex hull of \mathcal{G}_B . We then select the medoid of those 5 clusters created over the boundary points.

Successive Elimination. The main goal of batch active learning as described in the previous methods is to select b points that will maximize the average distance among them out of the B samples in \mathcal{G}_B . This problem is called *max-sum diversification* in literature, known to be NP-hard [33, 34].

What makes our batch active learning problem special and different from standard max-sum diversification is that we can compute the conditional entropy $\mathcal{H}(I_i|\mathbf{w})$ for each potential pair of trajectories,

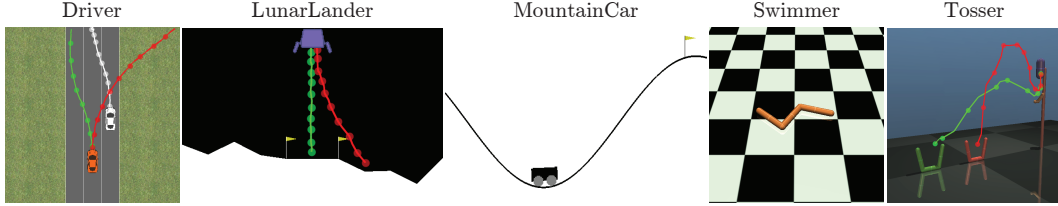


Figure 3: Views from each task. (a) Driver, (b) Lunar Lander, (c) Mountain Car, (d) Swimmer, and (e) Tosser. which corresponds to ψ_i . The conditional entropy is a metric that models how much a query is preferred to be in the final batch. We propose a novel method that leverages the conditional entropy to successively eliminate samples 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 points in \mathcal{G}_B , and remove the one with lower information entropy. We repeat this procedure until b points are left in the set resulting in the b samples in our final batch, which efficiently increases the diversity among queries. Fig. 2 (d) shows the successive pairwise comparisons between two samples based on their corresponding conditional entropy. In every pairwise comparison, we eliminate the sample shown with gray edge, keeping the point with the orange edge. The numbers show the order of comparisons done before finding $b=5$ optimally different sample points shown in orange.

3.2 Convergence Guarantees

Theorem 3.1. *Under the following assumptions:*

1. *The error introduced by the sampling of input space is ignored,*
2. *The function that updates the distribution of w , and the noise that human inputs have are $p(I_i|w)$ as given in Eq. (1); and the error introduced by approximation of noise model is ignored,*
3. *The errors introduced by the sampling of w 's and non-convex optimization is ignored,*

greedy selection and successive elimination algorithms remove at least $1-\epsilon$ times as much volume as removed by the best adaptive strategy after $b \ln(\frac{1}{\epsilon})$ times as many queries.

Proof. In greedy selection and successive elimination methods, the conditional entropy maximizer query (ξ_A^*, ξ_B^*) out of K possible queries will always remain in the resulting batch of size b , because the queries will be removed only if they have lower entropy than some other queries in the set. By assumption 1, we have (ξ_A^*, ξ_B^*) as the maximizer over the continuous control inputs set. In [6], it has been proven by using the ideas from submodular function maximization literature [35] that if we make the single query (ξ_A^*, ξ_B^*) at each iteration, at least $1-\epsilon$ times as much volume as removed by the best adaptive strategy will be removed after $\ln(\frac{1}{\epsilon})$ times as many iterations. The proof is complete with the pessimistic approach accepting other $b-1$ queries will remove no volume. \square

4 Simulations and Experiments

Experimental Setup. We have performed several simulations and experiments to compare the methods we propose and to demonstrate their performance. The code is available online¹. In our experiments, we set $b=10$, $B=20b$ and $M=1000$. We sample the input space with $K=5 \times 10^5$ and compute the corresponding ψ vectors once, and use this sampled set for every experiment and iteration. To acquire more realistic trajectories, we fix u_R when other agents exist in the experiment.

Alignment Metric. For our simulations, we generate a synthetic random w_{true} vector as our true preference vector. We have used the following alignment metric [6] in order to compare non-batch active, batch active and random query learning methods, where all queries are selected uniformly random over all feasible trajectories.

$$m = \frac{w_{\text{true}}^\top \hat{w}}{\|w_{\text{true}}\|_2 \|\hat{w}\|_2} \quad (5)$$

where \hat{w} is $\mathbb{E}[w]$ based on the estimate of the learned distribution of 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 w is very close to (aligned with) the true weight vector.

4.1 Tasks

We perform experiments in different simulation environments. Fig. 3 visualizes each of the experiments with some sample trajectories. We now briefly describe the environments.

¹See <http://github.com/Stanford-ILIAD/batch-active-preference-based-learning>

Linear Dynamical System (LDS). We assess the performance of our methods on an LDS:

$$x^{t+1} = Ax^t + Bu^t, \quad y^t = Cx^t + Du^t \quad (6)$$

For a fair comparison between the proposed methods independent of the dynamical system, we want $\phi(\xi)$ to uniformly cover its range when the control inputs are uniformly distributed over their possible values. We thus set A, B and C to be zeros matrices and D to be the identity matrix. Then we treat y^0 as $\phi(\xi)$. Therefore, the control inputs are equal to the features over trajectories, and optimizing over control inputs or features is equivalent. We repeat this simulation 10 times, and use non-parametric Wilcoxon signed-rank tests over these 10 simulations and 9 different N to assess significant differences [36].

Driving Simulator. We use the 2D driving simulator [37], shown in Fig. 3 (a). We use features corresponding to distance to the closest lane, speed, heading angle, and distance to the other vehicles. Two sample trajectories are shown in red and green in Fig. 3 (a). In addition, the white trajectory shows the state and actions (u_R) of the other vehicle.

Lunar Lander. We use OpenAI Gym’s continuous Lunar Lander [38]. We also use features corresponding to final heading angle, final distance to landing pad, total rotation, path length, final vertical speed, and flight duration. Sample trajectories are shown in Fig. 3 (b).

Mountain Car. We use OpenAI Gym’s continuous Mountain Car [38]. The features are maximum range in the positive direction, maximum range in the negative direction, time to reach the flag.

Swimmer. We use OpenAI Gym’s Swimmer [38]. Similarly we use features corresponding to horizontal displacement, vertical displacement, total distance traveled.

Tosser. We use MuJoCo’s Tosser [39]. The features we use are maximum horizontal range, maximum altitude, the sum of angular displacements at each timestep, final distance to closest basket. The two red and green trajectories in Fig. 3 (e) correspond to synthesized queries showing different preferences for what basket to toss the ball to.

4.2 Experiment Results

For the LDS simulations, we assume human’s preference is noisy as discussed in Eq. (1). For other tasks, we assume an oracle user who knows the true weights w_{true} and responds to queries with no error.

Figure 4 shows the number of queries that result in a corresponding alignment value m for each method in the LDS environment. While non-batch active version as described in [6] outperforms all other methods as it performs the optimization for each and every query, successive elimination method seems to improve over the remaining methods on average. The performance of batch-mode active methods are ordered from worst to best as *greedy*, *medoids*, *boundary medoids*, and *successive elimination*. While the last three algorithms are significantly better than greedy method ($p < 0.05$), and successive elimination is significantly better than medoid selection ($p < 0.05$); the significance tests for other comparisons are somewhat significant ($p < 0.13$). This suggests successive elimination increases diversity without sacrificing informative queries.

We show the results of our experiments in all 5 environments in Fig. 5 and Table 1. Fig. 5 (a) shows the convergence to the true weights w_{true} as the number of samples N increases (similar to Fig. 4). Interestingly, non-batch active learning performs suboptimally in *LunarLander* and *Tosser*. We believe this can be due to the non-convex optimization involved in non-batch methods leading to suboptimal behavior. The proposed batch active learning methods overcome this issue as they sample from the input space.

Figure 5 (b) and Table 1 evaluate the computation time required for querying. It is clearly visible from Fig. 5 (b) that batch active learning makes the process much faster than the non-batch active version and random querying. Hence, batch active learning is preferred over other methods as it

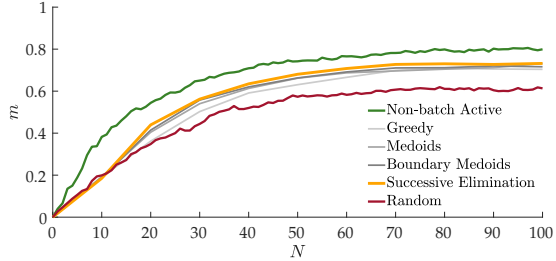


Figure 4: The performance of each algorithm was averaged over 10 different runs with LDS. The proposed batch methods perform better than the random querying baseline and worse than the non-batch active methods.

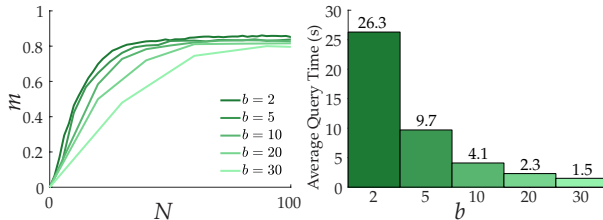


Figure 6: The performance of successive elimination algorithm with varying b values was averaged over 10 different runs with LDS.

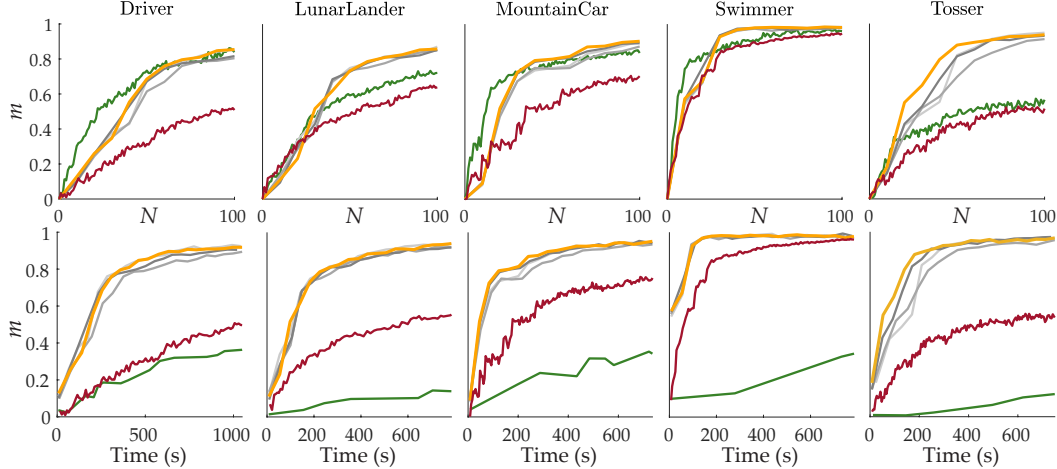


Figure 5: The performance of each algorithm is shown for all 5 tasks. This figure uses the same legend as Fig. 4. **Top row:** While it is difficult to compare batch active algorithms in the environments other than *MountainCar* and *Tossler*, where successive elimination is superior, we also note non-batch active method performs poorly on *LunarLander* and *Tossler*. **Bottom Row:** 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. Batch active methods clearly outperform both of them.

Table 1: Average Query Times (seconds)

Task Name	Non-Batch	Batch Active Learning			
		<i>Greedy</i>	<i>Medoids</i>	<i>Boundary Med.</i>	<i>Succ. Elimination</i>
<i>Driver</i>	79.2	5.4	5.7	5.3	5.5
<i>LunarLander</i>	177.4	4.1	4.1	4.2	4.1
<i>MountainCar</i>	96.4	3.8	4.0	4.0	3.8
<i>Swimmer</i>	188.9	3.8	3.9	4.0	4.1
<i>Tossler</i>	149.3	4.1	4.3	3.8	3.9

balances the tradeoff between the number of queries required and the time it takes to compute them. This tradeoff can be seen from Fig. 6 where we simulated LDS with varying b values.

4.3 User Preferences

In addition to our simulation results using a synthetic w_{true} , we perform a user study to learn humans’ preferences for the *Driver* and *Tossler* environments. This experiment is mainly designed to show the ability of our framework to learn humans’ preferences.

Setup. We recruited 10 users who responded to 150 queries generated by successive elimination algorithm for *Driver* and *Tossler* environments.

Driver Preferences. Using successive elimination, we are able to learn humans’ driving preferences. We use four features corresponding to the vehicle staying within its lane, having high speed, having a straight heading, and avoiding collisions. Our results show the preferences of users are very close to each other as this task mainly models natural driving behavior. This is consistent with results shown in [6], where non-batch techniques are used. We noticed a few differences between the driving behaviors as shown in Fig. 7. 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 style.

We observed 70 queries were enough for converging to safe and sensible driving in the defined scenario where we fix the speed and let the system optimize steering. The optimized driving with different number of queries can be watched on <https://youtu.be/MaswyWRep5g>.

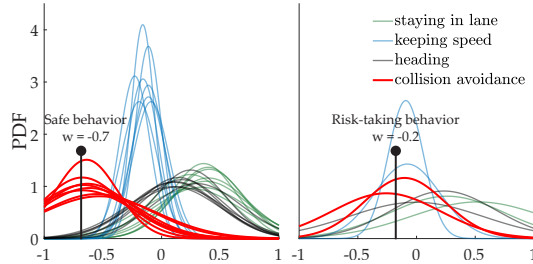


Figure 7: User preferences on *Driver* task are grouped into two sets. The first set shows the preferences conforming with the natural driving behavior. The second set is comprised of data from two users one of whom preferred collisions over leaving the road and the other regarded some collisions as near-misses and thought they can be acceptable to keep speed. It can be seen that the uncertainty in their learned preferences is higher.

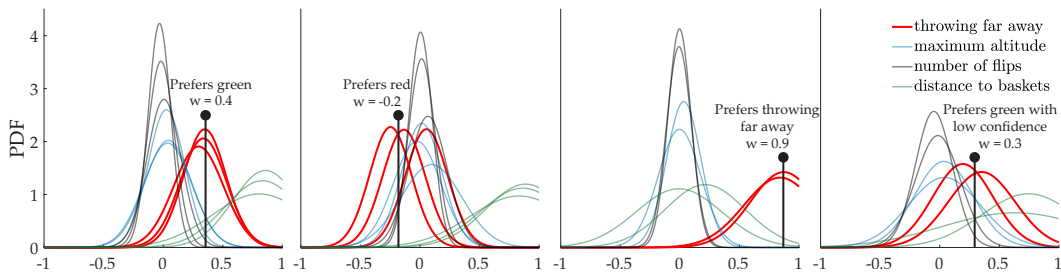


Figure 8: User preferences on *Tosser* task are grouped into four sets. The first set shows the preferences of people who aimed at throwing the ball into the green basket but accepted throwing into the other basket is better than not throwing into any baskets. The second set is comprised of data from three users who preferred the red basket. 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. 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.

Tosser Preferences. Similarly, we use successive elimination to learn humans’ preferences on the tosser task. The four features used correspond to: throwing the ball far away, maximum altitude of the ball, number of flips, and distance to the basket. These features are sufficient to learn interesting tossing preferences as shown in Fig. 8. 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 (see Fig. 3 (e)). We note 100 queries were enough to see reasonable convergence. The evolution of the learning can be watched on <https://youtu.be/cQ7vvUg9rU4>.

5 Discussion

Summary. In this work, we have proposed an end-to-end method 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 provide theoretical guarantees for convergence of our algorithm, and demonstrate its performance in simulation.

Limitations. In our experiments, we sample the control space in advance for batch-mode active learning methods, while we still employ the optimization formulation for the non-batch active version. It can be argued that this creates a bias on the computational times. However, there are two points that make batch techniques more efficient than the non-batch version. First, this sampling process can be easily parallelized. Second, even if we used predefined samples for non-batch method, it would be still inefficient due to adaptive Metropolis algorithm and discrete optimization running for each query, which cannot be parallelized across queries. It can also be inferred from Fig. 6 that non-batch active learning with sampling the control space would take a significantly longer running time compared to batch versions. We note the use of sampling would reduce the performance of non-batch active learning, while it is currently the best we can do for batch version.

Future directions. In this study, we used a fixed batch-size. However, we know the first queries are more informative than the following queries. Therefore, instead of starting with b random queries, 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.

The algorithms we described in this work can be easily implemented when appropriate simulators are available. For the cases where safety-critical dynamical systems are to be used, further research is warranted to ensure that the optimization is not evaluated with unsafe inputs.

We also note the procedural similarity between our successive elimination algorithm and Matérn processes [40], which also points out a potential use for determinantal point processes for diversity within batches [41, 42].

Lastly, we used handcrafted feature transformations in this study. In the future we plan to learn those transformations along with preferences, i.e. to learn the reward function directly from trajectories, by developing batch techniques that use as few queries as possible generated in a short amount of time.

Acknowledgments

The authors would like to acknowledge FLI grant RFP2-000. Toyota Research Institute (“TRI”) provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. Erdem Bıyık is partly supported by the Stanford School of Engineering James D. Plummer Graduate Fellowship.

References

- [1] B. Varadarajan, D. Yu, L. Deng, and A. Acero. Maximizing global entropy reduction for active learning in speech recognition. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4721–4724. IEEE, 2009.
- [2] H. Sugiyama, T. Meguro, and Y. Minami. Preference-learning based inverse reinforcement learning for dialog control. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.
- [3] N. V. Cuong, W. S. Lee, N. Ye, K. M. A. Chai, and H. L. Chieu. Active learning for probabilistic hypotheses using the maximum gibbs error criterion. In *Advances in Neural Information Processing Systems*, pages 1457–1465, 2013.
- [4] O. Sener and S. Savarese. A geometric approach to active learning for convolutional neural networks. *arXiv preprint arXiv:1708.00489*, 2017.
- [5] Y. Chen and A. Krause. Near-optimal batch mode active learning and adaptive submodular optimization. *ICML (1)*, 28:160–168, 2013.
- [6] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2017.
- [7] R. Akrou, M. Schoenauer, and M. Sebag. April: Active preference learning-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 116–131. Springer, 2012.
- [8] A. Jain, S. Sharma, T. Joachims, and A. Saxena. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research*, 34(10):1296–1313, 2015.
- [9] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4(4):343–355, 2012.
- [10] C. Basu, Q. Yang, D. Hungerman, M. Singhal, and A. D. Dragan. Do you want your autonomous car to drive like you? In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 417–425. ACM, 2017.
- [11] M. De Gemmis, L. Iaquinta, P. Lops, C. Musto, F. Narducci, and G. Semeraro. Preference learning in recommender systems. *Preference Learning*, 41, 2009.
- [12] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4302–4310, 2017.
- [13] K. Wei, R. Iyer, and J. Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963, 2015.
- [14] E. Elhamifar, G. Sapiro, and S. S. Sastry. Dissimilarity-based sparse subset selection. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2182–2197, 2016.
- [15] Y. Yang and M. Loog. Single shot active learning using pseudo annotators. *arXiv preprint arXiv:1805.06660*, 2018.
- [16] R. Holladay, S. Javdani, A. Dragan, and S. Srinivasa. Active comparison based learning incorporating user uncertainty and noise. In *RSS Workshop on Model Learning for Human-Robot Communication*, 2016.
- [17] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.
- [18] S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*, 2012.
- [19] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots*, pages 1–22, 2016.
- [20] D. Sadigh. *Safe and Interactive Autonomy: Control, Learning, and Verification*. PhD thesis, EECS Department, University of California, Berkeley, Aug 2017. URL <http://www2.eecs>.

berkeley.edu/Pubs/TechRpts/2017/EECS-2017-143.html.

- [21] C. Basu, Q. Yang, D. Hungerman, A. Dragan, and M. Singhal. Do you want your autonomous car to drive like you? In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2017.
- [22] R. Akrou, M. Schoenauer, and M. Sebag. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 12–27. Springer, 2011.
- [23] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.-H. Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1-2): 123–156, 2012.
- [24] A. Wilson, A. Fern, and P. Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems*, pages 1133–1141, 2012.
- [25] N. Ailon. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research*, 13(Jan):137–164, 2012.
- [26] H. Haario, E. Saksman, J. Tamminen, et al. An adaptive metropolis algorithm. *Bernoulli*, 7(2): 223–242, 2001.
- [27] D. Krueger, J. Leike, O. Evans, and J. Salvatier. Active reinforcement learning: Observing rewards at a cost. In *Future of Interactive Learning Machines, NIPS Workshop*, 2016.
- [28] G. Andrew and J. Gao. Scalable training of l_1 -regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM, 2007.
- [29] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *Advances in neural information processing systems*, pages 593–600, 2008.
- [30] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.
- [31] L. Kaufman and P. Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.
- [32] C. Bauckhage. Numpy/scipy recipes for data science: k-medoids clustering. *researchgate.net*, Feb, 2015.
- [33] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th international conference on World wide web*, pages 381–390. ACM, 2009.
- [34] A. Borodin, H. C. Lee, and Y. Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 155–166. ACM, 2012.
- [35] A. Krause and D. Golovin. Submodular function maximization., 2014.
- [36] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.
- [37] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, 2016.
- [38] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [39] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.
- [40] B. Matérn. *Spatial variation*, volume 36. Springer Science & Business Media, 2013.
- [41] A. Kulesza, B. Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.
- [42] C. Zhang, H. Kjellstrom, and S. Mandt. Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.