LEARNING PREFERENCES FOR INTERACTIVE AUTONOMY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Erdem Bıyık
May 2022

© 2022 by Erdem Biyik. All Rights Reserved.
Re-distributed by Stanford University under license with the author.

This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.
http://creativecommons.org/licenses/by-nc/3.0/us/

This dissertation is online at: https://purl.stanford.edu/vz918rc3628

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dorsa Sadigh, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Emma Brunskill

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Chelsea Finn

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

When robots enter everyday human environments, they need to understand their tasks and how they should perform those tasks. To encode these, reward functions, which specify the objective of a robot, are employed. However, designing reward functions can be extremely challenging for complex tasks and environments. A more promising approach is to learn reward functions from humans. Recently, several robot learning works embrace this approach and leverage human demonstrations to learn the reward functions. Known as inverse reinforcement learning, this approach relies on a fundamental assumption: humans can provide near-optimal demonstrations to the robot. Unfortunately, this is rarely the case – human demonstrations to the robot are often suboptimal due to various reasons, e.g., difficulty of teleoperation, robot having high degrees of freedom, or humans' cognitive limitations.

This thesis is an attempt towards learning reward functions from human users by using other data modalities that are more reliable. Specifically, this thesis studies how reward functions can be learned using *comparative feedback*, in which the human user compares multiple robot trajectories instead of (or in addition to) providing demonstrations. To this end, we first propose various forms of comparative feedback, e.g., pairwise comparisons, best-of-many choices, rankings, scaled comparisons; and describe how a robot can use these various forms of human feedback to infer a reward function, which may be parametric or non-parametric. We discuss the pros and cons of each comparative feedback modality in detail, and show how such feedback enables us to outperform standard inverse reinforcement learning that only utilizes demonstrations.

An important limitation of comparative feedback is that each comparison carries only a small amount of information: instead of observing the humans' actions at every time step of a demonstration, we only observe their comparison between trajectories. This harms data-efficiency, which is crucial in robotics due to the cost of collecting data (especially when it is coming from humans). To solve this, we propose *active learning* techniques to enable the robot to ask for comparison feedback that optimizes for the expected information that will be gained from that user feedback.

While showcasing the benefits of these various techniques for learning and active querying, we also demonstrate its applicability in a wide variety of domains. Our experiment domains range from autonomous driving simulations to home robotics, from standard reinforcement learning benchmarks to lower-body exoskeletons.

Acknowledgments

First and foremost, I would like to thank my advisor Dorsa Sadigh. It has been an absolute pleasure to learn from and work with her. Even though I had almost no experience in robotics when I started working with Dorsa, her passion, positivity and hard-work have always inspired and motivated me to learn more. Being her first Ph.D. student has been an honor and gave me invaluable experience. I would not have been where I am today without her guidance and support.

Many of my works in graduate school have been in close collaboration with other advisors. Great academic advice by Ramtin Pedarsani made a lot of positive impact not only in my studies but also in my life. Dylan P. Losey and Mykel J. Kochenderfer have always been role models with their dedication and time management skills. My collaborations with Nima Anari, Yisong Yue, Yanan Sui, Stephen L. Smith, and Aaron D. Ames contributed a great deal to this thesis. I would also like to thank my internship supervisors and collaborators at Google Research: Yinlam Chow, Mohammad Ghavamzadeh, Chih-wei Hsu, Alex Haig, and Craig Boutilier for giving me a perspective for preference-based learning algorithms outside robotics. It has also been an honor for me to collaborate and be co-authors with great mentors from both industry and academia: Adrien Gaidon, Guy Rosman, Judith E. Fan, Shahrouz Ryan Alimo, and Andrea Goldsmith. Although we have not collaborated on a research project yet, Scott Niekum, Emma Brunskill, and Chelsea Finn gave me a lot of guidance and support during my Ph.D.

My first research experience was at Bilkent University during my undergraduate studies with Tolga Çukur in 2016. His continuous support to this date has been invaluable and I am thankful to him, as well as his students Efe Ilıcak, Kübra Keskin, L. Kerem Şenel, Salman U. H. Dar. I was also lucky enough to work and be co-authors with Aykut Koç at ASELSAN as a research engineer, and with Mohamad Dia and Jean Barbier in Rüdiger Urbanke's lab at EPFL as a research intern. Last but not least, I enjoyed working with and learning from my mentors at Bilkent: Orhan Arıkan, Cem Tekin, Emine Ülkü Sarıtaş, İsmail Uyanık, and my collaborators: H. Can Baykara, Gamze Gül, Deniz Onural, Ahmet Safa Öztürk, and İlkay Yıldız.

I would also like to thank all my amazing co-authors who have been one of the main sources of learning and incredibly fun to work with. Daniel A. Lazar is not only an efficient collaborator with his math skills and sharpness, but also a great travel companion. Minae Kwon, who was already an

intern in the lab when I started, has always been welcoming and I learned a lot from chatting with her. I also had the privilege to work with Nils Wilde, Anusha Lalitha, Amir Maleki, Mark Beliaev, Zhangjie Cao, Malayandi Palan, Nicholas C. Landolfi, Sydney M. Katz, Kejun (Amy) Li, Maegan Tucker, Ellen Novoseller, Chandrayee Basu, Erik Brockbank, Rajarshi Saha, Zhixun (Jason) He, Vivek Myers, Woodrow Z. Wang, Nicolas Huynh, Aditi Talati, Suvir Mirchandani, Kenneth Wang, Gleb Shevchuk, Zheqing (Bill) Zhu, Karan Bhasin, Jonathan Margoliash, and Allan Raventos.

Although the Ph.D. programs are infamously said to be lonely, I never felt in that way thanks to my friends. Special thanks to my labmates at ILIAD: Mengxi Li, Sidd Karamcheti, Andy Shih, Megha Srivastava, Suneel Belkhale and Zhiyang (Jerry) He for making the lab a great place to work at. I also thank Turkish Student Association, especially Burak Bartan, Serhat Arslan, Atiye Cansu Erol Arslan, Süleyman Kerimov, Okan Atalar, Hüseyin İnan for our board game nights; as well as my close friends from Turkey: Melike Ersoy, Batuhan Sütbaş, Ömer Mert Aksoy, Görkem Ünlü, Naz Yetimoğlu, Fatih Karaoğlanoğlu, Ömer Arol for supporting me even from thousands of kilometers away and the virtual social activities we have had during the pandemic.

Finally, I would like to thank my family for all their love and support. My parents Reyhan Bıyık and İrfan Bıyık have always believed and put confidence in me. They brought up me in a way that made this thesis possible: they always taught and encouraged me to pursue what I enjoy, and do it with passion, responsibility and determination. And my sister, Begüm Bıyık, has been a "best friend instead of a sibling" as she always wanted when she was younger; and my main source of good songs, which have been essential while working long hours. Overall, I am thankful to all members of my family: their perspective allowed me to understand "Science is the most reliable guide for civilization, for life, for success in the world. Searching a guide other than the science is meaning carelessness, ignorance and heresy."

Contents

Abstract **iv**

Acknowledgments **v**

1 Introduction **1**
1.1 Thesis Approach . . . 2
1.2 Contributions . . . 2
1.3 Thesis Organization . . . 4

2 Background **6**
2.1 Reinforcement Learning (RL) . . . 6
2.2 Inverse Reinforcement Learning (IRL) . . . 8

3 Learning Reward Functions via Comparative Feedback **10**
3.1 Incorporating Comparisons into IRL . . . 10
3.1.1 Formulation . . . 11
3.1.2 Our Approach . . . 13
3.2 Learning Non-parametric Rewards via Pairwise Comparisons . . . 16
3.2.1 Related Work . . . 16
3.2.2 Formulation . . . 17
3.2.3 Our Approach . . . 18
3.3 Incorporating Ordinal Feedback along with Comparisons . . . 22
3.3.1 Formulation . . . 23
3.3.2 Learning Algorithm . . . 24
3.4 More Expressive Feedback: Scale Questions . . . 26
3.4.1 Formulation . . . 28
3.4.2 Our Approach . . . 29
3.4.3 Algorithm Design . . . 33
3.5 Learning Multimodal Rewards via Ranking Queries . . . 34

3.5.1 Formulation . . . 36
3.5.2 Our Approach . . . 38
3.6 Hierarchical Comparison Queries for Non-stationary Rewards . . . 38
3.6.1 Formulation . . . 40
3.6.2 Hierarchical Comparison Queries . . . 41
3.6.3 Reward Dynamics Model . . . 42
3.6.4 Learning Reward Dynamics . . . 44
3.6.5 Derivation and Simplifications . . . 45
3.7 Chapter Summary . . . 46

4 Active Querying for Comparative Feedback **47**
4.1 Choosing Queries with Volume Removal . . . 48
4.1.1 Maximum Volume Removal Optimization . . . 48
4.1.2 Experiments . . . 52
4.2 Choosing Queries with Mutual Information . . . 57
4.2.1 Maximum Mutual Information Optimization . . . 57
4.2.2 Additional Tools and Analysis . . . 58
4.2.3 Algorithm . . . 61
4.2.4 Experiments . . . 61
4.3 Active Querying for Preference-based GP Regression . . . 68
4.3.1 Formulation . . . 68
4.3.2 Experiments . . . 70
4.4 ROI Active Learning with Comparisons and Ordinal Feedback . . . 76
4.4.1 Formulation . . . 77
4.4.2 Simulations and Experiments . . . 78
4.5 Active Querying for Scale Feedback . . . 83
4.5.1 Two Acquisition Functions for Active Scale Feedback . . . 83
4.5.2 Experiments . . . 85
4.6 Active Querying for Multimodal Rewards . . . 89
4.6.1 Formulation . . . 89
4.6.2 Overall Algorithm . . . 89
4.6.3 Experiments . . . 91
4.6.4 User Studies . . . 95
4.7 Active Generation of Hierarchical Queries . . . 97
4.7.1 Active Querying based on Maximum Volume Removal . . . 97
4.7.2 Simulations and Experiments . . . 98
4.8 Batch-mode Active Querying for Time-Efficiency . . . 103
4.8.1 Formulation . . . 106

4.8.2 DPP-based Batch Active Learning . . . 108
4.8.3 Time-Efficient Batch Active Learning Methods . . . 111
4.8.4 Theoretical Guarantees . . . 114
4.8.5 Simulations and Experiments . . . 114
4.9 Chapter Summary . . . 121

5 Final Words **123**
5.1 Challenges . . . 123
5.1.1 Limitations and Future Work in Learning . . . 124
5.1.2 Limitations and Future Work in Active Querying . . . 125
5.2 Closing Thoughts . . . 126

A Proofs **127**
A.1 Proof of Proposition 1 . . . 127
A.2 Volume Removal Equivalence when $|Q^{(i)}| = 2$ . . . 127
A.3 Proof of Theorem 2 . . . 129
A.4 Proof of Corollary 1 . . . 129
A.5 Justification for Remark 2 . . . 130

B Derivations **131**
B.1 Mutual Information Derivation for Section 4.2 . . . 131
B.1.1 Extension to User-Specific and Unknown ς . . . 132
B.2 Mutual Information Derivation for Section 4.3 . . . 134
B.3 Mutual Information Derivation for Section 4.6 . . . 137

C Implementation Details **139**
C.1 Metropolis-Hastings for Section 4.6 . . . 139
C.2 Simulated Annealing for Section 4.6 . . . 140
C.3 Hyperparameters for Section 4.6 . . . 140
C.4 Hyperparameter Tuning for DPPs in Section 4.8.5 . . . 140

D Experiment Details **142**
D.1 Environment Features for Sections 4.1.2 and 4.2.4 . . . 142
D.1.1 *FetchReach* . . . 142
D.1.2 *Driver* . . . 142
D.1.3 *Tosser* . . . 143
D.2 Environment Features for Section 4.5.2 . . . 143
D.2.1 *ExtendedDriver* . . . 143
D.2.2 Original *Driver* . . . 144

D.2.3 *FetchDrink* . . . 144
D.3 Choice of σ_S in the User Studies for Section 4.5.2 . . . 144
D.4 Baselines for Section 4.6.3 . . . 145
D.4.1 Random . . . 145
D.4.2 Volume Removal . . . 145
D.5 Trajectory Generation in Section 4.6.3 . . . 145
D.5.1 *LunarLander* Trajectories . . . 145
D.5.2 *FetchBanana* Trajectories . . . 146
D.6 Metrics in Section 4.6.3 . . . 147
D.6.1 `MSE` . . . 147
D.6.2 `Log-Likelihood` . . . 147
D.6.3 `Learned Policy Reward` . . . 148
D.7 Experimental Setup in Section 4.6.3 . . . 148
D.7.1 Shelf Descriptions for *FetchBanana* Environment . . . 148
D.7.2 User Interface . . . 148

E Additional Results **149**
E.1 Additional Simulation Results for Section 4.2.4 . . . 149
E.1.1 Results with User-Specific and Unknown ς . . . 149
E.1.2 Results without Query Space Discretization . . . 150
E.1.3 Effect of Information from "About Equal" Responses . . . 150
E.1.4 Optimal Stopping under Query-Independent Costs . . . 151
E.2 Additional Simulation Results for Section 4.5.2 . . . 151
E.2.1 *ExtendedDriver* . . . 152
E.2.2 Original *Driver* . . . 153
E.2.3 *FetchDrink* . . . 153
E.3 Results with Test Set with Mixture Data for Section 4.5.2 . . . 154
E.4 Numerical Results for Section 4.5.2 . . . 157
E.5 Synthetic Experiment for Section 4.6.3 . . . 157
E.5.1 Testing $M > 2$ . . . 157
E.5.2 Testing Robustness to M Parameter . . . 159
E.6 Additional Unimodal Baseline for Section 4.6.3 . . . 159

List of Tables

4.1 Environment Properties 115
4.2 Average Query Generation Times (seconds) 118

C.1 Hyperparameters 140

D.1 Features of the *ExtendedDriver* Environment 143

E.1 Numerical results of the simulations at selected iterations i 157
E.2 Final numerical results of the user study 158
E.3 Additional User Study Reward Learning Baseline 159

List of Figures

1.1 **(top)** The robot should first learn a model of the agent it is interacting with. **(bottom)** It will then adapt to and influence the other agent in the task. 2

3.1 Example of a demonstration (top) and a best-of-many choice query with $|Q| = 2$, i.e., a pairwise comparison query (bottom). During the demonstration the robot is *passive*, and the human teleoperates the robot to produce trajectory ξ_D from scratch. By contrast, the preference query can be *active*: the robot chooses two trajectories ξ_1 and ξ_2 to show to the human, and the human answers by selecting their preferred option. 12

3.2 Overview of our DemPref approach. The human starts by providing a set of *high-level* demonstrations (left), which are used to initialize the robot's belief over the human's reward function through w. The robot then *fine-tunes* this belief by asking questions (right): the robot actively generates a set of trajectories, and asks the human to choose their favorite. 13

3.3 The user is trying to teach the robot how to play a variant of mini-golf, where the reward differs among eight targets. In preference-based learning, instead of trying to design a reward function by hand or controlling the robot to provide demonstrations, the user simply compares two demonstrated trajectories on the robot. Here, ξ_1 and ξ_2 demonstrate two trajectories that correspond to hitting the ball towards the blue or green targets. 18

3.4 The Atalante exoskeleton, designed by Wandercraft, has 12 actuated joints, 6 on each leg. The experiments explore four gait parameters: step length, step duration, pelvis roll, and pelvis pitch. 22

3.5 Scale feedback allows users to provide finely detailed comparisons between different options. 27

3.6 Different feasible sets learned from pairwise comparison and scale feedback under the linear reward model. Shown is the updated weight space (green) after observing user feedback for one (ξ_1, ξ_2) pair. If $\bar{q} = 1$, scale feedback enables us to learn a tighter half-space; when $\bar{q} \in (0, 1)$, scale feedback enables us to learn an equality, i.e., a hyperplane. 30
3.7 Noiseless user model. 31
3.8 Examples of why multimodal reward functions might be needed. 34
3.9 (a) 1-step comparison query. In any two iterations a user with bimodal preference may pick the trajectories optimal with respect to two different true weights w_1^* and w_2^*. (b) This ambiguity shows up as noise in 1-step comparison based learning where the goal is to learn a single reward function w^* (on the left). In reality the true preference function of the user changes between w_1^* and w_2^* depending on the environment, θ^* governs the transition. Our algorithm learns such a bimodal preference: w_1 close to w_1^* and a w_2 close to w_2^* (on the right). (c) Our proposed hierarchical query consists of 3 sub-queries. In iteration i of querying, $Q^{(i,0)}$ is a context sub-query, $Q^{(i,1)}$ is a comparison between two trajectories, each a different continuation of $Q^{(i,0)}$, and $Q^{(i,2)}$ continues the preferred trajectory from $Q^{(i,1)}$. 41

4.1 Sample queries generated with the volume removal and information gain methods on *Driver* and *Tosser* tasks. Volume removal generates queries that are difficult, because the options are almost equally good or equally bad. 50
4.2 Comparing preference queries that do not account for the human's ability to answer to queries generated using our information gain approach. Here the robot is attempting to learn the user's reward function, and demonstrates two possible trajectories. The user should select the trajectory that better aligns with their own preferences. While the trajectories produced by the state-of-the-art volume removal method are almost indistinguishable, our information theoretic approach results in questions that are easy to answer, which eventually increase the robot's overall learning efficiency. . . . 51
4.3 Views from simulation domains, with a demonstration in orange: **(a)** *LunarLander*, **(b)** *FetchReach* (simulated), **(c)** *FetchReach* (physical). 52
4.4 The results of our first experiment, investigating whether initializing with demonstrations improves the learning rate of the algorithm, on three domains. On the *Driver*, *LunarLander*, and *FetchReach* (simulated) environments, initializing with one demonstration improved the rate of convergence significantly. 54

4.5 (Left) Our testing domain, with two trajectories generated according to the reward functions learned by IRL and DemPref from a specific user in our study. (Right) The results of our usability study – the error bars correspond to standard deviation and significant results are marked with an asterisk. We find that users rated the robot trained with DemPref as significantly better at accomplishing the task and preferred to use our method for training the robot significantly more than they did IRL. However, we did not find evidence to suggest that users found our method easier to use than standard IRL. 55
4.6 Alignment values are plotted (mean ± standard error) to compare mutual information and volume removal formulations. Standard errors are so small that they are mostly invisible in the plots. Dashed lines show the weak pairwise comparison query variants. Mutual information provides a significant increase in learning rate in all cases. While weak pairwise comparison queries lead to a large amount of improvement under volume removal, mutual information formulation is still superior in terms of the convergence rate. 64
4.7 Wrong answer ratios on different queries are shown. The numbers at top show the average number of wrong responses and "About Equal" choices, respectively, for both strict and weak queries. Mutual information formulation yields smaller numbers of wrong and "About Equal" answers, especially in the early stages. 65
4.8 User study results. Error bars show std. Asterisks show statistical significance. **(a)** Easiness survey results averaged over all queries and users. Queries generated using the mutual information maximization method are rated significantly easier by the users than the volume removal queries. **(b)** The number of identical options in the experiments averaged over all users. In Driver and Tosser, users indicated significantly less indistinguishable queries with mutual information maximization compared to volume removal maximization. **(c)** Final preferences averaged over the users. 7 means the user strongly prefers the optimized trajectory w.r.t. the learned reward by the mutual information formulation, and 1 is the volume removal. Dashed line represents indifference between two methods. Users significantly prefer the robot who learned using the mutual information maximization method for active query generation. . . 66
4.9 Simulation results for the order of demonstrations and preference queries. `Alignment` values are plotted (mean±s.e.). It is consistently better to first utilize the passively collected demonstrations rather than actively generated preference queries. The differences in the `Alignment` value is especially small in the *FetchReach* simulations, which might be due to the fact that it is a simpler environment in terms of the number of trajectory features. 67

4.10 Simulation results for optimal stopping. Line plots show cumulative active learning rewards (cumulative difference between the mutual information values and the query costs), averaged over 100 test runs and scaled for visualization. Histograms show when optimal stopping condition is satisfied, which aligns with the desired cumulative rewards. 68
4.11 Sample trajectories are shown for the two simulation environments. In *Driver*, another car is cutting in front of the ego vehicle. In *Tosser*, the robot must hit the dropping capsule such that it will fall into one of the baskets. 70
4.12 Accuracies and average log-likelihoods for test set queries are shown for the *Driver* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two polynomial. **(c)** Data-efficiency results that compare ActiveGP with RandomGP. Accuracies and average log-likelihoods for test set queries are shown (mean±std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood. 71
4.13 Features of 1000 *Tosser* trajectories are visualized in two-dimensional plane (gray). Poisson disk sampling allows us to obtain a diverse set of 20 samples (orange), whereas sampling uniformly at random yields mostly uninteresting trajectories (blue). 72
4.14 Accuracies and average log-likelihoods for test set queries are shown for the *Tosser* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two polynomial. **(c)** Data-efficiency results that compare ActiveGP with RandomGP. Accuracies and average log-likelihoods for test set queries are shown (mean±std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood. 73
4.15 Top view of the eight targets in the variant of mini-golf user study. The users assign distinct scores from 2 to 9 to the targets. The figure shows an example of this ranking. While the robot is capable of hitting the ball into the entire shaded region, the maximizers of a linear reward always lie near the corners of the shaded region in blue. Therefore, while the GP reward model can query the user with better trajectories (e.g. the green trajectory), the linear model only explores the boundaries (e.g. the blue trajectory that throws the ball outside of this region). Crosses show where the ball hits the ground. 74

4.16 **(a)** Prediction accuracy results (mean±se). Each trained with 15 queries, ActiveGP achieves significantly higher prediction accuracy than both ActiveLinear and RandomGP ($p < 0.05$). **(b)** User ratings on the final robot performance (mean±se). ActiveGP accomplishes the task significantly better than both ActiveLinear and RandomGP ($p < 0.05$). 75

4.17 1D posterior illustration. The true objective function is shown in orange, and the algorithm's posterior mean is blue. Blue shading indicates the confidence region for $\varepsilon = 0.5$. The solid grey line indicates the true ordinal threshold B_1^o: the ROI is above this threshold, while the ROA is below it. The dotted grey line is the algorithm's B_1^o hyperparameter. The actions queried so far are indicated with "x"s. Utilities are normalized in each plot so that the posterior mean spans the range from 0 to 1. . . . 79

4.18 Impact of random subset size on algorithm performance. a) Example 3D synthetic objective function and posterior learned by ROIAL with subset size = 500 after 80 iterations. Values are averaged over the 3$^{\text{rd}}$ dimension and normalized to range from 0 to 1. b-c) Algorithm's error in predicting preferences and ordinal labels (mean ± std). Each simulation evaluated performance at 1000 randomly- selected points; the model posterior was used to predict preferences between consecutive pairs of points and ordinal labels at each point. 79

4.19 Effect of the confidence interval. All simulations are run over 50 reward synthetic functions with a random subset size of 500. a) Left: cumulative number of points in the ROA ($\mathcal{B}_1^o$) queried at each iteration (mean ± std). Note that as ε increases, more samples are required for the confidence interval to fall below the ROA threshold, at which point ROIAL starts avoiding the ROA. Middle and right: error in predicting comparison and ordinal labels for different values of ε; predictions are over 1,000 random actions (mean ± std). b) Confusion matrices (column-normalized) of ordinal label prediction over the entire action space at iterations 80 and 240 with ε = -0.45. The 2×2 confusion matrices for ROI prediction accuracy are outlined in green. Prediction accuracy increases with the number of iterations. 80

4.20 Effect of noisy feedback. The ordinal and pairwise comparison noise parameters, σ_O and σ_C, range from 0.1 to 0.3 and 0.02 to 0.06, respectively. All cases use a random subset size of 500 and $\varepsilon = -0.45$, and each simulation uses 1,000 random points to evaluate label prediction. Plots show means ± standard deviation. 81

4.21 Confusion matrix of the validation phase results for all three subjects. The first column is gray because trajectories in the ROA ($\mathcal{B}_1^o$) were purposefully avoided to prevent subject discomfort. Percentages are normalized across columns. Parentheses show the numbers of gait trials in each case. 81

4.22 4D posterior mean reward across exoskeleton gaits. Rewards are plotted over each pair of gait space parameters, with the values averaged over the remaining 2 parameters in each plot. Each row corresponds to a subject: Subject 1 is the most experienced exoskeleton user, Subject 2 is the second-most experienced user, and Subject 3 never used the exoskeleton prior to the experiment. 82
4.23 Comparison of scale feedback and weak pairwise comparisons for different active querying methods. 85
4.24 All results are shown for the first experiment (mean±s.e. over 18 subjects). 88
4.25 All results are shown for the second experiment (mean±s.e. over 14 subjects). 88
4.26 The *LunarLander* environment is visualized with the two tasks. Sample trajectories associated with these tasks are shown. 91
4.27 Unimodal and bimodal reward learning models are compared under `MSE`. Both mean and median values (over 100 runs) are shown. Shaded regions show the first and the third quartiles. 93
4.28 Different querying methods are compared with the (top) `MSE` and (bottom) `Log-Likelihood` metrics (mean±se over 75 runs). 94
4.29 Mutual information based and random querying methods are compared with the `Learned_Policy_Reward` values (mean±se over 75 runs which correspond to 150 randomly generated reward function parameters) in *LunarLander*. 95
4.30 User study results (mean±se over 24 users for *LunarLander* and 13 groups for *Fetch-Banana*). 96
4.31 `Alignment` value shows that our algorithm converges well for non-driving data with non-active query selection when the simulated user is *oracle*. Here we show an average `Alignment` over 5 different ground truth *reward dynamics*. 99
4.32 `Alignment` values show that our algorithm with active query selection (left) can learn reward dynamics faster than non-active query selection (right) when the simulated user is *oracle*. Here we show an average `Alignment` over 5 different ground truth reward dynamics. 100
4.33 `Alignment` value shows even when the users are noisy our algorithm can learn the true *reward dynamics* (left) and that as the probability being at mode 1 increases, w_1 converges faster (right). 101
4.34 Distribution of $\hat{w}_1$ and $\hat{w}_2$ across all users for individual features. (a) User preferences vary widely for adherence to lane center and distance to road boundaries, but are very similar for efficiency (speed) and safe driving (collision avoidance). (b) While we did not learn significantly different w_1 and w_2 for individual users, the average reward with respect to $\hat{w}_1$ and $\hat{w}_2$ differs slightly for some of our study participants. 102

4.35 Most users gave high ratings to the trajectories optimal with respect to $R_{\hat{w}_1}$ and $R_{\hat{w}_2}$ and low ratings to trajectories optimal with respect to their perturbed versions $R_{w_1^p}$ and $R_{w_2^p}$ and the lowest rating to the trajectories that were optimal with respect to a reward function that is parameterized randomly R_{w_r}. 103
4.36 Batches should be both diverse and informative in batch active 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. 105
4.37 The effect of λ is visualized. The columns of the matrix L have the same magnitude here; however $\{1, 3\}$ is a more diverse set than $\{1, 2\}$. When $\lambda = 1$, $\{1, 3\}$ is two times more likely to be sampled from the DPP distribution than $\{1, 2\}$. When we increase λ to 2, this ratio increases to 4, since more diverse sets are boosted against the less diverse sets. 109
4.38 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 volume removal. 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 volume removal. 112
4.39 Simulation view of each environment. (a) *FetchReach*, (b) *Driver*, (c) *Tosser*, (d) *LunarLander*, (e) *MountainCar*, (f) *Swimmer*. 115
4.40 Batch-active learning methods are compared. 117
4.41 The performance of each algorithm is averaged over 10 different runs on *LDS* where w^* is uniformly randomly generated. Successive elimination performs better than the random querying and worse than the non-batch active method. 119

4.42 The performance the algorithms is shown. The non-batch active method performs poorly on *LunarLander* and *Tosser*. 119
4.43 Convergence to w^* as a function of time is 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. Successive elimination clearly outperforms both of them. 120
4.44 The performance of successive elimination algorithm with varying k values was averaged over 10 different runs with *LDS* where w^* is uniformly randomly generated and $|\mathcal{X}| = 20k$. (a) The `Alignment` values, and (b) average query times. 120
4.45 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. 121
4.46 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. 122

C.1 Multi-chain Metropolis-Hastings sampling **(left)** gives more representative samples from the distribution compared to the single-chain variant **(right)**. 139
C.2 Tuning results for the DPP-based method for various γ under each environment. . . 141

D.1 Sample *LunarLander* trajectory (left) with extracted features (right). 146
D.2 Sample *FetchBanana* trajectory (left) with extracted features (right). 147
D.3 The user interface for the online studies with the real Fetch robot (*FetchBanana* environment). The user selected the 2^{nd} trajectory as their top choice and the 6^{th} trajectory as the second top. 148

E.1 The simulation results with mutual information formulation for unknown ς. Plots are mean$\pm$s.e. 149

E.2 Alignment values are plotted (mean±s.e.) for the experiments without query space discretization, i.e., with continuous trajectory optimization for active query generation. 150
E.3 The results (mean±s.e.) of the simulations with weak pairwise comparison queries where we use the information from "About Equal" responses (blue and red lines) and where we do not use (purple and orange lines). 150
E.4 Simulation results for optimal stopping under query-independent costs. Line plots show cumulative active learning rewards (cumulative difference between the information gain values and the query costs), averaged over 100 test runs and scaled for better appearance. Histograms show when optimal stopping condition is satisfied. 151
E.5 `Alignment` (left) and `Relative_Reward` (right) for *ExtendedDriver* with $\sigma_S = 0.3$. . . 152
E.6 `Log-Likelihood` for the *ExtendedDriver* simulations. 153
E.7 `Alignment` and `Relative_Reward` for the original *Driver* with $\sigma_S = 0.1$. 154
E.8 `Alignment` and `Relative_Reward` for the original *Driver* with $\sigma_S = 0.3$. 154
E.9 `Log-Likelihood` for the original *Driver*. 154
E.10 Fetch robot with drink serving experiment (*FetchDrink*) with $\sigma_S = 0.1$. 155
E.11 Fetch robot with drink serving experiment (*FetchDrink*) with $\sigma_S = 0.3$. 155
E.12 Log-Likelihood for the Fetch robot with drink serving experiment (*FetchDrink*). . . . 155
E.13 Additional analysis results are shown (mean±s.e. over 18 subjects). 156
E.14 Different querying methods are compared on a synthetic environment (mean±se over 250 runs). 157
E.15 Different values of M for the mutual information maximization approach are compared (mean±se over 100 runs). 158

Chapter 1

Introduction

In recent years, we have seen enormous effort to integrate robots and systems equipped with artificial intelligence (AI) into the society. While these agents are increasingly becoming part of our lives, most of their current interactions with the humans is one-way, e.g., a driver commands a vehicle to park autonomously, or the vehicle warns the driver about weather conditions. However, their successful integration will require them to intelligently *learn*, *adapt to*, and *influence* the humans and other AI agents.

These two-way interactions, where agents need to learn, adapt to, and influence each other; appear in almost all real-life scenarios. Human teams that are good at collaborating are often the ones where each individual adapted themselves to the others, e.g., sports teams train together rather than trying to improve individually. However, AI agents are not yet capable of this adaptation: their inability to model others led to problems in several occasions. For example, price-setting bots tried to sell a book for 23.7 million dollars on an online retail website after blindly competing with each other and not realizing that by increasing the price, the other bots will increase the price as well, while no human would be willing to pay this price [188]. Though this is an old example, we still see similar issues arise: autonomous cars fail to change lanes because they do not know the other drivers will slow down if they simply nudge in front of them [137]. The approach in this thesis to enable the robots to achieve the two-way interactions is inspired by how humans interact: we efficiently infer our partners' goals to optimize our behavior. For example, we move to one side of the sidewalk when we see a cyclist is approaching. If there is a mismatch between the inferred goal and our own goal, we try to influence our partners, e.g., if the cyclist moves to the same side, we stop for a second to imply we want to stay on this side and they should use the other side.

To achieve this human-like interaction, robots should understand the objective in the task, which encodes what they need to do and how they should do what they do. Designing these objective by hand, known as reward function, is extremely challenging. A more promising approach is to learn it from humans. Recent works dominantly focused on learning from human demonstrations of the

task. However, human demonstrations are often suboptimal due to various reasons, e.g., difficulty of teleoperation, robots' high degrees of freedom, humans' cognitive limitations, etc. Therefore, many questions arise: What are some other forms of human feedback that enables robots to more reliably learn reward functions? How can robots learn from multiple data modalities? How can these methods extend to the cases where the reward function is multimodal or non-stationary? How can robots optimize for data-efficiency to mitigate the high costs of data collection? In this thesis, we attempt to answer these questions.

1.1 Thesis Approach

Integrating robots and systems equipped with AI into the society requires a thorough understanding of how they may learn, adapt to and influence other agents. Our approach is to divide this problem into two parts (see Figure 1.1) [33]. First, machine learning techniques that we develop in this thesis will enable AI agents to model the behaviors and goals of the other agents by leveraging different forms of information they provide. Next, these learned behaviors and goals will enable these agents to better interact with the others to achieve online adaptation, e.g., an autonomous vehicle will adapt to both its driver and the other vehicles to better optimize its route and driving style. In this thesis, we focus on the first aspect and study how robots can learn from human feedback.

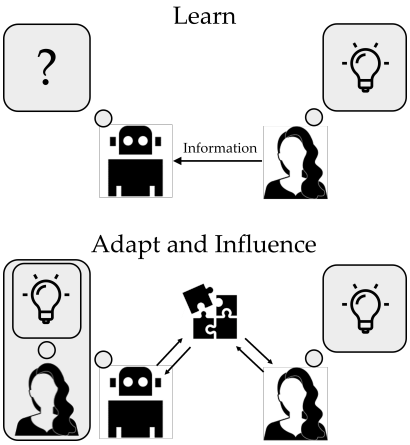


Figure 1.1: **(top)** The robot should first learn a model of the agent it is interacting with. **(bottom)** It will then adapt to and influence the other agent in the task.

Although recent works mostly focused on learning from demonstrations, they often suffer from the suboptimality of demonstrations. In this thesis, we propose using comparative feedback to learn the objectives, where human users are asked to compare multiple trajectories of a robot based on their preferences. We develop various forms of comparative feedback, and further study how they can be collected actively to improve data-efficiency, which is crucial in robotics, especially because the data are coming from human users. For these, we bridge ideas from machine learning, information theory, human-robot interaction, optimization and control theory.

1.2 Contributions

This thesis makes the following contributions.

The goal of this thesis is to develop rewards learning methods for robots that leverage

comparative feedback from humans in a data-efficient way.

Learning Reward Functions via Comparative Feedback

In Chapter 3, we study various forms of comparative feedback, how to learn reward functions using them, and how to utilize them along with user demonstrations that are possibly suboptimal. Specifically, we develop and analyze the following feedback modalities:

- **Pairwise comparisons:** The user selects their preferred trajectory among two options [34, 39, 40, 120].
- **Scale feedback:** The user uses a slider bar to indicate how much they prefer one trajectory over the other in a pairwise comparison setting [208].
- **Ordinal feedback:** In addition to pairwise comparisons, the user also labels each trajectory with an ordinal feedback. e.g., "Bad", "Neutral" and "Good" [135].
- **Best-of-many choices:** We extend the pairwise comparisons so that the user will now choose their most preferred trajectory out of multiple options, possibly more than two [43, 38, 36, 41, 26].
- **Hierarchical choices:** To handle non-stationary reward functions, we develop hierarchical choice queries in which the user first responds to a standard best-of-many choice query. After their response, a new best-of-many choice query is presented such that its trajectories start from the final state of the user's preferred trajectory in the first query [23].
- **Rankings.** The user ranks multiple (more than two) trajectories from their most preferred to the least preferred [154]. This helps robots learn multimodal reward functions, e.g., when the data are coming from multiple people.

In addition, we study different forms of reward functions that encode how a robot or an AI agent should perform the task:

- **Parametric reward functions:** Most of the thesis focuses on reward functions that are parametric [34, 39, 120, 43, 38, 36, 41, 26]. In principle, such functions may range from linear functions to neural networks. However, as we take a Bayesian learning approach, functions with large parameter spaces are difficult to learn in practice. This restricts us to simple functional forms.
- **Non-parametric reward functions:** To solve this issue and be able to learn more complex rewards, we employ Gaussian processes and learn non-parametric reward functions [40, 135].

- **Multimodal reward functions:** If the data are coming from multiple people with different objectives, or the same person with varying objectives, unimodal reward functions fail to encode their preferences. Hence, we model the multimodal reward as a mixture of multiple parametric unimodal reward functions [154].
- **Non-stationary reward functions:** As a specific case of multimodal reward functions, we study rewards that are non-stationary with some structure: the users' preferences change based on the history in the environment according to a parametric transition function [23].

Active Querying for Comparative Feedback

In Chapter 4, we address the problem of data inefficiency when learning from comparative feedback. As opposed to user demonstrations of the task, where each state in a trajectory receives an action label; comparisons contain very little information – they only say some trajectories are better than some others. This means a robot may require enormous amounts of data to learn useful reward functions via comparative feedback, especially when there are no demonstrations to warm-start the learning. To mitigate this problem, we develop active learning techniques to actively query the users for the most informative comparative feedback.

Specifically, we study active learning objectives that are based on volume removal [171, 34, 39, 36, 41], mutual information [38, 43, 41, 40, 135, 208, 154], and max-regret [207, 208]. While doing this, we follow a similar structure to Chapter 3: we describe how each section of Chapter 3 can be extended with active querying. As a result of this choice, we defer all simulation and experiment results to Chapter 4 where we not only investigate the learning performance but also analyze the benefits of active querying.

1.3 Thesis Organization

Chapter 2 is geared towards a reader who is inexperienced at robot learning: we present an introductory overview of reinforcement learning (RL) and inverse reinforcement learning (IRL) problems. While doing this, we do not focus on any particular solution – instead we keep the problem formulations general enough so that we can present learning from comparative feedback using the same formulation. In fact, the focus of this thesis, learning reward functions from comparative feedback is closely related to the inverse reinforcement learning problem. In both of these problems, the goal is to learn a reward function that encodes the desired behavior of the robot or the AI agent.

Chapter 3 then starts with building upon an existing IRL solution, namely Bayesian inverse reinforcement learning [164], where the reward function is learned using human demonstrations of the task. Again taking a Bayesian approach, we study how comparative feedback can be used to learn the reward function. For this, we start with best-of-many choice queries (Section 3.1, [38, 159, 43]). We then focus on a specific version of these queries with only two options to compare, i.e., pairwise

comparisons. Using this simpler query form enables us to learn non-parametric reward functions that are modeled via Gaussian processes (Section 3.2, [40]), which we later improve with ordinal feedback (Section 3.3, [135]). We then extend the pairwise comparisons by providing the users with a slider bar to indicate how much they prefer one trajectory over the other, which would not be practical with general best-of-many choice queries (Section 3.4, [208]). After this detour, we go back to queries with more than two options, and study ranking queries which enable us to learn multimodal reward functions (Section 3.5, [154]). Finally, we look at a specific case of multimodal rewards where users transition between different modes according to the latest behavior of the robot (Section 3.6, [23]). Section 3.7 summarizes Chapter 3.

In Chapter 4, we focus on how to improve data-efficiency when we use learning from comparative feedback where information is very sparse as opposed to demonstrations. We follow a similar structure to Chapter 3, i.e., we follow almost the same order to present the active querying techniques for each section in Chapter 3. We first introduce volume removal (Section 4.1, originally proposed in [171]) and mutual information (Section 4.2, [38, 43]) based active learning objectives for best-of-many choice queries. We then proceed with mutual information based active querying when the reward is non-parametric and modeled as a Gaussian process (Section 4.3, [40]), and extend it to the case where we also use ordinal feedback (Section 4.4, [135]). In addition to mutual information, we introduce max-regret based active querying in Section 4.5 where we focus on scale feedback [208]. Following the same order as in Chapter 3, we next present active methods for ranking queries when the reward is multimodal (Section 4.6, [154]) and for hierarchical choice queries when users transition between different reward modes (Section 4.7, [23]). All these active querying methods require solving an optimization problem for each and every query, which might be computationally expensive and makes the querying process non-parallelizable. Hence, in Section 4.8, we present various *batch* active learning methods where multiple queries are optimized together in batches [34, 39]. Finally, Section 4.9 summarizes the chapter.

Chapter 5 is the final chapter of the thesis in which we discuss the limitations and open challenges, and conclude the ideas presented throughout the thesis. Additional material, e.g., proofs, derivations, implementation and experiment details, and additional results, are presented in the appendices.

Chapter 2

Background

2.1 Reinforcement Learning (RL)

We first start with defining the reinforcement learning (RL) problem. The goal of reinforcement learning is to find how a dynamical system can be optimally controlled. For this, we will first mathematically define dynamical systems. As a running example, let's consider a robot trying to reach an object on a desk.

We use a discrete-time Markov decision process (MDP) to define a dynamical system. An MDP is a tuple $\mathcal{M} = \langle \mathcal{S}, \pi_0, \mathcal{A}, \mathcal{T}, T, r \rangle$ with the following variables. $\mathcal{S}$ denotes the state space. Each $s \in \mathcal{S}$ fully characterizes a state of the world, e.g., the robot's and the object's poses and velocities. Each episode in the system starts with an initial state $s_0 \in \mathcal{S}$ drawn randomly from the initial state distribution π_0:

$$s_0 \sim \pi_0(\cdot)\,. \tag{2.1}$$

$\mathcal{A}$ denotes the action space such that each $a \in \mathcal{A}$ is an action taken in the system, e.g., the robot is given some control input. The transition distribution $\mathcal{T}$ then governs how this system evolves. Based on the action a_t at time step t, the system transitions from state s_t to a new state s_{t+1} according to:

$$s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)\,. \tag{2.2}$$

It is important that each state *fully* characterizes the state of the world: knowing the current state and action is sufficient to best predict the next state, i.e.,

$$\mathcal{T}(\cdot \mid s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = \mathcal{T}(\cdot \mid s_t, a_t)\,. \tag{2.3}$$

This is known as the Markov property. The system evolves for $T < \infty$ time steps, known as the *horizon* of the MDP.[1]

At each time step t, the decision maker (the agent) receives a scalar *reward*, based on the reward function r. For example, the robot may experience some positive reward for getting close to the target object, or negative reward (cost, or penalty) for colliding with some obstacles around. The goal of the agent is to select its actions to maximize the expected cumulative reward over the time steps of an episode. In literature, there are different conventions about how a reward function is defined. The three options are to define them as a function of:

- only the current state: $r : \mathcal{S} \to \mathbb{R}$,
- the current state and action: $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,
- the current state, action, and the next state: $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$.

In this thesis, we will be focusing on learning from comparative feedback where multiple trajectories of a robot (or multiple episodes in an MDP) are compared. Therefore, we will use trajectory reward functions. For this, we first let a trajectory be a sequence of state-action pairs, i.e., $\xi = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$, and Ξ denote all possible trajectories of the system. This trajectory reward function $R : \Xi \to \mathbb{R}$ can then be defined for each of the reward function conventions above:

$$R(\xi) := \sum_{t=0}^{T} r(s_t), \tag{2.4}$$

$$R(\xi) := \sum_{t=0}^{T} r(s_t, a_t), \text{ or} \tag{2.5}$$

$$R(\xi) := \sum_{t=0}^{T-1} r(s_t, a_t, s_{t+1}) \,. \tag{2.6}$$

In fact, trajectory reward function can be defined more broadly: it does not have to be additive over time steps. Hence, it is more expressive and general. Consequently in our setup, the goal of the agent is to maximize the trajectory reward. This is the problem that reinforcement learning methods attempt to solve: how should an agent decide its actions (based on the state of the system) so that the trajectory will acquire as much reward as possible?

Reinforcement learning is still a very active area of research. Over the past decade, several methods have been successfully implemented for various versions or applications of this problem. Although the details of those methods are beyond the scope of this thesis, we include a list of widely-used methods: deep Q-networks (DQN) [151], deep deterministic policy gradient (DDPG) [138], asynchronous advantage actor-critic (A3C) [152], trust region policy optimization (TRPO)

[1] In this thesis we only consider finite-horizon MDPs. Extending to infinite-horizon MDPs requires an additional variable: discount factor. We refer to the standard text on reinforcement learning by Sutton and Barto for more details [189].

[175], proximal policy optimization (PPO) [176], hindsight experience replay (HER) [12], actor-critic using Kronecker-factored trust region method (ACKTR) [215], actor-critic with experience replay (ACER) [201], twin delayed DDPG (TD3) [90], and soft-actor critic (SAC) [104].

2.2 Inverse Reinforcement Learning (IRL)

Inverse reinforcement learning (IRL), as its name implies, tries to solve an inverse problem. In IRL, an agent who is already acting (near-)optimally in a system provides some data, i.e., they control the system. For example, an expert operator provides demonstrations of a task by teleoperating a robot. The goal in IRL is to use these expert demonstrations to identify the objective of the task, i.e., the reward function [1, 2, 155, 157, 87].[2]

Formally in IRL, we are given some trajectory demonstrations $\xi_D^{(1)}, \xi_D^{(2)}, \cdots \in \Xi$ (or more generally: state-action pairs, or transition tuples that also include the next state) that are known to be (near-)optimal with respect to the target task, encoded by an unknown reward function r. The goal is to learn this reward function r.

It might not be obvious why IRL is an important problem: if we already have an agent that is able to successfully control the system, why do we try to learn a reward function? The most common reason is automation. It is usually the case that the expert agent is a human, which means we need that expert human every time we need to control the system. However, if we can learn the reward function that encodes the task, then we can perform reinforcement learning in this system with the learned reward function to be able to control the system even in the absence of the expert. This is not the only use case of IRL. Another interesting application is behavior modeling [140]. Imagine we are trying to develop an autonomous vehicle that predicts the actions of the other cars around so that it will seamlessly interact with them in traffic. To do this, understanding the objective of the other cars is crucial: if our car can infer their objective, i.e., their reward function, then it may better predict their actions. IRL has also applications in recommendation systems: Given a user's browsing history (a demonstration), the goal is to learn their preferences (reward function) so that the system can make better recommendations in future (learn a better policy).

Similar to reinforcement learning, IRL is also an active research area. Arguably the most influential methods in IRL have been apprenticeship learning [1], maximum margin planning [167], Bayesian inverse reinforcement learning [164], and maximum entropy inverse reinforcement learning (MaxEnt-IRL) [226].

In this section, we reviewed the standard IRL problem where the goal is to learn a reward function given expert demonstrations. However in many cases, especially in robotics, expert demonstrations

[2] Another interesting and very related problem is imitation learning [160, 168, 109, 182, 88, 181], where the goal is to directly learn an optimal control policy from expert demonstrations. Although the research community does not have a consensus on the scope of these terms, we use this convention: IRL tries to learn the reward function, imitation learning tries to learn the optimal policy; both from expert demonstrations.

may not be available, or all users of the system might be providing suboptimal demonstrations [98]. Two common reasons for this are (1) good demonstrations might require a high level of expertise [197], and (2) it is often too difficult to manually operate robots, especially manipulators with high degrees of freedom (DoF) [5, 84, 115, 123]. Moreover, even when operating the high DoF of a robot is not an issue, people might have cognitive biases or habits that cause their demonstrations to not align with their actual reward functions. For example, in [131] we have shown that people tend to perform consistently risk-averse or risk-seeking actions in risky situations, depending on their potential losses or gains, even if those actions are suboptimal. As another example from the field of autonomous driving, Basu et al. [21] suggest that people prefer their autonomous vehicles to be more timid compared to their own demonstrations. These problems show that, even though demonstrations carry an important amount of information about what the humans want, one should either try to learn from suboptimal demonstrations [53, 64, 58, 99, 214] or go beyond demonstrations, e.g., corrections [19, 20, 142, 221, 136], rankings [52, 51, 53, 64], critiques [14, 78], trajectory assessments [178], or ordinal feedback [74], to properly capture the underlying reward functions. The latter approach is also the theme of this thesis: We will go beyond demonstrations and present methods that (actively) learn reward functions from comparative feedback where users compare multiple trajectories of the system. This type of feedback has been shown to be successful in several other domains such as classification [65], bandit problems [54], and reinforcement learning [212].

Chapter 3

Learning Reward Functions via Comparative Feedback

Having presented the reinforcement learning (RL) and inverse reinforcement learning (IRL) problems in Chapter 2, we are now ready to start presenting our learning methods. In this chapter, we present alternative IRL solutions in which we learn reward functions using comparative feedback. Although the novelty of our methods is due to the use of comparative feedback, we still allow the use of demonstrations as in the standard IRL framework. To this end, Section 3.1 presents how we can incorporate the information from best-of-many choices [38, 43] into Bayesian IRL [164] that originally learns from demonstrations. Although we reduce the emphasis on demonstrations in later sections, the same Bayesian approach easily extends to all methods in this chapter except Section 3.6 where we assume humans have non-stationary rewards, which makes the generation process of demonstrations ambiguous.

3.1 Incorporating Comparisons into IRL

Let's start with briefly going over the other works in the literature that attempt to incorporate comparative feedback into the IRL framework.

Learning reward functions from rankings and best-of-many choices. Two helpful and closely related sources of information that can be used to learn reward functions is rankings and best-of-many choices. In rankings, a human expert ranks a set of trajectories in the order of their preference [51] whereas in best-of-many choices they just pick their favorite trajectory [36]. A special case of both of these, which we also adopt in our experiments, is when these queries are pairwise [7, 133, 72, 114, 52, 205, 212, 6, 92, 184, 210]. While these works experimented their methods on some simulation environments, others leveraged pairwise comparison questions for various real-life

applications, including exoskeleton gait optimization [193], and trajectory optimization for robots in interactive settings [57, 159].

Learning reward functions from both demonstrations and comparisons. Ibarz et al. [114] have explored combining demonstrations and comparisons, where they take a model-free approach to learn a reward function in the Atari domain. Our motivation, physical autonomous systems, differs from theirs, leading us to a structurally different method. It is difficult and expensive to obtain data from humans controlling physical robots. Hence, model-free approaches are presently impractical. In contrast, we give special attention to data-efficiency as we will discuss in detail in Sections 4.1 and 4.2. As the resulting training process is not especially time-intensive, we efficiently learn personalized reward functions.

3.1.1 Formulation

Building on prior work, we integrate demonstrations and comparative feedback to learn the human's reward function. Here we formalize this problem setting, and introduce the two forms of human feedback that we will focus on: demonstrations and best-of-many choices. Our formulation revisits the definitions in Section 2.1 and extends it for comparative feedback.

MDP. Let us consider a fully observable dynamical system describing the evolution of the robot, which should ideally behave according to the human's preferences. We formulate this system as a discrete-time Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, T \rangle$. At time t, $s_t \in \mathcal{S}$ denotes the state of the system and $a_t \in \mathcal{A}$ denotes the robot's action. The robot transitions to a new state according to its dynamics distribution: $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$. At every time step the robot receives reward $r(s)$, and the task ends after a total of T time steps.

Trajectory. A trajectory $\xi \in \Xi$ is a finite sequence of state-action pairs, i.e., $\xi = \big((s_t, a_t)\big)_{t=0}^{T}$ over the time horizon T.

Reward. The reward function captures how the human wants the robot to behave. Similar to related works [1, 155, 226] where the reward is a linear combination of features, we assume the reward is a parametric function of some trajectory features. Consistent with prior work [45, 19, 226], we will assume that the trajectory features $\Phi(\xi)$ for any given trajectory ξ are known. In practice, they can be based on the state features along that trajectory. Or more generally, the trajectory features $\Phi(\xi)$ can be defined as any function over the entire trajectory ξ as we discussed in Section 2.1. To understand what the human wants, the robot must simply learn the true parameters of the reward function which we denote with w^*. Accordingly, we denote a trajectory reward function parameterized with w as:

$$R_w(\xi) = f_w(\Phi(\xi)) \,. \tag{3.1}$$

Demonstrations. One way that the human can convey their reward function parameters w to

Figure 3.1: Example of a demonstration (top) and a best-of-many choice query with $|Q| = 2$, i.e., a pairwise comparison query (bottom). During the demonstration the robot is *passive*, and the human teleoperates the robot to produce trajectory ξ_D from scratch. By contrast, the preference query can be *active*: the robot chooses two trajectories ξ_1 and ξ_2 to show to the human, and the human answers by selecting their preferred option.

the robot is by providing demonstrations. Each human demonstration is a trajectory ξ_D, and we denote a dataset of human demonstrations as $\mathcal{D}_D = \{\xi_D^{(1)}, \xi_D^{(2)}, \ldots, \xi_D^{(|\mathcal{D}_D|)}\}$. In practice, these demonstrations could be provided by kinesthetic teaching, by teleoperating the robot, or in virtual reality (see Figure 3.1, top).

Best-of-many Choices. Another way the human can provide information is by giving feedback about the trajectories the robot shows. We define a best-of-many choice query $Q = \{\xi_1, \xi_2, \ldots, \xi_{|Q|}\}$ as a set of $|Q|$ robot trajectories. The human answers this query by picking a trajectory $q \in Q$ that matches their personal preferences (i.e., maximizes their reward function). In practice, the robot could play $|Q|$ different trajectories, and let the human choose their favorite (see Figure 3.1, bottom).

Problem. Our overall goal is to accurately and efficiently learn the human's reward function from multiple sources of data. In this section, we will only focus on demonstrations and best-of-many choices. Our approach should learn the reward parameters w with a combination of demonstrations and best-of-many choice queries.

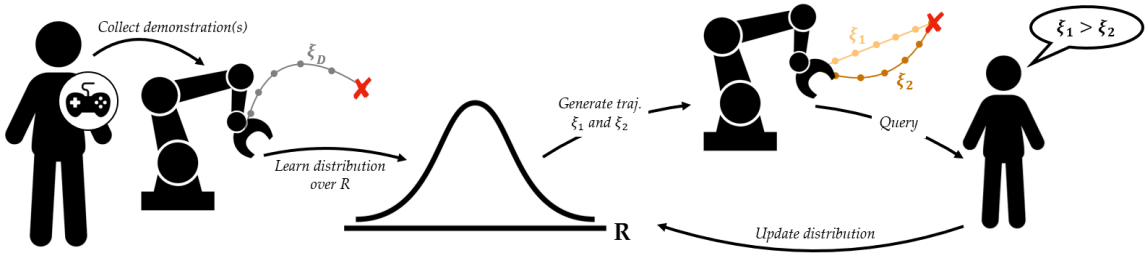


Figure 3.2: Overview of our DemPref approach. The human starts by providing a set of *high-level* demonstrations (left), which are used to initialize the robot's belief over the human's reward function through w. The robot then *fine-tunes* this belief by asking questions (right): the robot actively generates a set of trajectories, and asks the human to choose their favorite.

3.1.2 Our Approach

We now overview our approach for integrating demonstrations and best-of-many choices to efficiently learn the human's reward function. Intuitively, demonstrations provide an informative, *high-level* understanding of what behavior the human wants; however, these demonstrations are often noisy, and may fail to cover some aspects of the reward function. By contrast, preferences are *fine-grained*: they isolate specific, ambiguous aspects of the human's reward, and reduce the robot's uncertainty over these regions. It therefore makes sense for the robot to start with high-level demonstrations before moving to fine-grained preferences. Indeed — as we will show in Theorem 3 — starting with demonstrations and then shifting to actively collected comparative feedback is the most efficient order for gathering data. Our algorithm, which we call DemPref (short for demonstrations and preferences) leverages this insight to combine high-level demonstrations and low-level best-of-many choice queries (see Figure 3.2).

Initializing a Belief from Offline Demonstrations

DemPref starts with a set of offline trajectory demonstrations $\mathcal{D}_D$. These demonstrations are collected *passively*: the robot lets the human show their desired behavior, and does not interfere or probe the user. We leverage these passive human demonstrations to initialize an informative but imprecise prior over the true reward function parameters w.

Belief. Let the belief b be a probability distribution over w. We initialize b using the trajectory demonstrations, so that $b^0(w) = P(w \mid \mathcal{D}_D)$. Applying Bayes' Theorem:

$$\begin{aligned} b^0(w) &\propto P(\mathcal{D}_D \mid w)P(w) \\ &= P(\xi_D^{(1)}, \xi_D^{(2)}, \ldots, \xi_D^{(|\mathcal{D}_D|)} \mid w)P(w) \end{aligned} \tag{3.2}$$

We assume that the trajectory demonstrations are conditionally independent, i.e., the human does

not consider their previous demonstrations when providing a new demonstration. Hence, Equation (3.2) becomes:

$$b^0(w) \propto P(w) \prod_{\xi_D \in \mathcal{D}_D} P(\xi_D \mid w) \tag{3.3}$$

In order to evaluate Equation (3.3), we need a model of $P(\xi_D \mid w)$ — in other words, how likely is the demonstrated trajectory ξ_D given that the human's reward function parameters are w?

Boltzmann Rational Model. DemPref is not tied to any specific choice of the human model in Equation (3.3), but we do want to highlight the Boltzmann rational model that is commonly used in inverse reinforcement learning [226, 164]. Under this particular model, the probability of a human demonstration is related to the reward associated with that trajectory:

$$P(\xi_D \mid w) \propto \exp\left(\beta^D R_w(\xi_D)\right) \tag{3.4}$$

$$= \exp\left(\beta^D f_w(\Phi(\xi_D))\right). \tag{3.5}$$

Here $\beta^D \geq 0$ is a temperature hyperparameter, commonly referred to as the *rationality coefficient*, that expresses how noisy the human demonstrations are, and we substituted Equation (3.1) for R. Leveraging this human model, the initial belief over w given the offline demonstrations becomes:

$$b^0(\omega) \propto \exp\left(\beta^D \cdot \sum_{\xi_D \in \mathcal{D}_D} f_w(\Phi(\xi_D))\right) P(w) \tag{3.6}$$

Summary. Human demonstrations provide an informative but imprecise understanding of w. Because these demonstrations are collected passively, the robot does not have an opportunity to investigate aspects of w that it is unsure about. We therefore leverage these demonstrations to initialize b^0, which we treat as a high-level *prior* over the human's reward. Next, we will introduce how we update this belief using best-of-many choice questions to remove uncertainty and obtain a fine-grained posterior.

Updating the Belief with Proactive Queries

After initialization, DemPref iteratively performs two main tasks: *actively* choosing the right preference query Q to ask, and applying the human's answer to update b. In this section we focus on the second task: updating the robot's belief b. We will explore how robots should proactively choose the right question in Sections 4.1 and 4.2 under Chapter 4.

Posterior. The robot asks a new question at each iteration $i \geq 1$. Let $Q^{(i)}$ denote the i-th best-of-many choice query, and let $q^{(i)}$ be the human's response to this query.[1] Again applying Bayes'

[1]For the ease of notation, we will slightly abuse the notation and use $q^{(i)}$ as a random variable when the user's response has not been elicited yet, and as a constant after their response is revealed. Similarly, we will use $Q^{(i)}$ as an optimization variable when we are optimizing for the next query in Chapter 4, but it will be a constant as soon as the query is made to the user.

Theorem, the robot's posterior over w becomes:

$$b^i(w) \propto P(q^{(1)}, \ldots, q^{(i)} \mid Q^{(1)}, \ldots, Q^{(i)}, w) \cdot b^0(w)\,, \tag{3.7}$$

where b^0 is the prior initialized using human demonstrations. We assume that the human's responses q are conditionally independent, i.e., only based on the current query and reward function parameters. Equation (3.7) then simplifies to:

$$b^i(w) \propto b^0(w) \cdot \prod_{i'=1}^{i} P(q^{(i')} \mid Q^{(i')}, w) \tag{3.8}$$

We can equivalently write the robot's posterior over w after asking i questions as:

$$b^i(w) \propto P(q^{(i)} \mid Q^{(i)}, w) \cdot b^{i-1}(w) \tag{3.9}$$

Human Model. In Equation (3.9), $P(q^{(i)} \mid Q^{(i)}, w)$ denotes the probability that a human with reward function parameters w will answer query $Q^{(i)}$ by selecting trajectory $q^{(i)} \in Q^{(i)}$. Put another way, this likelihood function is a probabilistic human model, and previous work demonstrated the importance of modeling imperfect human responses [129]. One way to do this is to model a noisily optimal human as selecting $q^{(i)}$ from a *strict* best-of-many choice query $Q^{(i)}$ by

$$P(q^{(i)} = Q_j^{(i)} \mid Q^{(i)}, w) = \frac{\exp(\beta^C R_w(Q_j^{(i)}))}{\sum_{j'=1}^{|Q^{(i)}|} \exp(\beta^C R_w(Q_{j'}^{(i)}))}\,, \tag{3.10}$$

where β^C is the rationality coefficient for comparisons. We call the query strict because the human is required to select one of the trajectories. This model, backed by neuroscience and psychology [81, 147, 28, 146], is routinely used [36, 100, 196, 211]. It is also known as the multinomial logits (MNL) model [67].

Although we present this human choice model and use its variants in most of the subsequent sections, our DemPref approach is agnostic to the specific choice of $P(q^{(i)} \mid Q^{(i)}, w)$ — we test different human models in our experiments presented in Section 4.2. For now, we simply want to highlight that this human model defines the way users respond to queries.

Having defined the learning algorithm that incorporates both demonstrations and best-of-many choices, and a human model, we now have a computational method of learning reward functions from comparative feedback in addition to demonstrations. In Sections 4.1 and 4.2, we will boost the data-efficiency of this algorithm by developing active querying techniques. We also defer our experiments to those sections. Prior to them in the subsequent sections of this chapter, we will extend this framework to other forms of comparative feedback and relax some of the assumptions. The next section relaxes the parametric reward assumption.

3.2 Learning Non-parametric Rewards via Pairwise Comparisons

In Section 3.1, we showed a Bayesian learning approach that incorporates demonstrations and best-of-many choice queries. An important assumption we made is that the reward function is a *parametric* function of some known trajectory features. One might think that this is general enough, because we did not impose any other constraints on the functional form, so for example, deep neural networks could be good enough in most practical applications. However, computational issues often arise in practice: since we are taking a Bayesian learning approach, learning becomes infeasible as the number of parameters to learn increases. Given that deep neural networks often contain large number of learnable parameters, this means the approach is limited to simple functional forms. In fact, we will mostly focus on linear reward functions when we present our experiments as in the prior work [1, 155, 226].

Alternative approaches to Bayesian learning includes training a deep neural network using gradient-based optimization methods with a loss function that minimizes the log-likelihood where the likelihood simply comes from our belief distribution b [120]. However, these approaches are usually limited in the sense that they only give a point-estimate of the reward function, whereas the Bayesian learning approach enables us to model the uncertainty by learning a distribution over reward functions. Motivated by this, we relax the parametric reward function assumption by explicitly learning a distribution over reward functions using Gaussian processes (GP) in this section. To do this, we focus on a special case of best-of-many choice queries with $|Q| = 2$, i.e., pairwise comparisons where users just compare two trajectories and choose their favorite. Our results, which we again defer until we present the corresponding active querying methods in Section 4.3, show this significantly improves the expressive power of the learned reward function.

3.2.1 Related Work

Gaussian process regression. In the machine learning literature, González et al. [97] and Chu and Ghahramani [73] proposed methods for preference-based Bayesian optimization and GP regression, respectively, but they were not collecting data actively. Furthermore, González et al. [97] required to regress a GP over $2d$-dimensions to model a d-dimensional function, which causes a computational burden. More relevantly, Houlsby et al. [113] presented an active method for preference-based GP regression. However, similar to [97], the regression was over a $2d$-dimensional space where the learned model predicts the outcome of a comparison rather than outputting a reward value. Jensen and Nielsen [116] showed how to update a GP with preference data, but the active query generation was not an interest. Kapoor et al. [119] developed an active learning approach for classification with GPs. This is a specific case of our problem, as the labels in classification are consistent, i.e., the labels assigned to the samples in the dataset, even if they are incorrect, do not change during

training. In our case, the user can respond to the same pairwise comparison query inconsistently depending on their noise model. Houlsby et al. [112] and Daniel et al. [80] proposed active GP fitting methods for classification and reward learning, respectively. While the latter focused on robotics tasks, they were not preference-based. Hence, they may be infeasible in many applications as it is difficult for humans to assign actual reward values.

In this section, we study a method for preference-based GP regression that learns from pairwise comparisons, and extend it with active query generation in Section 4.3. This method does not require the humans to assign actual reward values to the trajectories for fitting the GP.

3.2.2 Formulation

We again model the environment the robot is going to operate in as a discrete-time finite-horizon MDP. We use $s_t \in \mathcal{S}$ to denote the state and $a_t \in \mathcal{A}$ for the action (control inputs) at time t. A trajectory $\xi \in \Xi$ within this MDP is a sequence that consists of the state and actions: $\xi = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$, where T is the finite time horizon.

We assume a reward function over trajectories, $R : \Xi \to \mathbb{R}$, that encodes the human user's preferences about how they want the system to operate.

We assume the reward function R can be formulated as $R(\xi) = f(\Phi(\xi))$, where $\Phi : \Xi \to \mathbb{R}^d$ defines a set of trajectory features, e.g. average speed and minimum distance to the closest car in a driving trajectory. However, we emphasize that this formulation of R enables a more general form of functions that does not require strong assumptions – such as linearity in the features – which is commonly put in place when learning reward functions (as in [1, 155, 226]). We use a GP to model f, which allows us to avoid strong assumptions about the form of f.[2]

Our goal is to learn this more general form of reward functions using preference data in the form of pairwise comparisons. The robot will demonstrate a query Q consisting of two trajectories, ξ_1 and ξ_2 as shown in Figure 3.3 with blue and green curves, to the user, and will ask which trajectory they prefer. The user will respond to this query based on their preferences. The user's response provides useful information about the underlying preference reward function R. Of course, we cannot assume human responses are perfect for every query, so we model the noise in their responses using the commonly adopted probit model (an alternative to the human choice model we presented in Equation (3.10)), which assumes humans make a binary decision between the two trajectories noisily based on the cumulative distribution function (cdf) of the difference between the two reward values:

$$P(q = \xi_1 \mid Q = \{\xi_1, \xi_2\}) = P\left(R(\xi_1) - R(\xi_2) > \epsilon\right), \tag{3.11}$$

where $q \in Q$ denotes the user's choice, and $\epsilon \sim \mathcal{N}(0, 2\sigma_C^2)$ for some standard deviation $\sigma_C\sqrt{2}$.

[2]Due to computation reasons, we assume d is small. Compared to previous works which assume R to be linear in features or a small dimensional parameter space for parametric reward functions, this is a very mild assumption.

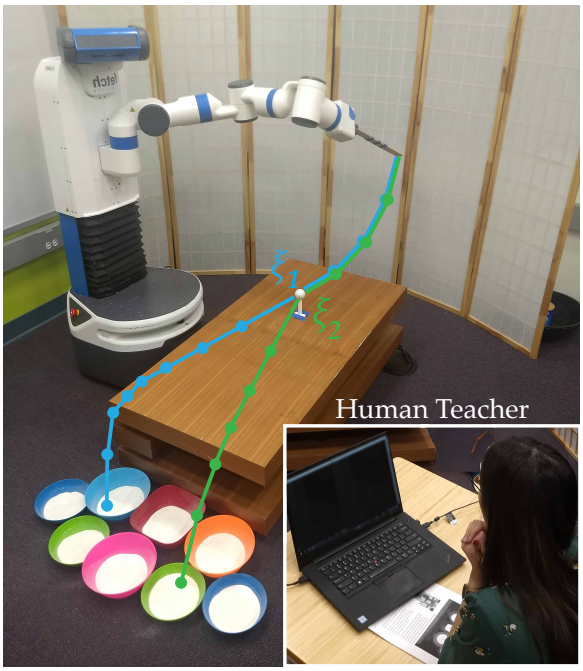


Figure 3.3: The user is trying to teach the robot how to play a variant of mini-golf, where the reward differs among eight targets. In preference-based learning, instead of trying to design a reward function by hand or controlling the robot to provide demonstrations, the user simply compares two demonstrated trajectories on the robot. Here, ξ_1 and ξ_2 demonstrate two trajectories that correspond to hitting the ball towards the blue or green targets.

Therefore, equivalently:

$$P(q = \xi_1 \mid Q = \{\xi_1, \xi_2\}) = \Phi\left(\frac{R(\xi_1) - R(\xi_2)}{\sqrt{2}\sigma_C}\right), \tag{3.12}$$

where Φ is the cumulative distribution function of the standard normal. This is an example of a Thurstonian model [149].

Having defined the problem setting, we are now ready to present our method for learning expressive reward functions using GPs.

3.2.3 Our Approach

In this section, we first give some background information about Gaussian Processes. We then introduce preference-based GP regression, where we show how to update a GP with the results of pairwise comparisons. We will present our approach to active preference query generation in Section 4.3, where we discuss how to find the most informative query that accelerates the learning.[3]

[3]We make our Python code for preference-based GP regression and active query generation publicly available at `https://github.com/Stanford-ILIAD/active-preference-based-gpr`.

To simplify the notation, we replace $\Phi(\xi)$ with Φ, with superscripts and subscripts when needed in this subsection and the related appendices.

Gaussian Processes

We start by introducing the necessary background on GPs for our work. We refer the readers to [166] for other uses of GPs in machine learning.

Suppose we have a finite trajectory space $\Xi = \{\Phi^{(i)}\}_{i=1}^{|\Xi|}$, where $\Phi^{(i)} \in \mathbb{R}^d$ is the features of the i^{th} trajectory according to some arbitrary indexing in Ξ. We employ a probabilistic point of view for f by modeling it using a GP, which enables us to model nonlinearities and uncertainties well without introducing parameters. We have:

$$P(\mathbf{f} \mid \boldsymbol{\mu}, \mathbf{K}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{f}-\boldsymbol{\mu})^\top \mathbf{K}^{-1}(\mathbf{f}-\boldsymbol{\mu})\right)}{(2\pi)^{|\Xi|/2}|\mathbf{K}|^{1/2}}, \tag{3.13}$$

where $\mathbf{f} = [f(\Phi^{(1)}), f(\Phi^{(2)}), \ldots, f(\Phi^{(|\Xi|)})]^\top$, $\boldsymbol{\mu}$ and $\mathbf{K}$ are the mean vector and the covariance matrix of the GP distribution for the $|\Xi|$ items in the dataset. Put it in another way, $\mathbf{f}$ follows a multivariate distribution. The covariance matrix depends on the used kernel. In this work, we use a variant of radial basis function (RBF) kernel with hyperparameter ϑ:

$$k(\Phi^{(i)}, \Phi^{(j)}) = \exp\left(-\vartheta\|\Phi^{(i)} - \Phi^{(j)}\|_2^2\right) - \bar{k}(\Phi^{(i)}, \Phi^{(j)}),$$
$$\bar{k}(\Phi^{(i)}, \Phi^{(j)}) = \exp\left(-\vartheta\|\Psi^{(i)} - \bar{\Phi}\|_2^2 - \vartheta\|\Phi^{(j)} - \bar{\Phi}\|_2^2\right),$$

where $\tilde{\Phi} \in \mathbb{R}^d$ is an arbitrary point for which we assume $f(\tilde{\Phi}) = 0$. This helps in practice because the query responses only depend on the relative difference between the two reward function values at the trajectories, i.e., $f(\Phi) + \epsilon$ for any $\epsilon \in \mathbb{R}$ would have the same likelihood for a dataset as $f(\Phi)$. By setting $f(\tilde{\Phi}) = 0$ for some arbitrary $\tilde{\Phi} \in \mathbb{R}^d$, we dissolve this ambiguity. It does not introduce an assumption because for any function f' and for any point $\tilde{\Phi}$, one can define $f(\Phi) = f'(\Phi) - f'(\tilde{\Phi})$ without loss of generality—both f' and f will encode the same preferences. Finally, this variant of the RBF kernel is still positive semi-definite, because it is equivalent to the covariance kernel of a GP which is initialized with an initial data point and a standard RBF kernel prior.

Preference-based GP Regression

Although the previous subsection was needed to explain how GPs work, we only focus on preference-based learning without any demonstrations in this section. In preference-based learning with pairwise comparisons, we have a dataset $\mathcal{D}_C = (Q^{(i)}, q^{(i)})_{i=1}^{|\mathcal{D}_C|} = ((\Phi_1^{(i)}, \Phi_2^{(i)}), q^{(i)})_{i=1}^{|\mathcal{D}_C|}$, consisting of pairs of trajectory features $\Phi_1^{(i)}, \Phi_2^{(i)} \in \mathbb{R}^d$, and user responses $\mathbf{q} = (q^{(i)})_{i=1}^{|\mathcal{D}_C|}$, where $q^{(i)} \in \{Q_1^{(i)}, Q_2^{(i)}\}$ indicates whether the user preferred $\Phi_1^{(i)}$ or $\Phi_2^{(i)}$. Accordingly, $\mathbf{K}$ is now a $2|\mathcal{D}_C| \times 2|\mathcal{D}_C|$ matrix,

whose rows and columns correspond to $\Phi_j^{(i)}, \forall i \in \{1, \ldots, |\mathcal{D}_C|\}, \forall j \in \{1, 2\}$. Similarly, $\boldsymbol{\mu}$ is a $2|\mathcal{D}_C|$-vector. Using a Bayesian approach to update the GP with new preference data $(Q^{(i)}, q^{(i)})$ gives:

$$\begin{aligned} P(f \mid \boldsymbol{\mu}, \mathbf{K}, Q^{(i)}, q^{(i)}) &\propto P(q^{(i)} \mid f, \boldsymbol{\mu}, \mathbf{K}, Q^{(i)}) P(f \mid \boldsymbol{\mu}, \mathbf{K}, Q^{(i)}) \\ &= P(q^{(i)} \mid f, Q^{(i)}) P(f \mid \boldsymbol{\mu}, \mathbf{K}). \end{aligned} \tag{3.14}$$

Here, the first term is just the probabilistic human response model given in Equation (3.12), and the second term is Equation (3.13). However, this posterior does not follow a GP distribution similar to Equation (3.13), and becomes analytically intractable [116].

Prior works have shown it is possible to perform some approximation such that the posterior is another GP [116, 166]. The most common approximation techniques are:

1. Laplace approximation, where the idea is to model the posterior as a GP such that the mode of the likelihood is treated as the posterior mean, and a second-order Taylor approximation around the maximum of the log-likelihood gives the posterior covariance. This technique is computationally very fast.

2. Expectation Propagation (EP), where the idea is to approximate each factor of the product by a Gaussian. EP is an iterative method that processes each factor iteratively to update the distribution to minimize its KL-divergence with the true posterior. While it is more accurate than Laplace approximation, it is slower in practice [156].

In this section, we use the former for its computational efficiency. Hence, we now show how to compute the quantities for Laplace approximation, i.e., posterior mean and covariance.

Finding the posterior mean. We use the mode of the posterior as an approximation to the posterior mean:

$$\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} \left(\log \left(P(\mathbf{q} \mid \mathbf{Q}, \mathbf{f}) \right) + \log \left(P(\mathbf{f} \mid \mathbf{Q}, \boldsymbol{\mu}, \mathbf{K}) \right) \right) , \tag{3.15}$$

where $\mathbf{Q}$ denotes the queries that correspond to the responses $\mathbf{q}$. Because the preference data are conditionally independent, the first term can be written as:

$$\begin{aligned} \log \left(P(\mathbf{q} \mid \mathbf{Q}, \mathbf{f}) \right) &= \sum_{i=1}^{|\mathcal{D}_C|} \log P(q^{(i)} \mid Q^{(i)}, \mathbf{f}) \\ &= \sum_{i=1}^{|\mathcal{D}_C|} \log \Phi \left(\frac{f(\Phi_1^{(i)}) - f(\Phi_2^{(i)})}{\sqrt{2}\sigma_C} \right) \end{aligned}$$

due to Equation (3.12). Adopting a zero-mean prior for f, we can write the second term of the

optimization (3.15) as:

$$\log\left(P(\mathbf{f} \mid \boldsymbol{\Phi}), \boldsymbol{\mu}, \mathbf{K}\right) = -\frac{1}{2}\log|\mathbf{K}| - |\mathcal{D}_C|\log 2\pi - \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}$$

Armed with these two expressions, we can now optimize the log-likelihood and thus find the mode of it to approximate the posterior mean.

Finding the posterior covariance matrix. Following [166], and omitting the derivation details for brevity, the posterior covariance matrix is $\hat{\mathbf{K}} = (\mathbf{K}^{-1} + \mathbf{W})^{-1}$ where $\mathbf{W}$ is the negative Hessian of the log-likelihood:

$$\mathbf{W}_{ij} = -\frac{\partial^2 \log\left(P(\mathbf{q} \mid \mathbf{Q}, \mathbf{f})\right)}{\partial f^{(i)} \partial f^{(j)}}.$$

Now, we know how to approximate the posterior mean and the posterior covariance for the Laplace approximation of Equation (3.14). This allows us to model and update the reward with preference data using a GP.

We also want to perform inference from this approximated GP. Inference is not only useful for active query generation as we will show in Section 4.3, but it also enables us to compute the reward expectations and variances given a trajectory.

Inference. Given two points $\Phi_1^*, \Phi_2^* \in \mathbb{R}^d$, we want to be able to compute the expected mean rewards $\boldsymbol{\mu}^*$ and also the covariance matrix between those two points $\mathbf{K}^*$, both of which will be useful for active query generation in Section 4.3. These are given by:

$$\boldsymbol{\mu}^* = \mathbb{E}\left[\mathbf{f}^* \mid \mathbf{Q}, \mathbf{q}, \Phi_1^*, \Phi_2^*\right] = {k^*}^\top \mathbf{K}^{-1}\mathbf{f}, \tag{3.16}$$

where k^* is a $2 \times 2|\mathcal{D}_C|$ matrix whose i^{th} row consists of $k(\Phi_i^*, \Phi_1^{(j)})$ and $k(\Phi_i^*, \Phi_2^{(j)})$ values for $j \in \{1, \ldots, |\mathcal{D}_C|\}$, and

$$\mathbf{K}_* = \mathbf{K_0} - k_*\left(\boldsymbol{I}_{2|\mathcal{D}_C|} + \mathbf{W}\mathbf{K}\right)^{-1}\mathbf{W}k_*^\top, \tag{3.17}$$

where $\mathbf{K_0}_{ij} = k\left(\Phi_i^*, \Phi_j^*\right)$ is a 2×2 matrix, $\boldsymbol{I}_{2|\mathcal{D}_C|}$ is the $2|\mathcal{D}_C| \times 2|\mathcal{D}_C|$ identity matrix.

Having a way to find the posterior mean and covariance as well as to perform inference means we now know how to learn a reward function modeled using a GP. In practice, the posterior mean can be used as a point estimate of the reward function, and the posterior covariance is useful for modeling the uncertainty over rewards. In the next section, we incorporate ordinal feedback on top of pairwise comparisons (as in [74]), which also enables us to define a *region of avoidance* for safety-critical applications.

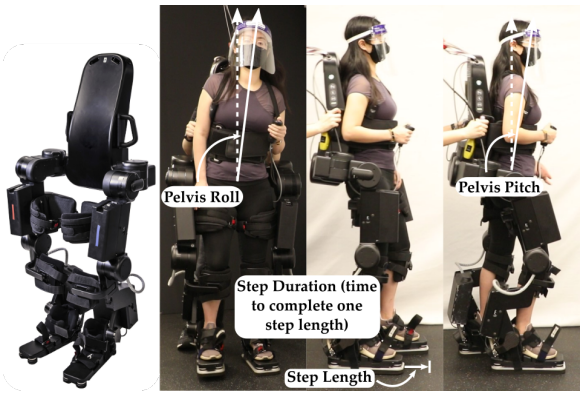


Figure 3.4: The Atalante exoskeleton, designed by Wandercraft, has 12 actuated joints, 6 on each leg. The experiments explore four gait parameters: step length, step duration, pelvis roll, and pelvis pitch.

3.3 Incorporating Ordinal Feedback along with Comparisons

Learning from comparative feedback naturally involves demonstrating some suboptimal trajectories to the human expert. In some cases, this might be problematic. For example, suppose we are trying to learn optimal gait parameters of a lower body exoskeleton (see Figure 3.4) where each gait corresponds to a trajectory. The human user who will give the comparison feedback will wear this exoskeleton and make comparisons about how comfortable they feel. Asking them comparison questions that involve highly suboptimal trajectories will cause them to feel uncomfortable and/or unsafe. In practice, we need to avoid this as much as possible.

Although we only focus on learning from offline datasets in this chapter and defer active comparison data collection until Chapter 4, we now present how we can define such regions in the trajectory space Ξ to avoid by utilizing ordinal feedback (in addition to pairwise comparisons) from humans.

More formally in this section, we denote this region of undesirable trajectories as the "Region of Avoidance" (ROA) and the region of remaining trajectories as the "Region of Interest" (ROI). In prior work on the highly-related area of safe exploration [185, 174, 30, 187], unsafe trajectories (or actions, or parameters) are considered to be catastrophically bad and therefore must be avoided completely. However, the resulting algorithms can be overly conservative in settings such as ours, where occasionally sampling from bad regions is tolerable.

This section, together with Section 4.4, proposes the Region of Interest Active Learning (ROIAL) algorithm, a novel active learning framework which queries the user for qualitative (ordinal) or preference (comparative) feedback to locate the ROI and estimate the reward function as accurately as possible over the ROI. In this section, we describe the learning algorithm, and Section 4.4 focuses on active querying.

The vast majority of prior work on preference learning obtains at most 1 bit of information per pairwise comparison query [112, 193, 192, 216, 91, 190, 207, 161, 186, 29]. ROIAL additionally

learns from ordinal labels [74], which assign trajectories to discrete ordered categories such as "bad," "neutral," and "good." Ordinal feedback enables ROIAL to both: 1) locate the ROI by learning the boundary between the least-preferred category (ROA) and remaining trajectories (ROI), and 2) estimate the reward function more efficiently within the ROI. Compared to the 1 bit of information obtained per pairwise comparison query, each ordinal query yields up to $\log_2(|\mathcal{B}^o|)$ bits of information where $\mathcal{B}^o$ is the set of ordinal labels. Since ordinal feedback is identical for trajectories within each ordinal category, pairwise comparisons provide finer-grained information about the reward function's shape within the categories.

We describe the learning algorithm that performs GP regression using both pairwise comparisons and ordinal feedback, and learns the ROA in the subsequent subsections. In Section 4.4, we extend it with active query generation to complete the description of the ROIAL algorithm, and validate it both in simulation and experimentally using the aforementioned lower-body exoskeleton task.

3.3.1 Formulation

We again consider a learning problem over a finite (but potentially-large) trajectory space Ξ, with a trajectory feature function $\Phi : \Xi \to \mathbb{R}^d$. Each trajectory $\xi \in \Xi$ is assumed to have an underlying reward to the user, $R(\xi) = f(\Phi(\xi))$. The algorithm aims to learn the unknown reward function $R : \Xi \to \mathbb{R}$. The trajectories' rewards can be written in the vectorized form $\boldsymbol{f} := [f(\Phi(\xi^{(1)})), f(\Phi(\xi^{(2)})), ..., f(\Phi(\xi^{(|\Xi|)}))]^\top$, where $\{\xi^{(j)} \mid j = 1, \ldots, |\Xi|\}$ are the trajectories in Ξ.

Specific to this section, we slightly change the comparison dataset structure: instead of having pairwise comparisons between two arbitrary trajectories, we assume the human user experiences (or watches) trajectories one by one and compares each trajectory to the previous one. This choice is made as it is more natural and time-efficient for the lower-body exoskeleton task we mentioned. However mathematically, this does not change anything: the learning algorithm could still handle pairwise comparison datasets with arbitrary trajectories as in the previous sections.

Accordingly, we let $\xi^{(i)} \in \Xi$ be the trajectory selected in trial i. We receive qualitative information about f after each trial i, consisting of an ordinal label $q_o^{(i)}$ and a comparison between $\xi^{(i)}$ and $\xi^{(i-1)}$ for $i \geq 2$. We use $\xi^{(i1)} \succ \xi^{(i2)}$ to denote a preference for trajectory $\xi^{(i1)}$ over $\xi^{(i2)}$, and following each trial i, collect these pairwise comparisons into a dataset $\mathcal{D}_C^{(i)} = \{\xi^{(i1)} \succ \xi^{(i2)} \mid i = 1, 2, ..., |\mathcal{D}_C|\}$. The ordinal labels are similarly collected into $\mathcal{D}_O^{(i)} = \{(\xi^{(i)}, q_o^{(i)}) \mid i = 1, 2, ..., |\mathcal{D}_O|\}$. Again assuming no expert demonstrations in this section, the full user feedback dataset after iteration i is defined as $\mathcal{D}^{(i)} := \mathcal{D}_C^{(i)} \cup \mathcal{D}_O^{(i)}$.

Ordinal feedback assigns one of pre-determined ordered labels to each sampled action. These (possibly noisy) labels are assumed to reflect ground truth ordinal categories (e.g., "bad," "neutral," "good," etc.), which partition Ξ into the sets that correspond to each ordinal label. We define the region of avoidance (ROA) as the trajectories that would fall into the set of lowest ordinal label. For

instance, in the lower-body exoskeleton setting, it consists of gaits that make the user feel unsafe or uncomfortable. Similarly, the ROA could be defined as the union of multiple sets that correspond to the bottom ordinal labels. We define the region of interest (ROI) as the complement of the ROA, i.e., $\Xi \setminus$ ROA.

3.3.2 Learning Algorithm

This subsection describes the learning algorithm, which leverages qualitative human feedback to estimate the ROI and reward function (code available at `https://github.com/kli58/ROIAL`). We first discuss Bayesian modeling of the reward function, and then explain the procedure for rendering it tractable in high dimensions. We then detail the process for estimating the ROI.

Bayesian Posterior Inference

To simplify notation, this section omits the iteration i from all quantities. Given the feedback dataset $\mathcal{D} = \mathcal{D}_C \cup \mathcal{D}_O$, the utilities $\boldsymbol{f}$ have posterior:

$$P(\boldsymbol{f} \mid \mathcal{D}_C, \mathcal{D}_O) \propto P(\mathcal{D}_C \mid \boldsymbol{f}) P(\mathcal{D}_O \mid \boldsymbol{f}) P(\boldsymbol{f}), \tag{3.18}$$

where $P(\boldsymbol{f})$ is a Gaussian prior over the utilities $\boldsymbol{f}$:

$$P(\boldsymbol{f}) = \frac{1}{(2\pi)^{|\Xi|/2} |\mathbf{K}|^{1/2}} \exp\left(-\frac{1}{2} \boldsymbol{f}^\top \mathbf{K}^{-1} \boldsymbol{f}\right),$$

in which $\mathbf{K} \in \mathbb{R}^{|\Xi| \times |\Xi|}$, $\mathbf{K}_{jj'} = k(\Phi(\xi^{(j)}), \Phi(\xi^{(j')}))$, and k is a kernel of choice. This section and Section 4.4 use the squared exponential kernel.

Comparison feedback. We assume that the users' preferences are corrupted by noise as in [73], such that:

$$P(\xi^{(1)} \succ \xi^{(2)} \mid \boldsymbol{f}) = g_C\left(\frac{f(\Phi(\xi^{(1)})) - f(\Phi(\xi^{(2)}))}{\sigma_C}\right), \tag{3.19}$$

where $g_C : \mathbb{R} \to (0, 1)$ is a monotonically-increasing link function, and $\sigma_C > 0$ quantifies noisiness in the comparisons. Note that this is a generalized version of the Thurstonian model we used in Equation (3.12).

Ordinal feedback. We define set of thresholds B^o such that $-\infty = B_0^o < B_1^o < B_2^o < \ldots < B_{|\mathcal{B}^o|}^o = \infty$. These thresholds partition the trajectory space into $|\mathcal{B}^o|$ ordinal categories $\mathcal{B}_1^o, \mathcal{B}_2^o, \ldots, \mathcal{B}_{|\mathcal{B}^o|}^o$. For any $\xi \in \Xi$, if $f(\Phi(\xi)) < B_1^o$, then $\xi \in \mathcal{B}_1^o$, and ξ has an ordinal label of 1. Similarly, if $B_j^o \leq f(\Phi(\xi)) < B_{j+1}^o$, then $\xi \in \mathcal{B}_{j+1}^o$, and ξ corresponds to an ordinal label of $j + 1$. We assume

that the users' ordinal labels are corrupted by noise as in [74], such that:

$$P(q_o \mid \boldsymbol{f}, \xi) = g_O\left(\frac{B^o_{q_o} - f(\Phi(\xi))}{\sigma_O}\right) - g_O\left(\frac{B^o_{q_o-1} - f(\Phi(\xi))}{\sigma_O}\right), \tag{3.20}$$

where $g_O : \mathbb{R} \to (0,1)$ is a monotonically-increasing link function, and $\sigma_O > 0$ quantifies the ordinal noise.

Assuming conditional independence of queries, the likelihoods $P(\mathcal{D}_C \mid \boldsymbol{f})$ and $P(\mathcal{D}_O \mid \boldsymbol{f})$ are:

$$P(\mathcal{D}_C \mid \boldsymbol{f}) = \prod_{i=1}^{|\mathcal{D}_C|} g_C\left(\frac{f(\Phi(\xi^{(i1)})) - f(\Phi(\xi^{(i2)}))}{\sigma_C}\right),$$
$$P(\mathcal{D}_O \mid \boldsymbol{f}) = \prod_{i=1}^{|\mathcal{D}_O|} \left[g_O\left(\frac{B^o_{q_o^{(i)}} - f(\Phi(\xi^{(i)}))}{\sigma_O}\right) - g_O\left(\frac{B^o_{q_O^{(i-1)}} - f(\Phi(\xi^{(i)}))}{\sigma_O}\right)\right].$$

Our simulations and experiments in Section 4.4.2 fix the hyperparameters σ_C, σ_O, and $\{B^o_j \mid j = 1, \ldots, |\mathcal{B}^o| - 1\}$ in advance. One could also estimate them during learning using strategies such as evidence maximization, but this can be computationally very expensive, especially with large trajectory spaces.

Common choices of link function (g_C and g_O) include the Gaussian cumulative distribution function [73, 74] and the sigmoid function, $g(x) = (1 + e^{-x})^{-1}$ [192]. We model feedback via the sigmoid link function because empirical results suggest that a heavier-tailed noise distribution improves performance. We use the Laplace approximation to approximate the posterior as Gaussian as in Section 3.2: $P(\boldsymbol{f} \mid \mathcal{D}^{(i)}) \approx \mathcal{N}(\hat{\boldsymbol{f}}^{(i)}, \hat{\mathbf{K}}^{(i)})$ [209].

High-Dimensional Tractability

Calculating the model posterior is the algorithm's most computationally expensive step, and is intractable for large trajectory spaces. Most existing work in high-dimensional Gaussian process learning requires quantitative feedback [118, 200]. Previous work in preference-based high-dimensional Gaussian process learning [192] restricts posterior inference to one-dimensional subspaces. However, the approach in [192] is more amenable to the regret minimization problem because each one-dimensional subspace is biased toward regions of high posterior reward. Instead, to increase the online computing speed over high-dimensional spaces, in each iteration i we select a subset $\Xi_S^{(i)} \subset \Xi$ of trajectories uniformly at random, and evaluate the posterior only over $\Xi_S^{(i)}$.

Estimating the Region of Interest

Since we lack prior knowledge about the ROI, it must be estimated during the learning process. In each iteration i, we model the ROI as the following set of trajectories: $\{\xi \in \Xi \mid \hat{f}^{(i-1)}(\Phi(\xi)) + \varepsilon\hat{\sigma}^{(i-1)}(\xi) > B^o_1\}$, where $\hat{\sigma}^{(i-1)}(\xi)$ is the posterior standard deviation associated with ξ. The variable

ε is a user-defined hyperparameter that determines the algorithm's conservatism in estimating the ROI; positive ε's are optimistic, while negative ε's are more conservative in avoiding the ROA. In practice, we evaluate trajectories in the randomly-selected subset $\Xi_S^{(i)}$ and define $\Xi_{\text{ROI}}^{(i)} = \{\xi \in \Xi_S^{(i)} \mid \hat{f}^{(i-1)}(\Phi(\xi)) + \varepsilon\hat{\sigma}^{(i-1)}(\xi) > B_1^o\}$ in each iteration i. Note that this definition is optimistic, whereas safe exploration approaches use pessimistic definitions [185, 174, 30, 187].

Summary

In Section 3.2, we studied a GP regression method that uses pairwise comparisons. In this section, we extended it with ordinal feedback, and defined region of avoidance (ROA) and region of interest (ROI) based on the ordinal categories. Together, these two sections present a computational method of learning non-parametric reward functions from pairwise comparisons and ordinal feedback. We will extend these methods with active querying in Chapter 4 and present experiment results that include the lower-body exoskeleton task we mentioned in the beginning of this section. The subsequent sections in this chapter goes back to parametric reward functions and focuses on reward learning algorithms that make use of other forms of comparative feedback. Since the Bayesian learning approach is maintained, they can be easily extended to non-parametric reward functions with GPs via Laplace approximation as long as the reward function is stationary and unimodal.[4]

3.4 More Expressive Feedback: Scale Questions

In Section 3.1, we introduced best-of-many choice queries and presented a Bayesian approach for learning parametric reward functions using them along with expert demonstrations. In Sections 3.2 and 3.3 we focused on a special case of best-of-many choice queries where the user is presented with only 2 options, making it a pairwise comparison question between the options. Using this special case, we showed how we can learn non-parametric reward functions by modeling them as Gaussian processes, optionally along with ordinal feedback.

Pairwise comparisons, or best-of-many choices in general, although simple to collect, are limiting in a number of ways. Consider the example shown in Figure 3.5, where a robot is tasked to serve a drink to a customer. The customer might have different preferences over the type of drink to have (milk, orange juice, or water), or the specifics of the trajectory the robot takes (e.g., if it goes over the stove or around it which can affect the temperature of the drink or the likelihood of the robot accidentally hitting the pan handle). A strict pairwise comparison between two trajectories, although minimizing interface complexity and mental effort for the user, does not really capture these intricacies of human preferences. In addition, when the user is indifferent towards both options,

[4] In fact, best-of-many choice queries as presented in Section 3.1 could also be used for GP regression. However, we intentionally focused on pairwise comparisons as we adopt them later when we introduce active query generation in Sections 4.3 and 4.4.

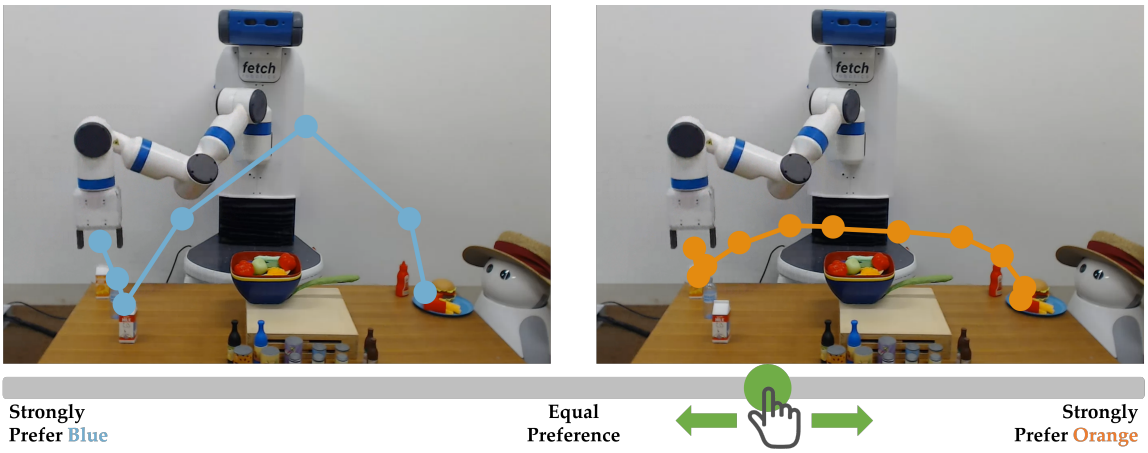


Figure 3.5: Scale feedback allows users to provide finely detailed comparisons between different options.

learning becomes difficult since users may become noisier in their responses. We thus need to have a more expressive way of collecting data from humans.

Several works, e.g., Holladay et al. [110] and Basu et al. [22], investigate modifications of learning from pairwise comparisons where users can also answer *About Equal* (which we also experiment with in Section 4.2). These two forms of pairwise comparison feedback are usually referred to as *strict* and *weak* pairwise comparisons. When the user chooses the neutral answer, the robot learns to assign about equal reward to the presented trajectories.

In the proposed scale feedback framework in this section, we take the weak pairwise comparisons approach one step further: Instead of three discrete values for feedback (prefer A, prefer B, neutral) users give quasi-continuous feedback. Our key insight is that allowing users to provide a scaled approach on a slider (as shown in Figure 3.5) can provide a more expressive medium for learning from humans and capture nuances in their preferences. This allows the user to indicate *how much* they prefer one option over the other.

Slider bars have been used in robotics for tuning parameters [163]. More related to our work, Cabi et al. [56] proposed using them for *reward sketching*. Instead of assigning a numerical preference between presented options, users continuously indicate the robot's progress towards some goal. However, this requires users to assign scores to different parts of trajectories. Developing the scale feedback for preference-based learning, we retain the ease of comparing trajectories.

To this end, we propose *scale feedback* as a new mode of interaction: Instead of a strict question on which of the two proposed trajectories the user prefers, we allow for more nuanced feedback using a slider bar. We design a Gaussian model for how users provide scale feedback, and learn a reward function capturing human preferences. Similar to Section 3.1 and prior work in robotics, we assume this reward is a parametric function of a set of trajectory features [1, 206, 159, 110], where the main task of learning from scale feedback is to recover the parameters of this reward function.

We demonstrate the performance benefit of scale feedback over pairwise comparisons in a driving simulation. Further, we investigate its practicality in two user studies with the real robot experiment shown in Figure 3.5. Our results suggest scale feedback leads to significant improvements in learning performance. We present these simulation and experiment results in Chapter 4 after we develop the active querying methods for scale feedback in Section 4.5.

3.4.1 Formulation

We now introduce the notation we use in this section and formulate the learning from scale feedback problem.

Reward function. We again consider the scenario where a robot needs to learn a reward function from a user, for example for customizing its behavior to the preferences of the user. We assume the user evaluates robot trajectories $\xi \in \Xi$ from a potentially infinite trajectory space Ξ based on a vector of features $\Phi(\xi) \in \mathbb{R}^d$. Similar to Section 3.1 and prior works in robotics [1, 206, 159, 110], we define a parametric reward function R that assigns a numerical value to any trajectory ξ:

$$R_w(\xi) = f_w(\Phi(\xi)) \,. \tag{3.21}$$

These features are usually provided by a domain expert incorporating the core factors that the reward needs to capture, e.g., collision with other objects, or distance to the goal.

Further, we assume in this section the robot has access to a motion planner that finds an optimal trajectory given reward function parameters, i.e., the planner is a (deterministic) function ρ where $\rho(w) = \arg\max_{\xi \in \Xi} R_w(\xi)$.

Regret. Similar to [207], we define the *regret* between any two parameter sets (w, w') as the difference in the reward w' assigns to the trajectories $\rho(w)$ and $\rho(w')$:

$$\mathcal{R}(w, w') = R_{w'}(\rho(w')) - R_{w'}(\rho(w)) \,, \tag{3.22}$$

which quantifies the suboptimality when the true weights are w', but the trajectory is optimized using w.

Learning. Let w^* denote the true weights for the reward function. These weights are not known to the robot; the only information initially available is a prior distribution $b^0 = P(w = w^*)$, which might be initialized using offline demonstrations as in Section 3.1. The robot learns w^* by iteratively presenting the user with two trajectories $Q^{(i)} = \left(\xi_1^{(i)}, \xi_2^{(i)}\right)$ for iterations $i \in \{1, 2, \ldots\}$. We extend the learning from pairwise comparisons framework, where users simply indicate the trajectory they prefer, to a setting where they instead provide a more finely detailed *scale feedback*.

Scale Feedback. Presented with two trajectories ξ_1 and ξ_2, the user returns numerical feedback $q \in [-1, 1]$. If $q = 0$, this means the user has no preference between the trajectories, $q = 1$ equals a strong preference for trajectory ξ_1 and $q = -1$ a strong preference for trajectory ξ_2.

From an interface design and expressiveness perspective, it is undesirable to have users give a numerical value for q. Instead, they can express such a feedback with a slider bar with a more fine-grained set of options. An example is illustrated in Figure 3.5. We let $\mathcal{D}_S = \{(Q^{(i)}, q^{(i)}\}_{i=1}^{|\mathcal{D}_S|} = \{(\xi_1^{(i)}, \xi_2^{(i)}, q^{(i)})\}_{i=1}^{|\mathcal{D}_S|}$ be the set of recorded scale feedback from the user.

Performance Measures. Let $\hat{w}$ be the robot's estimate of w^*. We consider two performance metrics. One is *alignment* of parameters [171, 38], $\texttt{Alignment} = \frac{\hat{w} \cdot w^*}{\|\hat{w}\| \cdot \|w^*\|}$, measuring the cosine similarity of vectors $\hat{w}$ and w^*, i.e., how well the parameters of the user's reward function are learned. Alternatively, Wilde et al. [207] proposed the relative error in *cost*. We adapt this as the $\texttt{Relative\_Reward} = \frac{R_{w^*}(\rho(\hat{w}))}{R_{w^*}(\rho(w^*))}$, measuring how much the user likes the trajectory optimized for $\hat{w}$ compared to the one optimized for w^*.

Problem Statement. Given a robot motion planner ρ and a user whose preferences come from the prior $b^0 = P(w = w^*)$, our goal in this section is to develop a learning model that maximizes either of the performance measures by performing inference of reward function parameters from scale feedback. Later in Section 4.5, we will develop an adaptive querying policy for querying the user with maximally informative scale feedback questions for some number of rounds.

3.4.2 Our Approach

We now briefly review learning from pairwise comparisons from a new perspective, and then extend the framework to scale feedback.

Pairwise Comparisons Feedback

When presented with two trajectories ξ_1 and ξ_2, a user returns an ordering $\xi_1 \succeq \xi_2$ (ξ_1 is preferred) or $\xi_1 \preceq \xi_2$ (ξ_2 is preferred). In a noiseless setting, we have

$$R_{w^*}(\xi_1) - R_{w^*}(\xi_2) \geq 0 \iff \xi_1 \succeq \xi_2 \,. \tag{3.23}$$

That is, the trajectory ξ_1 has a reward that is at least as high as that of ξ_2 with respect to the hidden true user weights w^*. Equation (3.23) already contains an observation model: If the user chooses trajectory ξ_1, the robot can infer that ξ_1 has a higher reward with respect to w^*. This inequality defines a subset in the parameter space: $\Lambda(\xi_1, \xi_2) = \{w \mid (R_w(\xi_1) - R_w(\xi_2)) \geq 0\}$ containing all weights that are *feasible* given the observed user choice. Over $|\mathcal{D}_C|$ iterations, we can intersect the subsets $\Lambda(\xi_1^{(1)}, \xi_2^{(1)}), \ldots, \Lambda(\xi_1^{(|\mathcal{D}_C|)}, \xi_2^{(|\mathcal{D}_C|)})$ to obtain the *feasible set* $\mathcal{F}^{(|\mathcal{D}_C|)}$. An example is shown in Figure 3.6a for a linear reward function.

Scale Feedback

Scale feedback allows the robot to gain more information: the robot can also infer by *how much* the user prefers ξ_1, allowing for learning tighter feasible sets. We extend the model in (3.23) and show

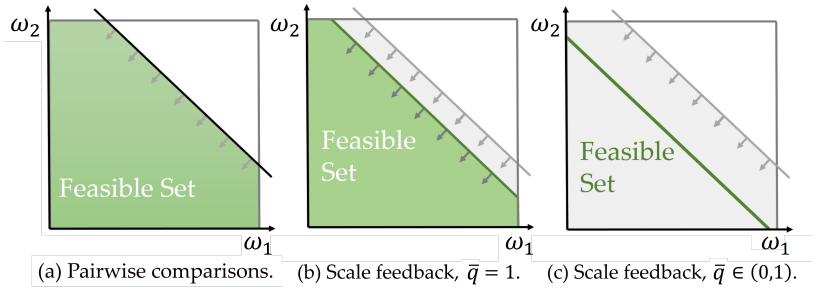


Figure 3.6: Different feasible sets learned from pairwise comparison and scale feedback under the linear reward model. Shown is the updated weight space (green) after observing user feedback for one (ξ_1, ξ_2) pair. If $\bar{q} = 1$, scale feedback enables us to learn a tighter half-space; when $\bar{q} \in (0, 1)$, scale feedback enables us to learn an equality, i.e., a hyperplane.

how a noiseless user would provide scale feedback and then study how a robot can learn from it.

Definition 1 (Maximum Reward Gap)**.** *Given true parameters w^* for a user, the maximum reward gap is*

$$\delta^* = \max_{\xi_1, \xi_2 \in \Xi} \left(R_{w^*}(\xi_1) - R_{w^*}(\xi_2) \right) . \tag{3.24}$$

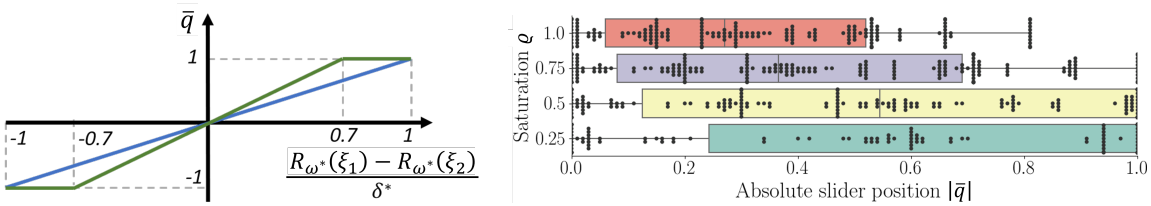
We notice that the maximum reward gap cannot be computed, since w^* is unknown to the robot. Nevertheless, we can formulate the user choice model and then derive an observation model.

User model. The maximum reward gap helps to define when a noiseless user would indicate a strong preference. We assume this occurs if and only if the difference in reward of ξ_1 and ξ_2 with respect to w^* is at least $\varrho^* \delta^*$ for some $0 < \varrho^* \leq 1$. Here ϱ^* is a saturation parameter which governs at what reward difference (w.r.t. to the maximum gap) the user's feedback gets saturated to a strong preference. For any other (ξ_1, ξ_2) where $|R_{w^*}(\xi_1) - R_{w^*}(\xi_2)| \in [0, \varrho^* \delta^*)$, we assume the user to linearly scale the noiseless response $\bar{q}$ between -1 and 1, which leads to the following model.

Definition 2 (Noiseless User Model)**.** *Presented with two trajectories ξ_1 and ξ_2, a noiseless user with the saturation parameter $\varrho^* \in (0, 1]$ will always provide the following feedback:*

$$\bar{q} = \begin{cases} 1 & \text{if } R_{w^*}(\xi_1) - R_{w^*}(\xi_2) \geq \varrho^* \delta^*, \\ -1 & \text{if } R_{w^*}(\xi_2) - R_{w^*}(\xi_1) \geq \varrho^{w^*} \delta^{w^*}, \\ \frac{R_{w^*}(\xi_1) - R_{w^*}(\xi_2)}{\varrho^* \delta^*} & \text{otherwise} . \end{cases} \tag{3.25}$$

We illustrate the noiseless user model in Figure 3.7a under different saturation parameters ϱ^*. In Figure 3.7b, we show a simulated example: for a fixed w^* we simulate how users with different values for ϱ^* would provide scale feedback to the same 20 queries. For larger ϱ^*, they position the



(a) User model for providing scale feedback with $\varrho^* = 1$ (blue) and $\varrho^* = 0.7$ (green).

(b) Example slider feedback for different ϱ. The boxplots indicate the four quartiles of the absolute slider values.

Figure 3.7: Noiseless user model.

slider closer to the neutral position. Finally, we derive an observation model for the noiseless user:

$$\begin{aligned} \bar{q} = -1 &\implies R_{w^*}(\xi_1) - R_{w^*}(\xi_2) \leq \bar{q}\varrho^*\delta^* \\ \bar{q} \in (-1, 1) &\implies R_{w^*}(\xi_1) - R_{w^*}(\xi_2) = \bar{q}\varrho^*\delta^*, \\ \bar{q} = 1 &\implies R_{w^*}(\xi_1) - R_{w^*}(\xi_2) \geq \bar{q}\varrho^*\delta^*. \end{aligned} \tag{3.26}$$

Figures 3.6b and 3.6c illustrate the resulting feasible sets from (3.26) for a linear reward model. Moreover, we notice the user-specific and unknown parameters ϱ^* and δ^* always appear as a product. Thus, this product can be seen as a single additional parameter, and the notion of feasible set $\mathcal{F}$ can be readily extended to this augmented parameter space.

Probabilistic User Feedback

In practice, users are often noisy; they might consider additional or slightly different features than the robot, not follow the parametric reward function, or simply be uncertain in some answers. Since we cannot expect users to always provide slider feedback following (3.25), we introduce a probabilistic model where we add uncertainty to the placement of the slider.

Another practical limitation is the fact that we cannot collect truly continuous feedback from the users. Instead, the slider bar has a step size $\nu \in (0, 1]$ such that the user provides feedback of the form $n\nu$ for $n \in \mathbb{Z}$ and $-\nu^{-1} \leq n \leq \nu^{-1}$. Note that $\nu \to 0$ retains the continuous scale feedback, whereas $\nu = 1$ gives the standard weak pairwise comparison model where the feedback is always in $\{-1, 0, 1\}$.

Definition 3 (Probabilistic User Model)**.** *Given a user w^* and a query (ξ_1, ξ_2), let $\bar{q}$ be the user feedback defined in the noiseless user model in* (3.25)*. A probabilistic user using a slider bar with a step size of ν then provides feedback*

$$q = \text{round}(\bar{q} + \epsilon, \nu) \tag{3.27}$$

where ϵ is a zero-mean Gaussian noise, i.e., $\epsilon \sim \mathcal{N}(0, \sigma_S^2)$ with standard deviation σ_S, and $\text{round}(x, \nu)$

outputs $n\nu$ closest to x such that $n \in \mathbb{Z} \cap [-\nu^{-1}, \nu^{-1}]$.

Probabilistic Observation Model. Given the probabilistic user model, we now show how a robot can infer about w^* from scale feedback. In the noiseless case, user feedback defines a feasible set. For the probabilistic case, we instead derive a distribution over w and ϱ. Let $\delta(w) = \max_{\xi_1,\xi_2 \in \Xi}(R_w(\xi_1) - R_w(\xi_2))$, similar to (3.24). Then for $0 < \varrho \leq 1$, the belief is defined

$$P(w, \varrho \mid \bar{q}, \xi_1, \xi_2) = \begin{cases} \tilde{P}(w, \varrho \mid \bar{q}, \xi_1, \xi_2) & \text{if } \bar{q} \in (-1, 1), \\ P^+(w, \varrho \mid \bar{q}, \xi_1, \xi_2) & \text{if } \bar{q} = 1, \\ P^-(w, \varrho \mid \bar{q}, \xi_1, \xi_2) & \text{if } \bar{q} = -1, \end{cases} \tag{3.28}$$

where

$$\tilde{P}(w, \varrho \mid \bar{q}, \xi_1, \xi_2) \propto \begin{cases} 1 \text{ if } R_w(\xi_1) - R_w(\xi_2) = \bar{q}\varrho\delta(w), \\ 0 \text{ otherwise .} \end{cases} \tag{3.29}$$

$$P^+(w, \varrho \mid \bar{q}, \xi_1, \xi_2) \propto \begin{cases} 1 \text{ if } R_w(\xi_1) - R_w(\xi_2) \geq \bar{q}\varrho\delta(w), \\ 0 \text{ otherwise .} \end{cases} \tag{3.30}$$

$$P^-(w, \varrho \mid \bar{q}, \xi_1, \xi_2) \propto \begin{cases} 1 \text{ if } R_w(\xi_1) - R_w(\xi_2) \leq \bar{q}\varrho\delta(w), \\ 0 \text{ otherwise.} \end{cases} \tag{3.31}$$

Given noisy user feedback q as in (3.27), we can define a probabilistic density function $P(\bar{q} \mid q)$. Together with (3.28) we derive a compound probability distribution

$$P(w, \varrho \mid q, \xi_1, \xi_2) = \int_{-1}^{1} P(w, \varrho \mid \bar{q}, \xi_1, \xi_2) P(\bar{q} \mid q) d\bar{q}. \tag{3.32}$$

where we can write $P(\bar{q} \mid q)$ for $\bar{q} \in [-1, 1]$ as

$$P(\bar{q} \mid q) \propto \begin{cases} \Phi\left(\frac{q - \bar{q} + \nu/2}{\sigma_S}\right) & \text{if } q = -1, \\ \Phi\left(\frac{\bar{q} - q + \nu/2}{\sigma_S}\right) - \Phi\left(\frac{\bar{q} - q - \nu/2}{\sigma_S}\right) & \text{if } q \in (-1, 1), \\ \Phi\left(\frac{\bar{q} - q + \nu/2}{\sigma_S}\right) & \text{if } q = 1, \end{cases} \tag{3.33}$$

and $P(\bar{q} \mid q) = 0$ for $\bar{q} \notin [-1, 1]$. Here, Φ denotes the cdf of the standard normal distribution. Finally, given a dataset $\mathcal{D}_S = \{(\xi_1^{(i)}, \xi_2^{(i)}, q^{(i)})\}_{i=1}^{|\mathcal{D}_S|}$ and some prior $b^0(w, \varrho)$, the joint posterior is

$$b^{|\mathcal{D}_S|}(w, \varrho) \propto b^0(w, \varrho) \prod_{i=1}^{|\mathcal{D}_S|} P(w, \varrho \mid q^{(i)}, \xi_1^{(i)}, \xi_2^{(i)}). \tag{3.34}$$

Here, we can factor $b^0(w, \varrho)$ as $P(w)P(\varrho)$ by assuming w and ϱ are independent and we also have a prior for ϱ^*. We can then take the expectation of the posterior $P(w, \varrho \mid \mathcal{D}_S)$ as a point estimate of the learned user model.

3.4.3 Algorithm Design

We now outline the learning algorithm. Over $|\mathcal{D}_S|$ iterations: (i) the robot generates a query $(\xi_1^{(i)}, \xi_2^{(i)})$ (active query generation for scale feedback will be presented in Section 4.5), (ii) the user provides scale feedback to the query in the form of the slider value $q^{(i)}$ (in the noiseless case, $q^{(i)} = \bar{q}^{(i)}$), and (iii) the robot updates its dataset and posterior using Equation (3.34). After iteration $|\mathcal{D}_S|$, the algorithm returns the expected parameters $\hat{w} = \mathbb{E}[w \mid \mathcal{D}_S]$.

Worst Case Error Bound

To compare scale feedback to pairwise comparisons, we establish a worst case bound on the performance measures for both frameworks. We introduce the *worst-case error* as the maximum negative performance measure, $1-\texttt{Alignment}(w, w^*)$ (the bounds also generalize to $1-\texttt{Relative\_Reward}(w, w^*)$, but we use only `Alignment` for brevity). The constant in front ensures a positive value, which we then discount with the posterior belief, given observations $\mathcal{D}_S$ (or $\mathcal{D}_C$ in the case of pairwise comparisons):

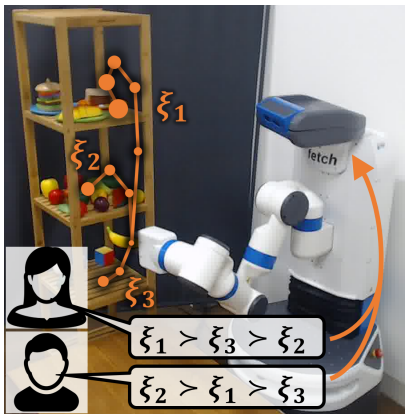
$$\texttt{Err}^{\max}(w^*, \mathcal{D}_S) = \max_w b^{|\mathcal{D}_S|}(w)(1 - \texttt{Alignment}(w, w^*)) \tag{3.35}$$

where $b^{|\mathcal{D}_S|}(w)$ is obtained by marginalizing the posterior $b^{|\mathcal{D}_S|}(w, \varrho)$ over ϱ. This describes the worst w the robot could pick, discounted by the posterior distribution learned from data $\mathcal{D}_S$. In the noiseless setting, this simplifies to $\max_{w \in \Lambda} 1 - \texttt{Alignment}(w, w^*)$ where Λ is the feasible set.

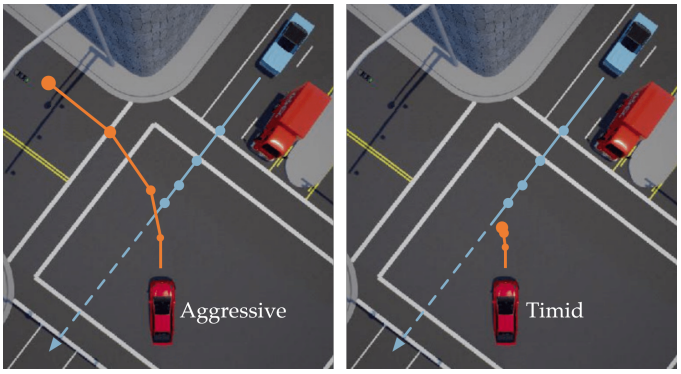
Proposition 1 (Upper error bound)**.** *Let $\mathcal{D}_S$ denote the dataset of scale feedback and $\mathcal{D}_C$ be the dataset of pairwise comparisons for the same set of queries (trajectory pairs). For any user weights w^*, it holds in the noiseless setting that $\texttt{Err}^{\max}(w^*, \mathcal{D}_S) \leq \texttt{Err}^{\max}(w^*, \mathcal{D}_C)$.*

The proof follows from the observation $\Lambda^{\texttt{Scale}} \subseteq \Lambda^{\texttt{Choice}}$, i.e., scale feedback removes more volume from the parameter space. Hence, the worst choice of an estimate $\hat{w}$ given observations is guaranteed to have a smaller worst-case error when using scale feedback. The full proof is in Appendix A.1.

We defer the simulation and user study results to Section 4.5 where we will also present an active querying approach for scale feedback. In the remainder of this chapter, we move to more complex settings where the reward functions to be learned are either multimodal or non-stationary.



(a) Fetch robot putting a banana on one of the three shelves. The two users have different preferences, and so they provide different rankings to the robot. The robot needs to be able to model multimodal reward functions for successfully achieving the task.



(b) The red car does not observe the blue car due to the occluding truck until it comes to the intersection. It is possible to avoid accident by **(left)** completing the turn aggressively or **(right)** making a hard-brake. An autonomous vehicle trying to learn from such mixture data must be able to model multimodal reward functions for safe and efficient driving.

Figure 3.8: Examples of why multimodal reward functions might be needed.

3.5 Learning Multimodal Rewards via Ranking Queries

Up to this point in the thesis, we focused on learning a *unimodal* reward function that models human preferences on a target task. However, this unimodality assumption does not always hold: human preferences are usually more complex and need to be captured via a multimodal representation. Further, even if the preferences of a human are truly unimodal, we often use a mixture of data from multiple humans, which can be difficult to disentangle, leading to multimodality.

As an example, consider a robot placing a banana on one of the three shelves (see Figure 3.8a). The middle shelf is often used for fruits, but it has no room left and if the robot tries to put the banana there, it may cause other fruits to fall. The top shelf has some space but it has been used for cooked meals. The bottom shelf has a lot of free space, but is usually used only for toys. In such a scenario, people may have very different preferences about what the robot should do. If we try to learn a unimodal reward using data collected from multiple people, the robot is likely to fail in the task, because the data will include inconsistent preferences.

As another example, consider the driving scenario presented in [59] (see Figure 3.8b), where the red car attempts to make an unprotected left turn, but fails to observe the blue car occluded behind a truck. An aggressive driver might accelerate and avoid the accident by completing the turn before the blue car reaches the collision point. Similarly, a timid driver would move slowly and brake sharply the moment it sees the blue car, which also prevents the accident. Even though both modes can avoid the accident, a driving policy learned by a mixture of this data is likely to fail while trying to comply with both modes. In fact, Cao et al. [59] demonstrated simply applying imitation

learning using such a mixture data fails in this case, and a separation of different modes is needed.

One solution is of course to label the different modes in the data. For example, one could separate the data based on the preferred shelf in the banana placing example, and learn different reward functions for each shelf. However, this separation is not always straightforward. For example in the driving example, it is unclear what should be labeled as aggressive or timid driving. Clustering the data based on the human who provided the data is also not viable as it will introduce data-inefficiency issues, and perhaps more importantly, humans are not always unimodal: a usually timid driver can drive more aggressively when in a hurry.

These examples motivate us to develop methods that can learn *multimodal* reward functions using datasets that are not specifically labeled with the modes. To this end, previous work proposed learning from demonstrations to learn multimodal policies [107, 86] or reward functions with multiple possible intentions [16, 165, 109]. However, learning from expert demonstrations is often extremely challenging in robotics as we discussed in earlier sections, e.g., providing demonstrations on a robot with high degrees of freedom is non-trivial [5], and humans have difficulty giving demonstrations that align with their preferences due to their cognitive biases [21, 131]. Thus, it is desirable to have methods that learn from other more reliable sources of data, e.g., pairwise comparisons of trajectories [51, 22].

While learning from pairwise comparisons provides a rich source of data for learning reward functions, the theoretical results by Zhao et al. [223] imply that extending the pairwise-comparison-based reward learning techniques to multimodal reward functions is not possible, i.e. failure cases can be constructed, where pairwise comparisons are not sufficient for identifying different modes of a multimodal function. Our insight is that it is possible to learn a multimodal reward function by going beyond pairwise comparisons and instead using *rankings*.

We want to emphasize this is a different problem from learning nonlinear rewards. Nonlinear reward functions allow users to have multiple sets of desired behavior: a user may prefer both aggressive and timid driving over a behavior that is in between which can cause an accident. However, the existing methods that handle nonlinearities assume unimodal human feedback, which means the user must have a consistent preference towards either of the modes. Therefore, either ($aggressive \succ timid \succ accident$) or ($timid \succ aggressive \succ accident$) is expected. In this work, we relax this assumption and learn multimodal reward functions without requiring consistent rankings in the dataset. Our framework allows both ($aggressive \succ timid \succ accident$) and ($timid \succ aggressive \succ accident$) to be in the dataset, and we recover both modes of the reward function.

To achieve this, we formulate multimodal reward learning as a mixture learning problem in this section, and use rankings from humans to learn the mixture.

Computational Models for Rankings

Before we move into formulation, we briefly review computational models for human rankings. In the previous sections, we introduced different examples of such models for best-of-many choice queries (Equation (3.10)), pairwise comparisons (Equations (3.10) and (3.12)), and scale queries (Equation (3.25)). In fact, our best-of-many choice model is known as the multinomial logits (MNL) model [67], and it has been widely used for human preferences in many fields [27, 211, 41, 26] including robotics [171, 34, 22]. Similarly, the model is known as the Bradley-Terry model for the special case of pairwise comparisons [66].

To extend these models to rankings, Plackett-Luce [150, 13] and Mallows models [144, 198, 145, 55] are commonly employed. In this section, we use the Plackett-Luce model as it is a natural extension of MNL, which we already used in Section 3.1. We formalize this model in Section 3.5.1.

Learning Mixture Models from Rankings.

Our approach to learning multimodal reward functions is through mixture models, where we assume the data come from different individual models with some unknown probabilities. Relatedly, previous works considered mixtures of MNLs [82, 70], Plackett-Luce models [223], and Mallows models [139]. Other works adopt different methods to model multimodality, such as by assuming latent state dynamics that transition between different modes [153] or by learning the different modes from labeled datasets [59, 162]. To avoid these modeling assumptions, we focus on directly learning the mixture model.

While Zhao et al. [223] have theoretically studied the mixture of Plackett-Luce choice models, which also informs our algorithm in terms of the query sizes, they only focus on learning the rewards of a discrete set of items. In this section, we deal with a continuous hypothesis space under a mixture of Plackett-Luce models.

3.5.1 Formulation

Setup. We again consider a fully-observable dynamical system. A trajectory ξ in this system is a series of states and actions, i.e., $\xi = (s_0, a_0, \ldots, s_T, a_T)$. The set of feasible trajectories is Ξ.

We assume there is a set of M individual reward functions that are possibly different, each of which encodes some preference between the trajectories in Ξ. For the rest of the formalism, we refer to each individual reward function as an *expert* for the clarity of the presentation.

Following the common assumption in reward learning [226, 43, 207], we assume each preference can be modeled as a parametric reward function over a known fixed feature space, so the reward associated with a trajectory ξ with respect to the m^{th} expert is $R_{w_m}(\xi) = f_{w_m}(\Phi(\xi))$, where w_m is the unknown vector of parameters. Across the expert population, there exists some unknown distribution, corresponding to the ratio of the data provided by the experts. We represent this

distribution with mixing coefficients α_m such that $\sum_{m=1}^{M} \alpha_m = 1$. We will then learn both the unknown reward functions $\{w_m\}_{m=1}^{M}$ and the mixing coefficients $\{\alpha_m\}_{m=1}^{M}$, using ranking queries made to the M experts.

Ranking Model. We define a *ranking query* to be a set of the form $Q = \{\xi_1, \ldots, \xi_{|Q|}\}$ for a fixed query size $|Q|$. The response to a ranking query is a ranking over the items contained therein, of the form $q = (\xi_{\iota_1}, \ldots, \xi_{\iota_{|Q|}})$, where ι_1 is the index of the expert's top choice, ι_2 is the second top choice, and so on. While it is not known which expert provided the response to the query, we know the prior that a response comes from expert m with some unknown probability α_m, i.e., $P(R = R_{w_m}) = \alpha_m$. Going back to our banana placing example, a ranking query of $|Q|$ robot trajectories is generated by the algorithm, and a user—whose identity is unknown to the algorithm—responds to this query.

We then capture how human experts respond to these ranking queries by modeling a ranking distribution through an iterative process using Luce's choice axiom [147]. In this process, the experts repeatedly select their top choice ι_1 with a probability distribution generated with the softmax rule to generate a ranking from the order items were selected:

$$P\left(q_1 = \xi_{\iota_1} \mid R = R_{w_m}, w_m\right) = \frac{\exp\left(\beta^C R_{w_m}(\xi_{\iota_1})\right)}{\sum_{j=1}^{|Q|} \exp\left(\beta^C R_{w_m}(\xi_{\iota_j})\right)} .$$

where β^C is the rationality coefficient for comparative feedback. In the following iterations, the experts select their top choice among the remaining trajectories:

$$P\left(q_j = \xi_{\iota_j} \mid q_1, \ldots, q_{j-1}, R = R_{w_m}, w_m\right) = \frac{\exp\left(\beta^C R_{w_m}(\xi_{\iota_j})\right)}{\sum_{j'=j}^{|Q|} \exp\left(\beta^C R_{w_m}(\xi_{\iota_{j'}})\right)} . \tag{3.36}$$

This is known as the Plackett-Luce ranking model [150, 13]. Together with the prior over experts α_m, the resulting distribution over rankings q is a mixture of Plackett-Luce models with mixing coefficients α_m and weights proportional to $\exp\left(R_{w_m}(\xi)\right)$.

Hence, the ranking distribution first selects the reward function R_{w_m} with probability α_m, and then selects trajectories from Q sequentially with probability proportional to the exponent of their reward, i.e., $\exp\left(R_{w_m}\right)$, among the remaining trajectories until none is left, generating a ranking of the trajectories.

So given knowledge of the true reward function weights w_m and mixing coefficients α_m, we have the following joint mass over responses q from a query Q:

$$P(q \mid Q, w, \alpha) = \sum_{m=1}^{M} \alpha_m \prod_{j=1}^{|Q|} \frac{\exp\left(\beta^C R_{w_m}(\xi_{\iota_j})\right)}{\sum_{j'=j}^{|Q|} \exp\left(\beta^C R_{w_m}(\xi_{\iota_{j'}})\right)} . \tag{3.37}$$

Objective. Our goal is to design a series of adaptive queries $Q^{(i)}$ to optimally learn the reward function parameters w_m and corresponding mixing coefficients α_m upon observing the query responses

$q^{(i)}$. We constrain all queries to consist of a fixed number of trajectories $|Q|$.

In the subsequent section, we present our learning framework. We defer the active query generation algorithm that makes the queries adaptive to Section 4.6 along with simulation and experiment results.

3.5.2 Our Approach

To learn the reward parameters w_m and mixing coefficients α_m, we again adopt a Bayesian learning approach. For this, we maintain a posterior over the parameters w_m and α_m. Given a dataset of rankings $\mathcal{D}_R$, this posterior can be written as

$$\begin{aligned} P(w, \alpha \mid \mathcal{D}_R) &= P(w, \alpha \mid Q^{(1)}, q^{(1)}, \ldots, Q^{(|\mathcal{D}_R|)}, q^{(|\mathcal{D}_R|)}) \\ &\propto P(w, \alpha) P(Q^{(1)}, q^{(1)}, \ldots, Q^{(|\mathcal{D}_R|)}, q^{(|\mathcal{D}_R|)} \mid w, \alpha) \\ &= P(w, \alpha) \prod_{i=1}^{|\mathcal{D}_R|} P(q^{(i)}, Q^{(i)} \mid w, \alpha, Q^{(1)}, q^{(1)}, \ldots, Q^{(i-1)}, q^{(i-1)}) \\ &\propto P(w, \alpha) \prod_{i=1}^{|\mathcal{D}_R|} P(q^{(i)} \mid w, \alpha, Q^{(i)}) \,, \end{aligned} \tag{3.38}$$

where we use the conditional independence of rankings $q^{(i)}$ given w, α and the conditional independence of the $Q^{(i)}$ on w, α given $Q^{(1)}, q^{(1)}, \ldots, Q^{(i-1)}, q^{(i-1)}$ in the last equation. To be able to compute this posterior, we assume some prior distribution over the reward parameters and the mixing coefficients, which is system-dependent and may come from domain knowledge, and use Equation (3.37) to calculate the likelihood terms. For example, in our simulations and user studies in Section 4.6, we adopted a Gaussian prior $w_m \sim \mathcal{N}(0, I)$ and a uniform prior over the unit $M - 1$ simplex for α. Learning this posterior distribution in Equation (3.38), one can compute a maximum likelihood estimate (MLE) or expectation as the predicted reward parameters and mixing coefficients.

Having presented the learning framework for multimodal rewards from rankings, we are now ready to proceed to the last section of Chapter 3, where we will relax the stationarity assumption of the reward function, assume a specific mode transition model, and develop a new query type (namely, hierarchical choice queries) for learning these nonstationary rewards.

3.6 Hierarchical Comparison Queries for Non-stationary Rewards

In the previous sections, we assumed a stationary reward function, which is not expressive enough to match human preferences in all environments. Real world is often non-stationary due to environment

complexity or changes in objectives in the environment. Surrounding agents continuously change their behavior which in turn requires the robot to adapt to these changes. For example in driving, people continuously adapt their reward functions in response to traffic complexity and behavior of other drivers. It is quite common for us to get impatient behind a slow driver and make drastic maneuvers different from our usual driving style. Here, we may weigh efficiency more than collision avoidance than we usually do.

As an important class of non-stationary environments, human-robot and robot-robot adaptation have recently attracted much attention, where the aim is to ensure robots adapt to their changing environments and other agents [158, 8]. In contrast, our goal in this section is to learn the reward functions that dynamically change depending on the interactions between the agents and the environment. We augment learning from comparative feedback to recover such multimodal reward functions.

Prior works theoretically investigated how to perform preference-based learning for multimodal reward functions [223, 139, 70], which we also discussed in Section 3.5. In this section, we relax this problem by assuming there is a structure between the modes, i.e., the mode from which the next comparison data will come can be estimated based on the current comparative feedback. Although we assume and explicitly model these transitions between modes, the problem is not necessarily easier, because we are also interested in learning how users' preferences transition between the modes.

Modeling behaviors in such environments is a well-studied problem especially for driving. For example, Dong et al. [83] characterize driving styles based on sensor data using deep learning. In a more related paper, Morton and Kochenderfer [153] modeled the drivers with a latent state space which can affect their driving behavior. While they stated these latent states might change over time, both of these works made the assumption that latent states remain unchanged over the trajectories of interest, so they did not address changing behaviors. Berndt et al. [31] modeled the latent states of the drivers using Hidden Markov Models (HMM) where they also allow adaptation. However, they did not specifically learn reward functions, and they focused on identifying the maneuvers the drivers will perform from a predefined database. With a similar objective, Kulic and Croft [130] used HMM for latent state estimation for human-robot interaction. We recently studied how trust of humans in a robot, a specific latent state, changes based on the performance in a task [49].

In this section, we propose to learn an expressive representation of preferences in non-stationary scenarios, where interactions and adaptations better reflect the real-world conditions. We assume that the non-stationary scenarios arise from changing behaviors of other agents interacting with our system, which in turn affect human preferences. We formalize the reward dynamics which encodes not only different human preferences but also *how* they change.

> *Our insight is that a dynamic reward model may match human preferences more accurately in a wide range of scenarios than a stationary reward function.*

We tailor best-of-many choice queries to capture longer term interactions between the robot and the surrounding agents, and develop a probabilistic model of user responses for any number of stationary reward functions and the transitions between them.

In this section, we make the following contributions:

Reward dynamics. User preferences may change based on the behaviors of other agents in the environment. We encode the momentary human preference by a static reward function and assume at any point of time the human has an internal *preference mode* (mood) which dictates what reward function the human will optimize next. We introduce the notion of *reward dynamics* as a tuple of reward functions and parameters governing transitions between those.

Hierarchical choice queries. We formalize the *hierarchical choice queries* as a sequence of best-of-many choice queries, each of which we call a *sub-query*. The sub-queries sequentially follow each other so that the user moods are reflected into their choices.

3.6.1 Formulation

Let us consider a robot that should match human preferences in an environment of interest that includes other agents (e.g. driving scenarios). These other agents can act differently at different times. For example, in the case of driving, some cars aggressively swerve through the traffic and others may follow a more cooperative strategy of allowing other cars to merge smoothly. Prior works on autonomous driving [170, 169, 172, 173, 18, 177, 89, 183] assume the robot should follow the same reward function over time in both of the above scenarios. We argue user preferences may vary in response to the changing behaviors of the environment agents in both driving and potentially other multi-agent environments. Our goal is to learn an expressive reward function corresponding to these dynamic preferences.

We model the environment as a fully-observable dynamical system. For driving, the continuous state of the system $s \in \mathcal{S}$ includes the positions and the velocities of the robot and the other agents. The state of the system changes based on actions of all the agents through the transition function $\mathcal{T}$, which we can now write as

$$s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_{\text{robot}t}, a_{\text{others}t}) \tag{3.39}$$

where $a_{\text{others}t}$ are the actions of the other agents at time step t, which affect the reward function and in turn the actions of the robot.[5] We define a finite trajectory $\xi \in \Xi$ as a sequence of continuous state-action pairs $\xi = (s_0, a_{\text{robot}0}, a_{\text{others}0}, \ldots, s_T, a_{\text{robot}T}, a_{\text{others}T})$ over a finite horizon T, and Ξ is the set of all feasible trajectories that satisfy the dynamics of the system.

Our goal is to learn human preferences for how the robot should behave in the presence of different

[5] More generally, the other agents in the environment can be thought of as part of the environment. All information regarding them, including their trajectories, can be encoded into the states. This will reduce the setup to the setup we used in earlier sections, e.g., Section 3.1

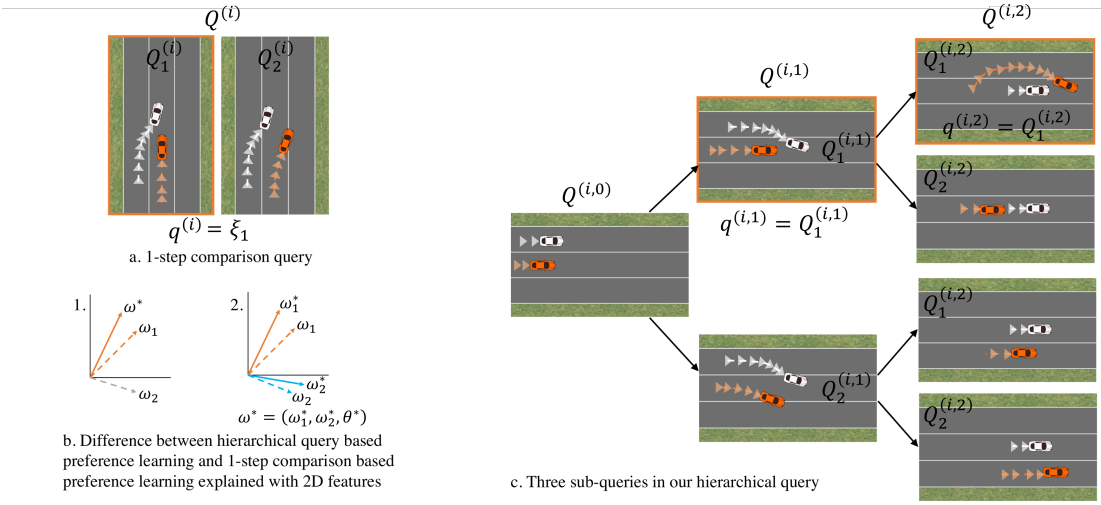


Figure 3.9: (a) 1-step comparison query. In any two iterations a user with bimodal preference may pick the trajectories optimal with respect to two different true weights w_1^* and w_2^*. (b) This ambiguity shows up as noise in 1-step comparison based learning where the goal is to learn a single reward function w^* (on the left). In reality the true preference function of the user changes between w_1^* and w_2^* depending on the environment, θ^* governs the transition. Our algorithm learns such a bimodal preference: w_1 close to w_1^* and a w_2 close to w_2^* (on the right). (c) Our proposed hierarchical query consists of 3 sub-queries. In iteration i of querying, $Q^{(i,0)}$ is a context sub-query, $Q^{(i,1)}$ is a comparison between two trajectories, each a different continuation of $Q^{(i,0)}$, and $Q^{(i,2)}$ continues the preferred trajectory from $Q^{(i,1)}$.

environment agents. We learn this reward function by making hierarchical comparison queries to the users.

3.6.2 Hierarchical Comparison Queries

Prior works learned static reward functions by asking people to compare between two different trajectories of robots. There, each query is a pair of short videos that demonstrate two trajectories of the system [171, 72]. Such short trajectories do not capture the nuances of interaction in a non-stationary multi-agent system. As an example, in a 1-step comparison query in Figure 3.9a, an environment agent (white car) aggressively merged in front of the ego agent (orange car). One option for the ego agent is to slow down (optimize a cooperative reward function). This sudden slow down may have frustrated the user, causing a mode change. So in a similar situation later (in another query) the user prefers a trajectory optimal with respect to a competitive reward function and prevents the other agents in the environment from merging in front. This change in preference manifests as noise in 1-step preference-based learning approaches (see Figure 3.9b). However, we would like to learn a composite reward function that not only captures both of these preferences but also *how* they have changed in such a non-stationary environment.

To do so, we allow the users to change their preferences within the same query. We present

each query $Q^{(i)}$ as a sequence of several sub-queries. Each sub-query $Q^{(i,j)}$ in the sequence is a continuation from the final state of the preferred trajectory of the previous sub-query $Q^{(i,j-1)}$. This allows us to learn how the behavior of other interacting agents in one sub-query affects user preference in the next sub-query. We assume that the users' next immediate *preference mode* depends only on their current experience. We, therefore, reset their preference mode at the beginning of each query with a demonstration, which we denote as $Q^{(i,0)}$. After $Q^{(i,0)}$, each sub-query is a best-of-many choice between trajectories from Ξ: $Q_1^{(i,1)}$, $Q_2^{(i,1)}$, ..., $Q_{|Q^{(i,1)}|}^{(i,1)}$ are all continuations of $Q^{(i,0)}$, where $|Q^{(i,1)}|$ denotes the number of trajectories (options) in sub-query $Q^{(i,1)}$. In general, for the rest of the sub-queries, $Q_{j'}^{(i,j)}$ for $j' = 1, 2, \ldots, |Q^{(i,j)}|$ are continuations from $q^{(i,j-1)}$, which is the answer for the $(j-1)^{\text{th}}$ sub-query, as shown in Figure 3.9c.

3.6.3 Reward Dynamics Model

Preliminaries

Throughout this section, we will use $[n]$ to denote the integer set $\{1, 2, \ldots, n\}$ for $n \in \mathbb{Z}_{>0}$.

We denote the j^{th} sub-query in query i as $Q^{(i,j)}$. $|Q^{(i)}|$ denotes the number of sub-queries in query i (as opposed to number of options in a question as in other sections where queries were not hierarchical), and $|Q^{(i,j)}|$ denotes the number of options in the sub-query.

We assume there is a finite set of modes $[M]$ where M is the number of modes. We also assume the mode of the user is stable during a sub-query. Slightly abusing the notation, we denote the mode in the j^{th} sub-query of query i as $M(i,j) \in [M]$.

Each sub-query $Q^{(i,j)}$ consists of $|Q^{(i,j)}|$ trajectories: $Q_1^{(i,j)}, Q_2^{(i,j)}, \ldots, Q_{|Q^{(i,j)}|}^{(i,j)} \in \Xi$. When $j = 0$, $|Q^{(i,j)}| = 1$, as the 0^{th} sub-query is only for setting the initial mode of the user. The user selects one of the trajectories in a sub-query as their response to that sub-query. The user's response to $Q^{(i,j)}$ is $q^{(i,j)} \in Q^{(i,j)}$.

In addition, we assume a trajectory features function $\Phi : \Xi \to \mathbb{R}^d$ that maps every trajectory to a d-dimensional feature space. This function may depend on both the robot and the other agents in the environment. We assume Φ is known. For example, some representative features for driving are distance to the closest environment car, distance to the road boundaries, the speed and the heading angle of the ego vehicle.

Human Preference Model

Reward functions under known modes. We define a user-specific reward function parameterized by the mode of the user, for example, two different reward functions representing calm and rushed driving: $R_{w_m} : \Xi \to \mathbb{R}$ for $m \in [M]$. With a parametric assumption, it is defined as: $R_{w_m}(\xi) = f_{w_m}(\Phi(\xi))$ where $\xi \in \Xi$ and w is a user-specific reward parameter matrix, and w_m is the m^{th} column of w, with each column corresponding to a particular mode for a user. Then, the user response to a

sub-query $Q^{(i,j)}$ is probabilistic based on the softmax model [147, 110], as we also used before:

$$P(q^{(i,j)} \mid Q^{(i,j)}, M(i,j) = m, w) = \frac{\exp(\beta^C R_m(q^{(i,j)}))}{\sum_{\xi \in Q^{(i,j)}} \exp(\beta^C R_m(\xi))} \tag{3.40}$$

for any $q^{(i,j)} \in Q^{(i,j)}$. This models the probability of the human making a choice given a sub-query, the humans' mode during that sub-query, and the user-specific preferences.

Prior on mode transitions. We also learn how people change modes. For example, how likely a person is to transition from aggressive driving to defensive driving or vice versa. We assume that a prior $G \in \mathbb{R}^{M \times M}$ over the mode transitions is given by the designer. The matrix G alone represents the natural propensity to transition between different modes and is independent of the sub-queries and the current state of the learning algorithm. For example, some mode transitions are naturally more likely than the others: If we have three modes that correspond to defensive, neutral and offensive moods, then it would be more likely for a defensive user to switch to the neutral mode than to the offensive mode. G captures this prior. Hence, it is constrained to be a proper Markov chain matrix. We note that Markov chains are employed similarly for mood changes by psychiatrists, e.g. [111]. We explain the formulation of the mode transitions next.

Mode transition model. The users change their mode based on what they experienced in the previous sub-query and their previous mode. We define a *mode-utility* function to capture this effect of the sub-queries. Specifically, we model the mode transitions as follows: The user has an underlying mode-utility function that quantifies the previous trajectories. If the user thinks they would have higher utility with mode m, then they transition to m. As an example, imagine you are driving in a very calm mood. If someone suddenly cuts in front of you, you would think "if I were aggressive, I could keep a shorter headway with the car in front and the other car would not have been able to cut in front of me", and you also switch to an aggressive mood. It is of course also possible that you keep calm. Hence, the transition should be stochastic.

We model the mode-utility as a function of trajectories: $R^u_{\theta_m} : \Xi \to \mathbb{R}$ for $m \in [M]$. Again with the assumption that it is a parametric function, it is defined as: $R^u_{\theta_m}(\xi) = f^u_{\theta_m}(\Phi(\xi))$ where θ is another user-specific parameter matrix and θ_m is the m^{th} column of θ.

The probability of transitioning from any mode m in sub-query $Q^{(i,j)}$ to any mode m' in the next sub-query $Q^{(i,j+1)}$ is given by multiplying the prior G with the likelihood computed using the mode-utility function:

$$\begin{aligned} P_{mm'}(Q^{(i,j)}, q^{(i,j)}, \theta) &:= P(M(i,j+1) = m' \mid M(i,j) = m, Q^{(i,j)}, q^{(i,j)}, \theta) \\ &= \frac{1}{Z} \frac{\exp(R^u_{m'}(q^{(i,j)}))}{\sum_{m'' \in [M]} \exp(R^u_{m''}(q^{(i,j)}))} G_{mm'} \end{aligned} \tag{3.41}$$

where Z is the normalization constant. In a completely "neutral case", when the likelihood (softmax) gives equal values for each mode, the transition is solely defined by the prior G. Some examples of

G are:

- $G_{mm'} = 1/M$ for $\forall(m, m) \in [M]^2$ means that the user may change from any mode to any other mode just based on the previous sub-query with a uniform prior. This is suitable when the modes are categorical, not sequential.
- $G = I$ means the user will not ever change her mode and will remain in her initial mode. Note that the initial mode will also be modeled in a probabilistic way.
- If G is a band matrix, then the user can only change between the modes that are "close". This is suitable for sequential modes.

While our learning model is valid for any feasible G, we will do simplifying assumptions to actively select the hierarchical queries for sample-efficient learning in Section 4.7.

Definition 4. Reward dynamics *of a user is a tuple of* (w, θ)*, which governs both the user preferences and how they transition between modes with the interactions the user is involved in.*

Therefore, our aim is to learn the *reward dynamics* rather than a static unimodal reward function.
Initial Mode. We do not know the initial mode $M(i, 0)$ of the user, which is the active mode during $Q^{(i,0)}$. One simple way is to assume uniform distribution over all modes. However, imagine G is such that transitioning to a mode m is very unlikely from any mode. Then, the uniform assumption will not hold, because the user is unlikely to be in mode m. Then a better model is the following:

$$P_m := P(M(i, j) = m) = \bar{G}_m \tag{3.42}$$

where $\bar{G}_m$ denotes the probability of mode m in the stationary distribution of the Markov chain G. If there exist several stationary distributions, the designer should pick one of them using domain knowledge.

3.6.4 Learning Reward Dynamics

To make the learning of reward dynamics effective and efficient, we should restrict the continuous space of reward dynamics. For that, we make assumptions on the norms of the columns of w and θ similar to [171, 34], i.e., we assume those norms are not larger than 1.

There is also the problem of *label switching*. That is, all the probabilities will remain the same if we switch the order of modes both in w and θ. Since this can completely disable the learning, we enforce another constraint on the ordering, as mentioned by [223], such that $\theta_{1,1} > \theta_{2,1} > \cdots > \theta_{M,1}$ where $\theta_{m,1}$ is the first element of m^{th} column of θ.

Our goal is to learn a distribution over the *reward dynamics* by using hierarchical choice queries. We start with a uniform prior over the space of all feasible (w, θ). After receiving all the responses

to a query $Q(i)$, $(q^{(i,1)}, q^{(i,2)}, \ldots, q^{(i,|Q^{(i)}|)})$, we perform a Bayesian update:

$$P(w, \theta \mid q^{(i,1)}, q^{(i,2)}, \ldots, q^{(i,|Q^{(i)}|)}, Q^{(i,1)}, Q^{(i,2)}, \ldots, Q^{(i,|Q^{(i)}|)}) \\ \propto P(q^{(i,1)}, q^{(i,2)}, \ldots, q^{(i,|Q^{(i)}|)} \mid w, \theta, Q^{(i,1)}, Q^{(i,2)}, \ldots, Q^{(i,|Q^{(i)}|)}) P(w, \theta) \tag{3.43}$$

Next we derive the expression for the update function, i.e., the first term in the right-hand side of Equation (3.43), and present some simplifications that we adopted for our implementation.

3.6.5 Derivation and Simplifications

In this section, we present how we compute the update function for the prior $p(w, \theta)$. We note $Q^{(i,0)}$ does not receive any response. For the simplicity of notation, we let $q^{(i,0)}$ denote the only trajectory in $Q^{(i,0)}$, so that $P_{mm'}(Q^{(i,j)}, q^{(i,j)}, \theta)$ is well-defined for $\forall (m, m') \in [M]^2$ when $j = 0$. We then derive

$$P(q^{(i,1)}, q^{(i,2)}, \ldots, q^{(i,|Q^{(i)}|)} \mid w, \theta, Q^{(i,1)}, Q^{(i,2)}, \ldots, Q^{(i,|Q^{(i)}|)}) \\ = \sum_{(m_0, \ldots, m_{|Q^{(i)}|}) \in [M]^{|Q^{(i)}|+1}} P_{m_0} P_{m_0 m_1}(Q^{(i,0)}, q^{(i,0)}, \theta) \ldots P_{m_{|Q^{(i)}|-1} m_{|Q^{(i)}|}}(Q^{(i,|Q^i|-1)}, q_{|Q^{(i)}|-1}, \theta) \\ \prod_{j=1}^{|Q^i|} P(q^{(i,j)} | w, Q^{(i,j)}, M(i,j) = m_j) \tag{3.44}$$

In our implementation, we restrict ourselves to the cases where $|Q^{(i)}| = 3$ for all iterations $i = 1, 2, \ldots$. Then, the above equation is simplified as

$$P(q^{(i,1)}, q^{(i,2)} \mid w, \theta, Q^{(i,0)}, Q^{(i,1)}, Q^{i,2}) \\ = \sum_{m_0 \in [M]} \sum_{m_1 \in [M]} \sum_{m_2 \in [M]} P_{m_0} P_{m_0 m_1}(Q^{(i,0)}, q^{(i,0)}, \theta) P_{m_1 m_2}(Q^{(i,1)}, q^{(i,1)}, \theta) \\ P(q^{(i,1)} \mid w, Q^{(i,1)}, M(i,1) = m_1) P(q^{(i,2)} \mid w, Q^{(i,2)}, M(i,2) = m_2) \tag{3.45}$$

To eliminate the normalization Z from the equation, we assume $G_{mm'} \in \{0, 1/c_m\}$ for $\forall (m, m') \in [M]^2$ where c_m is an appropriate constant. That is, we assume the model designer will just decide on whether or not it is possible to move between any two modes and will not assign specific prior probabilities. Then,

$$P_{mm'}(Q^{(i,0)}, q^{(i,0)}, \theta) = \frac{\exp(R^u_{\theta_{m'}}(q^{(i,0)}))}{\sum_{m'' \in [M]: G_{mm''} = 1/c_m} \exp(R^u_{\theta_{m''}}(q^{(i,0)}))} \tag{3.46}$$

If we further assume $M = 2$ and $G_{mm'} = 1/2$ for $\forall (m, m') \in [M]^2$, such as the case of cooperative

and competitive modes, we also have $P_m = \frac{1}{2}$, so we can write:

$$P(q^{(i,1)}, q^{(i,2)} \mid w, \theta, Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)})$$
$$= \sum_{(m_1,m_2)\in\{1,2\}^2} \prod_{j\in\{1,2\}} \frac{\exp(R_{w_{m_j}}(q^{(i,j)}))}{\sum_{\xi\in Q^{(i,j)}} \exp(R_{w_{m_j}}(\xi))} \frac{\exp(R^u_{\theta_{m_j}}(q^{(i,j-1)}))}{\exp(R^u_{\theta_1}(q^{(i,j-1)})) + \exp(R^u_{\theta_2}(q^{(i,j-1)}))} \tag{3.47}$$

This formulation enables us to update $P(w, \theta)$. We will present our active hierarchical choice query selection algorithm that improves data-efficiency in Section 4.7.

3.7 Chapter Summary

In this chapter, we developed reward learning techniques that use comparative feedback from humans instead of or in addition to expert demonstrations. This problem setup is closely related to inverse reinforcement learning as we discussed in Section 2.2. However, it brings an important advantage in practice: we do not need to collect human demonstrations of the task that are (almost) optimal, which is infeasible or extremely challenging in many domains, e.g., regular users of a lower-body exoskeleton (see Section 3.3) are people with paralysis (a group with nearly 5.4 million people in the US alone [15]), who cannot possibly give demonstrations to these systems.

On the other hand, comparative feedback is easy to collect and does not require expertise on controlling the system. In most cases, a human user who has the knowledge of the target task can easily compare two (or more) trajectories of a robot in terms of their task performance. Motivated by this, we developed and investigated various techniques for reward learning using comparative feedback. We studied different query types and functional forms for the reward function (as we outlined in Section 1.2).

An important limitation of comparative feedback, especially when we are working on trajectory level, i.e., human users comparing trajectories rather than individual states or actions, is the fact that each comparison query carries a small amount of information compared to expert demonstrations. Although we showed in Section 3.1 that demonstrations can still be used together with comparative feedback, this limitation hurts the practicality and scalability of the methods we developed. Therefore, techniques that actively query the users for the highest information gain are crucial. In the next chapter, we will focus on such approaches that are based on active learning optimizations. We will also present the simulation and experiment results that we deferred in Chapter 3.

Chapter 4

Active Querying for Comparative Feedback

Even though having humans provide comparative feedback does not suffer from similar problems to collecting demonstrations, each comparison question is much less informative than an expert demonstration. For example, each pairwise comparison query can provide at most 1 bit of information. A promising approach to tackle this problem is to actively generate the queries for comparative feedback [171, 121, 207].

In Chapter 3, we covered how the robot can update its understanding of the reward function parameters w given the human's comparative feedback; but how does the robot choose the right questions in the first place? Active query generation deals with this problem. Unlike demonstrations — where the robot is passive — here the robot is *active*, and purposely probes the human to get fine-grained information about specific parts of w that are unclear. By actively querying the user, the robot attempts to get as much information as possible, mitigating the data-inefficiency issue of learning from comparative feedback. At the same time, the robot needs to remember that a human is answering these questions, and so the options need to be easy and intuitive for the human to respond to. Proactively choosing intuitive queries is arguably the most challenging part of learning from comparative feedback. Accordingly, we will explore methods for actively generating queries Q in the subsequent sections.

Greedy Robot. Ideally, the robot must find the best *adaptive sequence* of queries to clarify the human's reward. Unfortunately, reasoning about a sequence of queries is — in general — NP-hard [4]. We therefore proceed in a *greedy* fashion throughout this chapter: at every iteration i, the robot chooses $Q^{(i)}$ while thinking only about the next posterior belief b^i (e.g., see Equation (3.9)).

In Section 4.1, we start with describing the maximum volume removal based active querying

method [171], which has been the dominant approach for active query generation to maximize data-efficiency [159]. However, in Section 4.2, we will show optimizing mutual information is a better approach for data-efficiency, and also helps with generating easier questions for the human users. We will then use this technique to extend the majority of the methods we presented in Chapter 3 with active querying in Sections 4.3 through 4.7. Finally, Section 4.8 will introduce various batch-active querying techniques that enable actively generating queries in batches for all the methods presented until that point. All sections in this chapter will also present simulation and experiment results both for the active querying techniques and the corresponding learning methods from Chapter 3.

4.1 Choosing Queries with Volume Removal

In this section, we introduce an active querying method that is called maximum volume removal. The objective in this method is to maximize the amount of space that will be removed from the hypothesis space of reward function parameters w, and the overall optimization problem is submodular, allowing this approach to enjoy some theoretical guarantees as shown by Sadigh et al. [171].

Although the volume removal objective can be used with any of the query types we introduced in Chapter 3, we will use the DemPref method we introduced in Section 3.1 for its simplicity. This will also enable us to show in the next section that active comparative feedback must be collected after the demonstrations are incorporated into the belief distribution via Equation (3.6). However, we would like to note once again that demonstrations can be used along with any, possibly actively collected, comparative feedback type, as long as the reward function is stationary and unimodal.

4.1.1 Maximum Volume Removal Optimization

Maximizing volume removal is a widely used strategy for selecting queries. The method attempts to generate the most-informative queries by finding the $Q^{(i)}$ that maximizes the expected difference between the prior and *unnormalized* posterior [171, 159]. Formally, the method generates a query of $|Q^{(i)}| \geq 2$ trajectories at iteration i by solving:

$$\underset{Q^{(i)}=\{\xi_1,\ldots,\xi_{|Q^{(i)}|}\}}{\arg\max} \mathbb{E}_{q^{(i)}} \left[\int \Big(b^{i-1}(w) - b^{i-1}(w) P(q^{(i)} \mid Q^{(i)}, w) \Big)\, dw \right] \tag{4.1}$$

where the prior is on the left and the unnormalized posterior from Equation (3.9) is on the right. The integration is over the all possible values of w. This optimization problem can equivalently be written as:

$$Q^{(i)}_* = \underset{Q^{(i)}=\{\xi_1,\ldots,\xi_{|Q^{(i)}|}\}}{\arg\max} \mathbb{E}_{q^{(i)}} \mathbb{E}_{w\sim b^{i-1}} \left[1 - P(q^{(i)} \mid Q^{(i)}, w) \right] , \tag{4.2}$$

or in the special case of pairwise comparison queries, i.e., $|Q^{(i)}| = 2$, as we show in Appendix A.2,

$$Q_*^{(i)} = \underset{Q^{(i)}=\{\xi_1,\xi_2\}}{\arg\max} \min_{q^{(i)}} \mathbb{E}_{w\sim b^{i-1}} \left[1 - P(q^{(i)} \mid Q^{(i)}, w)\right] . \tag{4.3}$$

The distribution b^{i-1} can get very complex and thus — to tractably compute the expectations in Equation (4.2) — we are forced to leverage sampling. Letting Ω denote a set of samples drawn from the prior b^{i-1}, and $\doteq$ denote asymptotic equality as the number of samples $|\Omega| \to \infty$, the optimization problem in Equation (4.2) becomes:

$$Q_*^{(i)} \doteq \underset{Q^{(i)}=\{\xi_1,\ldots,\xi_{|Q^{(i)}|}\}}{\arg\min} \sum_{q^{(i)}\in Q^{(i)}} \left(\sum_{w\in\Omega} P(q^{(i)} \mid Q^{(i)}, w)\right)^2 \tag{4.4}$$

In practice, we can use, for example, Metropolis-Hastings [69] for sampling from the prior belief b^{i-1}.

Intuition. When solving Equation (4.4), the robot looks for queries $Q^{(i)}$ where each answer $q^{(i)} \in Q^{(i)}$ is equally likely given the current belief over w. These questions appear useful because the robot is maximally uncertain about which trajectory the human will prefer.

When Does This Fail? Although prior works have shown that volume removal can work in practice, we here identify two key shortcomings. First, we point out a failure case: the robot may solve for questions where the answers are equally likely but *uninformative* about the human's reward. Second, the robot does not consider the human's ability to answer when choosing questions — and this leads to *challenging*, indistinguishable queries that are hard to answer!

Uninformative Queries

The optimization problem used to identify maximum volume removal queries fails to capture our original goal of generating informative queries. Consider a trivial query where all options are identical: $Q^{(i)} = \{\xi_1, \xi_1, \ldots, \xi_1\}$. Regardless of which answer $q^{(i)}$ the human chooses, here the robot gets no information about the right reward function; put another way, $b^i = b^{i-1}$. Asking a trivial query is a waste of the human's time — but we find that this uninformative question is actually a best-case solution to Equation (4.2).

Theorem 1. *The trivial query $Q = \{\xi_1, \xi_1, \ldots, \xi_1\}$ (for any $\xi_1 \in \Xi$) is a global solution to Equation* (4.2).

Proof. For a given $Q^{(i)}$ and w, $\sum_{q^{(i)}} P(q^{(i)} \mid Q^{(i)}, w) = 1$. Thus, we can upper bound the objective

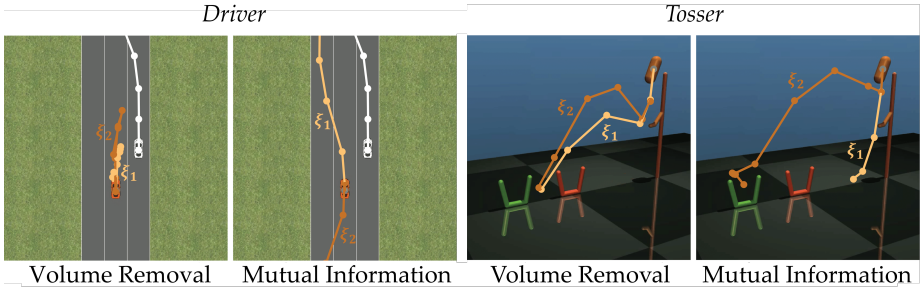


Figure 4.1: Sample queries generated with the volume removal and information gain methods on *Driver* and *Tosser* tasks. Volume removal generates queries that are difficult, because the options are almost equally good or equally bad.

in Equation (4.2) as follows:

$$\mathbb{E}_{q^{(i)}|Q^{(i)},b^{i-1}}\mathbb{E}_{w\sim b^{i-1}}[1-P(q^{(i)}\mid Q^{(i)},w)] \tag{4.5}$$

$$=1-\mathbb{E}_{w\sim b^{i-1}}\left[\sum_{q^{(i)}\in Q^{(i)}}P(q^{(i)}\mid Q^{(i)},w)^2\right]\leq 1-1/|Q^{(i)}|\,, \tag{4.6}$$

recalling that $|Q^{(i)}|$ is the total number of options in $Q^{(i)}$. For the trivial query $Q=\{\xi_1,\xi_1,\ldots,\xi_1\}$, the objective in Equation (4.2) has value $\mathbb{E}_{q^{(i)}|Q^{(i)},b^{i-1}}\mathbb{E}_{w\sim b^{i-1}}\left[1-P(q^{(i)}\mid Q^{(i)},w)\right] = 1-1/|Q^{(i)}|$. This is equal to the upper bound on the objective, and thus the trivial, uninformative query of identical options is a global solution to Equation (4.2). □

Challenging Queries

Volume removal prioritizes questions where each answer is equally likely. Even when the options are not identical (as in a trivial query), the questions may still be very challenging for the user to answer. We explain this issue through a concrete example (also see Figure 4.1):

Example 1. Let the robot query the human while providing $|Q|=2$ different answer options, i.e., with a pairwise comparison query, ξ_1 and ξ_2.

Question A. Here the robot asks a question where both options are equally good choices. Consider query Q_A such that $P(q=\xi_{A,1}\mid Q_A,w)=P(q=\xi_{A,2}\mid Q_A,w)\quad\forall w\in\Omega$. Responding to Q_A is difficult for the human, since both options $\xi_{A,1}$ and $\xi_{A,2}$ equally capture their reward function.

Question B. Alternatively, this robot asks a question where only one option matches the human's

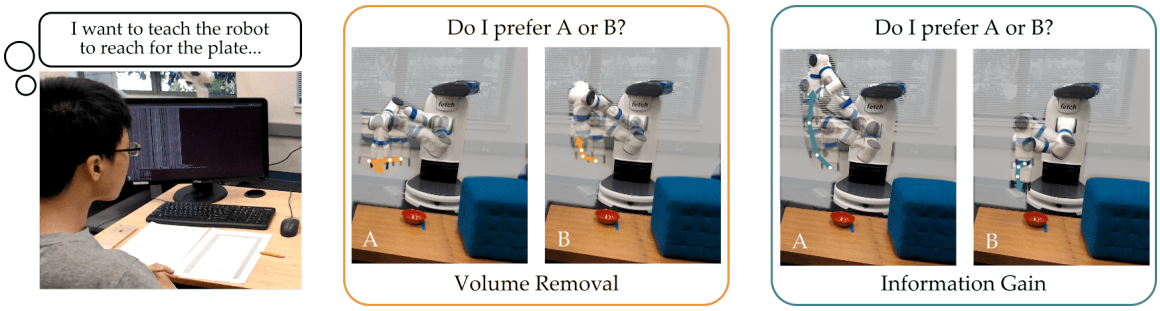


Figure 4.2: Comparing preference queries that do not account for the human's ability to answer to queries generated using our information gain approach. Here the robot is attempting to learn the user's reward function, and demonstrates two possible trajectories. The user should select the trajectory that better aligns with their own preferences. While the trajectories produced by the state-of-the-art volume removal method are almost indistinguishable, our information theoretic approach results in questions that are easy to answer, which eventually increase the robot's overall learning efficiency.

true reward. Consider a query Q_B such that:

$$P(q = \xi_{B,1} \mid Q_B, w) \approx 1 \quad \forall w \in \Omega^{(1)} \tag{4.7}$$

$$P(q = \xi_{B,2} \mid Q_B, w) \approx 1 \quad \forall w \in \Omega^{(2)} \tag{4.8}$$

$$\Omega^{(1)} \cup \Omega^{(2)} = \Omega, \quad |\Omega^{(1)}| = |\Omega^{(2)}| \tag{4.9}$$

If the human's weights w lie in $\Omega^{(1)}$, the human will always answer with $\xi_{B,1}$, and — conversely — if the true w lies in $\Omega^{(2)}$, the human will always select $\xi_{B,2}$. Intuitively, this query is easy for the human: regardless of what they want, one option stands out when answering the question.

Incorporating the Human. Looking at Example 1, it seems clear that the robot should ask question Q_B. Not only does Q_A fail to provide any information about the human's reward (because their response could be equally well explained by any w), but it is also hard for the human to answer (since both options seem equally viable). Unfortunately, when maximizing volume removal the robot thinks Q_A is *just as good* as Q_B: they are both global solutions to its optimization problem! Here volume removal gets it wrong because it fails to take the human into consideration. Asking questions based only on how uncertain the robot is about the human's answer can naturally lead to confusing, uninformative queries. Figure 4.1 demonstrates some of these hard queries generated by the volume removal formulation.

Theorem 1 and Example 1 make it clear that volume removal is not the true objective we should be optimizing for. It works in practice not despite the local optima, but thanks to them! Therefore, in the next section, we will introduce a new active querying approach that is based on maximizing the mutual information. We will show this approach does not suffer from similar issues. It will further enable us to show demonstrations must be incorporated into the belief *before* actively querying the user for comparative feedback. Before we move into this new objective, we present our experiment

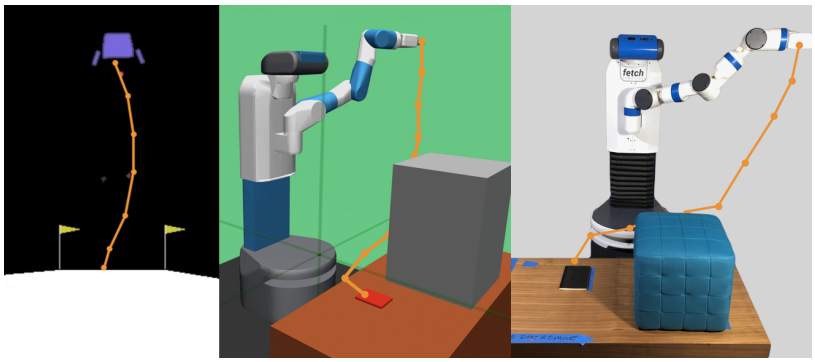


Figure 4.3: Views from simulation domains, with a demonstration in orange: **(a)** *LunarLander*, **(b)** *FetchReach* (simulated), **(c)** *FetchReach* (physical).

results with the volume removal optimization.

4.1.2 Experiments

We conduct two sets of experiments to assess the performance of DemPref with volume removal maximization under various metrics. In all experiments, we assume a reward function that is linear in trajectory features, i.e., $R_w(\xi) = w^\top \Phi(\xi)$ for any trajectory $\xi \in \Xi$ with $\|w\| \leq 1$.[1] We start by describing the simulation domains and the user study environment. Each subsequent subsection presents a set of experiments and tests the relevant hypotheses.

Simulation Domains

In each experiment, we use a subset of the following domains, shown in Figures 4.1 and 4.3, as well as a linear dynamical system:

LunarLander. We use the continuous "LunarLander" environment from OpenAI Gym [50], where the lander has to safely reach the landing pad. The trajectory features correspond to the lander's average distance from the landing pad, its angle, its velocity, and its final distance to the landing pad.

FetchReach. Inspired by [19], we use a modification of the "FetchReach" environment from OpenAI Gym (built on top of MuJoCo), where the robot has to reach a goal with its arm, while keeping its arm as low-down as possible (see Figure 4.3). The trajectory features correspond to the robot gripper's average distance to the goal, its average height from the table, and its average distance to a box obstacle in the domain. See Appendix D.1 for the formal feature definitions.

For our user studies, we employ a version of the *FetchReach* environment with the physical Fetch robot (see Figure 4.3) [213].

[1] In this section, unless otherwise noted, we adopt $\beta^D = 0.02$, $\beta^C = 1$, and assume a uniform prior over reward parameters w, i.e., $P(w)$ is constant for any $\|w\|_2 \leq 1$. We use Metropolis-Hastings algorithm [69] for sampling the set Ω from belief distribution over w.

Evaluation Metric

To judge convergence of the inferred reward function parameters to true parameters in simulations, we adopt the *alignment metric* from [171]:

$$\texttt{Alignment} = \frac{1}{|\Omega|} \sum_{\bar{w} \in \Omega} \frac{w^* \cdot \bar{w}}{\|w^*\|_2 \|\bar{w}\|_2} \,, \tag{4.10}$$

where w^* is the true reward function parameters.

We are now ready to present our two sets of experiments each of which demonstrates a different aspect of the proposed DemPref framework:

1. The utility of initializing with demonstrations,
2. The advantages comparative feedback provide over using only demonstrations,

Initializing with Demonstrations

We first investigate whether initializing the learning framework with user demonstrations is helpful. Specifically, we test the following hypotheses:

H1. *DemPref accelerates learning by initializing the prior belief b^0 using user demonstrations.*

H2. *The convergence of DemPref improves with the number of demonstrations used to initialize the algorithm.*

To test these two claims, we perform simulation experiments in *Driver*, *LunarLander* and *FetchReach* environments. For each environment, we simulate a human user with hand-tuned reward function parameters w, which gives reasonable performance. We generate demonstrations by applying model predictive control (MPC) to solve: $\max_\xi R_{w^*}(\xi)$. After initializing the belief with varying number of such demonstrations ($|\mathcal{D}_D| \in \{0, 1, 3\}$), the simulated users in each environment respond to 25 pairwise comparison queries ($|Q| = 2$) according to Equation (3.10), each of which is actively synthesized with the volume removal optimization.[2] We repeat the same procedure for 8 times to obtain confidence bounds.

The results of the experiment are presented in Figure 4.4. On all three environments, initializing with demonstrations improves the convergence rate of the preference-based algorithm significantly; to match the `Alignment` value attained by DemPref with only one demonstration in 10 pairwise comparison queries, it takes the pure preference-based algorithm, i.e., without any demonstrations, 30 pairwise comparisons on *Driver*, 35 on *LunarLander*, and 20 on *FetchReach*. These results provide strong evidence in favor of **H1**.

The results regarding **H2** are more complicated. Initializing with three instead of one demonstration improves convergence significantly only on the *Driver* and *LunarLander* domains. (The improvement on *Driver* is only at the early stages of the algorithm, when fewer than 10 pairwise

[2]The environments we use are deterministic, i.e., state transitions are not stochastic. Hence, we fix the initial state and simply optimize over the sequence of actions to solve the volume removal maximization problem.

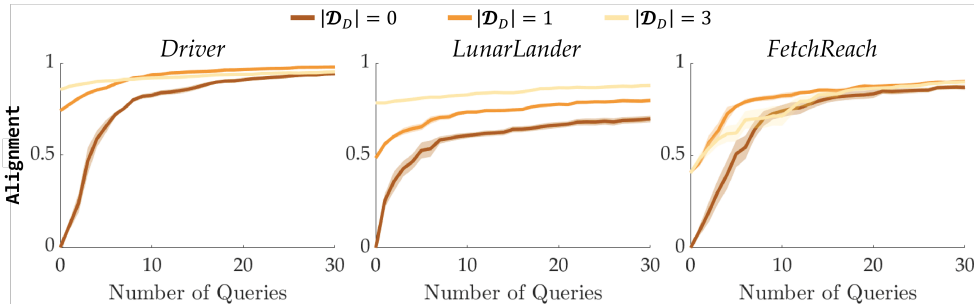


Figure 4.4: The results of our first experiment, investigating whether initializing with demonstrations improves the learning rate of the algorithm, on three domains. On the *Driver*, *LunarLander*, and *FetchReach* (simulated) environments, initializing with one demonstration improved the rate of convergence significantly.

comparisons are used.) However, on the *FetchReach* domain, initializing with three instead of one demonstration hurts the performance of the algorithm. (Although, we do note that the results from using three demonstrations are still an improvement over the results from not initializing with demonstrations). This is unsurprising. It is much harder to provide good demonstrations on the *FetchReach* environment than on the *Driver* or *LunarLander* environments, and therefore the demonstrations are of lower quality. Using more demonstrations when they are of lower quality leads to the prior being more concentrated further away from the true reward function, and can cause the the learning algorithm to slow down.

In practice, we find that using a single demonstration to initialize the algorithm leads to reliable improvements in convergence, regardless of the complexity of the domain.

DemPref vs. IRL

Next, we analyze if preference elicitation improves learning performance. To do that, we conduct a within-subjects user study where we compare our DemPref algorithm with Bayesian IRL [164]. The hypotheses we are testing are:

H3. *The robot which uses the reward function learned by DemPref will be more successful at the task (as evaluated by the users) than the IRL counterpart.*

H4. *Participants will prefer to use the DemPref framework as opposed to the IRL framework.*

For these evaluations, we use the *FetchReach* domain with the physical Fetch robot. Participants were told that their goal was to get the robot's end-effector as close as possible to the goal, while (1) avoiding collisions with the block obstacle and (2) keeping the robot's end-effector low to the ground (so as to avoid, for example, knocking over objects around it). Participants provided demonstrations via teleoperation (using end-effector control) on a keyboard interface; each user was given some time to familiarize themselves with the teleoperation system before beginning the experiment.

Participants trained the robot using two different systems. (1) IRL: Bayesian IRL with 5 demonstrations. (2) DemPref: our DemPref framework (with the volume removal optimization) with 1

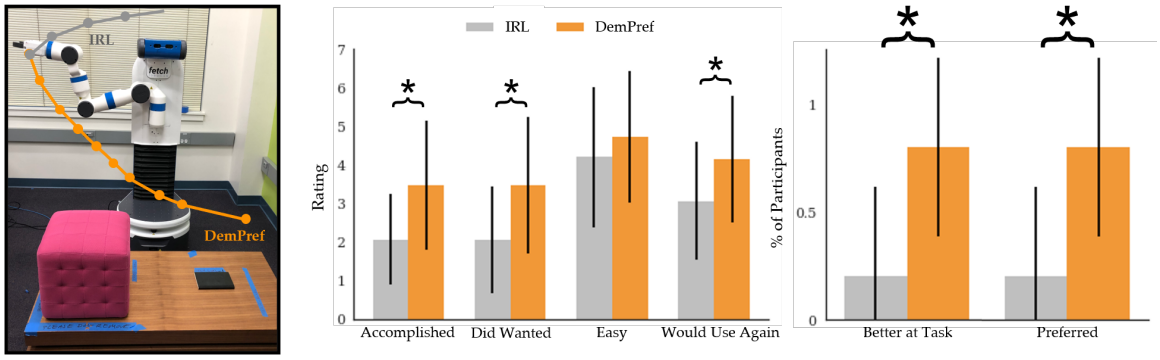


Figure 4.5: (Left) Our testing domain, with two trajectories generated according to the reward functions learned by IRL and DemPref from a specific user in our study. (Right) The results of our usability study – the error bars correspond to standard deviation and significant results are marked with an asterisk. We find that users rated the robot trained with DemPref as significantly better at accomplishing the task and preferred to use our method for training the robot significantly more than they did IRL. However, we did not find evidence to suggest that users found our method easier to use than standard IRL.

demonstration and 15 proactive pairwise comparison queries[3]. We counter-balanced across which system was used first, to minimize the impact of familiarity bias with our teleoperation system.

After learning from human feedback, the robot was trained in simulation using Proximal Policy Optimization (PPO) with the reward function learned from each system [176]. To ensure that the robot was not simply overfitting to the training domain, we used different variants of the domain for training and testing the robot. We used two different test domains (and counter-balanced across them) to increase the robustness of our results against the specific testing domain. Figure 4.5 (left) illustrates one of our testing domains. We rolled out three trajectories in the test domains for each algorithm on the physical Fetch robot. After observing each set of trajectories, the users were asked to rate the following statements on a 7-point rating scale:

1. The robot accomplished the task well. (Accomplished)
2. The robot did what I wanted. (Did Wanted)
3. It was easy to train the robot with this system. (Easy)
4. I would want to use this system to train a robot in the future. (Would Use Again)

They were also asked two comparison questions:

1. Which robot accomplished the task better? (Better at Task)
2. Which system would you prefer to use if you had to train a robot to accomplish a similar task? (Preferred)

They were finally asked for general comments.

For this user study, we recruited 15 participants (11 male, 4 female), six of whom had prior

[3]The number of demonstrations and pairwise comparisons used in each system were chosen such that a simulated agent achieves similar convergence to the true reward on both systems.

experience in robotics but none of whom had any prior exposure to our system.

We present our results in Figure 4.5 (right). When asked which robot accomplished the task better, users preferred the DemPref system by a significant margin ($p < 0.05$, Wilcoxon paired signed-rank test); similarly, when asked which system they would prefer to use in the future if they had to train the robot, users preferred the DemPref system by a significant margin ($p < 0.05$). This provides strong evidence in favor of both **H3** and **H4**.

As expected, many users struggled to teleoperate the robot. Several users made explicit note of this fact in their comments: "I had a hard time controlling the robot", "I found the [IRL system] difficult as someone who [is not] kinetically gifted!", "Would be nice if the controller for the [robot] was easier to use." Given that the robot that employs IRL was only trained on these demonstrations, it is perhaps unsurprising that DemPref outperforms IRL on the task.

We were however surprised by the extent to which the IRL-powered robot fared poorly: in many cases, it did not even attempt to reach for the goal. Upon further investigation, we discovered that IRL was prone to, in essence, "overfitting" to the training domain. In several cases, IRL had overweighted the users' preference for obstacle avoidance. This proved to be an issue in one of our test domains where the obstacle is closer to the robot than in the training domain. Here, the robot does not even try to reach for the goal since the loss in value (as measured by the learned reward function) from going near the obstacle is greater than the gain in value from reaching for the goal. Figure 4.5 (left) shows this test domain and illustrates, for a specific user, a trajectory generated according to reward function learned by each of IRL and DemPref.

While we expect that IRL would overcome these issues with more careful feature engineering and increased diversity of the training domains, it is worth noting DemPref was affected much less by these issues. These results suggest learning from comparative feedback methods may be more robust to poor feature engineering and a lack of training diversity than IRL; however, a rigorous evaluation of these claims is beyond the scope of this thesis.

It is interesting that despite the challenges that users faced with teleoperating the robot, they did not rate the DemPref system as being "easier" to use than the IRL system ($p = 0.297$). Several users specifically referred to the time it took to generate each query ($\sim$45 seconds) as negatively impacting their experience with the DemPref system: "I wish it was faster to generate the preference [queries]", "The [DemPref system] will be even better if time cost is less." Additionally, one user expressed difficulty in evaluating the preference queries themselves, commenting "It was tricky to understand/infer what the preferences were [asking]. Would be nice to communicate that somehow to the user (e.g. which [trajectory] avoids collision better)!", which highlights the fact that volume removal formulation may generate queries that are extremely difficult for the humans. Hence, we analyze in the next section how mutual information objective improves the experience for the users.

4.2 Choosing Queries with Mutual Information

As we just showed both mathematically and empirically, maximizing volume removal sometimes fails to generate *informative* queries, and also does not consider the ease and intuitiveness of every query for the human-in-the-loop. This can lead to queries that are *difficult* for the human to answer, e.g., two queries that are equally good (or bad) from the human's perspective.

We *resolve* this issue with a second active querying method, mutual information: here the robot balances (a) how much information it will get from a correct answer against (b) the human's ability to answer that question confidently. We also present a *set of tools* that can be used to enhance the user's experience, including an optimal condition for determining when the robot should stop asking questions.

4.2.1 Maximum Mutual Information Optimization

At each iteration, we find the query $Q^{(i)}$ that maximizes the mutual information about w. We do so by solving the following optimization problem:

$$\begin{aligned} Q^{(i)}_* &= \arg\max_{Q^{(i)}} I(w; q^{(i)} \mid Q^{(i)}, b^{i-1}) \\ &= \arg\max_{Q^{(i)}} H(w \mid Q^{(i)}, b^{i-1}) - \mathbb{E}_{q^{(i)}|Q^{(i)},b^{i-1}} H(w \mid q^{(i)}, Q^{(i)}, b^{i-1}), \end{aligned} \tag{4.11}$$

where I is the mutual information and H is Shannon's information entropy [77]. Approximating the expectations via sampling, we re-write this optimization problem below (see Appendix B.1 for the full derivation):

$$Q^{(i)}_* \doteq \arg\max_{Q^{(i)}=\{\xi_1,\ldots,\xi_{|Q^{(i)}|}\}} \frac{1}{|\Omega|} \sum_{q^{(i)} \in Q^{(i)}} \sum_{w \in \Omega} \left(P(q^{(i)} \mid Q^{(i)}, w) \log_2 \left(\frac{|\Omega| \cdot P(q^{(i)} \mid Q^{(i)}, w)}{\sum_{w' \in \Omega} P(q^{(i)} \mid Q^{(i)}, w')} \right) \right), \tag{4.12}$$

where Ω again denotes the samples from the prior belief b^{i-1}.

Intuition. To see why accounting for the human is naturally part of the mutual information solution, re-write Equation (4.11):

$$Q^{(i)}_* = \arg\max_{Q^{(i)}} \; H(q^{(i)} \mid Q^{(i)}, b^{i-1}) - \mathbb{E}_{w \sim b^{i-1}} H(q^{(i)} \mid w, Q^{(i)}) \tag{4.13}$$

Here the first term in Equation (4.13) is the *robot's uncertainty* over the human's response: given a query $Q^{(i)}$ and the robot's understanding of w, how confidently can the robot predict the human's answer? The second entropy term captures the *human's uncertainty* when answering: given a query and their true reward, how confidently will they choose option $q^{(i)}$? Optimizing for mutual

information with Equations (4.11) or (4.12) naturally considers both robot and human uncertainty, and favors questions where (a) the robot is unsure how the human will answer but (b) the human can answer easily. We contrast this to maximum volume removal, where the robot purely focused on questions where the human's answer was unpredictable.

Why Does This Work? To highlight the advantages of this method, let us revisit the shortcomings of volume removal. Below we show how mutual information optimization successfully addresses the problems described in Theorem 1 and Example 1. Further, we emphasize that the computational complexity of computing objective (4.12) is equivalent — in order — to the volume removal objective from Equation (4.4). Thus, the mutual information based method avoids the previous failures while being at least as computationally tractable.

Uninformative Queries

Recall from Theorem 1 that any trivial query $Q = \{\xi_1, \ldots, \xi_1\}$ is a global solution for volume removal. In reality, we know that this query is a worst-case choice: no matter how the human answers, the robot will gain no insight into w. Mutual information ensures that the robot will not ask trivial queries: under Equation (4.11), $Q = \{\xi_1, \ldots, \xi_1\}$ is actually the *global minimum*!

Challenging Questions

Revisiting Example 1, we remember that Q_B was a much easier question for the human to answer, but volume removal values Q_A as highly as Q_B. Under mutual information, the *robot* is equally uncertain about how the human will answer Q_A and Q_B, and so the first term in Equation (4.13) is the same for both. But the robot using mutual information additionally recognizes that the *human* is very uncertain when answering Q_A: here Q_A attains the global maximum of the second term while Q_B attains the global minimum! Thus, the overall value of Q_B is higher and — as desired — the robot recognizes that Q_B is a better question.

4.2.2 Additional Tools and Analysis

We introduced how robots can generate proactive questions to maximize mutual information. Below we highlight some additional tools that designers can leverage to improve the computational performance and applicability of these methods. In particular, we draw the reader's attention to an optimal stopping condition, which tells the DemPref robot when to stop asking the human questions.

Optimal Stopping

We propose a novel extension — specifically for mutual information — that tells the robot when to stop asking questions. Intuitively, the DemPref querying process should end when the robot's questions become more costly to the human than informative to the robot.

Let each query Q have an associated cost $c(Q) \in \mathbb{R}_{\geq 0}$. This function captures the *cost* of a question: e.g., the amount of time it takes for the human to answer, the number of similar questions that the human has already seen, or even the interpretability of the question itself. We subtract this cost from our mutual information objective in Equation (4.11), so that the robot (greedily) maximizes mutual information while biasing its search towards low-cost questions:

$$\max_{Q=\{\xi_1,\ldots,\xi_{|Q|}\}} I(w; q \mid Q, b^{i-1}) - c(Q) \tag{4.14}$$

Now that we have introduced a cost into the query selection problem, the robot can reason about when its questions are becoming prohibitively expensive or redundant. We find that the best time to stop asking questions in expectation is when their cost exceeds their value:

Theorem 2. *A robot using mutual information to perform active preference-based learning should stop asking questions if and only if the global solution to Equation* (4.14) *is negative at the current iteration.*

See the Appendix A.3 for our proof. We emphasize that this result is valid only for mutual information, and adapting Theorem 2 to volume removal is not trivial.

The decision to terminate our DemPref algorithm is now fairly straightforward. At each iteration i, we search for the question $Q_*^{(i)}$ that maximizes the trade-off between mutual information and cost. If the value of Equation (4.14) is non-negative, the robot shows this query to the human and elicits their response; if not, the robot cannot find any sufficiently important questions to ask, and the process ends. This automatic stopping procedure makes the active learning process more user-friendly by ensuring that the user does not have to respond to unnecessary or redundant queries.

Why Demonstrations First?

Now that we have a user-friendly strategy for generating queries and stopping, we want to determine *in what order* the robot should leverage the demonstrations and the comparisons.

Recall that demonstrations provide coarse, high-level information, while comparison queries hone-in on isolated aspects of the human's reward function. Intuitively, it seems like we should start with high-level demonstrations before probing low-level preferences: but is this really the right order of collecting data? What about the alternative — a robot that instead waits to utilize the demonstrations dataset $\mathcal{D}_D$ until after asking questions?

When leveraging mutual information maximization to generate queries, we here prove that the robot will gain *at least as much* information about the human's preferences as any other order of demonstrations and queries. Put another way, starting with demonstrations *in the worst case* is just as good as any other order; and *in the best case* we obtain more information.

Theorem 3. *Under the Boltzmann rational human model for demonstrations presented in Equation* (3.5)*, our DemPref approach — where best-of-many choice queries are actively generated after collecting demonstrations — results in at least as much information about the human's preferences as would be obtained by reversing the order of queries and demonstrations.*

Proof. Let $Q_*^{(i)}$ be the (greedily) optimal query with respect to the mutual information optimization *after collecting demonstrations*. From Equation (4.14), $Q_*^{(i)} = \arg\max_{Q^{(i)}} \left(I(w; q^{(i)} \mid Q^{(i)}, b^{i-1}) - c(Q^{(i)})\right)$. We let $q_*^{(i)}$ denote the human's response to query $Q_*^{(i)}$. Similarly, let $\tilde{Q}^{(i)}$ be the mutual information query *before incorporating the demonstrations into the belief*, so that $\tilde{Q}^{(i)} = \arg\max_{Q^{(i)}} \left(I(w; q^{(i)} \mid Q^{(i)}, (\tilde{Q}^{(j)}, \tilde{q}^{(j)})_{j=0}^{i-1}\right.$ Again, we let $\tilde{q}^{(i)}$ denote the human's response to query $\tilde{Q}^{(i)}$. Noting the total cost of queries will not change (and hence, the theorem and the proof extend to the case where $c(Q) = 0$ for all queries Q), we can compare the overall mutual information for each order of questions and demonstrations:

$$\begin{aligned}
&I\Big(w; \mathcal{D}_D, (q_*^{(1)}, q_*^{(2)}, \ldots) \mid (Q_*^{(1)}, Q_*^{(2)}, \ldots)\Big) \\
&\quad = I(w; \mathcal{D}_D) + I\left(w; (q_*^{(1)}, q_*^{(2)}, \ldots) \mid (b^0, Q_*^{(1)}, Q_*^{(2)}, \ldots)\right) \\
&\quad \geq I(w; \mathcal{D}_D) + I\left(w; (\tilde{q}^{(1)}, \tilde{q}^{(2)}, \ldots) \mid (b^0, \tilde{Q}^{(1)}, \tilde{Q}^{(2)}, \ldots)\right) \\
&\quad = I\left(w; (\tilde{q}^{(1)}, \tilde{q}^{(2)}, \ldots, \mathcal{D}_D) \mid (\tilde{Q}^{(1)}, \tilde{Q}^{(2)}, \ldots)\right) \qquad (4.15)
\end{aligned}$$

□

Intuition. We can explain Theorem 3 through two main insights. First, the mutual information from a passively collected demonstration dataset is the same regardless of when that demonstration is incorporated as long as it is conditioned on the same variables. Second, proactively generating questions based on a prior leads to more incisive queries than choosing questions from scratch. In fact, Theorem 3 can be generalized to show that active information resources should be utilized after passive resources.

Bounded Regret

At the start of this chapter we mentioned that — instead of looking for the optimal sequence of future questions — our techniques will greedily choose the best query at the current iteration. Prior work has shown that this greedy approach is reasonable for volume removal, where it is guaranteed to have bounded suboptimality in terms of the volume removed [171]. However, this volume is defined in terms of the unnormalized distribution, and so this result does not say much about the learning performance. Unfortunately, the mutual information also does not provide theoretical guarantees, as it is only submodular, but not *adaptive* submodular.

Algorithm 1 DemPref with a Human-in-the-Loop

Collect human demonstrations: $\mathcal{D}_D = \{\xi_D^{(1)}, \xi_D^{(2)}, \ldots, \xi_D^{(|\mathcal{D}_D|)}\}$
Initialize belief over the human's reward weights w: $b^0(w) \propto \exp\left(\beta^D w \cdot \sum_{\xi_D \in \mathcal{D}_D} \Phi(\xi_D)\right) P(w)$
for $i \leftarrow 1, 2, \ldots$ **do**
    Choose proactive question $Q^{(i)}$: $Q_*^{(i)} \leftarrow \arg\max_{Q^{(i)}} \; I(w; q^{(i)} \mid Q^{(i)}, b^{i-1}) - c(Q^{(i)})$
    if $I(w; q^{(i)} \mid Q^{(i)}, b^{i-1}) - c(Q^{(i)}) < 0$ **then**
        return b^{i-1}
    end if
    Elicit human's answer $q^{(i)}$ to query $Q^{(i)}$
    Update belief over w given query and response: $b^i(w) \propto P(q^{(i)} \mid Q^{(i)}, w) b^{i-1}(w)$
end for

4.2.3 Algorithm

We present the complete DemPref pseudocode with active querying in Algorithm 1. This algorithm involves two main steps: first, the robot uses the human's offline trajectory demonstrations $\mathcal{D}_D$ to initialize a high-level understanding of the human's preferred reward. Next, the robot actively generates user-friendly questions Q to fine-tune its belief b over w. These questions can be selected using volume removal or mutual information objectives (we highlight the mutual information approach in Algorithm 1). As the robot asks questions and obtains a precise understanding of what the human wants, the informative value of new queries decreases: eventually, asking new questions becomes suboptimal, and the DemPref algorithm terminates.

Advantages. Before moving to the simulation and experiment results, we conclude our presentation of DemPref with mutual information maximization by summarizing its two main contributions:

1. The robot learns the human's reward by synthesizing two types of information: high-level demonstrations and fine-grained best-of-many choice queries.
2. The robot generates questions while accounting for the human's ability to respond, naturally leading to user-friendly and informative queries.

4.2.4 Experiments

We conduct three sets of experiments to evaluate the performance of DemPref with mutual information maximization. Similar to Section 4.1, we assume a reward function that is linear in trajectory features in all experiments, i.e., $R_w(\xi) = w^\top \Phi(\xi)$ for any trajectory $\xi \in \Xi$.[4]

[4]Unless otherwise noted, we adopt $\beta^D = 0.02$, $\beta^C = 1$, constant $c(Q)$ for $\forall Q$, and assume a uniform prior over reward parameters w, i.e., $P(w)$ is constant for any $\|w\|_2 \leq 1$. We use Metropolis-Hastings algorithm [69] for sampling the set Ω from belief distribution over w.

Simulation Domains

In each experiment in addition to the experiment domains presented in Section 4.1.2, we use a subset of the following domains. These domains are shown in Figure 4.1 (and see Figure 4.3 for the domains adopted from Section 4.1.2).

Linear Dynamical System (LDS). We use a linear dynamical system with six dimensional state and three dimensional action spaces. State values are directly used as state features without any transformation.

Driver. We use a 2D driving simulator [170], where the agent has to safely drive down a highway. The trajectory features correspond to the distance of the agent from the center of the lane, its speed, heading angle, and minimum distance to other vehicles during the trajectory (white in Figure 4.1 (top)). See Appendix D.1 for the formal feature definitions.

Tosser. We use a "Tosser" robot simulation built in MuJoCo [191] that tosses a capsule-shaped object into baskets. The trajectory features are the maximum horizontal distance forward traveled by the object, the maximum altitude of the object, the number of flips the object does, and the object's final distance to the closest basket. See Appendix D.1 for the formal feature definitions.

For our user studies, we again employ the same version of the Fetch environment as in Section 4.1.2 with the physical Fetch robot (see Figure 4.3) [213].

Human Choice Models

As we described in Section 3.1, we require a probabilistic model for the human's response $q^{(i)}$ in a query $Q^{(i)}$ conditioned on their reward function parameters w. In the results we presented in this section, we use two specific models. One is the softmax model we introduced in Equation (3.10), re-stating:

$$P(q^{(i)} = Q_j^{(i)} \mid Q^{(i)}, w) = \frac{\exp(\beta^C R_w(Q_j^{(i)}))}{\sum_{j'=1}^{|Q^{(i)}|} \exp(\beta^C R_w(Q_{j'}^{(i)}))} \,. \tag{4.16}$$

As the second model, we generalize pairwise comparison queries and this preference model to include an "About Equal" option. This is similar to scale queries we introduced in Section 3.4. However, we are using a different, simpler choice model as we are not allowing any choice other than "About Equal" and the trajectory choices. We denote this new "About Equal" option by Υ and define a *weak pairwise comparison query* $Q^+ := Q \cup \{\Upsilon\}$ when $|Q| = 2$.

Building on prior work by Krishnan [126], we incorporate the information from the "About

Equal" option by introducing a minimum perceivable difference parameter $\varsigma \geq 0$, and defining:

$$P(q^{(i)} = \Upsilon \mid {Q^{(i)}}^+, w) = (\exp(2\varsigma) - 1)\, P(q^{(i)} = Q_1^{(i)} \mid {Q^{(i)}}^+, w) P(q^{(i)} = Q_2^{(i)} \mid {Q^{(i)}}^+, w)\,,$$
$$P(q^{(i)} = Q_j^{(i)} \mid {Q^{(i)}}^+, w) = \frac{1}{1 + \exp(\varsigma + R_w(Q_{j'}^{(i)}) - R_w(Q_j^{(i)}))}, \quad \{Q_j^{(i)}, Q_{j'}^{(i)}\} = {Q^{(i)}}^+ \setminus \{\Upsilon\}\,. \tag{4.17}$$

Notice that Equation (4.17) reduces to Equation (3.10) when $\varsigma = 0$; in which case we model the human as always perceiving the difference in options. All derivations in Sections 3.1, 4.1 and 4.2 hold with weak pairwise comparison queries. In particular, we include a discussion of extending our formulation to the case where ς is user-specific and unknown in Appendices B.1.1 and E.1.1. The additional parameter causes no trouble in practice. For all our experiments in this section, we set $|Q| = 2$, and $\varsigma = 1$ (whenever relevant).

We note that there are alternative choice models compatible with our framework for weak pairwise comparisons (e.g., [110] and our scale feedback model in Section 3.4). Additionally, one may generalize the weak pairwise comparison queries to $|Q| > 2$, though it complicates the choice model as the user must specify which of the trajectories create uncertainty.

We are now ready to present our three sets of experiments each of which demonstrates a different aspect of the proposed active DemPref framework:

1. The advantages of mutual information formulation over volume removal,
2. The order of demonstrations and preferences, and
3. Optimal stopping condition under the mutual information objective.

We again use the `Alignment` metric to quantitatively evaluate the performance (see Equation (4.10)).

Mutual Information vs. Volume Removal

To investigate the performance and user-friendliness of the mutual information and volume removal methods for learning from comparative feedback, we conduct experiments with simulated users in *LDS*, *Driver*, *Tosser* and *FetchReach* environments; and real user studies in *Driver*, *Tosser* and *FetchReach* (with the physical robot). We are particularly interested in the following three hypotheses:

H5. *Mutual information formulation outperforms volume removal in terms of data-efficiency.*

H6. *Mutual information queries are easier and more intuitive for the human than those from volume removal.*

H7. *A user's preference aligns best with reward parameters learned via mutual information maximization.*

To enable faster computation, we discretized the search space of the optimization problems by generating 500,000 random pairwise comparison queries and precomputing their trajectory features. Each call to an optimization problem then performs a loop over this discrete set.

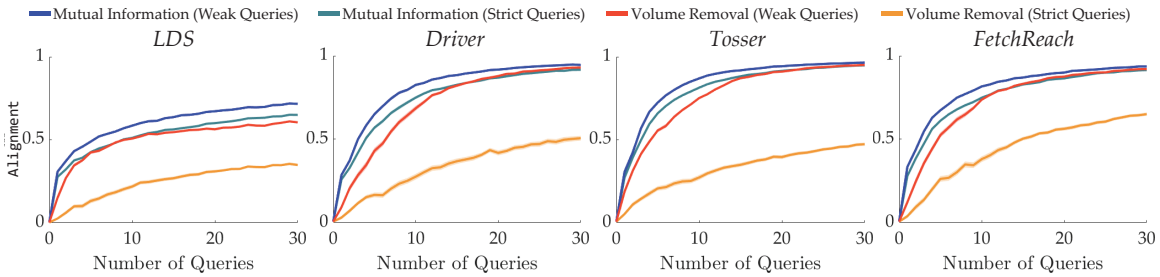


Figure 4.6: Alignment values are plotted (mean ± standard error) to compare mutual information and volume removal formulations. Standard errors are so small that they are mostly invisible in the plots. Dashed lines show the weak pairwise comparison query variants. Mutual information provides a significant increase in learning rate in all cases. While weak pairwise comparison queries lead to a large amount of improvement under volume removal, mutual information formulation is still superior in terms of the convergence rate.

In simulation experiments, we learn the randomly generated reward functions via both strict and weak pairwise comparison queries where the "About Equal" option is absent and present, respectively. We repeat each experiment 100 times to obtain confidence bounds. Figure 4.6 shows the `Alignment` value against query number for the 4 different tasks. Even though the "About Equal" option improves the performance of volume removal by preventing the trivial query, $Q = \{\xi_1, \xi_1, \dots\}$, from being a global optimum, mutual information gives a significant improvement on the learning rate both with and without the "About Equal" option in all environments.[5] These results strongly support **H5**.

The numbers given within Figure 4.7 count the wrong answers and "About Equal" choices made by the simulated users. The mutual information formulation significantly improves over volume removal. Moreover, weak pairwise comparison queries consistently decrease the number of wrong answers, which can be one reason why it performs better than strict queries.[6] Figure 4.7 also shows when the wrong responses are given. While wrong answer ratios are higher with volume removal formulation, it can be seen that mutual information reduces wrong answers especially in early queries, which leads to faster learning. These results support **H6**.

In the user studies for this part, we used *Driver* and *Tosser* environments in simulation and the *FetchReach* environment with the physical robot. We began by asking participants to rank a set of features (described in plain language) to encourage each user to be consistent in their preferences. Subsequently, we queried each participant with a sequence of 30 questions generated actively; 15 from volume removal and 15 via mutual information. We prevent bias by randomizing the sequence of questions for each user and experiment: the user does not know which algorithm generates a question.

Participants responded to a 7-point rating scale survey after each question:

[5] See Appendix E.1.2 for results without query space discretization.

[6] Another possible explanation is the information acquired by the "About Equal" responses. We analyze this in Appendix E.1.3 by comparing the results with what would happen if this information was discarded.

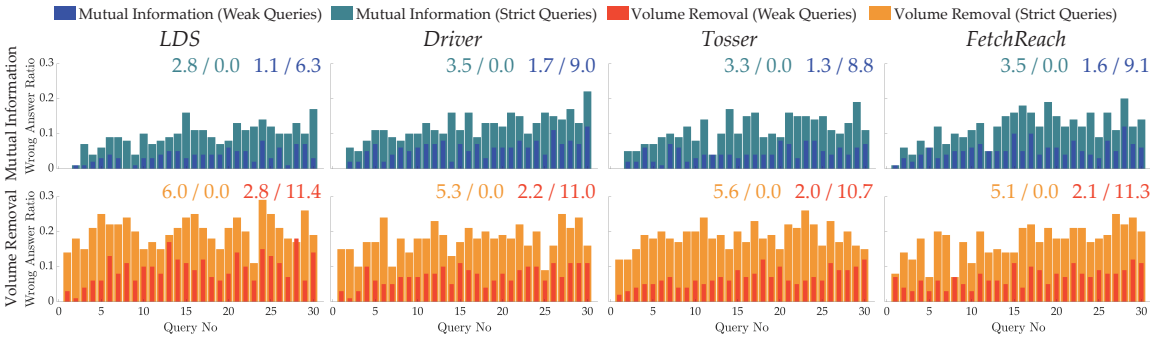


Figure 4.7: Wrong answer ratios on different queries are shown. The numbers at top show the average number of wrong responses and "About Equal" choices, respectively, for both strict and weak queries. Mutual information formulation yields smaller numbers of wrong and "About Equal" answers, especially in the early stages.

- It was easy to choose between the trajectories that the robot showed me.

They were also asked the Yes/No question:

- Can you tell the difference between the options presented?

In concluding the *Tosser* and *Driver* experiments, we showed participants two trajectories: one optimized using reward function parameters from mutual information (trajectory A) and one optimized using reward parameters from volume removal (trajectory B)[7]. Participants responded to a 7-point rating scale survey:

- Trajectory A better aligns with my preferences than trajectory B.

We recruited 15 participants (8 female, 7 male) for the simulations (*Driver* and *Tosser*) and 12 for the *FetchReach* (6 female, 6 male). We used strict pairwise comparison queries. A video demonstration of these user studies is available at `http://youtu.be/JIs43cO\_g18`.

Figure 4.8a shows the results of the easiness surveys. In all environments, users found mutual information queries easier: the results are statistically significant ($p < 0.005$, two-sample t-test). Figure 4.8b shows the average number of times the users stated they cannot distinguish the options presented. The volume removal formulation yields several queries that are indistinguishable to the users while the mutual information formulation avoids this issue. The difference is significant for *Driver* ($p < 0.05$, paired-sample t-test) and *Tosser* ($p < 0.005$). Taken together, these results support **H6**.

Figure 4.8c shows the results of the survey the participants completed at the end of experiment. Users significantly preferred the mutual information trajectory over that of volume removal in both environments ($p < 0.05$, one-sample t-test), supporting **H7**.

[7]We excluded *FetchReach* for this question to avoid prohibitive trajectory optimization (due to large action space).

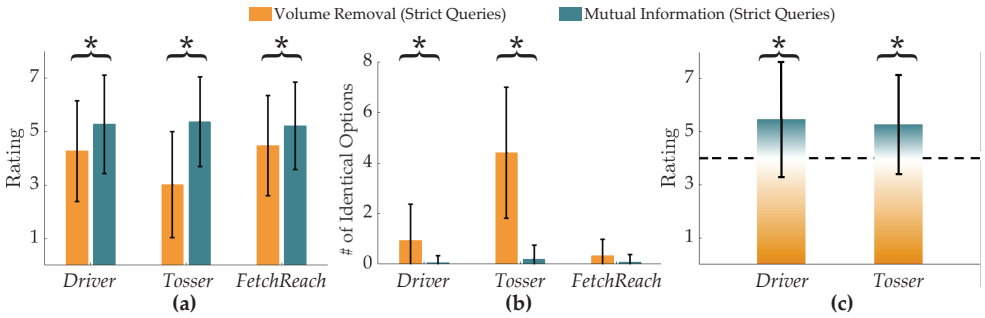


Figure 4.8: User study results. Error bars show std. Asterisks show statistical significance. **(a)** Easiness survey results averaged over all queries and users. Queries generated using the mutual information maximization method are rated significantly easier by the users than the volume removal queries. **(b)** The number of identical options in the experiments averaged over all users. In Driver and Tosser, users indicated significantly less indistinguishable queries with mutual information maximization compared to volume removal maximization. **(c)** Final preferences averaged over the users. 7 means the user strongly prefers the optimized trajectory w.r.t. the learned reward by the mutual information formulation, and 1 is the volume removal. Dashed line represents indifference between two methods. Users significantly prefer the robot who learned using the mutual information maximization method for active query generation.

The Order of Information Sources

Having seen mutual information maximization provides a significant boost in the learning rate, we checked whether the passively collected demonstrations or the actively queried preferences should be given to the model first. Specifically, we tested:

H8. *If passively collected demonstrations are used before the actively collected comparison query responses, the learning becomes faster.*

While Theorem 3 asserts that we should first initialize DemPref via demonstrations, we performed simulation experiments to check this notion in practice. Using *LDS*, *Driver*, *Tosser* and *FetchReach*, we ran three sets of experiments where we adopted weak pairwise comparison queries: (i) We initialize the belief with a single demonstration and then query the simulated user with 15 pairwise comparison questions, (ii) We first query the simulated user with 15 pairwise comparison questions and we add the demonstration to the belief independently after each question, and (iii) We completely ignore the demonstration and use only 15 pairwise comparison queries. The reason why we chose to have only a single demonstration is because having more more demonstrations tends to increase the alignment value for both (i) and (ii), thereby making the difference between the methods' performances very small. We ran each set of experiment 100 times with different, randomly sampled, true reward functions. We again used the same dataset of 500,000 queries for query generation. We also used the trajectory that gives the highest reward to the simulated user out of this dataset as the demonstration in the first two sets of experiments. Since the demonstration is not subject to noises or biases due to the control of human users, we set $\beta^D = 0.2$.

Figure 4.9 shows the `Alignment` value against the number of queries. The last set of experiments

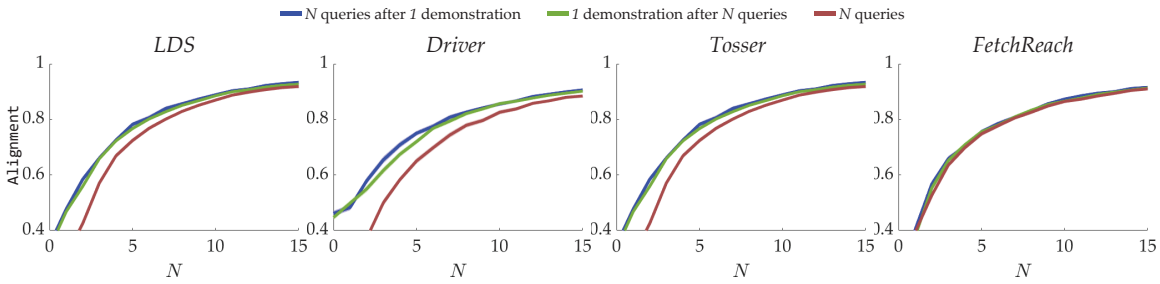


Figure 4.9: Simulation results for the order of demonstrations and preference queries. `Alignment` values are plotted (mean±s.e.). It is consistently better to first utilize the passively collected demonstrations rather than actively generated preference queries. The differences in the `Alignment` value is especially small in the *FetchReach* simulations, which might be due to the fact that it is a simpler environment in terms of the number of trajectory features.

has significantly lower `Alignment` values than the first two sets especially when the number of pairwise comparison queries is small. This indicates the demonstration has carried an important amount of information. Comparing the first two sets of experiments, the differences in the `Alignment` values are small. However, the values are consistently higher when the demonstrations are used to initialize the belief distribution. This supports **H8** and numerically validates Theorem 3.

Optimal Stopping

Finally, we experimented our optimal stopping extension for mutual information based active querying algorithm in *LDS*, *Driver*, *Tosser* and *FetchReach* environments with simulated users. Again adopting query discretization, we tested:

H9. *Optimal stopping enables cost-efficient reward learning under various costs.*

As the query cost, we employed a cost function to improve interpretability of queries, which may have the associated benefit of making learning more efficient [20]. We defined a cost function:

$$c(Q) = \varpi - |\psi_{j^*}| + \max_{j' \in \{1,\dots\} \setminus \{j^*\}} |\psi_{j'}|,\ j^* = \arg\max_j |\psi_j|,$$

where $\psi = \Phi(Q_1) - \Phi(Q_2)$. This cost favors queries in which the difference in one feature is larger than that between all other features. Such a query may prove more interpretable. We first simulate 100 random users and tune ϖ accordingly: For each simulated user, we record the ϖ value that makes the objective zero in the i^{th} query (for smallest i) such that $\texttt{Alignment}_i, \texttt{Alignment}_{i-1}, \texttt{Alignment}_{i-2} \in [x, x + 0.02]$ for some x. We then use the average of these ϖ values for our tests with 100 different random users. Figure 4.10 shows the results.[8] Optimal stopping rule enables terminating the process with near-optimal cumulative active learning rewards (the cumulative difference between the mutual information and the cost as in Equation (4.14)) in all environments, which supports **H9**.

[8] We found similar results with query-independent costs minimizing the number of queries. See Appendix E.1.4.

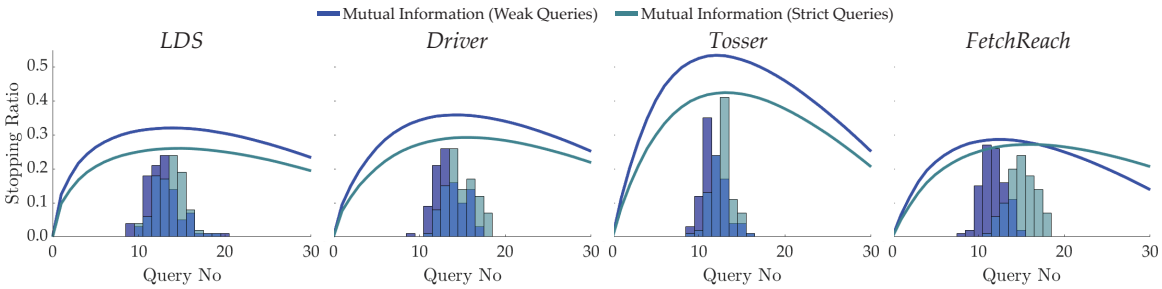


Figure 4.10: Simulation results for optimal stopping. Line plots show cumulative active learning rewards (cumulative difference between the mutual information values and the query costs), averaged over 100 test runs and scaled for visualization. Histograms show when optimal stopping condition is satisfied, which aligns with the desired cumulative rewards.

Following the same order as in Chapter 3, we now proceed with preference-based GP regression where a Gaussian process model is trained using pairwise comparisons. We extend this learning framework with mutual information based active querying to improve data-efficiency.

4.3 Active Querying for Preference-based GP Regression

While we know how to learn a non-parametric reward function f using only pairwise comparisons from Section 3.2, this endeavor can require tremendous amount of data, because each query will give at most 1 bit of information. Furthermore, we can expect a decreasing trend in the information gain as we learn the reward function. Therefore, it is important to select the queries for the human such that each query gives as much information as possible. For parametric reward models, Section 4.2 has shown that this can be done by maximizing the mutual information, which also makes the queries easy for the user. Extending this formulation to the reward functions modeled with a GP is not trivial, because one needs to sample from the GP many times for each trajectory, whereas a parametric reward form allows the reward prediction after sampling the parameters only once.

Hence, in this section, our goal is to perform mutual information maximization with GPs.

4.3.1 Formulation

Formally, we want to solve the following problem, using the same notation as in Section 3.2:

$$Q_* = (\Phi_*^{(1)}, \Phi_*^{(2)}) = \underset{\Phi^{(1)}, \Phi^{(2)}}{\arg\max}\, I(f; q \mid \Phi^{(1)}, \Phi^{(2)}, \mathbf{Q}, \mathbf{q}),$$

where I is the mutual information and q is the response to the query $Q = (\Phi^{(1)}, \Phi^{(2)})$. This optimization is equivalent to

$$\underset{\Phi^{(1)},\Phi^{(2)}}{\arg\max} \left(H(q \mid \Phi^{(1)}, \Phi^{(2)}, \mathbf{Q}, \mathbf{q}) - \mathbb{E}_{f|\mathbf{Q},\mathbf{q}} \left[H(q \mid \Phi^{(1)}, \Phi^{(2)}, f) \right] \right), \tag{4.18}$$

where H is the information entropy.

This optimization can be interpreted as follows: On one hand, maximizing the first entropy term $H(q \mid \Phi^{(1)}, \Phi^{(2)}, \mathbf{Q}, \mathbf{q})$ encourages fast convergence by maximizing the uncertainty of the outcome of every query for the learned GP model. On the other hand, minimizing the second entropy term $H(q \mid \Phi^{(1)}, \Phi^{(2)}, f)$ encourages the ease of responding to the queries by the user meaning the user should be certain about their choices.

We defer the full derivation of (4.18) to Appendix B.2, but here we give an easy-to-implement formulation of the optimization. Denoting the posterior mean of $f(\Phi^{(i)})$, which is obtained using Equation (3.16), with $\mu^{(i)}$, the objective function can be written as

$$h\left(\Phi\left(\frac{\mu^{(1)} - \mu^{(2)}}{\sqrt{2\sigma_C^2 + g(\Phi^{(1)}, \Phi^{(2)})}} \right) \right) - m\left(\Phi^{(1)}, \Phi^{(2)} \right) \tag{4.19}$$

where σ_C is the noise parameter of the human response model we introduced in Equation (3.12),

$$g(\Phi^{(1)}, \Phi^{(2)}) = \mathrm{Var}\left(f(\Phi^{(1)}) \right) + \mathrm{Var}\left(f(\Phi^{(2)}) \right) - 2\,\mathrm{Cov}\left(f(\Phi^{(1)}), f(\Phi^{(2)}) \right),$$

whose terms can be computed using Equation (3.17); h is the binary entropy function, i.e.,

$$h(p) = -p \log_2(p) - (1-p) \log_2(1-p),$$

and

$$m\left(\Phi^{(1)}, \Phi^{(2)} \right) = \frac{\sqrt{\pi \ln(2) \sigma_C^2}\, \exp\left(-\frac{(\mu^{(1)} - \mu^{(2)})^2}{\pi \ln(2) \sigma_C^2 + 2g(\Psi^{(1)}, \Psi^{(2)})} \right)}{\sqrt{\pi \ln(2) \sigma_C^2 + 2g(\Phi^{(1)}, \Phi^{(2)})}}.$$

Synthesizing queries that maximize this objective will give us very informative data points for preference-based GP regression and improve data-efficiency.

Previously, we have shown in Section 4.2 for the parametric reward models that using a mutual information based formulation accelerates the learning whereas volume removal based methods (as in Section 4.1) rely on local optima and can produce trivial queries that compare the exact same trajectory and so gives no information. In the following, we show our formulation in this section also does not suffer from this trivial query problem.

Remark 1. *The trivial query $Q = \{\Phi^{(1)}, \Phi^{(1)}\}$ does not maximize our acquisition function given in*

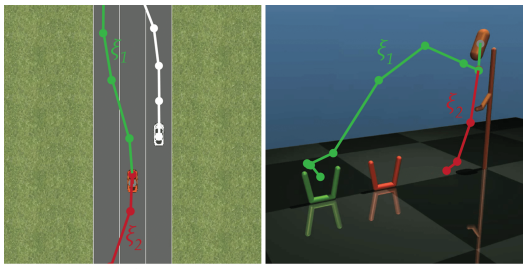


Figure 4.11: Sample trajectories are shown for the two simulation environments. In *Driver*, another car is cutting in front of the ego vehicle. In *Tosser*, the robot must hit the dropping capsule such that it will fall into one of the baskets.

(4.19), *and is in fact a global minimizer.*

Proof. For the query $Q = \{\Phi^{(1)}, \Phi^{(1)}\}$, we re-write (4.19) as

$$h\left(\Phi\left(\frac{\mu^{(1)} - \mu^{(1)}}{\sqrt{2\sigma_C^2 + g(\Phi^{(1)}, \Phi^{(1)})}}\right)\right) - m\left(\Phi\right) = 1 - m(\Phi^{(1)}, \Phi^{(1)})$$

where $\mathrm{Var}\left(f(\Phi^{(1)})\right) = \mathrm{Cov}\left(f(\Phi^{(1)}), f(\Phi^{(1)})\right)$, and so $g(\Phi^{(1)}, \Phi^{(1)}) = 0$, and

$$m\left(\Phi\right) = \frac{\sqrt{\pi \ln(2)\sigma_C^2}\exp\left(-\frac{(\mu^{(1)}-\mu^{(1)})^2}{\pi \ln(2)\sigma_C^2 + 2g(\Phi^{(1)}, \Phi^{(1)})}\right)}{\sqrt{\pi \ln(2)\sigma_C^2 + 2g(\Phi^{(1)}, \Phi^{(1)})}} = 1$$

which makes the objective value 0. Since the mutual information has to be nonnegative, this completes the proof that the trivial query is a global minimizer of the objective. □

We now proceed with our simulations and experiments on GP regression using actively collected pairwise comparison feedback.

4.3.2 Experiments

Simulation Experiments

In this subsection, we present our experiments in two simulation domains to demonstrate how (i) GP rewards improve expressiveness over linear reward functions (which is often assumed as we did in Sections 4.1.2 and 4.2.4, also see [1, 155, 226]), and (ii) active query generation improves data-efficiency over random querying.

Environments. To validate our framework on robotics tasks, we used two simulation environments from Section 4.2.4 with slight modifications: a 2D *Driver* simulation [170] and a MuJoCo [191] environment to simulate a *Tosser* robot that tries to throw an object into a basket. For reader's convenience, we again show visuals from these environments with sample trajectories in Figure 4.11.

For example in *Driver*, the user is asked whether they would move forward or backward in the given scenario. While the users would have a common response to this query, some questions may differ among the users. For instance in *Tosser*, the query asks the user whether to throw the ball into the green basket or to drop it instead. Depending on the users' preferences about the green basket, different users may have different responses.

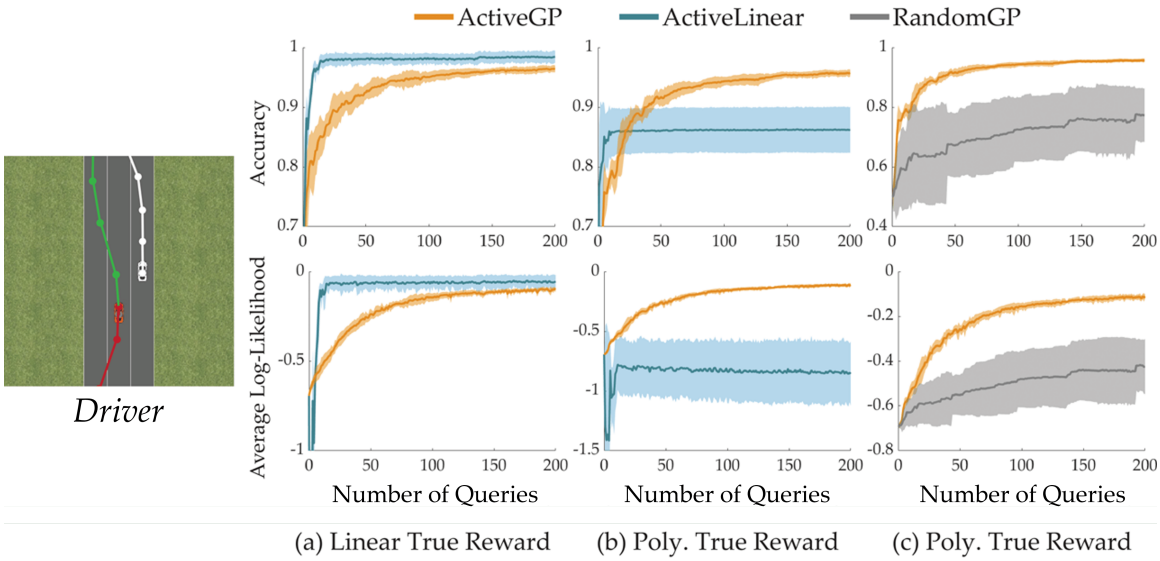


Figure 4.12: Accuracies and average log-likelihoods for test set queries are shown for the *Driver* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two polynomial. **(c)** Data-efficiency results that compare ActiveGP with RandomGP. Accuracies and average log-likelihoods for test set queries are shown (mean±std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood.

In these two environments, we use the following simple features for the function Φ:

- *Driver*: Distance to the other car, speed, heading angle, distance to the closest lane center.
- *Tosser*: The maximum horizontal range, and the number of capsule flips.

In contrast to the other sections and the prior work, here we do not need to fine-tune the feature hyperparameters to learn the reward functions because GPs can effectively capture nonlinearities.

Simulated Human Model. We simulated human responses with an underlying true reward function R with some Gaussian noise, in accordance with the probit model presented in Equation (3.12). We modeled the true R as either a degree-of-two polynomial or a linear function. In both cases, we selected the parameters of true R as i.i.d. random samples from the standard normal distribution. We repeated each simulation experiment 5 times with varying underlying true reward functions.

Baselines. For our analyses, we compared three methods:

- RandomGP: The reward is modeled using a Gaussian process. The two distinct trajectories selected in each training query are sampled from a training dataset uniformly at random.

- ActiveLinear: The reward is modeled as a linear combination of features, and the active query generation method of Section 4.2 selects the most informative comparison queries at every step of training.
- ActiveGP: The reward is modeled as a Gaussian process. We will use our active query generation method to generate the most informative comparison queries to efficiently learn the reward function.

We generated a training dataset of trajectories with uniformly randomly selected actions, as in Section 4.2.4. At every iteration of ActiveGP and ActiveLinear, we computed the mutual information of each possible query from this dataset to select the most informative query. This approach decreases the computation time compared to solving a continuous optimization over all possible trajectories as it was done in Section 4.1.2 and by [171, 159].

Evaluation. We compare GP reward with linear reward in terms of *expressiveness* (ActiveGP vs. ActiveLinear), and compare active query generation with random querying baseline in terms of *data-efficiency* (ActiveGP vs. RandomGP).

Test Set Generation. For both analyses on the expressiveness and data-efficiency, we also generated test sets of trajectories from the same distribution as the training set. However, it would not be fair to use the test set as is. Obtained with uniformly random action sequences, the majority of the training set is uninteresting trajectories, e.g. the ego car moves slightly forward and backward (similar to a random walk) in *Driver*, or the robot does not hit the capsule in *Tosser*. Using the test set without further modifications would mean we give more importance to these uninteresting behaviors as they form the majority in the datasets. Obviously, this is not the case. We want to learn the reward function everywhere in the dynamically feasible region with equal importance.

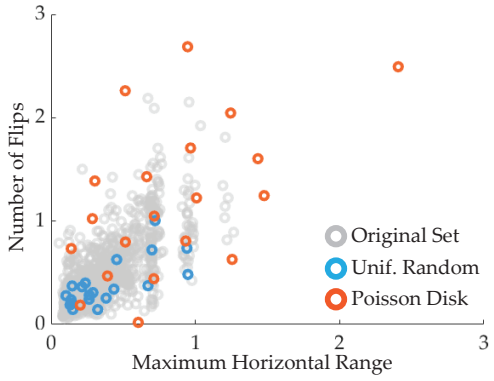


Figure 4.13: Features of 1000 *Tosser* trajectories are visualized in two-dimensional plane (gray). Poisson disk sampling allows us to obtain a diverse set of 20 samples (orange), whereas sampling uniformly at random yields mostly uninteresting trajectories (blue).

Hence, we adopted Poisson disk sampling [48] to get a diverse set of trajectories from the test

set. Poisson disk sampling makes sure the difference between trajectories[9] is above some threshold by rejecting the samples that violate this constraint. A small example set of samples is compared to uniformly random samples in Figure 4.13 for the *Tosser* environment.

After obtaining the diverse test set, we stored the true (noiseless) response of the simulated user for each possible query in this set. For the analysis on expressiveness, we computed the accuracy and the log-likelihood of the true responses under the reward functions that are learned with $|\mathcal{D}_C|$ actively chosen queries (up to $|\mathcal{D}_C| = 200$). For data-efficiency analysis, we again used the true human responses to the queries in the diverse test set (only from the polynomial reward functions) to calculate the accuracy and the log-likelihood under the learned reward functions.

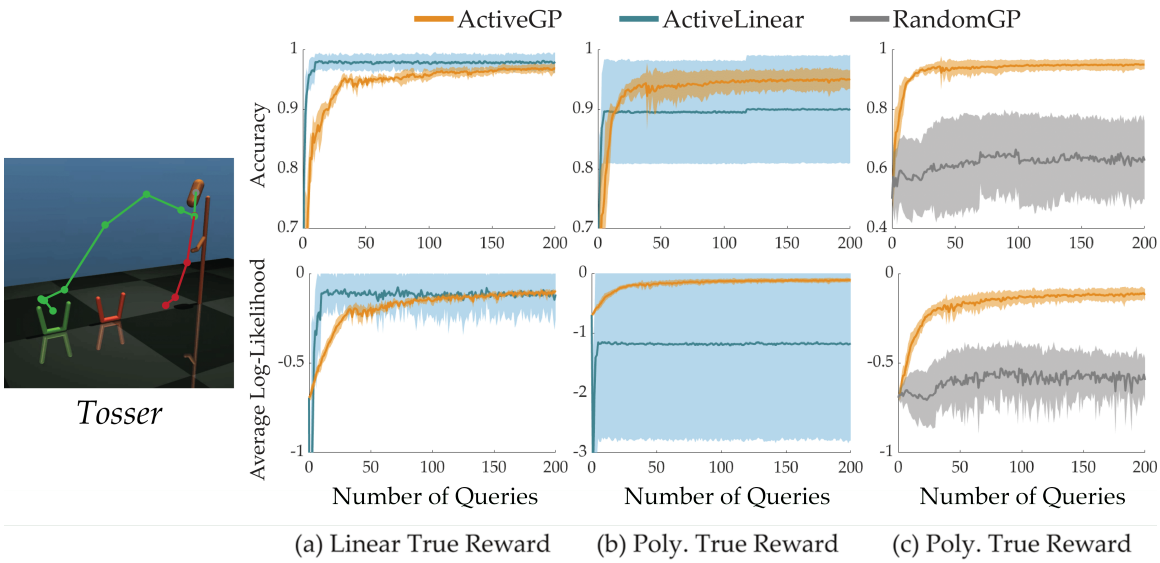


Figure 4.14: Accuracies and average log-likelihoods for test set queries are shown for the *Tosser* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two polynomial. **(c)** Data-efficiency results that compare ActiveGP with RandomGP. Accuracies and average log-likelihoods for test set queries are shown (mean±std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood.

Expressiveness. Figures 4.12a, 4.12b, 4.14a, and 4.14b show the results of expressiveness simulations. When the true reward is polynomial, the linear model results in very high variance in both accuracy and likelihood, because its performance relies on how good a linear function can explain the true nonlinear reward. In this case, the GP model captures nonlinearities better than the linear model and provides better learning (Figures 4.12b and 4.14b). When the true reward function is linear in features, a linear model naturally learns faster. However, as shown in Figures 4.12a and 4.14a, even in that case, GP model can achieve linear model's performance. To further improve the reward model, one can consider an approach to combine the linear and GP models by keeping

[9] We used L_2 distance between the feature vectors.

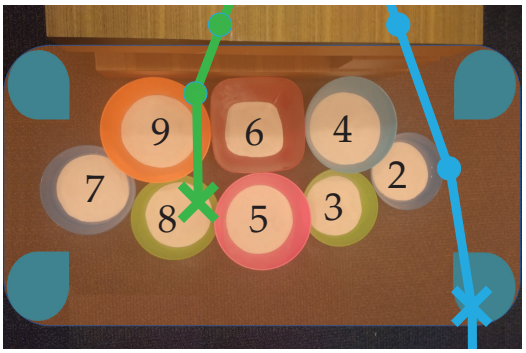


Figure 4.15: Top view of the eight targets in the variant of mini-golf user study. The users assign distinct scores from 2 to 9 to the targets. The figure shows an example of this ranking. While the robot is capable of hitting the ball into the entire shaded region, the maximizers of a linear reward always lie near the corners of the shaded region in blue. Therefore, while the GP reward model can query the user with better trajectories (e.g. the green trajectory), the linear model only explores the boundaries (e.g. the blue trajectory that throws the ball outside of this region). Crosses show where the ball hits the ground.

a belief distribution over whether the true reward is linear or not, and actively querying the user according to this belief. We leave this extension as future work.

Data-Efficiency. We then evaluated how our active query generation helps with data-efficiency. Figures 4.12c and 4.14c compare ActiveGP and RandomGP for the simulation environments. It can be seen that active querying significantly accelerates learning over random querying. It should be noted that the number of samples taken via Poisson disk sampling matters: While choosing a very small number will increase the variance in the results, choosing a very large number will make random querying seem like it performs comparable to (or even better than) the active querying as the test set will mostly consist of uninteresting trajectories, which are also abundant in the training set, as we stated earlier.

User Studies

Experiment Setup. We also compare our method ActiveGP with ActiveLinear and RandomGP on a user study with a Fetch mobile manipulator robot [213]. In this study, the human subjects teach the Fetch robot how to play a variant of mini golf where the robot can achieve different scores by hitting the ball to different targets (see Figures 3.3 and 4.15 for the setup). However, these scores are only known to the human. In fact, the robot does not even know the locations of the targets, and it tries to learn the reward as a function of its control inputs. Fixing some of the joints, we let the robot vary only its shot speed and angle, which are also the features of the reward function.

This experiment setting is interesting because a linear reward function can only encode whether the robot must hit the ball to the right or to the left, or whether it must hit with high or low speed. It cannot particularly encourage (or discourage) hitting with a modest angle and/or speed.

Therefore, as we show in Figure 4.15, the targets that are around the middle region cannot be the maximizers of a linear reward function.

Subjects and Procedure. We recruited 10 users (6 male, 4 female) with an age range from 19 to 28. Each user first assigned their distinct scores (from 2 to 9) to the eight targets. The robot then queried them with 50 pairwise comparison questions: 15 for ActiveGP, 15 for ActiveLinear, 15 for RandomGP and 5 queries generated uniformly at random to create a test set. We shuffled the order of queries to avoid any bias. We used the reward models, each of which is learned with 15 queries, to predict the user responses in the test set. The prediction score on the test set provides an accuracy metric.

In addition to the accuracy, we assessed whether the robot could successfully learn how to perform a good shot. For this, after the subjects responded to 50 queries, the robot demonstrated 3 more trajectories each of which corresponds to the optimal trajectory of one method, the trajectory that maximizes the learned reward function. Again, the order of these trajectories was shuffled. After watching each demonstration, the subjects assigned a score to the shot from a 9-point rating scale (1-very bad, 9-very good).

Results and Discussion We provide a video that gives an overview of user studies and their results at `https://youtu.be/SLSO2lBj9Mw`.

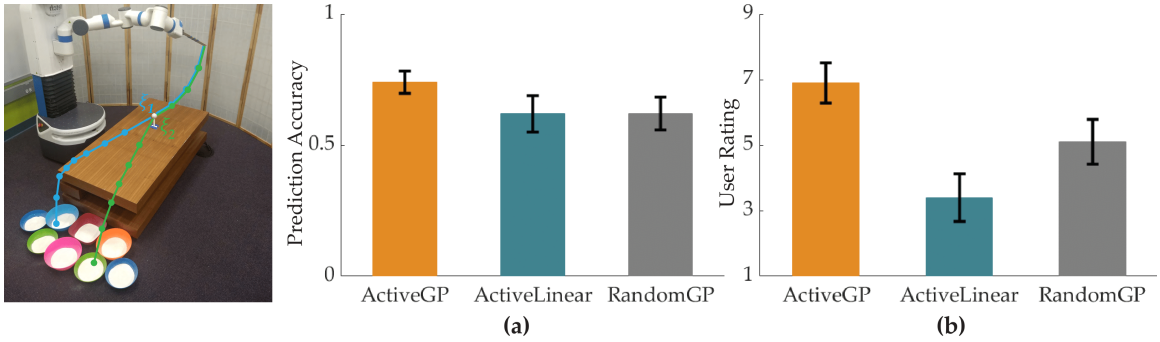


Figure 4.16: **(a)** Prediction accuracy results (mean±se). Each trained with 15 queries, ActiveGP achieves significantly higher prediction accuracy than both ActiveLinear and RandomGP ($p < 0.05$). **(b)** User ratings on the final robot performance (mean±se). ActiveGP accomplishes the task significantly better than both ActiveLinear and RandomGP ($p < 0.05$).

Figure 4.16a shows the prediction accuracy values on the test sets collected from the subjects (averaged over the subjects). By modeling the reward using a GP and querying the users with the most informative questions, ActiveGP achieves significantly higher prediction accuracy (0.74 ± 0.04, mean±se) compared to both ActiveLinear (0.62 ± 0.07) and RandomGP (0.62 ± 0.06) with $p < 0.05$ (Wilcoxon signed rank test). The results from this user study are aligned with our simulation studies.

In reward learning, it is crucial to validate whether the learned reward function can encode the desired behavior or not. Figure 4.16b shows the user ratings to the trajectories that the robot showed

after learning the user preferences via 3 different methods. ActiveGP obtains significantly higher scores (6.9 ± 0.6) than both ActiveLinear (3.4 ± 0.7) and RandomGP (5.1 ± 0.7) with $p < 0.05$. While ActiveLinear occasionally achieves high scores when the users' preferred target is near the edge, it generally fails to produce the desired behavior due to its low expressive power.

The next section will present more simulation and experiment results with active preference-based GP regression, after discussing active querying in the presence of ordinal feedback.

4.4 ROI Active Learning with Comparisons and Ordinal Feedback

Having seen the success of mutual information maximization based active querying for non-parametric reward functions in Section 4.3.2, we want to extend it in the same way as we did in Chapter 3: we want to be able to utilize ordinal feedback in addition to pairwise comparisons. Furthermore, in Section 3.3, we established that it is sometimes desired to define a "region of avoidance" (ROA) in the trajectory space Ξ such that we should not query the user with the trajectories in that space. This is especially relevant when we are actively querying the user: we should constrain our active querying optimization to avoid ROA.

Therefore in this section, we extend the GP regression model we presented in Section 3.3 with active querying. We again use mutual information maximization. However, differently from Section 4.3, we now select the trajectories such that:

1. Each trajectory is compared with the previous trajectory in the querying sequence (as opposed to optimizing for a couple of trajectories for pairwise comparisons),
2. We want to maximize the information from not only the comparative feedback, but also the ordinal feedback,
3. We try to avoid querying the user with trajectories from ROA.

We call our algorithm ROIAL, short for region of interest active learning.

At the end of this section, we will demonstrate in simulation that ROIAL estimates both the region of interest (ROI) and the reward function within the ROI with high accuracy. We experimentally demonstrate ROIAL on the lower-body exoskeleton Atalante (Figure 3.4) to learn the reward functions of three non-disabled users over four gait parameters. The obtained landscapes highlight both agreement and disagreement in preferences among the users. Previous algorithms for exoskeleton gait optimization were incapable of drawing such conclusions; thus, this work represents progress towards establishing a better understanding of the science of walking with respect to exoskeleton gait design.

4.4.1 Formulation

ROIAL selects samples by modeling a Bayesian posterior over the reward function using Gaussian processes and maximizing the mutual information (over the ROI) with respect to this posterior. In this section, we continue to use the notation in Section 3.3, as we are now making it active.

Defining $\hat{\boldsymbol{f}}^{(i)} := [\hat{f}^{(i)}(\Phi(\xi^{(1)})), \ldots, \hat{f}^{(i)}(\Phi(\xi^{(|\Xi|)}))]^\top$ as the maximum a posteriori (MAP) estimate of the rewards $\boldsymbol{f}$ given $\mathcal{D}^{(i)}$, we aim to adaptively select the next trajectories $\xi^{(1)}, \xi^{(2)}, \ldots \in \Xi$ that minimize the error in estimating $\boldsymbol{f}$ over the ROI. We model the error as $\text{Error}(i) := \sum_{\xi \in \text{ROI}} |\boldsymbol{f} - \hat{\boldsymbol{f}}^{(i)}|$, where the absolute value is taken element-wise.

Trajectory Selection via Mutual Information Maximization

To learn the reward function in as few trials as possible, we select trajectories to maximize the mutual information between the reward function and the comparison-based and ordinal human feedback. We again adopt the greedy approach as in the previous sections to solve the following optimization in each iteration i:

$$\max_{\xi^{(i)} \in \text{ROI}^{(i)}} I(\boldsymbol{f}; q_o^{(i)}, q^{(i)} \mid \mathcal{D}^{(i-1)}, \xi^{(i)}), \tag{4.20}$$

where $q^{(i)}$ denotes the user's response to a pairwise comparison query between $\xi^{(i)}$ and $\xi^{(i-1)}$, and $q_o^{(i)}$ denotes the ordinal feedback for trajectory $\xi^{(i)}$. One can re-write (4.20) in terms of information entropy:

$$\max_{\xi^{(i)}} H(q_o^{(i)}, q^{(i)} \mid \mathcal{D}^{(i-1)}, \xi^{(i)}) - \mathbb{E}_{\boldsymbol{f} \mid \mathcal{D}^{(i-1)}} \left[H(q_o^{(i)}, q^{(i)} \mid \mathcal{D}^{(i-1)}, \xi^{(i)}, \boldsymbol{f}) \right].$$

Again, we can interpret the first term as the uncertainty about trajectory $\xi^{(i)}$'s ordinal label and preference relative to $\xi^{(i-1)}$. We aim to maximize this term, because queries with high model uncertainty could potentially yield significant information. The second term is conditioned on $\boldsymbol{f}$, and so represents the user's expected uncertainty. If the user is very uncertain about their feedback, then the action $\xi^{(i)}$ gives only a small amount of information. Hence, we aim to minimize this second term. In this way, mutual information maximization produces queries that are both informative and easy for users.

The second term is estimated via sampling from the Laplace-approximated Gaussian posterior $P(\boldsymbol{f} \mid \mathcal{D}^{(i-1)})$. Computing the first term requires the probability $P(q_o^{(i)}, q^{(i)} \mid \mathcal{D}^{(i-1)}, \xi^{(i)})$. We derive it as:

$$P(q_o^{(i)}, q^{(i)} \mid \mathcal{D}^{(i-1)}, \xi^{(i)}) = \int_{\mathbb{R}^{|\Xi|}} P(\boldsymbol{f} \mid \mathcal{D}^{(i-1)}, \xi^{(i)}) P(q_o^{(i)}, q^{(i)} \mid \mathcal{D}^{(i-1)}, \xi^{(i)}, \boldsymbol{f}) d\boldsymbol{f} \tag{4.21}$$

$$= \mathbb{E}_{\boldsymbol{f} \mid \mathcal{D}^{(i-1)}} \left[P(q_o^{(i)}, q^{(i)} \mid \mathcal{D}^{(i-1)}, \xi^{(i)}, \boldsymbol{f}) \right], \tag{4.22}$$

Algorithm 2 ROIAL Algorithm

Require: Reward prior parameters; ordinal thresholds $B_1^o, \ldots, B_{|B^o|}^o$; subset size $|\Xi_S^{(i)}|$ for $\forall i$; confidence parameter ε

1: $\mathcal{D}^{(0)} = \emptyset$, ▷ $\mathcal{D}^{(i)}$: user feedback dataset including iteration i
2: Select a trajectory $\xi^{(1)}$ at random
3: Add ordinal feedback to data to obtain $\mathcal{D}^{(1)}$
4: **for** i = 2,... **do**
5: Update the model posterior $P(\boldsymbol{f} \mid \mathcal{D}^{(i-1)})$ ▷ Equation (3.18)
6: Determine $\Xi_S^{(i)}$ by randomly selecting $|\Xi_S^{(i)}|$ actions
7: Determine $\text{ROI}^{(i)} \subset \Xi_S^{(i)}$
8: $\xi^{(i)} \leftarrow \arg\max_{\xi \in \text{ROI}^{(i)}} I(\boldsymbol{f}; q_o^{(i)}, q^{(i)} \mid \mathcal{D}^{(i-1)}, \xi)$
9: Add preference and ordinal feedback to data to obtain $\mathcal{D}^{(i)}$
10: **end for**

which we approximate with samples from $P(\boldsymbol{f} \mid \mathcal{D}^{(i-1)})$.

ROIAL Algorithm

Algorithm 2 presents the pseudocode for the ROIAL algorithm we develop. Line 8 solves the mutual information maximization problem, whereas the procedures for lines 5-7, which include learning and estimating ROI, were presented in Section 3.3.

4.4.2 Simulations and Experiments

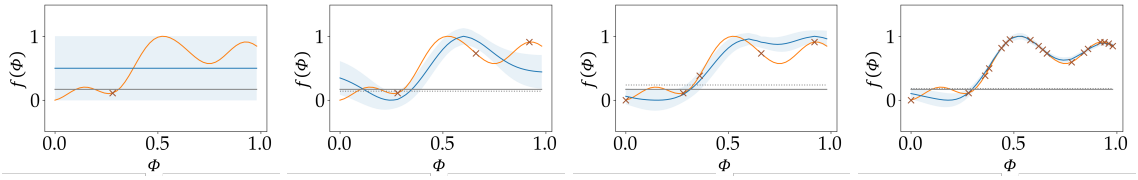
Simulation Results

We evaluate ROIAL's performance on the Hartmann3 function—which is a standard benchmark for learning non-convex, smooth functions—and on 3-dimensional synthetic functions, sampled from a Gaussian process prior over a $20 \times 20 \times 20$ grid. As evaluation metrics, we use the algorithm's errors in pairwise comparison and ordinal label prediction; these allow us to quantify performance when the true reward function is unknown. The average ordinal prediction error is defined as $\overline{\text{Error}}(i) := \frac{1}{i} \sum_{i'=1}^{i} |\hat{q}_o^{(i')} - {q_o^{(i')}}^*|$, and all simulations use 5 ordinal categories.[10]

1D illustration of ROIAL. Figure 4.17 illustrates the algorithm for a 1D objective (reward) function. Initially, ROIAL samples widely across the space (Figure 4.17a-4.17c). As seen by comparing iterations 5 and 20 (Figure 4.17c-4.17d), the algorithm stops querying points in the ROA (points in $\mathcal{B}_1^o$) because the upper confidence bound (top of the blue shaded region) there falls below the hyperparameter B_1^o (dotted gray line).

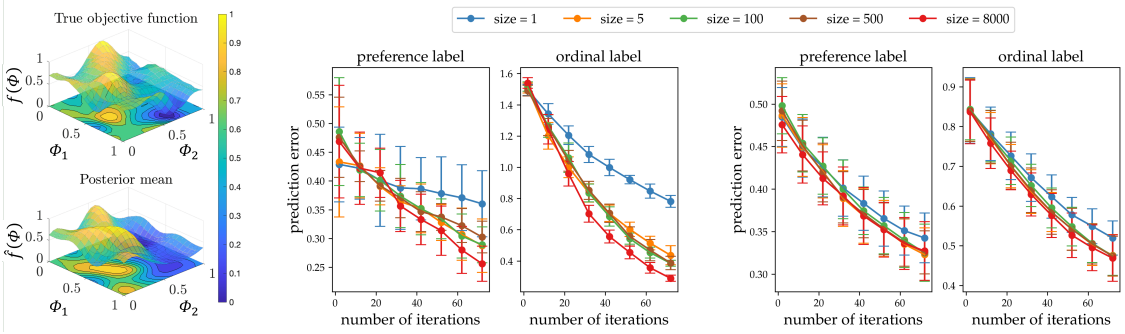
Extending to higher dimensions. To characterize the impact of the random subset size on

[10] Unless otherwise stated, hyperparameters are held constant across simulations and experiments, and their values can be found in `https://github.com/kli58/ROIAL`.



(a) Iteration 1 (b) Iteration 3 (c) Iteration 5 (d) Iteration 20

Figure 4.17: 1D posterior illustration. The true objective function is shown in orange, and the algorithm's posterior mean is blue. Blue shading indicates the confidence region for $\varepsilon = 0.5$. The solid grey line indicates the true ordinal threshold B_1^o: the ROI is above this threshold, while the ROA is below it. The dotted grey line is the algorithm's B_1^o hyperparameter. The actions queried so far are indicated with "x"s. Utilities are normalized in each plot so that the posterior mean spans the range from 0 to 1.

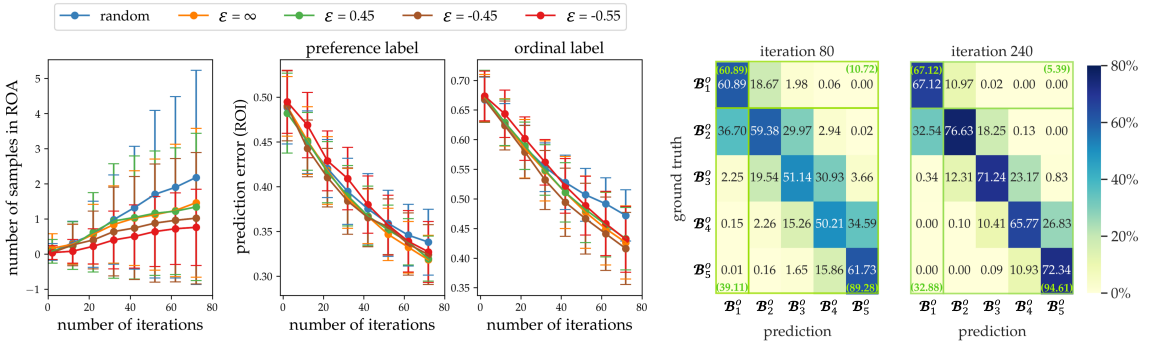


(a) Synthetic function posterior (b) Hartmann3 prediction error (c) Synthetic function prediction error

Figure 4.18: Impact of random subset size on algorithm performance. a) Example 3D synthetic objective function and posterior learned by ROIAL with subset size = 500 after 80 iterations. Values are averaged over the 3$^{\text{rd}}$ dimension and normalized to range from 0 to 1. b-c) Algorithm's error in predicting preferences and ordinal labels (mean ± std). Each simulation evaluated performance at 1000 randomly- selected points; the model posterior was used to predict preferences between consecutive pairs of points and ordinal labels at each point.

algorithmic performance, we compare performance of different sizes in simulation for both the Hartmann3 and synthetic reward functions. We calculate the posterior over the entire space only every 10 steps to reduce computation time, and then use this posterior to evaluate the algorithm's error in predicting preference and ordinal labels. Figure 4.18a provides an example of a 3D posterior, Figure 4.18b depicts the average performance for Hartmann3 over 10 simulation repetitions, and Figure 4.18c shows the average performance over a set of 50 unique synthetic functions. We find that a subset size of at least 5 yields performance close to using all points.

Estimating the region of interest. We demonstrate the effect of the confidence parameter ε on the number of points sampled from the ROA and on prediction error in the ROI. Figure 4.19a demonstrates that across various values of ε, visits to the ROA decrease as ε decreases. To confirm that restricting queries to the estimated ROI does not harm performance, we also compare label



(a) Number of samples in the ROA and prediction error in the ROI (b) Confusion matrices

Figure 4.19: Effect of the confidence interval. All simulations are run over 50 reward synthetic functions with a random subset size of 500. a) Left: cumulative number of points in the ROA ($\mathcal{B}_1^o$) queried at each iteration (mean ± std). Note that as ε increases, more samples are required for the confidence interval to fall below the ROA threshold, at which point ROIAL starts avoiding the ROA. Middle and right: error in predicting comparison and ordinal labels for different values of ε; predictions are over 1,000 random actions (mean ± std). b) Confusion matrices (column-normalized) of ordinal label prediction over the entire action space at iterations 80 and 240 with ε = -0.45. The 2×2 confusion matrices for ROI prediction accuracy are outlined in green. Prediction accuracy increases with the number of iterations.

prediction error in the ROI across values of ε. When $\varepsilon = -0.45$, ROIAL achieves similar preference prediction accuracy and slightly-improved ordinal label prediction within the ROI compared to $\varepsilon = \infty$, which permits sampling over the entire space (Figure 4.19a). Additionally, the confusion matrix (Figure 4.19b) shows that the algorithm usually predicts either the correct ordinal label or an adjacent ordinal category. The ROI prediction accuracy (green text in Figure 4.19b) indicates that ROIAL predicts whether points belong to the ROI with relatively-high accuracy.

Robustness to noisy feedback. Since user feedback is expected to be noisy, we evaluate the algorithm's robustness to noisy feedback generated from the distributions $P(q_o \mid \boldsymbol{f}, \xi) = g_O\left(\frac{B^o_{q_o} - f(\Phi(\xi))}{\sigma_O}\right) - g_O\left(\frac{B^o_{q_o-1} - f(\Phi(\xi))}{\sigma_O}\right)$ and $P(\xi^{(1)} \succ \xi^{(2)} \mid \boldsymbol{f}) = g_C\left(\frac{f(\Phi(\xi^{(1)})) - f(\Phi(\xi^{(2)}))}{\sigma_C}\right)$ for ordinal and pairwise comparison feedback, respectively, with true ordinal thresholds $\{B^o_j \mid j = 0, \ldots, |\mathcal{B}^o|\}$ and simulated noise parameters σ_C and σ_O. We set $\sigma_O > \sigma_C$ because we expect ordinal labels to be noisier than pairwise comparisons, as they require users to recall all past experience to give consistent feedback, whereas a pairwise comparison only involves the previous and current points (or trajectories). The algorithm learns more slowly with noisier feedback (Figure 4.20).

Exoskeleton Experiments

After demonstrating ROIAL's performance in simulation, we experimentally deployed it on the lower-body exoskeleton Atalante, developed by Wandercraft (video: `https://youtu.be/04lMJmKmZrQ`, ROIAL hyperparameters: `https://github.com/kli58/ROIAL`). Atalante, shown in Figure 3.4, is

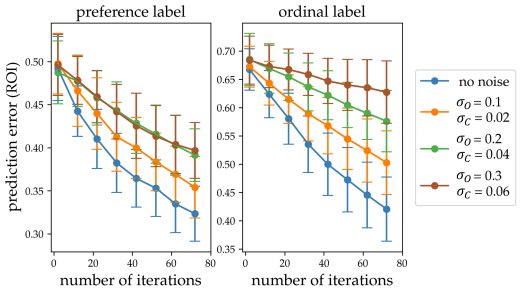


Figure 4.20: Effect of noisy feedback. The ordinal and pairwise comparison noise parameters, σ_O and σ_C, range from 0.1 to 0.3 and 0.02 to 0.06, respectively. All cases use a random subset size of 500 and $\varepsilon = -0.45$, and each simulation uses 1,000 random points to evaluate label prediction. Plots show means $\pm$ standard deviation.

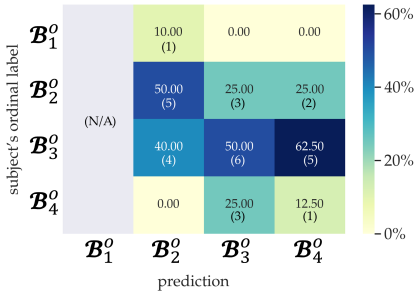


Figure 4.21: Confusion matrix of the validation phase results for all three subjects. The first column is gray because trajectories in the ROA ($\mathcal{B}_1^o$) were purposefully avoided to prevent subject discomfort. Percentages are normalized across columns. Parentheses show the numbers of gait trials in each case.

an 18 degree of freedom robot designed to restore assisted mobility to patients with motor complete paraplegia through the control of 12 actuated joints: 3 joints at each hip, 1 joint at each knee, and 2 degrees of actuation in each ankle. For more details on Atalante, refer to [3, 106, 102].

Dynamically stable crutch-less exoskeleton walking gaits are generated through nonconvex optimization techniques (see Section II of Tucker et al. [193]), based on the theory of hybrid zero dynamics (HZD) introduced by Ames [9] and the HZD-based optimization method presented in [108]. These periodic gaits are parameterized by various features, and this studies focuses on four: step length (SL) in meters, step duration (SD) in seconds, maximum pelvis roll (PR) in degrees, and maximum pelvis pitch (PP) in degrees (Figure 3.4). These parameters were selected because exoskeleton users frequently suggested modifications to SL, SD, and PR in prior work (see `https://sites.google.com/view/roial-icra2021`), and we wanted to further study the relationship between PR and PP. We discretized these parameters into bins of sizes 10, 7, 5, and 5, respectively, resulting in 1,750 trajectories within a 4D gait (or feature) space. ROIAL randomly selected 500 trajectories in each iteration and used $\varepsilon = 0.45$ to estimate the ROI.

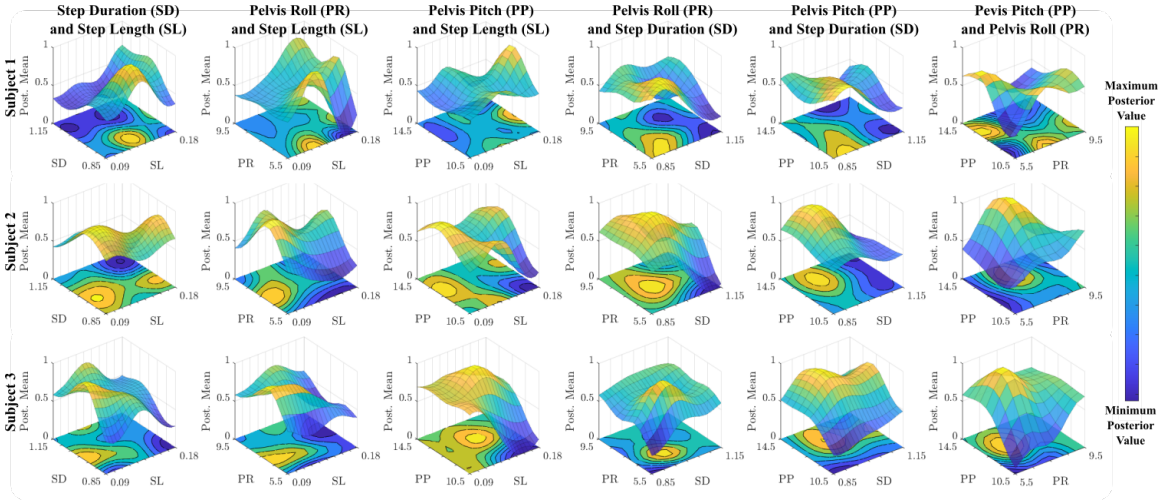


Figure 4.22: 4D posterior mean reward across exoskeleton gaits. Rewards are plotted over each pair of gait space parameters, with the values averaged over the remaining 2 parameters in each plot. Each row corresponds to a subject: Subject 1 is the most experienced exoskeleton user, Subject 2 is the second-most experienced user, and Subject 3 never used the exoskeleton prior to the experiment.

The experimental procedure was conducted for three non-disabled subjects and consisted of 40 trials divided into a *training phase* (30 trials) and a *validation phase* (10 trials). Subjects were not informed of when the validation phase began. Subjects provided ordinal labels for all 40 gaits, and optional pairwise comparisons between the current and previous gaits for all but the first trial.[11] Four ordinal categories were considered and described to the users as:

1. **Very Bad** ($\mathcal{B}_1^o$): User feels unsafe or uncomfortable to the point that the user never wants to repeat the gait.
2. **Bad** ($\mathcal{B}_2^o$): User dislikes the gait but does not feel unsafe or uncomfortable.
3. **Neutral** ($\mathcal{B}_3^o$): User neither dislikes nor likes the gait and would be willing to try the gait again.
4. **Good** ($\mathcal{B}_4^o$): User likes the gait and would be willing to continue walking with it for a long period of time.

While including additional ordinal categories could increase the potential information gain from each query, it also increases the cognitive burden for the users and thus makes the labels less reliable. Validation gaits were selected so that at least two samples were predicted to belong to $\mathcal{B}_2^o, \mathcal{B}_3^o$, and $\mathcal{B}_4^o$, with the remaining four validation gaits sampled at random. Gaits predicted to belong in $\mathcal{B}_1^o$ were excluded because they are likely to make the user feel uncomfortable or unsafe, and gaits sampled during the training phase were explicitly excluded from the validation trials.

Experimental results. Figure 4.21 depicts the results of the validation phase for all three subjects.

[11] The users were given chance to not give pairwise comparisons, in which case we simply update the posterior only with the ordinal feedback.

These results show a reliable correlation between the predicted categories and the users' reported ordinal labels, in which the majority of the predicted ordinal labels are within one category of the true ordinal labels. Since less than 2% of the gait space was explored during the experiment, we expect that the prediction accuracy would increase with additional exoskeleton trials as observed in simulation (Figure 4.19b). Overall, these results suggest that ROIAL can yield reliable preference landscapes within a moderate number of samples.

Figure 4.22 depicts the final posterior mean for each of the subjects. These reward functions highlight both regions of agreement and disagreement among the subjects. For example, all subjects strongly dislike gaits at the lower bound of PP and lower bound of PR. However, all subjects disagree in their reward landscapes across SL and SD. This type of insight could not be derived from direct gait optimization, which mostly obtains information near the optimum.

We also evaluated the effect of each gait parameter on the posterior rewards using the permutation feature importance metric. The results of this test for each respective subject across the four gait parameters (SL, SD, PR, PP) are: (0.20, 0.30, 0.33, 0.27), (0.26, 0.36, 0.38, 0.29), and (0.23, 0.16, 0.21, 0.45). These values suggest that the preferences of more experienced users (Subjects 1 and 2) may be most influenced by SD and PR, while the least-experienced user's feedback may be most weighted by PP (Subject 3). The code for this test is available on GitHub: `https://github.com/kli58/ROIAL`. These results demonstrate that ROIAL is capable of obtaining preference landscapes within relatively-few exoskeleton trials while avoiding gaits that make users feel unsafe or uncomfortable.

4.5 Active Querying for Scale Feedback

After presenting how to actively learn a non-parametric reward function using GPs and mutual information maximization, we go back to parametric reward functions and continue our presentation with other forms of comparative feedback, just like we did in Chapter 3. Following the same structure, we proceed with scale feedback. In this section, we again use the mutual information based active querying method, but we also introduce the max regret method (originally used by Wilde et al. [207]) to generate the scale queries. Afterwards, we present our simulation and experiment results with these two acquisition functions in Section 4.5.2.

4.5.1 Two Acquisition Functions for Active Scale Feedback

To learn the true reward parameters w^* efficiently, the robot actively chooses the query $Q^{(i)}$ it presents to the user at every iteration i. Two approaches for learning from pairwise comparisons are mutual information maximization (Sections 4.2 through 4.4) and max regret optimization [207].

Mutual information based active querying method seeks to reduce the robot's uncertainty over w while choosing queries that are easy to answer for the user. Max regret optimization, on the other

hand, minimizes the maximum regret (as defined in Equation (3.22)) by showing mutual worst case trajectories, which also results in easy queries. We leverage both of these methods for our active query generation in scale feedback.

Mutual Information Maximization

We start with the mutual information. Letting H denote Shannon's information entropy [202], a greedy step takes the expectation over the user's response $q^{(i)}$ to the query $Q^{(i)}$ being optimized to actively learn both the reward parameters w and the sensitivity threshold ϱ (see Definition (3)):

$$Q^{(i)} = \arg\max_{Q^{(i)}} H(w, \varrho \mid Q^{(i)}, b^i) - \mathbb{E}_{q^{(i)}|Q^{(i)},b^i}\big[H(w, \varrho \mid q^{(i)}, Q^{(i)}, b^i)\big]. \tag{4.23}$$

Noting the belief distribution b^i was defined over both w and ϱ when we use scale feedback, we approximate the computation of entropies by summing over a set Ω of samples of $(w, \varrho) \sim b^i$. Thus, following the derivation in Section 4.2 (thereby Appendix B.1), the new query $Q^{(i)}_*$ solves

$$Q^{(i)}_* = \arg\max_{Q^{(i)}} \sum_{q^{(i)}} \sum_{(w,\varrho)\in\Omega} \frac{P(q^{(i)} \mid Q^{(i)}, w, \varrho)}{|\Omega|} \log_2 \left(\frac{|\Omega| \cdot P(q^{(i)} \mid Q^{(i)}, w, \varrho)}{\sum_{(w',\varrho')\in\Omega} P(q^{(i)} \mid Q^{(i)}, w', \varrho')} \right). \tag{4.24}$$

Max Regret Optimization

The max regret policy generates queries $Q^{(i)} = (Q^{(i)}_1, Q^{(i)}_2)$ such that if the robot learned $Q^{(i)}_1$ as the optimal trajectory but the user optimal solution would be $Q^{(i)}_2$ is a worst case. With a symmetric perspective over $Q^{(i)}_1$ and $Q^{(i)}_2$, we solve

$$\max_{w,\varrho,w',\varrho'} b^i(w, \varrho) b^i(w', \varrho') \Big(\mathcal{R}(w, w') + \mathcal{R}(w', w) \Big), \tag{4.25}$$

where $\mathcal{R}(\cdot, \cdot)$ is the reward difference (regret) defined in Equation (3.22). The optimal query with respect to max regret optimization is then $Q^{(i)} = (Q^{(i)}_1, Q^{(i)}_2)$ such that $Q^{(i)}_1$ and $Q^{(i)}_2$ are the optimal trajectories with respect to w and w', respectively. By observing feedback to such queries it greedily improves the probabilistic worst case error. In contrast to the mutual information based approach, obtaining queries through max regret optimization requires $Q^{(i)}_1$ and $Q^{(i)}_2$ to be optimal trajectories for some users (w, ϱ) and (w', ϱ'). On the other hand, maximum regret does not require a one-step look-ahead and thus no summation over potential feedback values $q^{(i)}$, making it computationally lighter.

Optimizations in (4.24) and (4.25) now give us two different policies for actively solving the initial reward learning via scale feedback problem posed in Section 3.4.1. In the simulations, we compare how the performance of both benefits from scale feedback.

4.5.2 Experiments

Simulation Results

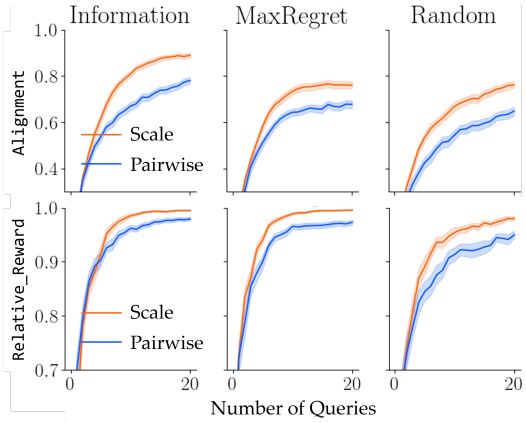


Figure 4.23: Comparison of scale feedback and weak pairwise comparisons for different active querying methods.

We now present our main simulation results. Additional results can be found in Appendix E.2. In all simulations and experiments, we assume a parametric reward function that is linear in trajectory features, similar to Sections 4.1.2 and 4.2.4.

Experiment Setup. We simulate the presented framework using the *Driver* experiment used in [171, 207, 22] and Sections 4.2.4 and 4.3.2. We modify the setup by adding 6 new features, obtaining a more challenging 10-dimensional problem (details on the features and the results for the original *Driver* can be found in Appendices D.2 and E.2, respectively). 71 distinct true reward parameters w^* are drawn uniformly at random, and each user is simulated with $\varrho^* \in \{0.25, 0.5, 0.75, 1.00\}$, making it 284 runs for each method. We set $\sigma_S = 0.1$ for the noise level. We generate a set of 200 distinct sample trajectories by drawing random reward parameters w and then computing their optimal trajectories. The active query generation methods then optimize over this set. We evaluate learning using the `Alignment` metric and the `Relative_Reward` (see Section 3.4 for their definitions).

As a baseline we use weak pairwise comparisons (strict pairwise comparisons showed a slightly poorer performance). To ensure a fair comparison, we emulate weak pairwise comparisons by setting the step size to $\nu = 1$ and use the same noise model for both forms of feedback (as opposed to using an alternative model, such as the one presented in Equation (4.17)).

Results. Figure 4.23 shows the `Alignment` and `Relative_Reward` for the *Driver* experiment for mutual information, max regret and random query generation. We observe that in all cases scale feedback significantly improves the performance over weak pairwise comparisons in both metrics ($p < 0.001$ in all cases with two-sample t-test). When using the proposed scale feedback, the `Alignment` after 20 iterations improves from 0.77 to 0.86 for mutual information, from 0.67 to 0.76

for max regret, and from 0.64 to 0.75 for random queries. The `Relative_Reward` improves for mutual information and max regret similarly from 0.97 to 1.00, i.e., the learned solution is optimal. Both methods make most progress during the first 10 iterations. Random queries improve the final relative reward from 0.94 to 0.97. Overall, the simulation showcases that scale feedback improves learning, independent of the query selection method. For mutual information and max regret, scale feedback allows for finding optimal solution within a small budget of iterations. Appendix E.4 presents the numerical results for all plots in this section. In Appendix E.2, we show additional simulation results for higher noise.

User Study

Next, we analyze the scale feedback in comparison with pairwise comparisons and under different active querying methods with two user studies.[12] In both studies, we used $\nu = 0.1$ for scale queries.

Experiment Setup. We designed a serving task with a Fetch robot [213] as shown in Figure 3.5, which we call *FetchDrink*. We generated a dataset of 120 distinct trajectories. Human subjects were told they should train the robot to bring the drink to the customer in the manner they prefer, paying attention to the following five factors: the drink (out of 3 options) to be served, the orientation of the pan in front of the robot, moving the drink behind or over the pan, the maximum height of the path, and the speed. The subjects were also informed about the types of queries they will respond to.

Independent Variables. In the first experiment, we wanted to compare scale feedback and weak pairwise comparisons under random querying, and scale feedback under random and mutual information based querying. Hence, we varied the query type and the querying algorithm among: (i) weak pairwise comparisons with random querying, (ii) scale feedback with random querying, and (iii) scale feedback with mutual information based querying. In the second experiment, we wanted to compare scale and weak pairwise comparisons under mutual information based querying. Hence, we employed: (i) weak pairwise comparisons with mutual information based querying, and (ii) scale feedback with mutual information based querying. For all, we took $\sigma_S = 0.35$ based on pilot trials with different users (see Appendix D.3).

Procedure. We recruited 18 participants (5 female, 13 male, ages 20 – 55) for the first, and 14 participants (5 female, 9 male, ages 20 – 56) for the second experiment. Due to the pandemic conditions, the subjects participated in the study remotely with an online interface as in Figure 3.5. The study started with an instructions page with a two-question quiz to make sure the participants understood how to use the interface. After reading the instructions, we had the subjects fill a form where they indicated their preferences for each of the five individual factors described above, to encourage them to be consistent in their responses during the data collection.

[12] A summary video is at `https://sites.google.com/view/reward-learning-scale-feedback`, and the code is at `https://github.com/Stanford-ILIAD/reward-learning-scale-feedback`.

In the experiments, each participant responded to 10 queries generated with each of the algorithms. After each of these 10-query sets, they were shown the optimal trajectory from the dataset with respect to their learned reward function. The participants responded to a 5-point rating scale survey (1-Strongly Disagree, 5-Strongly Agree) for this trajectory: "The displayed trajectory fits my preferences on the task." We also collected scale feedback for 10 more randomly-generated queries (called the test set) to measure performance in each experiment. We randomized the order of these sets (of 10 queries) to prevent any bias. The interface provided a "Sync Videos" button to restart both videos for easier comparison.

Dependent Measures. As an objective measure of the learning performance, we calculated the log-likelihood of the test set (of 10 scale queries[13]) under the posterior $b^{|\mathcal{D}|}(w, \varrho)$ learned using the 10 queries generated via each algorithm, i.e., we calculated:

$$\texttt{Log}-\texttt{Likelihood} = \log P(\mathcal{D}_{\text{test}} \mid \mathcal{D}) = \log \mathbb{E}_{w|\mathcal{D}} \left[P(\mathcal{D}_{\text{test}} \mid w) \right] \tag{4.26}$$

We also used the responses to the 5-point rating scale survey questions to measure how well the learned rewards achieve the task. Finally, the users took a post-experiment survey where they rated (from 1 to 5) the easiness and expressiveness of weak pairwise comparison and scale feedback questions.

Hypotheses. We test the following hypotheses.

H10. *Scale feedback leads to faster learning than weak pairwise comparisons.*

H11. *Querying based on mutual information accelerates learning compared to random querying.*

H12. *Users will prefer mutual information based querying over random querying in terms of the optimized trajectories.*

H13. *Users will prefer scale feedback over weak pairwise comparisons in terms of the optimized trajectories.*

H14. *Users will rate the scale feedback questions as easy as weak pairwise comparison questions.*

H15. *Users will rate the scale feedback questions as expressive as weak pairwise comparison questions.*

Results. We present results of the first and the second experiments in Figures 4.24 and 4.25, respectively. It can be seen that the log-likelihood of the test set after learning the reward function via scale feedback is higher than learning via weak pairwise comparisons, under both random and mutual information based querying. Besides, mutual information based query generation accelerates the learning and leads to higher log-likelihood values compared to random querying. All of these comparisons are statistically significant with $p < 0.001$ (paired-sample t-test), so they strongly support **H10** and **H11**.

In Figure 4.24b, it can be seen active querying led to learning reward functions that better

[13]We present results with a test set that consists of both scale feedback and weak pairwise comparisons in Appendix E.3.

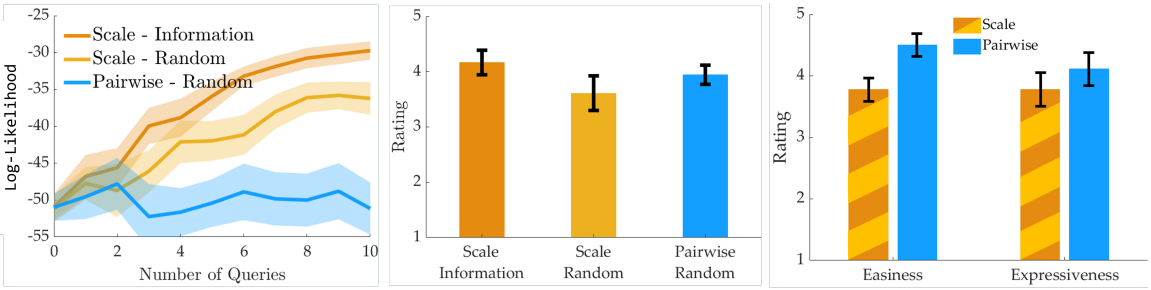


Figure 4.24: All results are shown for the first experiment (mean±s.e. over 18 subjects).

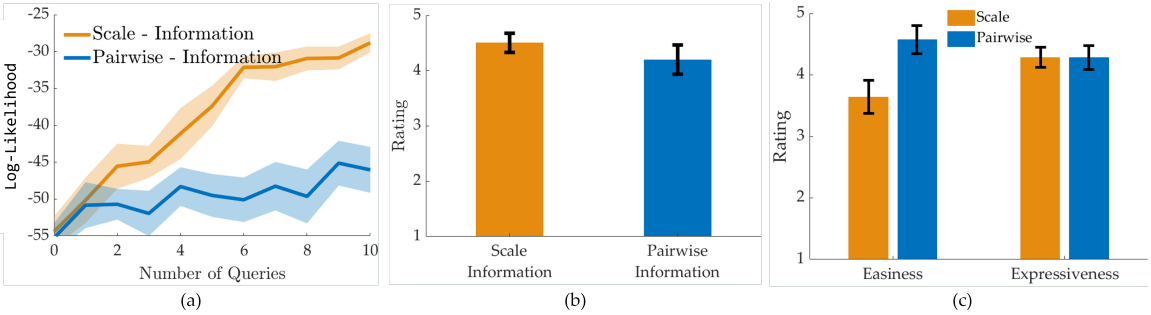


Figure 4.25: All results are shown for the second experiment (mean±s.e. over 14 subjects).

optimize trajectories compared to random querying – this comparison was somewhat significant with $p \approx 0.05$, supporting **H12**. In fact, when we fit a Gaussian distribution to the ratings, we observe that it is 1.95 times as likely to get a better rating with mutual information based querying than random querying. Surprisingly, learning via weak pairwise comparisons achieved slightly higher reward than learning via scale feedback when queries were randomly selected, and slightly lower reward when queries were generated based on mutual information maximization. However, these comparisons are not statistically significant. This is indeed analogous to the `Relative_Reward` comparisons in Figure 4.23: more complex tasks might be needed to better analyze the difference between the two methods. Thus, we neither reject nor accept **H13**.

Finally, the subjective results in Figure 4.24c and 4.25c suggest that users find the weak pairwise comparisons slightly, but consistently, easier than the scale feedback ($p < 0.01$), rejecting **H14**. This is not surprising, as it is often easier to make a pairwise comparison and the "About Equal" option in the weak pairwise comparison questions makes them even easier (see Section 4.2). On the other hand, there was no statistically significant difference in terms of expressiveness of scale and weak pairwise comparison questions, partially supporting **H15**. In summary, it is interesting that our users perceived the weak pairwise comparison questions as easier and even more expressive at times; even though quantitatively, the scale feedback significantly outperforms the weak pairwise

comparisons.

Appendix E.4 presents the numerical results for all plots in this section.

4.6 Active Querying for Multimodal Rewards

The results in the previous sections in this chapter imply the comparative feedback queries made to the experts, $Q^{(i)}$'s, affect how well the posterior will be learned when the reward function is unimodal. In the case of multimodal rewards where we learn via ranking queries as we discussed in Section 3.5, this is still true, and can be seen from Equation (3.38).

Specifically, a query $Q^{(i)}$ is desirable if observing the (anonymous) response to that, $q^{(i)}$, yields high information about the underlying model parameters, w_m and α_m for all $m \in [M]$. Therefore, we again propose using a mutual information objective to adaptively select the most informative query at each querying step i, generalizing the approach we presented in Section 4.2.

4.6.1 Formulation

Assume at a fixed round i, we have made past ranking query observations $\mathcal{D}_R = \left\{Q^{(i')}, q^{(i')}\right\}_{i'=1}^{i-1}$, possibly with other types of feedback to obtain the belief distribution b^{i-1}. The desired query is then

$$Q_*^{(i)} = \arg\max_{Q} I(q; w, \alpha \mid Q, b^{i-1}) \tag{4.27}$$

where $I(\cdot;\cdot)$ denotes mutual information and q is the response to the query Q. Equivalently,

$$Q^{(i)} = \arg\min_{Q} \mathbb{E}_{q',w',\alpha' \sim q,w,\alpha|Q,b^{i-1}} \log \frac{\mathbb{E}_{w'',\alpha'' \sim w,\alpha|b^{i-1}} P\left(q = q' \mid Q, w = w'', \alpha = \alpha''\right)}{P\left(q = q' \mid Q, w = w', \alpha = \alpha'\right)}. \tag{4.28}$$

The details of this derivation are presented in Appendix B.3.

4.6.2 Overall Algorithm

To efficiently solve the optimization in Equation (4.28), we first note that we should use a Monte Carlo approximation since the expectations are taken over continuous variables w, α and a discrete variable q over an intractably large set of $|Q|!$ alternatives. To perform this Monte Carlo integration, we require samples from the posterior $P(q, w, \alpha \mid Q, b^{i-1})$.

Our key insight is that we can obtain joint samples from both posteriors by first sampling $\bar{w}, \bar{\alpha} \sim P(w, \alpha \mid \mathcal{D}_R)$ and then sampling $\bar{q} \sim P(q \mid Q, w = \bar{w}, \bar{\alpha} = \alpha)$ since $(w, \alpha) \perp Q \mid b^{i-1}$ and $q \perp b^{i-1} \mid Q, w, \alpha$. We perform the sampling $\bar{q} \sim P(q \mid Q, w = \bar{w}, \alpha = \bar{\alpha})$ efficiently using Equation (3.37). In general, exact sampling from the posterior $P(w, \alpha \mid b^{i-1})$ is intractable. However,

we note Equation (3.38) can be directly evaluated (using Equation (3.37)) and gives $P(w, \alpha \mid b^{i-1})$ up to a proportionality constant factor.

With this unnormalized posterior of Equation (3.38), we use the Metropolis-Hastings algorithm as described in Appendix C.1 to generate samples from the posterior $P(w, \alpha \mid b^{i-1})$.

We see our optimization problem simplifies to finding, for $|\Omega|$ fixed samples $\bar{w}_j, \bar{\alpha}_j \sim P(w, \alpha \mid b^{i-1})$ and corresponding samples: $\bar{q}_j \sim P(q \mid Q, \bar{w}_j, \bar{\alpha}_j)$

$$Q^{(i)} = \arg\min_{Q} \sum_{j=1}^{|\Omega|} \left[\log \sum_{j'=1}^{|\Omega|} P\left(q = \bar{q}_j \mid Q, w = \bar{w}_{j'}, \alpha = \bar{\alpha}_{j'}\right) - \log P\left(q = \bar{q}_j \mid Q, w = \bar{w}_j, \alpha = \bar{\alpha}_j\right) \right]. \tag{4.29}$$

We solve this optimization using simulated annealing [32] (see Appendix C.2).

Algorithm 3 Active Querying for Rankings via Mutual Information Maximization

Require: Current belief distribution b^{i-1}

1: $\{\bar{w}_j, \bar{\alpha}_j\}_{j=1}^{|\Omega|} \sim P(w, \alpha \mid b^{i-1})$ with Equation (3.38) via Metropolis-Hastings
2: $\forall j, \bar{q}_j \sim P(q_j \mid Q, \bar{w}_j, \bar{\alpha}_j)$
3: $Q_*^{(i)} \leftarrow \arg\min_{Q} \sum_{j=1}^{|\Omega|} \left[\log \sum_{j'=1}^{|\Omega|} P\left(q = \bar{q}_j \mid Q, w = \bar{w}_{j'}, \alpha = \bar{\alpha}_{j'}\right) - \log P\left(q = \bar{q}_j \mid Q, w = \bar{w}_j, \alpha = \bar{\alpha}_j\right) \right]$
4: select query $Q_*^{(i)}$

Algorithm 3 goes over the pseudocode of our approach, and we discuss the hyperparameters in our experiments in Appendix C.3.

Analysis

We start the analysis by stating the bounds on the required number of trajectories in each ranking query to achieve *generic identifiability*. A Plackett-Luce model over Ξ is generically identifiable if for any sets of parameters (w, α) and (w', α') inducing the same distribution over the responses to all queries of size $|Q|$ on Ξ, the mixing coefficients of (w, α) and (w', α') are the same and the induced rewards $R_m(\xi)$ are identical across Ξ up to a constant additive scaling factor.

Theorem 4 (Zhao et al. [223])**.** *A mixture of M Plackett-Luce models with query size $|Q|$ and $|\Xi| = |Q|$ is generically identifiable if $M \leq \left\lfloor \frac{|Q|-2}{2} \right\rfloor!$.*

This statement follows directly from [223], which proves the above bound assuming that each query to the Plackett-Luce mixture is a full ranking over the set of items (i.e. $|\Xi| = |Q|$). However, the assumption $|\Xi| = |Q|$ is untenable in the active learning context, as it prevents any active query selection. To apply this result for our active learning algorithms, we relax the condition to $|\Xi| \geq |Q|$.

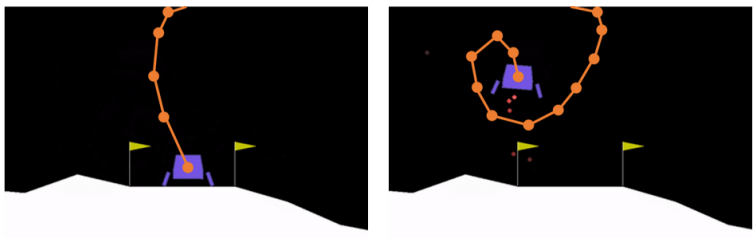
Corollary 1. *A mixture of M Plackett-Luce models with query size $|Q|$ is* generically identifiable *if $M \leq \left\lfloor \frac{|Q|-2}{2} \right\rfloor!$.*

We prove Corollary 1 in Appendix A.4. In our context, generic identifiability implies if the human response is modelled by a Plackett-Luce mixture, our Algorithm 3 will be able to recover its true parameters (up to a constant additive factor for the rewards) in the limit of infinite queries.

Remark 2. *Greedy selection of queries maximizing mutual information in Equation* (4.28) *is not necessarily within a constant factor of optimality.*

Appendix A.5 justifies Remark 2. In fact, greedy optimization of mutual information for adaptive active learning can be significantly worse than a constant factor of optimality in pathological settings [96]. Despite its lack of theoretical guarantees, mutual information is a commonly used effective approach in adaptive active learning [112, 224]. Although other approaches like volume removal satisfy adaptive submodularity [171], they fail in settings with noisy responses by selecting high-noise low-information queries as we showed in Section 4.1, and in practice result in far worse performance than mutual information.

4.6.3 Experiments



(a) Landing softly on the landing pad *(b) Staying in the air as long as possible*

Figure 4.26: The *LunarLander* environment is visualized with the two tasks. Sample trajectories associated with these tasks are shown.

Having presented our learning and active querying algorithms, we now evaluate their performance in comparison with other alternatives. In all simulations and experiments, we assume $\beta^C = 1$, and $R_{w_m}(\xi) = w_m^\top \Phi(\xi)$, i.e., the trajectory reward function of each individual expert is linear in trajectory features. We start with describing the two tasks we experimented with:

LunarLander. We used 1000 trajectories in OpenAI Gym's discrete action space LunarLander environment [50] shown in Figure 4.26 (see Appendix D.5.1 for details on how those trajectories were generated).

FetchBanana. We generated 351 distinct trajectories of the Fetch robot [213] putting the banana on the shelves as shown in Fig. 3.8a (see Appendix D.5.2 for details).

Methods

We compare our active querying via mutual information (MI) discussed in Algorithm 3 with two baselines. A simple benchmark for active learning is *random* query selection without replacement. We also benchmark against the *volume removal* (VR) method, which we described in Section 4.1. See Appendix D.4 for further details of how we implemented these two baselines.

Metrics

We want to evaluate both the active querying and the learning performance. The former requires metrics that assess the quality of the algorithm's selected queries $\mathcal{D}_R = \{(Q^{(i)}, q^{(i)})\}_i$ in terms of the information they provide on the model parameters (w, α). We use two such metrics: `Mean Squared Error (MSE)` and `Log-Likelihood`. Since both active and non-active methods are expected to reach the same performance with a large number of queries, we look at the area under the curve (AUC) of these two metrics over number of queries. To evaluate the learning performance, we quantify the success of a robot, which learned a multimodal reward, via the `Learned_Policy_Rewards` on the actual task.

`MSE.` Suppose we know the human is truly modeled by (w^*, α^*) adhering to the assumed model class of Section 3.5.1. Given a set of queries and responses in $\mathcal{D}_R$, we can compute a maximum likelihood estimate $(\hat{w}, \hat{\alpha})$ of the model parameters using Equation (3.37). The `MSE` is then the squared error between $(\hat{w}, \hat{\alpha})$ and (w^*, α^*) (see Appendix D.6.1).

While this metric cannot be evaluated with real humans, we can use this metric with synthetic human models (model with known parameters (w^*, α^*)) in simulation. A lower `MSE` score means the selected queries $\mathcal{D}_R$ allow us to better learn a multimodal Plackett-Luce model close to the true model (w^*, α^*).

`Log-Likelihood.` This metric measures the `Log-Likelihood` of the response to a random query given the past observations $\mathcal{D}_R$. If the past observations $\mathcal{D}_R$ are informative, the true response to a random query Q will in expectation be more likely, meaning the `Log-Likelihood` metric will be greater. See Appendix D.6.2 for details on how we compute this metric.

`Learned_Policy_Reward.` We take the maximum likelihood estimate of each reward parameters vector and train a DQN policy using them [151].[14] We then run these learned policies on the actual environment with the corresponding true reward functions (see Appendix D.6.3) to obtain the `Learned_Policy_Reward` values.

Results

Multimodal Learning is Necessary. We first compare unimodal and multimodal models to show the insufficiency of unimodal rewards when the data come from a mixture. To leave out any possible

[14] As we are using a real Fetch robot for our experiments and it would be infeasible and unsafe to train DQN on Fetch, this metric is limited to our simulations, i.e., *LunarLander* in our experiments.

bias due to active querying, we make this comparison using random querying.

We let the true reward function have $M = 2$ modes and set a query size of $|Q| = 6$ items for identifiability as Section 4.6.2 suggests, and for acquiring high information from each query. We simulate 100 pairs of experts whose reward function parameters w_m and the mixing coefficients α_m are sampled from the prior $P(w, \alpha)$. Having these simulated experts respond to 15 queries, we report the `MSE` in Figure 4.27.

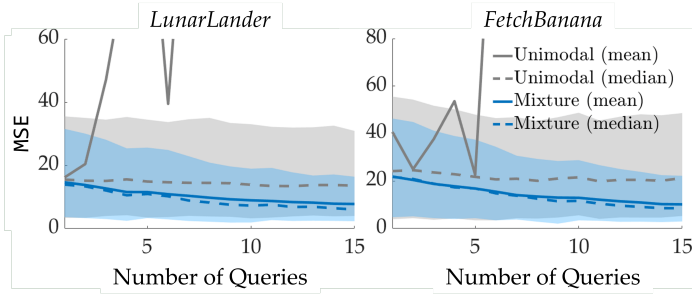


Figure 4.27: Unimodal and bimodal reward learning models are compared under `MSE`. Both mean and median values (over 100 runs) are shown. Shaded regions show the first and the third quartiles.

The unimodal reward model causes an unstably increasing `MSE`. This is mostly due to the outliers where the reward function parameters w_1 and w_2 are far away from each other and the unimodal reward fails to learn any of them. We therefore also plot the median values and quartiles in Figure 4.27. While the bimodal reward model learned using our proposed approach decreases the `MSE` over time, the unimodal model has a roughly constant `MSE`, which suggests it is unable to learn when the data come from a mixture.

We present an additional unimodal learning baseline evaluated on the user study data in Appendix E.6.

Active Querying with Mutual Information is Data-Efficient. We next compare our mutual information based active querying approach with the other baselines. For this, we use the same experiment setup as above with $M = 2$ reward function modes and ranking queries of size $|Q| = 6$, and simulate 75 pairs of human experts. We present the results in terms of `MSE` in Figure 4.28. In *LunarLander*, the mutual information objective significantly outperforms both random querying and volume removal in terms of the AUC `MSE` ($p < 0.005$, paired-sample t-test). Notably, volume removal performs even worse than the random querying method, which might be due to the known issues of volume removal optimization as briefly discussed in Appendix D.4.2 and in more detail in Section 4.1. On the other hand, the difference is not statistically significant in the *FetchBanana* experiment, which might be due to the small trajectory dataset, or because almost all trajectories in the dataset minimize or maximize some of the trajectory features, accelerating and simplifying learning under the linear reward assumption. See Appendix D.5.2 for details about the trajectory features and how we generated the trajectory dataset.

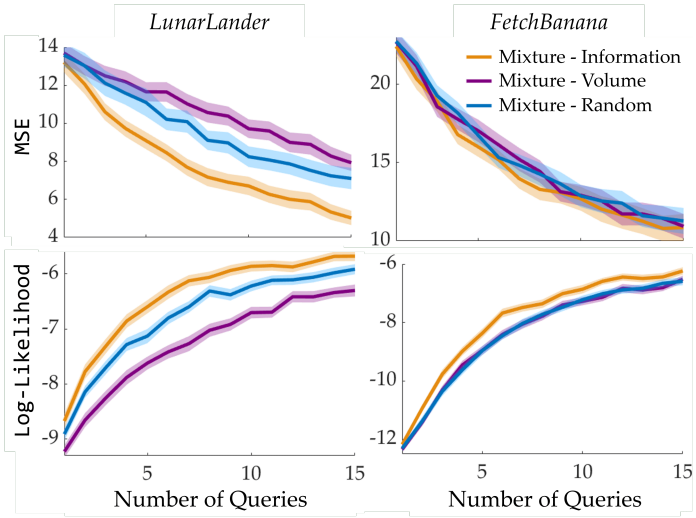


Figure 4.28: Different querying methods are compared with the (top) `MSE` and (bottom) `Log-Likelihood` metrics (mean±se over 75 runs).

We further analyze the querying methods in this multimodal setting under the `Log-Likelihood` metric in Figure 4.28. Mutual information based querying significantly outperforms random querying and volume removal based querying in both experiments with respect to the AUC `Log-Likelihood` ($p < 0.005$). With respect to the final `Log-Likelihood`, mutual information reduces the amount of required data in *LunarLander* by about 35% compared to random querying and about 60% to volume removal. Similarly in the *FetchBanana*, the improvement is approximately 25% over both baselines.

Appendix E.5 presents two additional experiments: one which clearly shows the effectiveness of our approach for learning a mixture of more than two reward functions (specifically, $M = 5$), and one which studies the robustness against misspecified M.

Mutual Information Maximization Leads to Better Learning. Having seen the superior predictive performance of the reward learned via mutual information maximization, we next assess its performance in the actual environment. As random querying outperforms volume removal in terms of `Log-Likelihood` and `MSE` as in Figure 4.28, we compare the mutual information based method with random querying.

For this, we run the multimodal reward learning with 75 pairs of randomly generated reward function parameters ($M = 2$ and $|Q| = 6$). For each of the 150 individual reward functions, we compute the `Learned_Policy_Reward` values. Figure 4.29 shows the results. While the standard errors in the plots seem high, this is mostly because optimal trajectories for different reward parameters differ substantially in terms of rewards, which causes an irreducible variance. However, since the underlying true rewards are the same between the mutual information based and random querying

methods, we ran the paired sample t-test between the results and observed statistical significance ($p < 0.05$). This means although the `Learned_Policy_Reward` values between different runs differ substantially, the reward function learned via the mutual information method leads to better task performance compared to random querying.

4.6.4 User Studies

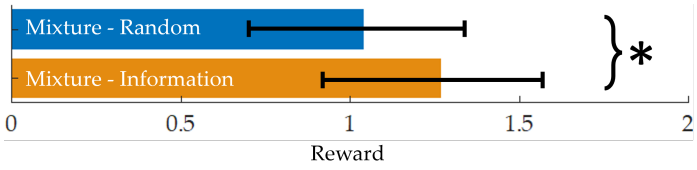


Figure 4.29: Mutual information based and random querying methods are compared with the `Learned_Policy_Reward` values (mean±se over 75 runs which correspond to 150 randomly generated reward function parameters) in *LunarLander*.

We now empirically analyze the performance of our algorithm with two online user studies. We again used the *LunarLander* and *FetchBanana* environments. We provide a summary and a video of the user studies and their results at `https://sites.google.com/view/multimodal-reward-learning`.

Experimental Setup. For *LunarLander*, subjects were presented with either of the following instructions at every ranking query: "Land softly on the landing pad" or "Stay in the air as long as possible". We randomized these instructions such that users get one of them with 0.6 and the other with 0.4 probability. We kept the presented instructions hidden from the learning algorithms so that they need to learn a multimodal reward without knowing which mode each ranking belongs to.

For the *FetchBanana* environment, we recorded the 351 trajectories on the real robot as short video clips so that the experiment can be conducted online under the pandemic regulations. Human subjects participated in the experiment as groups of two to test learning from multiple users. Each participant was instructed that the robot needs to put the banana in one of the shelves and different shelves have different conditions (the same as in our running example in Section 3.5, see Figure 3.8a, Appendix D.7.1).

After emphasizing there is no one correct choice and it only depends on their preferences, we asked each participant to indicate their preferences between the shelves on an online form. Afterwards, each group of two subjects responded to 30 ranking queries in total where each query consisted of 6 trajectories. We selected who responds to each query randomly, with probabilities 0.6 and 0.4.

Appendix D.7.2 presents details on the user interface used in our experiments.

Independent Variables. We varied the querying algorithm: active querying with mutual information and random querying. We excluded the volume removal based method to reduce the

experiment completion time for the subjects, as it already performed worse than random querying in our simulations.

Procedure. We conducted the experiments as a within-subjects study. We recruited 24 participants (ages 19 – 56; 9 female, 15 male) for *LunarLander* and 26 participants (ages 19 – 56; 11 female, 15 male) for the *FetchBanana*. Each subject in the *LunarLander*, and each group of two subjects in the *FetchBanana* experiment responded to 40 ranking queries; 15 with each algorithm and 10 random queries for evaluation at the end. The order of the first 30 queries was randomized to prevent bias.

Dependent Measures. Learning the multimodal reward functions via the 15 rankings collected by each algorithm, we measured the `Log-Likelihood` of the final 10 rankings collected for evaluation.

Hypotheses. With *LunarLander* and *FetchBanana*, we test the following hypotheses respectively:

H16. *Querying the participants, who are trying to teach two different tasks, actively with mutual information maximization will lead to faster learning than random querying.*

H17. *While learning from two people with different preferences, active querying with mutual information maximization will lead to faster learning than random querying.*

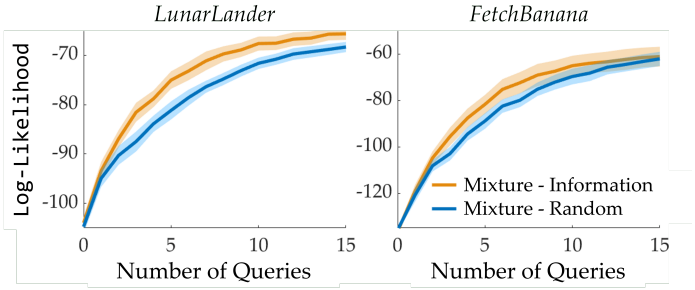


Figure 4.30: User study results (mean±se over 24 users for *LunarLander* and 13 groups for *FetchBanana*).

Results. Figure 4.30 visualizes how `Log-Likelihood` of the evaluation queries changes over the course of learning by both algorithms. Active querying with mutual information maximization leads to significantly faster learning compared to random querying in *LunarLander*. Indeed, the difference in AUC `Log-Likelihood` is statistically significant ($p < 0.05$). Furthermore, the active querying method enabled reaching the final performance of random querying after only 9 or 10 queries, for around a 35% reduction in the amount of data needed, supporting **H16**.

As *FetchBanana* experiments have an easier task with a small number of variables between the trajectories, both querying methods converge to similar performances by the end of 15 queries. However, active querying accelerates learning in the early stages—the difference in AUC `Log-Likelihood` is again statistically significant ($p < 0.05$). Looking at the final performance with random querying, improvement in data efficiency is about 10%, supporting **H17**.

4.7 Active Generation of Hierarchical Queries

A special case of multimodal rewards we studied in Section 3.6 is a setting where the user's reward function transitions from one mode to another based on a parametric model that depends on the behavior observed in the previous trajectories. We showed that using hierarchical queries, such non-stationary rewards could be learned along with the transition models. In this section, we extend that learning algorithm with active querying based on the maximum volume removal approach we presented in Section 4.1. One could also use the mutual information maximization based active querying technique from Section 4.2 to avoid the shortcomings of maximum volume removal, which we discussed in detail in Section 4.1.

Similar to Sections 4.2 through 4.6, and [22], we will actively select hierarchical choice queries from a query database to learn a probability distribution over *reward dynamics*: a collection of unimodal reward functions representing different moods and a set of parameters for the transitions between the moods.

In this section, we provide this algorithm which actively selects hierarchical choice queries in order to efficiently learn reward dynamics. We evaluate our active querying algorithm on an autonomous driving example, which we also used as a running example in Section 3.6. We show in simulations that we can efficiently learn changes in preferences when preferences indeed vary based on behavior and interactions in the environment.

4.7.1 Active Querying based on Maximum Volume Removal

In this section, $\mathcal{D}_H$ denotes the dataset of all the hierarchical choice queries and their responses up to the current iteration i. Each response tuple $(q^{(i,1)}, q^{(i,2)})$ removes some volume from the hypothesis space of (w, θ), where, volume removed is given as the difference between the unnormalized posterior distribution over (w, θ), and its prior distribution. We have a belief over what the user responses could be. We leverage this probabilistic model to *actively* select a query at each iteration that will maximize the expected volume removal. We again restrict our implementation to the case where $|Q^{(i)}| = 3$, i.e., we first show a trajectory to the user, and then ask 2 best-of-many choice questions where the trajectories in the first question start from the last state of the first trajectory and the trajectories in the second question start from the last state of the user's choice in the first question. Then, we need to solve the following optimization problem:

$$(Q_*^{(i,0)}, Q_*^{(i,1)}, Q_*^{(i,2)}) = \underset{Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)}}{\arg\max} \mathbb{E}_{q^{(i,1)}, q^{(i,2)} | \mathcal{D}_H, Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)}} \Big[\mathbb{E}_{w,\theta|\mathcal{D}_H} \left[1 - P(q^{(i,1)}, q^{(i,2)} \mid w, \theta, Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)}) \right] \Big] \tag{4.30}$$

To compute the inner expectation, we first sample (w, θ) from $P(w, \theta \mid \mathcal{D}_H)$ using Markov Chain Monte Carlo methods. Similar to previous sections, we use Metropolis-Hastings algorithm [69]. Now, we let $\bar{w}$ and $\bar{\theta}$ represent those samples, so:

$$(Q_*^{(i,0)}, Q_*^{(i,1)}, Q_*^{(i,2)}) \doteq \underset{Q^{(i,0)},Q^{(i,1)},Q^{(i,2)}}{\arg\min} \mathbb{E}_{q^{(i,1)},q^{(i,2)}|\mathcal{D}_H,Q^{(i,0)},Q^{(i,1)},Q^{(i,2)}} \Big[\sum_{\bar{w},\bar{\theta}} P(q^{(i,1)}, q^{(i,2)} \mid \bar{w}, \bar{\theta}, Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)}) \Big]$$

where $\doteq$ denotes asymptotic equality as the number of samples $(\bar{w}, \bar{\theta})$ goes to infinity. Expanding $\mathbb{E}_{q^{(i,1)},q^{(i,2)}|\mathcal{D}_H,Q^{(i,0)},Q^{(i,1)},Q^{(i,2)}}$, the minimization objective is

$$\sum_{q^{(i,1)},q^{(i,2)}} P(q^{(i,1)}, q^{(i,2)} \mid Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)}, \mathcal{D}_H) \sum_{\bar{w},\bar{\theta}} P(q^{(i,1)}, q^{(i,2)} \mid \bar{w}, \bar{\theta}, Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)})$$

where the first sum is over all possible response sequences, i.e., $Q^{(i,1)} \times Q^{(i,2)}$. By the law of large numbers, we can write the optimization as:

$$\underset{Q^{(i,0)},Q^{(i,1)},Q^{(i,2)}}{\arg\min} \sum_{(q^{(i,1)},q^{(i,2)}) \in Q^{(i,1)} \times Q^{(i,2)}} \left(\sum_{\bar{w},\bar{\theta}} p(q^{(i,1)}, q^{(i,2)} \mid \bar{w}, \bar{\theta}, Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)}) \right)^2, \quad (4.31)$$

because $P(q^{(i,1)}, q^{(i,2)} \mid Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)}, \mathcal{D}_H) = \mathbb{E}_{\bar{w},\bar{\theta}} \left[P(q^{(i,1)}, q^{(i,2)} \mid \bar{w}, \bar{\theta}, Q^{(i,0)}, Q^{(i,1)}, Q^{(i,2)}) \right]$ where the probability expression in the objective function is already derived in Section 3.6.3.

4.7.2 Simulations and Experiments

Problem Domain

We focus on learning non-stationary reward functions for driving, using a version of the *Driver* environment presented earlier in Section 4.2.4. Each mode of the learned *reward dynamics* weighs 5 features for driving that attempt to encode safety and traffic rules: one feature for penalizing closeness to the edges of the road, one for velocity, and three more features of proximity to lane centers, to other cars and alignment with the road, similar to [171].[15] Similar to what we did in Section 4.6 with multimodal reward functions, we assume each mode corresponds to a reward function that is linear in these trajectory features.

Each sub-query consists of the driving environment and a pair of trajectories of the ego car whose preferred behavior we are learning (i.e., $|Q^{(i,j)}| = 2$ for all iterations i and $j \in \{1, 2\}$), and another car in the scenario. Our query database consists of 10,000 randomly generated hierarchical choice

[15] While traffic rules are not explicitly modeled in our simulation, many of the features can be weighed appropriately to encode them. For example, the feature for velocity measures the deviation from the maximum allowed speed which can be easily adapted to the road type.

queries.

Dependent Measures

In our implementation, we learned $\Delta\theta := \theta_1 - \theta_2$, instead of θ, as it has fewer parameters.[16] The same approach generalizes to any M — we can simply subtract θ_M from θ_m for $\forall m \in [M-1]$ to reduce the number of parameters to be learned.

We measure the performance of hierarchical preference learning in terms of expected dot product between learned weights and true weight as in previous sections and [171], separately for each component of $(w, \Delta\theta)$:

$$\texttt{Alignment}(w_1) = \mathbb{E}\left[\frac{\hat{w_1}.w_1^*}{\|\hat{w_1}\|_2\|w_1^*\|_2}\right] \tag{4.32}$$

$\hat{w_1}$ and w_1^* are the estimated and true weights, respectively, and the expectation is taken over the sampled $\hat{w_1}$ values. We define `Alignment` similarly for w_2 and $\Delta\theta$. Hence, it is a measure of convergence, as its value being close to 1 indicates the learned weights are close to the true weights.

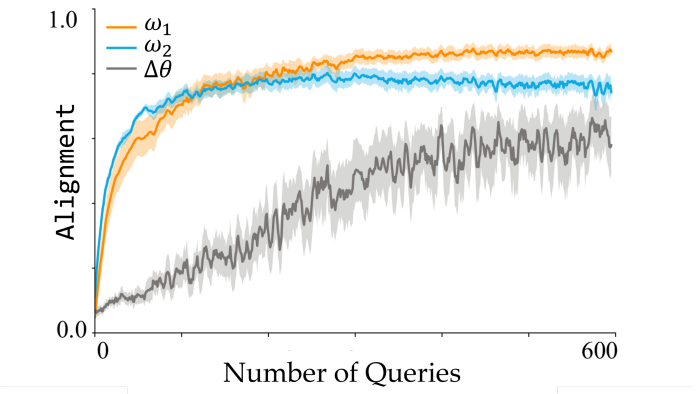


Figure 4.31: `Alignment` value shows that our algorithm converges well for non-driving data with non-active query selection when the simulated user is *oracle*. Here we show an average `Alignment` over 5 different ground truth *reward dynamics*.

Experiments with Random Data

We first conduct experiments with completely random and independent sub-queries without the driving environment. We assume we can generate queries in an unconstrained way such that any Φ-vector is possible, i.e. there is no dynamics constraint in the generation of queries. This is similar to the *LDS* environment we used in Section 4.2.4. Here, we simulate *oracle users*: users

[16] Note we cannot do the same trick for w, because while θ in Equation (3.47) can be simplified to $\Delta\theta$, there is no such simplification on w.

who are perfectly aware of their true reward dynamics. That is, they always behave (change mode and respond) with respect to the higher probability out of response and mode transition models. In Figure 4.31, the average results of 5 different simulated oracle users show convergence of $(w_1, w_2, \Delta\theta)$ whose true values were independently drawn from standard normal distribution independently for each entry.

Experiments with Driving Data

Active versus non-active query selection. We compare the performance of our active query selection algorithm with a non-active baseline where we uniformly sample the queries from the discrete database of 10,000 queries. Here our simulated users are always oracle. We test the following hypothesis:

H18. *The reward dynamics learned with our active query selection algorithm converges to the true parameters faster compared to the non-active baseline.* Our results in Figure 4.32 support this hypothesis by demonstrating that active query selection accelerated the learning of one of the modes (w_2) compared to the non-active baseline.

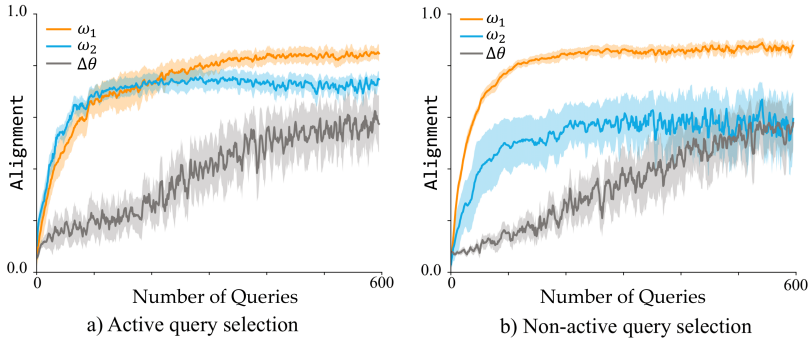


Figure 4.32: `Alignment` values show that our algorithm with active query selection (left) can learn reward dynamics faster than non-active query selection (right) when the simulated user is *oracle*. Here we show an average `Alignment` over 5 different ground truth reward dynamics.

Testing different mode preferences. Next we simulate 5 noisy users, who choose between options with respect to $P(q \mid w, \theta, Q)$ as we defined in Equation (3.47). Our algorithm actively selects queries from the same discrete dataset of size 10,000 as in the case of oracle users. We first set the following hypothesis:

H19. *Our algorithm learns the reward dynamics even when the users are noisy.*

We also test the performance of our algorithm for different mode likelihoods, i.e. probability of transitioning to a given mode. We manipulated the ground truth reward dynamics to reflect different mode likelihoods. For example, one user might be in one mode 80% of the time while another user has equal chances of being in one of the two modes. Although this might actually affect the priors P_1 and P_2 as we explained in Section 3.6.3, we still adopted the derivations based on uniform prior

to test the robustness of our framework. Therefore, we test the following hypothesis:

H20. *Our algorithm learns the reward function parameters w that correspond to both modes, and it converges faster for w_m if the user is more likely to be in mode m.*

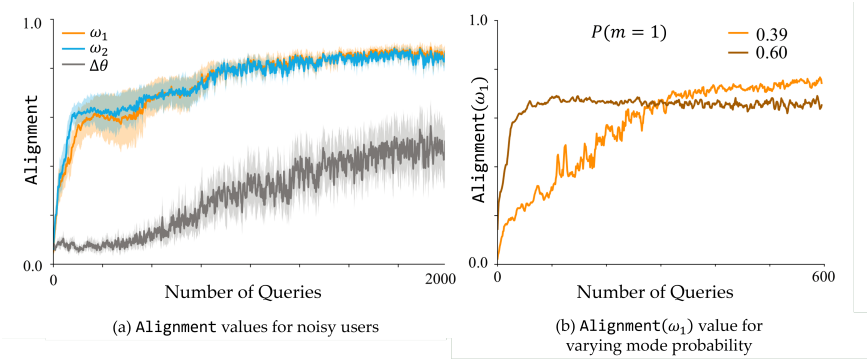


Figure 4.33: `Alignment` value shows even when the users are noisy our algorithm can learn the true *reward dynamics* (left) and that as the probability being at mode 1 increases, w_1 converges faster (right).

Figure 4.33a shows that our algorithm was able to learn w_1, w_2 and $\Delta\theta$ even when the users are noisy, supporting **H19**. We note that in general we learn $\Delta\theta$ slowly and we need more queries for its convergence. The second plot shows that we are able to learn the reward weights w_m of a mode m regardless of its likelihood probability being high or low. The same plot also shows the algorithm converges faster for the modes that are visited more often. This is intuitive, as the algorithm is able to gather more information about those modes, even though it does not perfectly know that the user is in the corresponding mood. Hence, **H20** has strong empirical support.

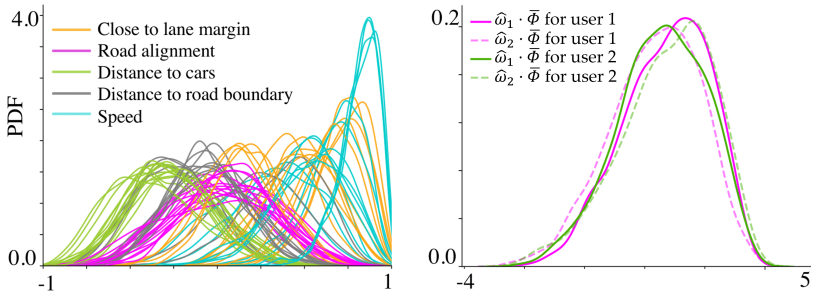
User Study

Hypotheses. We test the following hypotheses with the user study:

H21. *Our algorithm learns weights that can represent the driving behavior of the users.*

H22. *Some people indeed change preferences depending on the driving behaviors of the interacting agents.*

Study Design. To validate our hypotheses, we collected data from 10 real users in a within-subjects study. We first learn a general reward dynamics $(\hat{w}, \hat{\theta})$ using 50 hierarchical queries. We then use the posterior distributions over these parameters to jump-start the process for each subject with a reasonable prior that better represents legally correct driving. During validation, we ask users to provide ratings for trajectories locally optimized with respect to the learned reward dynamics. We compare the expressiveness of the learned reward parameters $(\hat{w}_1, \hat{w}_2)$ against their perturbed versions (w_1^p, w_2^p). We sampled these perturbed versions from Gaussian distributions centered around $(\hat{w}_1, \hat{w}_2)$ and a standard deviation of $0.5 \times \|\hat{w}_1\|$ and $0.5 \times \|\hat{w}_2\|$. While creating perturbed versions of $\hat{w}_1$ and $\hat{w}_2$, as an attempt to ensure legally correct driving, we constrain the weights for the features



(a) Distribution of learned $\hat{\omega}_1$ and $\hat{\omega}_2$ combined across all users

(b) Average rewards with respect to $\hat{\omega}_1$ and $\hat{\omega}_2$ for two users showing minor difference

Figure 4.34: Distribution of $\hat{w}_1$ and $\hat{w}_2$ across all users for individual features. (a) User preferences vary widely for adherence to lane center and distance to road boundaries, but are very similar for efficiency (speed) and safe driving (collision avoidance). (b) While we did not learn significantly different w_1 and w_2 for individual users, the average reward with respect to $\hat{w}_1$ and $\hat{w}_2$ differs slightly for some of our study participants.

that correspond to staying within the road and avoiding collision with cars. We also compare with w_{r} sampled from a Gaussian distribution centered on either $\hat{w}_1$ or $\hat{w}_2$ with a standard deviation of $2 \times \|\hat{w}_m\|$ with the corresponding mode index m. Each rating question consists of two parts. The first part is similar to $Q^{(i,0)}$ of the learning step, where we show user one trajectory demonstration of the robot as an attempt to set their initial mode. In the next part, we show users 5 trajectories continued from the first part, optimal with respect to 5 reward functions parameterized with: $\hat{w}_1$, $\hat{w}_2$, their perturbed versions w_1^p and w_2^p, and w_{r}. For each of the 5 trajectories, we ask users a 7-point rating scale question: *Indicate your level of agreement with the following statement: I would like to ride this car.*

In **H21** we claim 1) users will give the highest overall rating to the trajectories that are optimal with respect to $R_{\hat{w}_1}$ and/or $R_{\hat{w}_2}$ most of the time, and 2) if probability of being at mode m is very high, we expect people to give the highest rating to trajectories that suit to $R_{\hat{w}_m}$. To validate the first part, we repeat the same demonstration across several rating queries preserving the trajectory of the other car in the environment alike and changing the trajectory of the ego agent, varying between different local optima with respect to w_1, w_2 and the other weights. We randomize demonstration trajectories across the rating questions. In **H22** we hypothesize that subject to different interactions in the environment, users will sometimes give higher rating to trajectory optimal with respect to $R_{\hat{w}_1}$ and sometimes to those optimal with respect to $R_{\hat{w}_2}$.

Results. We found that the users have somewhat similar preferences: proximity to cars has high negative weight, and speed has high positive weight showing that people generally prefer safe and efficient driving (see Figure 4.34). On the other hand, features that encode staying on the road and alignment with the road vary more. While the general direction of the feature weights is similar between $\hat{w}_1$ and $\hat{w}_2$ for each user, there is some difference in the magnitudes. We computed the

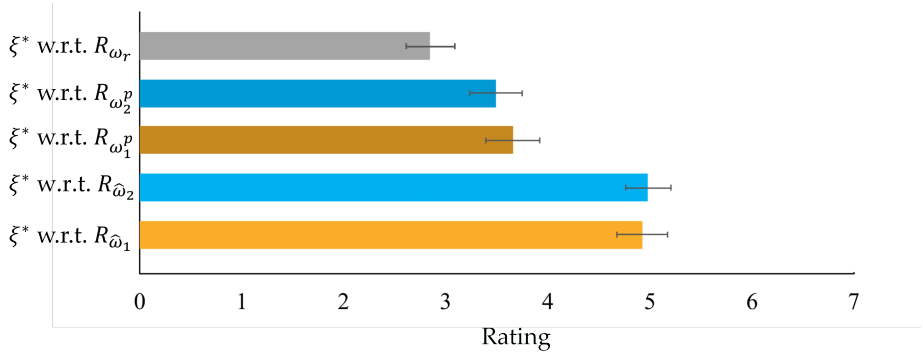


Figure 4.35: Most users gave high ratings to the trajectories optimal with respect to $R_{\hat{w}_1}$ and $R_{\hat{w}_2}$ and low ratings to trajectories optimal with respect to their perturbed versions $R_{w_1^p}$ and $R_{w_2^p}$ and the lowest rating to the trajectories that were optimal with respect to a reward function that is parameterized randomly R_{w_r}.

percentage difference between average reward with respect to $R_{\hat{w}_1}$ and $R_{\hat{w}_2}$ as $\frac{\hat{w}_1 \cdot \bar{\Phi} - \hat{w}_2 \cdot \bar{\Phi}}{\hat{w}_1 \cdot \bar{\Phi}}$, where $\bar{\Phi}$ is the average feature values of the trajectories in our query database. This gave us the percentage difference in the average reward. We found that of all the users the maximum difference is 12% and the minimum difference is 6%. While we also learned $\Delta\theta$, it becomes relatively unimportant here, as w_1 and w_2 are very close.

As it can be seen in Figure 4.35, the users gave the highest scores to the trajectories that are optimal with respec to $R_{\hat{w}_1}$ and $R_{\hat{w}_2}$ with statistical significance. This suggests an empirical evidence for **H21**. While we also observed that users sometimes gave high ratings to the trajectories of $R_{\hat{w}_1}$ and sometimes to those of $R_{\hat{w}_2}$, we have not observed a significant dependence on the modes. This is due to the fact that the learned weights were very close to each other as they represent the legal driving behavior, which is a very small subset of all the reward space. Further, our simulation environment may not be realistic enough to elicit emotions like anger, frustration etc. that cause behavioral changes in different traffic situations [179, 222, 143]. Therefore, our results are inconclusive about **H22**.

With this section, we completed extending the learning methods we presented in Chapter 3 with active querying techniques. In the next section, we describe how these techniques can be executed in batch settings where multiple questions are actively generated at the same time.

4.8 Batch-mode Active Querying for Time-Efficiency

Two important drawbacks of active query generation are the following: (i) the robot needs to optimize for each and every query, (ii) the querying process cannot be parallelized, i.e., even if multiple users are available to give comparative feedback, the robot needs to query them sequentially because each query is actively generated based on the responses to all the previous questions. Therefore,

even though active querying leads to significant gains in terms of data-efficiency, it might hurt time-efficiency, especially when optimizing queries is difficult. In some cases, it is desirable to be able to quickly generate queries and ask them to multiple users in parallel.

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 (as in Sections 4.3 and 4.4), 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.

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 generate queries actively for the highest data-efficiency while generating each query time-efficiently. In this section, we propose a new set of algorithms — *batch active comparison-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 query.

To this end, we actively generate each batch based on the data collected so far. We focus on pairwise comparison queries due to their simplicity, but the same techniques can be easily extended to any query type we discussed in this thesis. In our framework with pairwise comparisons, 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 querying 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 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 [218, 61]. Therefore, a good batch active learning method must produce batches that consist of both dissimilar and informative queries. This is visualized in Figure 4.36.

The problem of actively generating a batch of data is well-studied in other machine learning problems such as classification [203, 85, 217], where decision boundaries may inform the active learning algorithms. While this may simplify the problem, it is not applicable in our setting, where

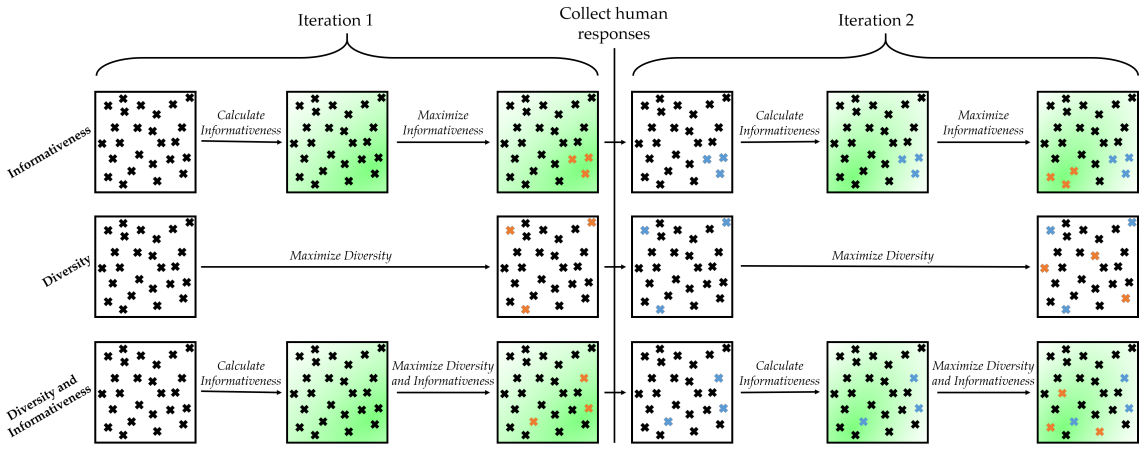


Figure 4.36: Batches should be both diverse and informative in batch active learning. Here, a hypothetical batch selection problem is visualized. Each cross represents a query. Similar queries are close to each other. Orange shows the queries selected in that iteration, and blue shows the queries for 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.

we attempt to actively learn a reward function for dynamical systems using pairwise comparison queries as opposed to data point - label pairs where the labels are directly associated with the corresponding data points.

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 [128] and are used to generate batches in other machine learning applications, such as for improving the convergence of stochastic gradient descent [219, 220]. Here, we propose using DPPs to generate not only diverse but also informative batches in active preference-based reward learning.

We summarize our contributions in this section as:

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 approximation algorithms for efficient batch active learning to learn about human preferences from pairwise comparison queries.
3. Experimenting and comparing approximation methods for batch active learning in complex comparison based learning tasks.

4. Showcasing our framework in predicting human users' preferences in simulated autonomous driving and robotics tasks.

For the rest of the section, we will start with formalizing the problem. We will then present how standard active methods select queries. After introducing the general batch-mode active learning idea, we propose our methods for batch selection. First, we propose a method based on determinantal point processes. Next, we propose more time-efficient alternatives which might be preferable when batch sizes are large or to avoid hyperparameter tuning. After proposing these different approaches, we give theoretical guarantees for three of them. Finally, we present our experiments with both simulated and real users.

4.8.1 Formulation

We consider the setup we presented in Section 3.1, for the special case of pairwise comparisons. which we later extended with maximum volume removal based active querying in Section 4.1. In this section, we will again use the maximum volume removal based method, but the batch generation algorithms we propose are agnostic to the choice of the acquisition function; so for example, mutual information (see Section 4.2) or max regret (see Section 4.5) can also be used in practice. Similarly, although we focus on parametric reward functions in this section, the algorithms we propose can also be used for non-parametric reward functions. THe batch active querying algorithms require only two things: (i) a score for each query that represents queries' informativeness, and (ii) a similarity metric between the queries.

In the setup for this section, we start with a prior belief b^0 over the reward function parameters w. This prior might be initialized with some domain knowledge and/or some expert demonstrations. Most active querying techniques, as we discussed in previous sections, relies on samples from this belief; and then in later querying iterations, samples from the updated beliefs b^i. However, we do not know the shape of this distribution. As a result, we used Metropolis-Hastings [69] in the previous sections to get the samples. In this section, since our goal is to increase the time-efficiency of active querying, our first change in the algorithm is to use a more efficient sampling technique. For this, we approximate $P(q \mid Q, w)$, which we modeled in Equation 3.10 with a log-concave function whose mode always evaluates to 1:

$$P(q \mid Q = (\xi_1, \xi_2), w) = \min(1, \exp((-1)^{\mathbb{I}(q=\xi_2)}(R(\xi_1) - R(\xi_2)))) \,, \tag{4.33}$$

where $\mathbb{I}$ denotes the indicator function. This allows us to efficiently use an adaptive Metropolis algorithm [103] for sampling.

Our goal is to learn the human's reward function, or equivalently w, in a both data-efficient and time-efficient way. To this end, we develop batch-active comparison-based reward learning methods, which actively generate a batch of pairwise comparison queries based on the previous queries and

the human's responses to them.

Our insight is that we can in fact balance between the number of queries required for convergence to R_w 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 w. The batch approach can significantly reduce the total time required for the satisfactory estimation of w at the expense of increasing the number of queries needed for convergence to true R_w.

To obtain a batch of queries that are informative, we need to find queries that optimize an acquisition function, e.g., volume removal as computed by the objective of Equation (4.4). We again fall back to a discretization method for generating batches of queries: we discretize the space of all possible pairwise comparison queries by randomly sampling K pairs of feasible trajectories from Ξ. While increasing K may lead to more accurate optimization results, the computation time also increases linearly with K.

The batch active learning problem we are trying to solve is then an optimization that attempts to find the k pairwise comparison queries out of K that will maximize the volume removal in the worst case in terms of the human's responses (or the expected volume removal — see the equivalence in Appendix A.2). However, such a problem is often computationally hard (see [79] and [68] for the proofs with similar objectives), requiring an exhaustive search which is intractable in practice as the search space is exponentially large [101].

Algorithm 4 Batch Active Comparison-based Learning

1: Generate query dataset $\mathcal{K} = (\xi_{i1}, \xi_{i2})_{i=1}^{K}$ where each trajectory comes from Ξ
2: $\mathcal{D}_C \leftarrow \emptyset$
3: **for** iteration $i = 1, 2, \ldots$ **do**
4: Get samples $\bar{w} \sim b^{(i-1)k}(w)$
5: Generate a query batch of size k using $\bar{w}$ from the query set
6: Get the human response for each query in the batch
7: Update the belief $b^{(i-1)k}(w)$ with the new data to get $b^{ik}(w)$
8: **end for**
9: **return** $\mathbb{E}[w \mid \mathcal{D}_C]$

In the subsequent sections, we present our batch generation algorithms that attempt to find approximately optimal batches using various techniques and time-efficient heuristics. Algorithm 4 gives an overview of the overall batch active comparison-based learning approach: line 1 discretizes the space of queries, line 4 samples a set of w from the belief distribution $b^{(i-1)k}(w) = P(w \mid \mathcal{D}_C)$. 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, the posterior belief is updated in line 7 with respect to Equation (3.9).

4.8.2 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.

Background

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

An L-ensemble defines a DPP through a real, symmetric and positive semidefinite (PSD) K-by-K kernel matrix L [46]. Then, sampling a subset $X = A \subseteq \mathcal{K}$ has the probability

$$P(X = A) \propto \det L_A \tag{4.34}$$

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 $\mathcal{K}$, 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 Equation (4.34) remains valid [127]. In this section, we employ k-DPPs and refer to them as DPPs for brevity.

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

$$P(X = A) \propto \det L_A = \text{Vol}(\{L_i\}_{i \in A}) \,, \tag{4.35}$$

so the probability is proportional to the square of the associated volume.[17] In fact, by using a generalized version of DPP, we can approximately achieve [11, 148]:

$$P(X = A) \propto \text{Vol}^{\lambda}(\{L_i\}_{i \in A}) \,, \tag{4.36}$$

[17] Volume here refers to the volume of the parallelepiped spanned by the columns of L, whereas the volume removal in the previous sections referred to the change in the belief distribution between the prior and the unnormalized posterior.

for $\lambda \geq 0$. One can note that higher λ 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 4.37.

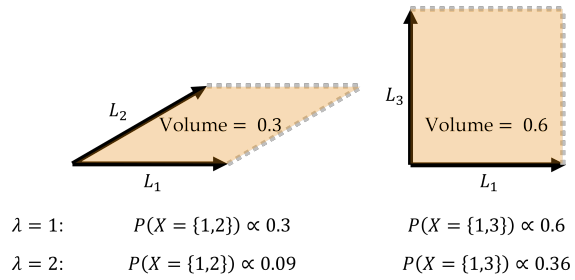


Figure 4.37: The effect of λ is visualized. The columns of the matrix L have the same magnitude here; however $\{1, 3\}$ is a more diverse set than $\{1, 2\}$. When $\lambda = 1$, $\{1, 3\}$ is two times more likely to be sampled from the DPP distribution than $\{1, 2\}$. When we increase λ to 2, this 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, every query i has a feature difference vector $\psi_i = \Phi(\xi_{i_1}) - \Phi(\xi_{i_2})$, and close ψ's (in terms of Euclidean distance) correspond to similar queries in terms of the information they provide. Therefore, we let

$$S_{ij} = \exp\left(-\frac{\|\psi_i - \psi_j\|_2^2}{2\sigma_{\text{DPP}}^2}\right), \tag{4.37}$$

where σ_{DPP} is a hyperparameter. However, we are not restricted to this choice — we could use distance metrics other than Euclidean distance. We then define the matrix L as

$$L_{ij} = u_i^{\gamma/\lambda} S_{ij} u_j^{\gamma/\lambda}, \tag{4.38}$$

which is guaranteed to be PSD by the construction of S. Here, γ is another hyperparameter and $u_i \in \mathbb{R}_{\geq 0}$ 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 how much volume they will remove from the belief distribution, as computed by the objective of Equation (3.9). By increasing γ for fixed λ, 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 λ 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 [10, 134, 148, 11].

Approximating the Mode of a DPP

Finding the mode of a DPP exactly is NP-hard [124]. It is hard to even approximate it better than a factor of 2^{xk} for some $x > 0$, under a cardinality constraint of size k [76]. 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

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

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

$$A^{(j+1)} = A^{(j)} \cup \{\arg\max_{i'} \mathrm{Vol}^\lambda(\{L_i\}_{i\in A^{(j)}\cup\{i'\}})\},$$

which we repeat until we obtain k queries in A. Çivril and Magdon-Ismail [75] 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 λ becomes irrelevant, as it is just an exponent in the objective in every iteration of batch generation, unless trivially $\lambda = 0$. This reduces the burden of hyperparameter tuning.

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 comparison-based learning algorithm.

As noted earlier, we work with a discretized set of queries. While this set has K queries, it might be computationally prohibitive to approximate the DPP mode (even greedily) if K is large. In such cases, we first reduce the query set into a smaller set $\mathcal{X}$ by picking the queries which will individually remove the highest volume. Algorithm 5 presents the pseudocode for this procedure.

Algorithm 5 ReduceDataset($\bar{w}, \mathcal{K}, |\mathcal{X}|$)

Input: $\bar{w}_1, \bar{w}_2, \ldots$ ▷ Sampled w estimates
Input: $\mathcal{K} := ((\xi_{1,1}, \xi_{1,2}), \ldots, (\xi_{K,1}, \xi_{K,2}))$ ▷ Dataset of queries
Input: $|\mathcal{X}|$ ▷ Desired size of the reduced query set
for $j = 1, \ldots, K$ **do**
  $\psi_j \leftarrow \Phi(\xi_{j,1}) - \Phi(\xi_{j,2})$
  $u_j \leftarrow \min_{q\in\{\xi_{j,1},\xi_{j,2}\}} \mathbb{E}_{\bar{w}}\left[1 - P(q \mid \bar{w})\right]$ ▷ Volume removal of query j (see Equation (4.3))
end for
$\mathcal{X} \leftarrow \psi_j$'s with $|\mathcal{X}|$ highest u_j values ▷ Reduction
$\mathbf{u} \leftarrow u_j$ values corresponding to $\mathcal{X}$
return $\mathcal{X}, \mathbf{u}$

Afterwards, we approximately compute the mode of the DPP distribution over this reduced set $\mathcal{X}$ as our batch. Algorithm 6 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.

Algorithm 6 DPP-based Batch Generation

Require: DPP hyperparameters σ_{DPP} and γ, sampled w estimates $\bar{w}_1, \bar{w}_2, \dots$

1: $\mathcal{X}, \mathbf{u} \leftarrow \textsc{ReduceDataset}(\bar{w}, \mathcal{K}, |\mathcal{X}|)$
2: **for** $i = 1, \dots, |\mathcal{X}|$ **do**
3: **for** $j = 1, \dots, |\mathcal{X}|$ **do**
4: $S_{ij} \leftarrow \exp\left(-\frac{\|\psi_i - \psi_j\|_2^2}{2\sigma_{\text{DPP}}^2}\right)$
5: $L_{ij} \leftarrow u_i^\gamma S_{ij} u_j^\gamma$
6: **end for**
7: **end for**
8: $A \leftarrow \emptyset$ ▷ Initialize the batch
9: **for** $j = 1, \dots, k$ **do**
10: $A \leftarrow A \cup \{\arg\max_{j'} \det L_{A \cup \{j'\}}\}$
11: **end for**
12: **return** A

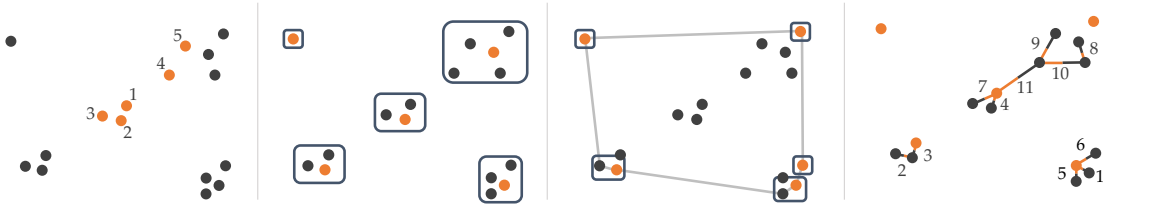
4.8.3 Time-Efficient Batch Active Learning Methods

DPP-based batch active learning method enables us to systematically tune the tradeoff between diversity and informativeness. This approach leads to the best learning performance as we will present in our experiments. However, the DPP method has two important drawbacks: (i) it requires tuning of the hyperparameters σ_{DPP} and γ, and (ii) even approximating the mode of the DPP might take too much time depending on the batch size k and the reduced set size $|\mathcal{X}|$, as we need to compute the matrix L and determinants of some of its submatrices at every greedy iteration. Although the former problem may be tolerated as it can be performed offline, the latter may cause problems in practice. DPP-based method is useful for the cases when parallelization in data collection is desired, but the time-efficiency is not crucial.

However in many cases, we want our approach to be not only parallelizable but also time-efficient. For this purpose, we now describe a set of methods that do not rely on DPPs in increasing order of complexity to provide alternative approximations to the batch active learning problem. Figure 4.38 visualizes each approach for a small set of queries.

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



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

Figure 4.38: 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 volume removal. 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 volume removal.

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 Equation (4.3) among the K discrete queries.

This method is a specific case of the DPP-based approach with $\lambda = 0$ or with $|\mathcal{X}| = k$. 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 volume removal values. For instance, as shown in Figure 4.38a, the 5 orange queries chosen are all going to be very close to the center where volume removal values are high.

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 $\mathcal{X}$ 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 [141] 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 volume removal 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 [122, 24] 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 [195], efficient approximate algorithms exist [17]. Figure 4.38b shows the medoids selection approach, where 5 orange queries are selected from the 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 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 dimensionality 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 4.38c, 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.

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 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 [94, 47]. However, there exists a set of algorithms that provide approximate solutions [62].

What makes our batch generation problem special and different from standard max-sum diversification is that we can compute the volume removal for each query. As in the DPP-based method, volume removal 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 volume removal 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, but again, other distance metrics between queries could also be used) in the reduced set $\mathcal{X}$, and remove the one with lower volume removal 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 7. Figure 4.38d shows the successive pairwise comparisons between two queries based on their corresponding volume removal. 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.

Algorithm 7 Successive Elimination

$\mathcal{X}, \mathbf{u} \leftarrow \text{REDUCEDATASET}(\bar{w}, \mathcal{K}, |\mathcal{X}|)$
$A \leftarrow \mathcal{X}$ ▷ Initialize the batch
while $|A| > k$ **do**
  $(\psi_i, \psi_j) \leftarrow \arg\min_{\psi_i, \psi_j \in A} \|\psi_i - \psi_j\|_2$
  if $u_i < u_j$ **then**
    Remove ψ_i from A
  else
    Remove ψ_j from A
  end if
end while
return A

We make the code for our batch active learning methods available at `https://bit.ly/381brBK`.

4.8.4 Theoretical Guarantees

Theorem 5. *Under the following assumptions:*

1. *The error introduced by the approximation given by Equation* (4.33) *is ignored,*
2. *The error introduced by the sampling of $\bar{w}$'s via adaptive metropolis algorithm is ignored,*

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

Proof. In the DPP-based method, greedy selection and successive elimination, the volume removal maximizer query Q_* out of K possible queries will always remain in the resulting batch of size k, because: (i) the greedy DPP mode approximation will first add this query to the batch, (ii) greedy selection algorithm will first add this query to the batch, and (iii) the queries will be removed in successive elimination only if they have lower volume removal than some other queries in the set. Sadigh et al. [171] proved, by using the ideas from adaptive submodular function maximization literature [125], that if we make the single query Q_* at each iteration, then at least $1 - \epsilon$ times as much volume as removed by the best adaptive non-batch strategy will be removed after $\ln(\frac{1}{\epsilon})$ times as many iterations. The proof is then complete with the pessimistic approach that accepts other $k - 1$ queries will not remove any volume at all. □

4.8.5 Simulations and Experiments

Experimental Setup. We performed several simulations and experiments to compare the methods we propose and to demonstrate their performance. In all experiments, we set batch size $k = 10$,

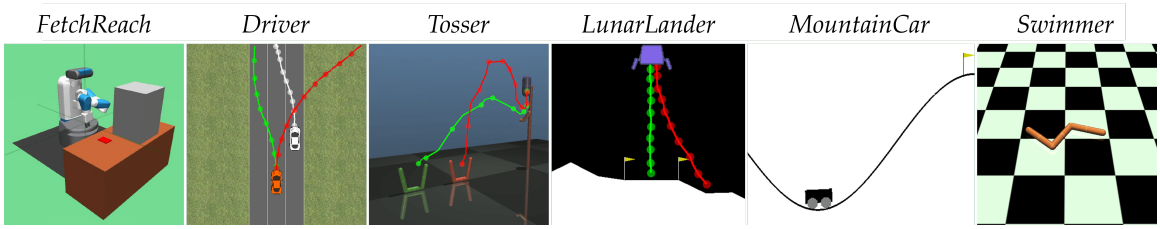


Figure 4.39: Simulation view of each environment. (a) *FetchReach*, (b) *Driver*, (c) *Tosser*, (d) *LunarLander*, (e) *MountainCar*, (f) *Swimmer*.

Table 4.1: Environment Properties

Task Name	$\dim(a_t)$	T	d
LDS	5	1	5
FetchReach	7	19	4
Driver	2	5	4
Tosser	2	2	4
*LunarLander**	2	5	6
*MountainCar**	1	12	3
Swimmer	2	12	3

* Continuous versions

reduced query set size $|\mathcal{X}| = 200$, number of w samples $|\Omega| = 1000$, and assumed a linear reward function: $R_w(\xi) = w^\top \Phi(\xi)$.

Alignment Metric. For our simulations, we generate synthetic random w^* vectors as our true reward function parameters. We again used the `Alignment` metric in order to compare non-batch active, batch active and random query selection methods, where all queries are selected randomly over all feasible trajectories. As a reminder,

$$\texttt{Alignment} = \frac{w^* \cdot \hat{w}}{\|w^*\|_2 \|\hat{w}\|_2}, \tag{4.39}$$

where $\hat{w}$ is $\mathbb{E}[w \mid \mathcal{D}_C]$, the expectation of the learned belief distribution over w. We remind that this `Alignment` metric can be used to test convergence, because its value being close to 1 means the estimate of w is very close to (aligned with) the true reward parameters vector. In our experiments, we compare the methods using `Alignment` and the number of queries made.

Tasks

We perform experiments in different simulation environments that are summarized in Table 4.1 with a list of the variables associated with every environment, where T is the number of time steps in each trajectory, also known as the horizon of the task. The optimization of queries in the non-batch active method (of Section 4.1), is hence over $2 \times (T \dim(a))$ with a fixed initial state s_0, where the

factor 2 is because we generate 2 trajectories for each query. Figure 4.39 visualizes each of the experiment environments with some sample trajectories. Most of these environment were also used in the previous sections, but we now go over them for the the convenience of the reader.

Linear Dynamical System (LDS). We assess the performance of our methods on a simple simulated linear dynamical system:

$$\tilde{s}_{t+1} = A\tilde{s}_t + Ba_t, \quad s_t = C\tilde{s}_t + Da_t \tag{4.40}$$

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 zero matrices and D to be identity matrix in this section. Then a single step simulation of the system results in the observation s_0, which can be treated as $\Phi(\xi)$. Therefore, the control inputs are equal to the features over trajectories, and optimizing over control inputs or features is equivalent. Despite its name, this environment is not meant to be a real dynamical system – instead it measures how our batch active learning algorithms would perform in the simplest linear regression via pairwise comparisons problem.

FetchReach. We use the simulator for Fetch mobile manipulator robot [213], visualized in Figure 4.39a. 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 [170], shown in Figure 4.39b. 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 4.39b. In addition, the white line shows the fixed trajectory of the other vehicle on the road.

Tosser. We use MuJoCo's "Tosser" [191] 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 time step and final distance to closest basket of the object. The two red and green trajectories in Figure 4.39c correspond to synthesized queries showing different preferences for what basket to toss the object to.

LunarLander. We use OpenAI Gym's continuous version of the "LunarLander" environment [50] 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 4.39d.

MountainCar. We use OpenAI Gym's "MountainCar" [50] where a simple 1D car model is controlled on a hill. The features are maximum range in the positive direction, maximum range in the negative direction, and time to reach the flag (or T if not reached). The environment is shown in Figure 4.39e.

Swimmer. We use OpenAI Gym's "Swimmer" [50]. We use features corresponding to horizontal

displacement, vertical displacement, and total distance traveled. The environment is shown in Figure 4.39f.

Comparison of Batch-Active Learning Methods

We first quantitatively compare the batch-active methods we proposed with each other. We use the *LDS*, *FetchReach*, *Driver* and *Tosser* to demonstrate this comparison. For each of these environments, we create a dataset of $K = 100{,}000$ queries.

Independently for each environment, we randomly generated 200 different reward functions (w^* vectors), 100 of which are for tuning γ in the DPP-based method and the remaining 100 are for tests of all methods. The same approach can be employed in practice: One can simulate random reward functions for tuning and then deploy the system to learn the reward functions from real users. For both tuning and tests, we simulated noiseless users, who always reveal their true preferences in order to eliminate the effect of noise in the results. We present the further details and results of hyperparameter tuning in the Appendix C.4.

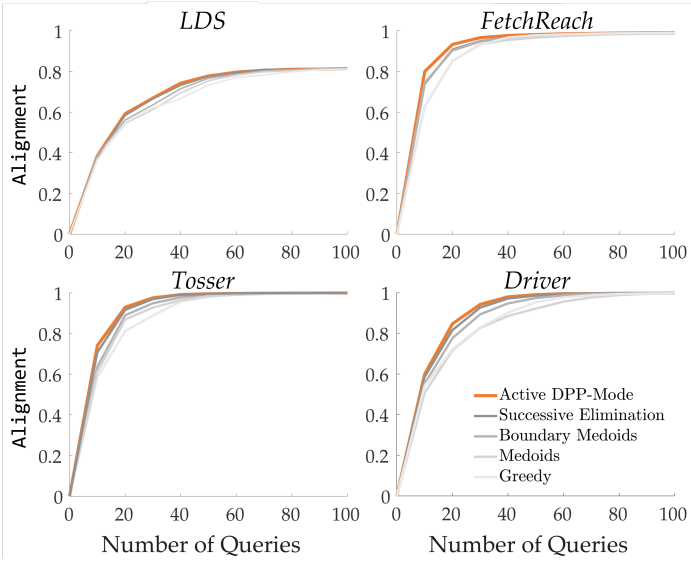


Figure 4.40: Batch-active learning methods are compared.

For each simulated reward function during our tests, we ran 10 batch generations with each method, summing up to 100 pairwise comparison queries. We demonstrate the results in Figure 4.40. Our results suggest that the DPP-based method significantly outperforms all other methods in all environments ($p < 0.05$, Wilcoxon signed-rank test [204]) except for successive elimination in LDS where both algorithms perform comparably.

Among the time-efficient batch active learning methods we proposed, successive elimination method significantly outperforms the others ($p < 0.05$) in all environments except *FetchReach* where

Table 4.2: Average Query Generation Times (seconds)

Environment	**Non-Batch**	**Batch Active Learning**				
		Active DPP-Mode	*Greedy*	*Medoids*	*Boundary Medoids*	*Successive Elimination*
Driver	79.2	5.5	5.4	5.7	5.3	5.5
Tosser	149.3	5.5	4.1	4.3	3.8	3.9
LunarLander	177.4	5.6	4.1	4.1	4.2	4.1
MountainCar	96.4	7.1	3.8	4.0	4.0	3.8
Swimmer	188.9	10.8	3.8	3.9	4.0	4.1

it performs comparably to boundary medoids, significantly outperforms medoids selection ($p < 0.05$), and marginally significantly outperforms greedy selection ($p \approx 0.06$). Similarly, boundary medoids approach significantly outperformed medoids selection and greedy selection in all environments ($p < 0.05$). Finally, medoids selection and greedy selection performed comparably in all environments, except *Tosser* where medoids selection significantly outperformed the greedy approach ($p < 0.05$).

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
3. Boundary Medoids
4. Medoids
5. Greedy

Comparison to Non-Batch Active Learning

We next investigated the average time it required to generate one query. For this, we took a dataset of $K = 500{,}000$ queries. We recorded the batch generation times, and divided it by $k = 10$. To show the advantage of batch-active learning methods, we also ran the same analysis on the non-batch active learning approach that synthesizes trajectories by optimizing over their action spaces along the time horizon. We ran this study for *Driver*, *Tosser*, *LunarLander*, *MountainCar* and *Swimmer* environments. The results are shown in Table 4.2. It can be seen that batch active learning methods lead to a great decrease in query generation times compared to the non-batch method, and the DPP-based method is slightly slower than the other batch algorithms. This slowness could be even more significant for larger batches.

As we observed that successive elimination generates highly informative queries in a time-efficient way without any hyperparameter tuning, 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.

We again conducted our simulation experiments on all environment but *FetchReach* due to its

large action space, which makes the non-batch active learning impractical as it optimizes over the action space along the horizon. For the simulations with *LDS*, we assume that human's preferences are noisy as discussed in Equation (4.33). For all other environments, we again assume an oracle user who responds to queries with no error to avoid perturbations due to noise in responses.

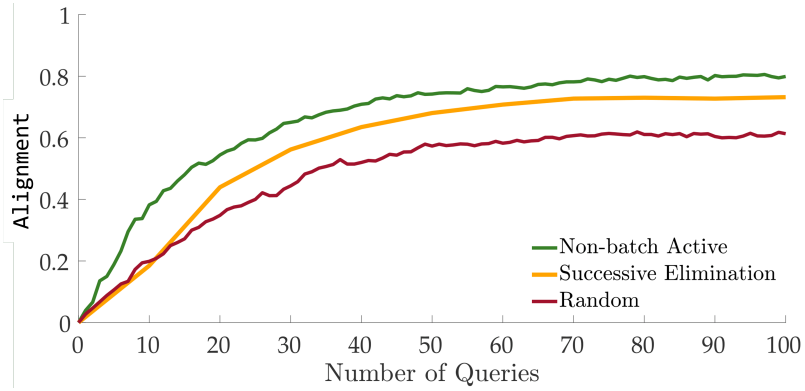


Figure 4.41: The performance of each algorithm is averaged over 10 different runs on *LDS* where w^* is uniformly randomly generated. Successive elimination performs better than the random querying and worse than the non-batch active method.

Figure 4.41 shows the number of queries that result in a corresponding `Alignment` value for each method in the *LDS* environment, averaged over 10 runs. The non-batch active version significantly outperforms successive elimination ($p < 0.05$), as it performs the optimization for each and every query. As expected, both active methods significantly outperform random querying ($p < 0.05$).

We show the results of our experiments on the other five environments in Figures 4.42 and 4.43. Figure 4.42 shows the convergence to the true reward function parameters w^* as the number of queries increases (similar to Figure 4.41). It is interesting to note that non-batch active learning performs suboptimally in *LunarLander* and *Tosser*. We believe this can be due to the non-convex optimization being solved in non-batch methods leading to suboptimal behavior, or because of the

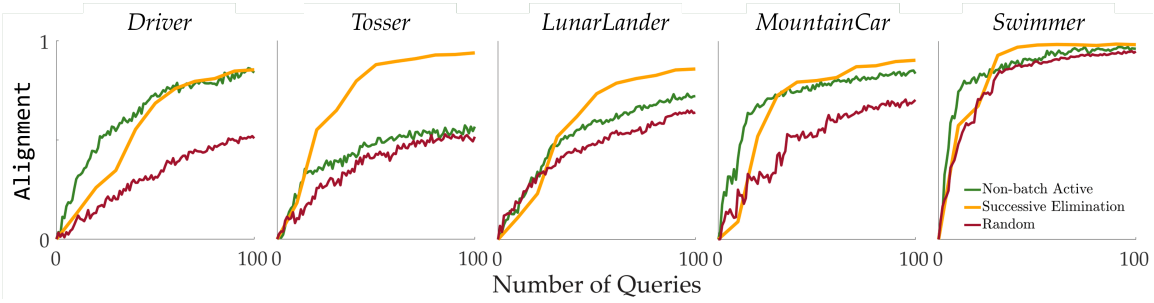


Figure 4.42: The performance the algorithms is shown. The non-batch active method performs poorly on *LunarLander* and *Tosser*.

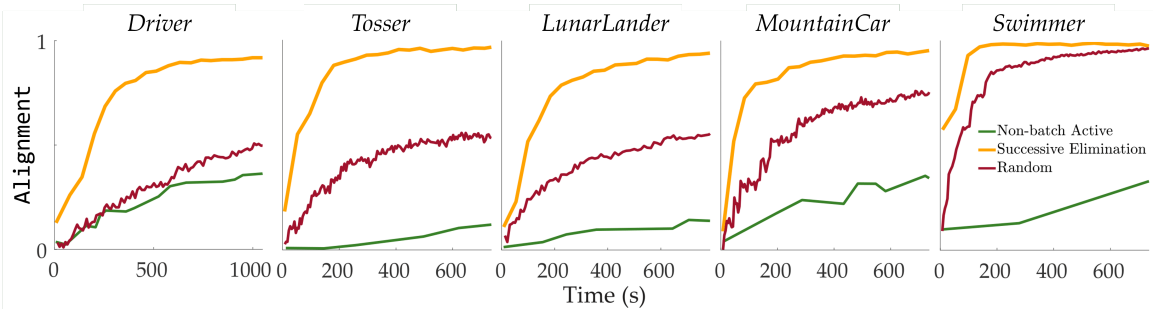


Figure 4.43: Convergence to w^* as a function of time is plotted for each environment. Non-batch active learning method is slow due to the optimization and adaptive metropolis algorithm involved in each iteration, whereas random querying performs poorly due to redundant queries. Successive elimination clearly outperforms both of them.

known failure cases of the volume removal objective as we discussed in Section 4.1. The proposed batch active learning methods overcome this issue thanks to query space discretization and the fact that it does not rely merely on the volume removal values and tries to incorporate diversity among the queries.

Figure 4.43 evaluates the computation time required for querying. It is clearly visible that batch active learning makes the process much faster than the non-batch active method and random querying.

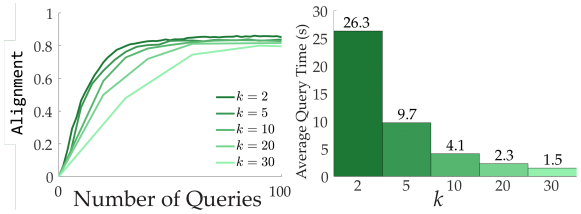


Figure 4.44: The performance of successive elimination algorithm with varying k values was averaged over 10 different runs with *LDS* where w^* is uniformly randomly generated and $|\mathcal{X}| = 20k$. (a) The `Alignment` values, and (b) average query times.

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 4.44 where we simulated *LDS* with varying k values. For this simulation, we set $|\mathcal{X}| = 20k$ in accordance with other experiments.

User Preferences

In addition to our simulation results using synthetic w^* 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.

Setup. We recruited 10 users who responded to 150 queries generated by successive elimination algorithm for each environment (*Driver* or *Tosser*).

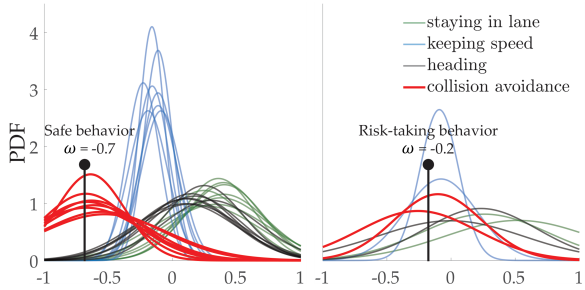


Figure 4.45: User preferences on *Driver* task are grouped into two sets. While the first set shows the preferences conforming with the natural driving behavior, the second set is comprised of data from two users one of whom preferred collisions with the other car over leaving the road and the other 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.

Driver Preferences. Using successive elimination, we are able to learn humans' driving preferences. Our results show that the reward functions of users are very close to each other as this task mainly models natural driving behavior. This is consistent with results shown by Sadigh et al. [171], where non-batch techniques are used. We noticed a few differences between the driving behavior as shown in Figure 4.45. 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 may correspond to more aggressive driving behavior. We observed that 70 queries were enough for 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`.

Tosser Preferences. Similarly, we use successive elimination to learn humans' preferences on the *Tosser* task. Figure 4.46 shows we learn interesting tossing preferences. 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 Figure 4.39c). We note that 100 queries were enough to see reasonable convergence in this task. The evolution of the learning can be watched on `https://youtu.be/cQ7vvUg9rU4`.

4.9 Chapter Summary

In this chapter, we built on the techniques we presented in Chapter 3: we showed how different comparative feedback types, which robots can leverage to learn reward functions, can be actively collected from humans for better data- and time-efficiency. We first started with reviewing an existing acquisition function from the literature, namely volume removal [171], and showed its drawbacks and

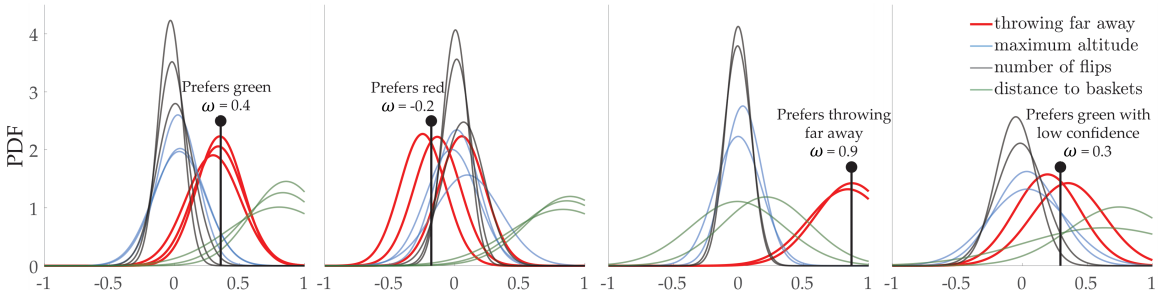


Figure 4.46: User preferences on *Tosser* task are grouped into four sets. The first set shows the preferences of people who aimed at throwing the ball into the green basket (the distant one) but accepted throwing into the other basket is better than not throwing into any baskets. The second set is comprised 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.

failure cases in Section 4.1. We then proposed an alternative objective in Section 4.2 to optimize for data-efficiency: mutual information, a widely used notion from information theory [77]. We showed it does not suffer from the same problems as volume removal even though it has the same computational complexity. We then adopted the same structure as in Chapter 3 to extend each comparative query and reward function type (parameteric and non-parametric) with active querying techniques, one by one from Section 4.3 to 4.7. Finally in Section 4.8, we showed how one can actively generate multiple queries at the same time, i.e., in batches, for parallelizability and better time-efficiency as it avoids solving an optimization problem for each and every query.

Chapter 5

Final Words

We envision a world where robots and agents powered with artificial intelligence seamlessly interact with humans and each other, which may include collaborating, competing, teaching and influencing. We are convinced that they need to be able to learn the objectives or the preferences of other agents to achieve such interactions. Because the information about an agent's objective enables a robot to better predict their behavior, which in turn, enables the robot to condition their interaction on these predictions. Our preliminary works empirically proved this concept in various settings such as human-robot collaboration [42], autonomous driving [60], traffic network optimization [132, 35, 41], or multi-agent learning [199, 25, 225]. Perhaps more importantly, learning humans' objectives in a task means learning how to perform the task itself. This has also been the theme of the simulations and experiments we conducted in this thesis.

Overall, this thesis is an important step towards the goal of reliably learning humans' objectives in various tasks. For more and easier accessibility, we released a software library that implements many of the methods we proposed in this thesis [44]. However, both in that library and in this thesis, we only focused on learning from comparative feedback (possibly in addition to demonstrations) and how to elicit human preferences using such feedback. In real world, when people interact with each other, they often use many more sources of information, such as gaze, gestures, language, facial expressions, etc. Robots are still not fully capable of using these other forms of feedback. While such high-level challenges exist and are yet to be solved, we would like to conclude this thesis by focusing on a discussion of the limitations and future directions of the methods we developed.

5.1 Challenges

Although we proposed several techniques for actively learning from comparative feedback and empirically showed they can be used to learn humans' preferences in different tasks, those techniques are limited in various ways. In this section, we would like to focus on these limitations for both

learning and active querying.

5.1.1 Limitations and Future Work in Learning

In this thesis, we worked with two different assumptions about the reward function that we are trying to learn: it is either a parametric or a non-parametric reward function. In the former, we were constrained to work with functions that have a small number of parameters in practice, because our learning scheme is fully Bayesian. However, many applications in robotics may require more complex reward functions, such as those modeled with deep neural networks. We are not able to efficiently (in terms of both data and time) learn such complex functions using the techniques in this thesis. Even though alternative learning methods that rely on gradient based learning exist, e.g. [52], they often cannot give reliable uncertainties about the learned reward which limits their usability and makes active querying difficult.

In the latter, we used Gaussian processes to model non-parametric reward functions. Although this relaxes some of the assumptions about the functional form of the reward, it brings its own challenges in practice: how large can the input to the Gaussian process be? Even though this approach enabled us to learn complex rewards, we were now limited in terms of the dimensionality of the input (the number of trajectory features), because Gaussian processes are difficult to fit with large amounts of data and increasing the input dimensionality makes it more data-hungry.

Both of these limitations point out another challenge: where do the trajectory features come from? Often, we rely on experts to design such features, but these reward models are prune to errors in feature functions. Although our preliminary works show new features could be discovered using comparative feedback [120], we still rely on the initial set of hand-designed features. Future work should investigate supervised and unsupervised techniques for learning features for reward functions.

Moreover, we adopted and used different models about how humans respond to comparative feedback queries. All of these models implicitly assume humans are rational decision makers: in all models, the most likely human response is the true response based on their underlying reward function. However, humans are bounded rational in various settings and conditions [180, 93, 105]. Our work based on cumulative prospect theory [117, 194] showed humans take consistently suboptimal actions when the system involves some risk [131], even when there are only two action choices, just like the pairwise comparisons setting we have in this thesis. Future work should incorporate this suboptimality or irrationality of humans into the reward learning from comparative feedback framework.[1]

An important limitation of our multimodal reward learning techniques presented in Sections 3.5 and 3.6 is the fact that we assumed we know the number of modes in the reward function. Although

[1] See the recent work by Chan et al. [63] that attempts to model various reasons of irrationality and incorporate them for reward inference.

this might be the case when learning rewards from multiple humans with different objectives, it is unrealistic if we are trying to learn a multimodal reward from one person who has non-stationary preferences. Our techniques could be easily modified to handle such scenarios: similar to clustering algorithms, one can experiment with varying number of modes and then take the simplest model that gives reasonable performance. However, this approach will prevent the robots from using active querying techniques as they rely on a fixed number of modes.

Finally, an interesting research direction is about the interfaces that can be used for collecting comparative feedback. In Section 3.4, we showed we can use a slider bar to collect scale feedback, which gives more information than pairwise comparisons. However, extending this to higher number of trajectories within a query is challenging. For three trajectories, one could think of a 2D plane on which the user selects a point whose distance to the corners indicate how much the user prefers each trajectory. Going beyond three trajectories requires more and more complex interfaces, and perhaps hardware. Even more interestingly, one could leverage the fact that the systems we are trying to teach via comparative feedback are robots that are embodied. This may open new possibilities such as giving feedback to the robot about different parts of the space it is operating in or about different segments of its trajectory (see [78]).

5.1.2 Limitations and Future Work in Active Querying

All of the techniques we proposed for active querying had an implicit assumption: the robot is in an offline training phase. Thanks to this assumption, we are able to optimize our queries to humans specifically for data-efficiency. In other words, we do not have to worry about whether the trajectories we are demonstrating to humans are good or bad: we can show bad trajectories just for the sake of learning. However, in many cases, it is desirable to ask questions while, at the same time, trying to perform the task. Such an online setting would require optimizing when to ask a question during the task, and other acquisition functions for deciding what question to ask.

It is not only that we did not have to worry about the quality of trajectories, but we also did not formulate any hard safety constraints (even though we formulated some soft constraints in Section 3.3 by constructing a region of avoidance in the trajectory space). This prevents one from using learning from comparative feedback techniques in safety critical systems. One possibility is to first utilize safe exploration techniques, e.g., [30, 185, 37], to construct the space of trajectories, but this requires a well-defined safety function. Future work should investigate how safety could also be learned using comparative feedback, similar to the related works on constraint learning from demonstrations [71].

5.2 Closing Thoughts

This thesis is only a step that brings ideas from robotics, machine learning, information theory and control theory to address the reward learning problem in artificial intelligence which we believe to be an important challenge to achieve seamless human-robot (or more generally human-AI) interaction. As we discussed in this chapter, there are still many limitations and challenges that need to be addressed for achieving such interactions. We believe these challenges will require collaborations between researchers from a wide range of fields, such as machine learning, robotics, computational psychology, human-robot interaction, behavioral economics, formal methods and control theory.

Appendix A

Proofs

A.1 Proof of Proposition 1

Proof. To prove the statement, we show the feasible set obtained from scale feedback is a subset of the feasible set from choice feedback. We note $\delta^* > 0$ for any non-trivial problem instance, as otherwise every trajectory would be equally optimal for any w^*. For one of the queries that form $\mathcal{D}_S$ and $\mathcal{D}_C$, say query i, we assume the user prefers $\xi_1^{(i)}$ over $\xi_2^{(i)}$ without loss of generality, implying $\bar{q}^{(i)} \geq 0$. For this query, pairwise comparison feedback defines a feasible set $\Lambda^{(i)}_{\texttt{Comparison}} = \{w \mid R_w(\xi_1^{(i)}) - R_w(\xi_2^{(i)}) \geq 0\}$. First, we consider $\bar{q}^{(i)} = 1$. This yields $\Lambda^{(i)}_{\texttt{Scale}} = \{w \mid R_w(\xi_1^{(i)}) - R_w(\xi_2^{(i)}) \geq \varrho\delta(w)\}$. Since both $\varrho > 0$ and $\delta(w) \geq 0$, we obtain $\Lambda^{(i)}_{\texttt{Scale}} \subseteq \Lambda^{(i)}_{\texttt{Comparison}}$. For the case $\bar{q}^{(i)} \in [0, 1)$, we have $\Lambda^{(i)}_{\texttt{Scale}} = \{w \mid R_w(\xi_1^{(i)}) - R_w(\xi_2^{(i)}) = \bar{q}^{(i)}\varrho\delta(w)\}$; the right hand side is non-negative and thus any w satisfying the equality must satisfy $R_w(\xi_1^{(i)}) - R_w(\xi_2^{(i)}) \geq 0$. This also implies $\Lambda^{(i)}_{\texttt{Scale}} \subseteq \Lambda^{(i)}_{\texttt{Comparison}}$. As $\texttt{Err}^{\max}(w^*, \mathcal{D}_S)$ maximizes over $\Lambda_{\texttt{Scale}}$, which is the intersection of $\Lambda^{(i)}_{\texttt{Scale}}$'s over queries, while $\texttt{Err}^{\max}(w^*, \mathcal{D}_C)$ maximizes over $\Lambda_{\texttt{Comparison}}$, $\texttt{Err}^{\max}(w^*, \mathcal{D}_S)$ cannot attain a larger value than $\texttt{Err}^{\max}(w^*, \mathcal{D}_C)$. □

A.2 Volume Removal Equivalence when $|Q^{(i)}| = 2$

While presenting the volume removal optimization for active learning in Equations (4.2) and (4.3), we stated we could maximize the worst-case volume removal when $|Q^{(i)}| = 2$ so that a large amount of volume will be removed regardless of the human's response. This objective is in fact equal to the expected volume removal objective while learning from pairwise comparisons.

Theorem 6. *While learning from pairwise comparisons, the worst-case volume removal maximization in Equation* (4.3) *is equivalent to expected volume removal maximization presented in Equation* (4.2)*:*

$$\max_{Q^{(i)}=\{\xi_1,\xi_2\}} \mathbb{E}_q^{(i)} \left[\int_w \left(b^{i-1}(w) - P(q^{(i)} \mid Q^{(i)} w) b^{i-1}(w) \right) dw \right]$$

Proof. We first work on the worst-case optimization objective:

$$\begin{aligned}
&\min_{q^{(i)} \in Q^{(i)}} \int_w \left(b^{i-1}(w) - P(q^{(i)} \mid Q^{(i)}, w) b^{i-1}(w) \right) dw \\
&\quad = \min_{q^{(i)} \in Q^{(i)}} \left(1 - \int_w P(q^{(i)} \mid Q^{(i)}, w) b^{i-1}(w) dw \right) \\
&\quad = \min_{q^{(i)} \in Q^{(i)}} \left(1 - \mathbb{E}_{w \sim b^{i-1}(w)} \left[P(q^{(i)} \mid Q^{(i)}, w) \right] \right) \\
&\quad = \min_{q^{(i)} \in Q^{(i)}} \left(1 - P(q^{(i)} \mid Q^{(i)}, b^{i-1}) \right) \\
&\quad = \min \left\{ P(q^{(i)} = Q_1^{(i)} \mid Q^{(i)}, b^{i-1}), P(q^{(i)} = Q_2^{(i)} \mid Q^{(i)}, b^{i-1}) \right\} ,
\end{aligned}$$

where the last equation is because the queries are pairwise and $P(q^{(i)} = Q_1^{(i)} \mid Q^{(i)}, b^{i-1}) + P(q^{(i)} = Q_2^{(i)} \mid Q^{(i)}, b^{i-1}) = 1$. Hence, we are trying to find queries for which the minimum of those two terms is maximized. An optimal query is therefore one for which our model predicts $P(q^{(i)} = Q_1^{(i)} \mid Q^{(i)}, b^{i-1}) = P(q^{(i)} = Q_2^{(i)} \mid Q^{(i)}, b^{i-1}) = 0.5$. Intuitively, we are looking for queries where our model is highly unsure about the human's response. We know that $0 \leq P(q^{(i)} = Q_1^{(i)} \mid Q^{(i)}, b^{i-1}), P(q^{(i)} = Q_2^{(i)} \mid Q^{(i)}, b^{i-1}) \leq 1$. Hence, we note that this optimization is equivalent to optimizing $P(q^{(i)} = Q_1^{(i)} \mid Q^{(i)}, b^{i-1}) P(q^{(i)} = Q_2^{(i)} \mid Q^{(i)}, b^{i-1})$.

Next, we work on the expected volume removal objective:

$$\begin{aligned}
&\mathbb{E}_{q^{(i)} \mid Q^{(i)}, b^{i-1}} \int_w \left(b^{i-1}(w) - P(q^{(i)} \mid Q^{(i)}, w) b^{i-1}(w) \right) dw \\
&\quad = \mathbb{E}_{q^{(i)} \mid Q^{(i)}, b^{i-1}} \left[1 - \int_w P(q^{(i)} \mid Q^{(i)}, w) b^{i-1}(w) dw \right] \\
&\quad = \mathbb{E}_{q^{(i)} \mid Q^{(i)}, b^{i-1}} \left[1 - \mathbb{E}_{w \sim b^{i-1}(w)} \left[P(q^{(i)} \mid Q^{(i)}, w) \right] \right] \\
&\quad = \mathbb{E}_{q^{(i)} \mid Q^{(i)}, b^{i-1}} \left[1 - P(q^{(i)} \mid Q^{(i)}, b^{i-1}) \right] \\
&\quad = 2 P(q^{(i)} = Q_1^{(i)} \mid Q^{(i)}, b^{i-1}) P(q^{(i)} = Q_2^{(i)} \mid Q^{(i)}, b^{i-1}) ,
\end{aligned}$$

and so optimizing this objective is equivalent to optimizing $P(q^{(i)} = Q_1^{(i)} \mid Q^{(i)}, b^{i-1}) P(q^{(i)} = Q_2^{(i)} \mid Q^{(i)}, b^{i-1})$ again. □

A.3 Proof of Theorem 2

Proof. We need to show if the global optimum is negative, then any longer-horizon optimization will also give negative reward (difference between information gain and the cost) in expectation. Let $Q_*^{(i)}$ denote the global optimizer. For any $i' \geq 0$,

$$
\begin{aligned}
&I(q^{(i)}, \ldots, q^{(i+i')}; w \mid Q^{(i)}, \ldots, Q^{(i+i')}) - \sum_{j=0}^{i'} c(Q^{(i+j)}) \\
&\quad = I(q^{(i)}; w \mid Q^{(i)}) + \ldots + \\
&\qquad I(q^{(i+i')}; w \mid q^{(i)}, \ldots, q^{(i+i'-1)}, Q^{(i)}, \ldots, Q^{(i+i')}) - \sum_{j=0}^{i'} c(Q^{(i+j)}) \\
&\quad \leq I(q^{(i)}; w \mid Q^{(i)}) + \ldots + I(q^{(i+i')}; w \mid Q^{(i+i')}) - \sum_{j=0}^{i'} c(Q^{(i+j)}) \\
&\quad \leq (i'+1) \left[I(q^{(i)}; w \mid Q_*^{(i)}) - c(Q_*^{(i)}) \right] < 0
\end{aligned} \tag{A.1}
$$

where the first inequality is due to the submodularity of the mutual information, and the second inequality is because $Q_*^{(i)}$ is the global maximizer of the greedy objective. The other direction of the proof is very clear: If the global optimizer is nonnegative, then querying $Q_*^{(i)}$ will not decrease the cumulative active learning reward in expectation, so stopping is not optimal. □

A.4 Proof of Corollary 1

Proof. Suppose we have such a mixture of M Plackett-Luce models that is not identifiable. Then, there must exist two distinct sets of parameters (w, α) and (w', α') such that for every query Q, the induced ranking distributions q_1 and q_2 respectively are identical. But since (w, α) and (w', α') are distinct, there is either (1) two mixing coefficients in (w, α) and (w', α') that disagree or (2) two trajectories ξ_1 and ξ_2 that have a different difference in rewards across (w, α) and (w', α') under one of the reward functions. Let $\bar{Q}$ with corresponding ranking distribution $\bar{q}$ be an arbitrary query in case (1) and an arbitrary query containing ξ_1 and ξ_2 in case (2). Note that $\bar{q}$ is the marginal distribution of the overall Plackett-Luce distribution, which by construction is a mixture of M Plackett-Luce models with parameters (w, α) and (w', α'), restricted to the trajectories in $\bar{Q}$. But now there are two distinct sets of parameters representing the distribution over the full ranking of Q since we know (w, α) and (w', α') differ on the restricted set of trajectories $\Xi' = Q$ (either because they have differing mixing coefficients or because their induced rewards on Ξ' are not a within a constant additive factor of each other since ξ_1 and ξ_2 are in Ξ'). But we know $|\Xi'| = |Q|$, so this finding contradicts the fact that $\bar{Q}$ must be identifiable by Theorem 4. We conclude every mixture of M Plackett-Luce models is identifiable subject to the query size bounds in the statement of this

corollary. □

A.5 Justification for Remark 2

Here, we define the optimal adaptive set of queries $\mathcal{D}_R^*$ to be the one which, in expectation, minimizes the uncertainty over model parameters $H(w, \alpha \mid \mathcal{D}_R^*)$. It is a well-known result that for *adaptive submodular* functions, greedy optimization yields results that are within a constant factor $(1 - \frac{1}{e})$ of optimality [95]. While our mutual information objective in Eqn. (4.28) is adaptive submodular in the non-adaptive setting (where all queries Q are selected before observing their results), in our adaptive setting these guarantees no longer hold (conditional entropy is only submodular with respect to conditioned variables if those variables are unobserved).

Appendix B

Derivations

B.1 Mutual Information Derivation for Section 4.2

We first present the full derivation of Equation (4.12),

$$Q_*^{(i)} = \underset{Q^{(i)}=\{\xi_1,\ldots,\xi_{|Q^{(i)}|}\}}{\arg\max} I(q^{(i)}; w \mid Q^{(i)}, b^{i-1}) \,.$$

We first write the mutual information as the difference between two entropy terms:

$$I(q^{(i)}; w \mid Q^{(i)}, b^{i-1}) = H(w \mid Q^{(i)}, b^{i-1}) - \mathbb{E}_{q^{(i)}|Q^{(i)},b^{i-1}}\left[H(w \mid q^{(i)}, Q^{(i)}, b^{i-1})\right] \,. \tag{B.1}$$

Next, we expand the entropy expressions and use $P(q^{(i)} \mid Q^{(i)}, b^{i-1})P(w \mid q^{(i)}, Q^{(i)}, b^{i-1}) = P(w, q^{(i)} \mid Q^{(i)}, b^{i-1})$ to combine the expectations for the second term to get:

$$\begin{aligned} &H(w \mid Q^{(i)}, b^{i-1}) - \mathbb{E}_{q^{(i)}|Q^{(i)},b^{i-1}}\left[H(w \mid q^{(i)}, Q^{(i)}, b^{i-1})\right] \\ &\quad = -\mathbb{E}_{w|Q^{(i)},b^{i-1}}\left[\log_2 P(w \mid Q^{(i)}, b^{i-1})\right] + \mathbb{E}_{w,q^{(i)}|Q^{(i)},b^{i-1}}\left[\log_2\left(P(w \mid q^{(i)}, Q^{(i)}, b^{i-1})\right)\right] \,. \end{aligned} \tag{B.2}$$

Since the first term is independent from $q^{(i)}$, we can write this expression as

$$\begin{aligned} &\mathbb{E}_{w,q^{(i)}|Q^{(i)},b^{i-1}}\left[\log_2 P(w \mid q^{(i)}, Q^{(i)}, b^{i-1}) - \log_2 P(w \mid Q^{(i)}, b^{i-1})\right] \\ &= \mathbb{E}_{w,q^{(i)}|Q^{(i)},b^{i-1}}\left[\log_2 P(q^{(i)} \mid Q^{(i)}, b^{i-1}, w) - \log_2 P(q^{(i)} \mid Q^{(i)}, b^{i-1})\right] \\ &= \mathbb{E}_{w,q^{(i)}|Q^{(i)},b^{i-1}}\left[\log_2 P(q^{(i)} \mid Q^{(i)}, w) - \log_2\left(\int P(q^{(i)} \mid Q^{(i)}, w')P(w' \mid Q^{(i)}, b^{i-1})dw'\right)\right] , \end{aligned} \tag{B.3}$$

where the integral is taken over all possible values of w.

Having Ω as a set of samples drawn from the prior b^{i-1},

$$\begin{aligned}
I(q^{(i)}; w \mid Q^{(i)}) &\doteq \mathbb{E}_{w,q^{(i)}|Q^{(i)},b^{i-1}}\left[\log_2 P(q^{(i)} \mid Q^{(i)}, w) - \log_2\left(\frac{1}{|\Omega|}\sum_{w'\in\Omega} P(q^{(i)} \mid Q^{(i)}, w')\right)\right] \\
&= \mathbb{E}_{w,q^{(i)}|Q^{(i)},b^{i-1}}\left[\log_2 \frac{|\Omega| \cdot P(q^{(i)} \mid Q^{(i)}, w)}{\sum_{w'\in\Omega} P(q^{(i)} \mid Q^{(i)}, w')}\right] \\
&= \mathbb{E}_{w|Q^{(i)},b^{i-1}}\left[\mathbb{E}_{q^{(i)}|Q^{(i)},w}\left[\log_2 \frac{|\Omega| \cdot P(q^{(i)} \mid Q^{(i)}, w)}{\sum_{w'\in\Omega} P(q^{(i)} \mid Q^{(i)}, w')}\right]\right] \\
&= \mathbb{E}_{w|Q^{(i)},b^{i-1}}\left[\sum_{q^{(i)}\in Q^{(i)}} P(q^{(i)} \mid Q^{(i)}, w)\log_2 \frac{|\Omega| \cdot P(q^{(i)} \mid Q^{(i)}, w)}{\sum_{w'\in\Omega} P(q^{(i)} \mid Q^{(i)}, w')}\right] \\
&\doteq \frac{1}{|\Omega|}\sum_{q^{(i)}\in Q^{(i)}}\sum_{\bar{w}\in\Omega} P(q^{(i)} \mid Q^{(i)}, \bar{w})\log_2 \frac{|\Omega| \cdot P(q^{(i)} \mid Q^{(i)}, \bar{w})}{\sum_{w'\in\Omega} P(q^{(i)} \mid Q^{(i)}, w')}\,, \qquad \text{(see 4.12)}
\end{aligned}$$

where, in the last step, we use the sampled w's to compute the expectation over $w \mid Q^{(i)}, b^{i-1}$, which is equivalent to $w \mid b^{i-1}$. This completes the derivation.

B.1.1 Extension to User-Specific and Unknown ς

We now derive the mutual information optimization when the minimum perceivable difference parameter ς of the extended human model (for weak pairwise comparison queries) we introduced in Section 4.2.4 is unknown. One can also attempt to learn the rationality coefficient β^C. Therefore, for generality, we denote all human model parameters that will be learned as a vector κ. Furthermore, we denote the belief over (w, κ) before iteration i as b_+^{i-1}. Since our true goal is to learn w, the optimization now becomes:

$$Q_*^{(i)} = \underset{Q^{(i)}=\{\xi_1,\ldots,\xi_{|Q^{(i)}|}\}}{\arg\max}\ \mathbb{E}_{\kappa|Q^{(i)},b_+^{i-1}}\left[I(q^{(i)}; w \mid Q^{(i)}, b_+^{i-1})\right] \tag{B.4}$$

We now work on this objective as follows:

$$
\begin{aligned}
&\mathbb{E}_{\kappa|Q^{(i)},b_+^{i-1}}\left[I(q^{(i)};w \mid Q^{(i)},b_+^{i-1})\right] \\
&= \mathbb{E}_{\kappa|Q^{(i)},b_+^{i-1}}\left[H(w \mid \kappa,Q^{(i)},b_+^{i-1}) - \mathbb{E}_{q^{(i)}|\kappa,Q^{(i)},b_+^{i-1}}\left[H(w \mid q^{(i)},\kappa,Q^{(i)},b_+^{i-1})\right]\right] \\
&= \mathbb{E}_{\kappa|Q^{(i)},b_+^{i-1}}\left[H(w \mid \kappa,Q^{(i)},b_+^{i-1})\right] - \mathbb{E}_{\kappa,q^{(i)}|Q^{(i)},b_+^{i-1}}\left[H(w \mid q^{(i)},\kappa,Q^{(i)},b_+^{(i)})\right] \\
&= -\mathbb{E}_{\kappa,w|Q^{(i)},b_+^{i-1}}\left[\log_2 P(w \mid \kappa,Q^{(i)},b_+^{i-1})\right] + \mathbb{E}_{\kappa,q^{(i)},w|Q^{(i)},b_+^{i-1}}\left[\log_2 P(w \mid q^{(i)},\kappa,Q^{(i)},b_+^{i-1})\right] \\
&= \mathbb{E}_{\kappa,q^{(i)},w|Q^{(i)},b_+^{i-1}}\left[\log_2 P(w \mid q^{(i)},\kappa,Q^{(i)},b_+^{i-1}) - \log_2 P(w \mid \kappa,Q^{(i)},b_+^{i-1})\right] \\
&= \mathbb{E}_{\kappa,q^{(i)},w|Q^{(i)},b_+^{i-1}}\left[\log_2 P(q_i \mid w,\kappa,Q^{(i)},b_+^{i-1}) - \log_2 P(q^{(i)} \mid \kappa,Q^{(i)},b_+^{i-1})\right] \\
&= \mathbb{E}_{\kappa,q^{(i)},w|Q^{(i)},b_+^{i-1}}\left[\log_2 P(q^{(i)} \mid w,\kappa,Q^{(i)},b_+^{i-1}) - \log_2 P(\kappa,q(i) \mid Q^{(i)},b_+^{i-1}) + \log_2 P(\kappa \mid Q^{(i)},b_+^{i-1})\right]
\end{aligned}
\tag{B.5}
$$

Noting that $P(\kappa \mid Q^{(i)},b_+^{i-1}) = P(\kappa \mid b_+^{i-1})$, we drop the last term because it does not involve the optimization variable $Q^{(i)}$. Also noting $P(q^{(i)} \mid w,\kappa,Q^{(i)},b_+^{i-1}) = P(q^{(i)} \mid w,\kappa,Q^{(i)})$, the new objective is:

$$
\begin{aligned}
&\mathbb{E}_{\kappa,q^{(i)},w|Q^{(i)},b_+^{i-1}}\left[\log_2 P(q^{(i)} \mid w,\kappa,Q^{(i)}) - \log_2 P(\kappa,q^{(i)} \mid Q^{(i)},b_+^{i-1})\right] \\
&\doteq \frac{1}{|\Omega^+|}\sum_{(\bar{w},\bar{\kappa})\in\Omega^+}\sum_{q^{(i)}\in Q^{(i)}} P(q^{(i)} \mid \bar{w},\bar{\kappa},Q^{(i)})\left[\log_2 P(q^{(i)} \mid \bar{w},\bar{\kappa},Q^{(i)}) - \log_2 P(\bar{\kappa},q^{(i)} \mid Q^{(i)},b_+^{i-1})\right]
\end{aligned}
\tag{B.6}
$$

where Ω^+ is a set containing samples from b_+^{i-1}. Since $P(\bar{\kappa},q^{(i)} \mid Q^{(i)},b_+^{i-1}) = \int P(q^{(i)} \mid \bar{\kappa},w',Q^{(i)})P(\bar{\kappa},w' \mid Q^{(i)},b_+^{i-1})dw'$ where the integration is over all possible values of w, we can write the second logarithm term as:

$$
\log_2\left(\frac{1}{|\Omega^+|}\sum_{w'\in\Omega(\bar{\kappa})} P(q^{(i)} \mid \bar{\kappa},w',Q^{(i)})\right) \tag{B.7}
$$

with asymptotic equality, where $\Omega(\bar{\kappa})$ is the set that contains samples from b_+^{i-1} with fixed $\bar{\kappa}$. Note that while we can actually compute this objective, it is computationally much heavier than the case without κ, because we need to sample w for each $\bar{\kappa}$ sample.

One property of this objective that will ease the computation is the fact that it is parallelizable. An alternative approach is to actively learn (w,κ) instead of just w. This will of course cause some performance loss, because we are only interested in w. However, if we learn them together, the derivation follows the derivation of Equation (4.12), which we already presented, by simply replacing

w with (w, κ), and the final optimization becomes:

$$\underset{Q^{(i)}=\{\xi_1,\ldots,\xi_{|Q^{(i)}|}\}}{\arg\max} \frac{1}{|\Omega^+|} \sum_{q^{(i)}\in Q^{(i)}} \sum_{(\bar{w},\bar{\kappa})\in\Omega^+} P(q^{(i)} \mid Q^{(i)}, \bar{w}, \bar{\kappa}) \log_2 \frac{|\Omega^+| \cdot P(q^{(i)} \mid Q^{(i)}, \bar{w}, \bar{\kappa})}{\sum_{(w',\kappa')\in\Omega^+} P(q_i \mid Q_i, w', \kappa')}$$

B.2 Mutual Information Derivation for Section 4.3

Let Σ be the posterior covariance matrix between $f(\Phi^{(1)})$ and $f(\Phi^{(2)})$. And let

$$\Sigma^{-1} = \begin{bmatrix} c & d \\ d & c' \end{bmatrix}.$$

Note that the c here is not related to the cost function c we used in Section 4.2. Throughout the derivation, all integrals are calculated over $\mathbb{R}$, but we drop it to simplify the notation. h denotes the binary entropy function, and Φ is the cdf of the standard normal distribution. We use f_1 and f_2 to denote $f(\Phi^{(1)})$ and $f(\Phi^{(2)})$, respectively. We write the first entropy term in the optimization (4.18) as:

$$\begin{aligned}
&H(q \mid \Phi^{(1)}, \Phi^{(2)}, \mathbf{Q}, \mathbf{q}) \\
&= h\left(\int\int \Phi\left(\frac{f_1 - f_2}{\sqrt{2}\sigma_C}\right) \mathcal{N}([f_1, f_2] \mid [\mu^{(1)}, \mu^{(2)}], \Sigma) df_2 df_1\right) \\
&= h\left(\frac{\sqrt{cc' - d^2}}{2\pi} \int\int \Phi\left(\frac{f_1 - f_2}{\sqrt{2}\sigma_C}\right) e^{-\frac{1}{2}(c(f_1-\mu^{(1)})^2 + c'(f_2-\mu^{(2)})^2 + 2d(f_1-\mu^{(1)})(f_2-\mu^{(2)}))} df_2 df_1\right) \\
&= h\left(\frac{\sqrt{cc' - d^2}}{2\pi} \int\int \Phi\left(\frac{f_1 - f_2}{\sqrt{2}\sigma_C}\right) e^{-\frac{1}{2}(c((f_1-\mu^{(1)})^2 + \frac{2d}{c}(f_1-\mu^{(1)})(f_2-\mu^{(2)})) + c'(f_2-\mu^{(2)})^2)} df_1 df_2\right) \\
&= h\left(\frac{\sqrt{cc' - d^2}}{2\pi} \int\int \Phi\left(\frac{f_1 - f_2}{\sqrt{2}\sigma_C}\right) e^{-\frac{1}{2}(c(f_1-\mu^{(1)} + \frac{d}{c}(f_2-\mu^{(2)}))^2 - \frac{d^2}{c}(f_2-\mu^{(2)})^2 + c'(f_2-\mu^{(2)})^2)} df_1 df_2\right) \\
&= h\left(\frac{\sqrt{cc' - d^2}}{2\pi} \int e^{-\frac{1}{2}c'(f_2-\mu^{(2)})^2} e^{\frac{1}{2}\frac{d^2}{c}(f_2-\mu^{(2)})^2} \int \Phi\left(\frac{f_1 - f_2}{\sqrt{2}\sigma_C}\right) e^{-\frac{1}{2}(c(f_1-\mu^{(1)} + \frac{d}{c}(f_2-\mu^{(2)}))^2)} df_1 df_2\right) \\
&= h\left(\frac{\sqrt{cc' - d^2}}{2\pi} \int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f_2-\mu^{(2)})^2} \int \frac{\Phi\left(\frac{f_1 - f_2}{\sqrt{2}\sigma_C}\right) e^{-\frac{1}{2}(c(f_1-\mu^{(1)} + \frac{d}{c}(f_2-\mu^{(2)}))^2)}}{\frac{\sqrt{2\pi}}{\sqrt{c}}} \frac{\sqrt{2\pi}}{\sqrt{c}} df_1 df_2\right) \\
&= h\left(\frac{\sqrt{cc' - d^2}}{\sqrt{2\pi}\sqrt{c}} \int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f_2-\mu^{(2)})^2} \int \frac{\Phi\left(\frac{f_1 - f_2}{\sqrt{2}\sigma_C}\right) e^{-\frac{1}{2}(c(f_1-\mu^{(1)} + \frac{d}{c}(f_2-\mu^{(2)}))^2)}}{\frac{\sqrt{2\pi}}{\sqrt{c}}} df_1 df_2\right) \qquad \text{(B.8)}
\end{aligned}$$

Using the mathematical identity $\int_x \phi(x)N(x|\mu,{\sigma_C}^2)dx = \phi(\frac{\mu}{\sqrt{1+{\sigma_C}^2}})$, we obtain

$$
\begin{aligned}
&H(q \mid \Phi^{(1)}, \Phi^{(2)}, \mathbf{Q}, \mathbf{q}) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}} \int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f_2-\mu^{(2)})^2} \Phi\left(\frac{\mu^{(1)} - \frac{d}{c}f_2 + \frac{d}{c}\mu^{(2)} - f_2}{\sqrt{2}\sigma_C\sqrt{1+\frac{1}{2c{\sigma_C}^2}}}\right) df_2\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}} \int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f_2-\mu^{(2)})^2} \Phi\left(\frac{\mu^{(1)} + \frac{d}{c}\mu^{(2)} - (\frac{d}{c}+1)f_2}{\sqrt{2}\sigma_C\sqrt{1+\frac{1}{2c{\sigma_C}^2}}}\right) df_2\right) \\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}} \int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f_2-\mu^{(2)})^2} \frac{\Phi\left(\frac{-(\frac{d}{c}+1)(f_2-\frac{\mu^{(1)}+\frac{d}{c}\mu^{(2)}}{\frac{d}{c}+1})}{\sqrt{2}\sigma_C\sqrt{1+\frac{1}{2c{\sigma_C}^2}}}\right)}{\frac{\sqrt{2\pi c}}{\sqrt{c'c-d^2}}} \frac{\sqrt{2\pi c}}{\sqrt{c'c-d^2}} df_2\right)
\end{aligned}
\tag{B.9}
$$

Using the same identity again,

$$
H(q \mid \Phi^{(1)}, \Phi^{(2)}, \mathbf{Q}, \mathbf{q}) = h\left(\Phi\left(\frac{\mu^{(1)} - \mu^{(2)}}{\sqrt{2}\sigma_C\sqrt{1+\frac{1}{2c{\sigma_C}^2}}\sqrt{1+\frac{c}{2{\sigma_C}^2+\frac{1}{c}}\frac{(1+\frac{d}{c})^2}{c'c-d^2}}}\right)\right) \tag{B.10}
$$

One can then expand the expression in the denominator and use the facts that $\mathrm{Var}(f(\Phi^{(1)})) = \frac{c'}{cc'-d^2}$, $\mathrm{Var}(f(\Phi^{(2)})) = \frac{c}{cc'-d^2}$ and $\mathrm{Cov}(f(\Phi^{(1)}), f(\Phi^{(2)})) = \frac{-d}{cc'-d^2}$ to obtain

$$
H(q \mid \Phi^{(1)}, \Phi^{(2)}, \mathbf{Q}, \mathbf{q}) = h\left(\Phi\left(\frac{\mu^{(1)} - \mu^{(2)}}{\sqrt{2{\sigma_C}^2 + g(\Phi^{(1)}, \Phi^{(2)})}}\right)\right). \tag{B.11}
$$

where $g(\Phi^{(1)}, \Phi^{(2)}) = \mathrm{Var}(f(\Phi^{(1)})) + \mathrm{Var}(f(\Phi^{(2)})) - 2\mathrm{Cov}(f(\Phi^{(1)}), f(\Phi^{(2)}))$

We next make the derivation for the second entropy term. To simplify the notation, we let ${\sigma_C}'^2 = \frac{\pi ln(2)}{2}, {\sigma_C}''^2 = {\sigma_C}'^2 + \frac{1}{c}$, and $\sigma_{C_b}^2 = \frac{c(1+\frac{d}{c})^2}{c'c-d^2}$. By performing a linearization over the

logarithm of the second entropy term as in [112],

$$\begin{aligned}
&\mathbb{E}_{f\sim P(f|\mathbf{Q},\mathbf{q})}\left[H(q \mid \Phi^{(1)},\Phi^{(2)},f)\right] \\
&\approx \frac{\sqrt{c'c-d^2}}{2\pi}\int\int e^{-\frac{(f_1-f_2)^2}{\pi ln(2)}} e^{-\frac{1}{2}(c(f_1-\mu^{(1)})^2+c'(f_2-\mu^{(2)})^2+2d(f_1-\mu^{(1)})(f_2-\mu^{(2)}))} df_1 df_2 \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'(f_2-\mu^{(2)})^2}\int e^{-\frac{(f_1-f_2)^2}{2\sigma_{C'}^2}} e^{-\frac{1}{2}(c(f_1-\mu^{(1)})^2+2d(f_1-\mu^{(1)})(f_2-\mu^{(2)}))} df_1 df_2 \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'{f_2}^2}\int e^{-\frac{(f_1+\mu^{(1)}-f_2-\mu^{(2)})^2}{2\sigma_{C'}^2}} e^{-\frac{1}{2}c{f_1}^2-df_1f_2} df_1 df_2 \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'{f_2}^2}\int e^{-\frac{(f_1+\mu^{(1)}-f_2-\mu^{(2)})^2}{2\sigma_{C'}^2}} e^{-\frac{1}{2}c(f_1+\frac{d}{c}f_2)^2+\frac{1}{2}\frac{d^2}{c}{f_2}^2} df_1 df_2 \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}{f_2}^2}\int e^{-\frac{(f_1-f_2+\mu^{(1)}-\mu^{(2)})^2}{2\sigma_{C'}^2}} e^{-\frac{1}{2}c(f_1+\frac{d}{c}f_2)^2} df_1 df_2
\end{aligned} \tag{B.12}$$

By the change of variables for the inner integral with $u = f_1 + \frac{d}{c}f_2$,

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{Q},\mathbf{q})}\left[H(q \mid \Phi^{(1)},\Phi^{(2)},f)\right] &= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{c'c-d^2}{2c}{f_2}^2}\int_u e^{-\frac{(u-\frac{d}{c}f_2+\mu^{(1)}-f_2-\mu^{(2)})^2}{2\sigma_{C'}^2}} e^{-\frac{1}{2}cu^2} du df_2 \\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{c'c-d^2}{2c}{f_2}^2}\int_u e^{-\frac{((1+\frac{d}{c})f_2-u-\mu^{(1)}+\mu^{(2)})^2}{2\sigma_{C'}^2}} e^{-\frac{1}{2}cu^2} du df_2
\end{aligned} \tag{B.13}$$

By another change of variables for the outer integral with $v = \frac{f_2}{1+d/c}$,

$$\mathbb{E}_{f\sim P(f|\mathbf{Q},\mathbf{q})}\left[H(q \mid \Phi^{(1)},\Phi^{(2)},f)\right] = \frac{1}{1+\frac{d}{c}}\frac{\sqrt{c'c-d^2}}{2\pi}\int_v e^{-\frac{c'c-d^2}{2c}\frac{v^2}{(1+\frac{d}{c})^2}}\int_u e^{-\frac{(v-u+\mu^{(2)}-\mu^{(1)})^2}{2\sigma_{C'}^2}} e^{-\frac{1}{2}cu^2} du dv\,. \tag{B.14}$$

By identifying the inner integral as a convolution of two Gaussians, we get

$$\begin{aligned}
&\mathbb{E}_{f\sim P(f|\mathbf{Q},\mathbf{q})}\left[H(q \mid \Phi^{(1)},\Phi^{(2)},f)\right] \\
&= \frac{1}{1+\frac{d}{c}}\frac{\sqrt{c'c-d^2}}{2\pi}2\pi{\sigma_C}'\frac{1}{\sqrt{c}}\int_v e^{-\frac{1}{2}\frac{c'c-d^2}{c}\frac{v^2}{(1+\frac{d}{c})^2}}\frac{1}{\sqrt{2\pi}\sqrt{{\sigma_{C'}}^2+\frac{1}{c}}} e^{-\frac{1}{2}\frac{(v-(\mu^{(1)}-\mu^{(2)}))^2}{{\sigma_{C'}}^2+\frac{1}{c}}} dv \\
&= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}{\sigma_C}'\frac{1}{\sqrt{c}}\frac{1}{\sqrt{2\pi}\sqrt{{\sigma_{C'}}^2+\frac{1}{c}}}\int_v e^{-\frac{1}{2}\frac{v^2}{\sigma_{C_b}^2}} e^{-\frac{1}{2}\frac{(v-(\mu^{(1)}-\mu^{(2)}))^2}{{\sigma_{C''}}^2}} dv\,.
\end{aligned} \tag{B.15}$$

By repeating the same convolution trick for the second integral,

$$\begin{aligned}
&\mathbb{E}_{f\sim P(f|\mathbf{Q},\mathbf{q})}\left[H(q \mid \Phi^{(1)}, \Phi^{(2)}, f)\right] \\
&= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}\sigma_{C}{}' \frac{1}{\sqrt{c}} \frac{1}{\sqrt{2\pi}\sqrt{\sigma_{C}{}'^2+\frac{1}{c}}} 2\pi\sigma_{Cb}\sigma_{C}{}'' \frac{1}{\sqrt{2\pi}\sqrt{\sigma_{C_b}^2+\sigma_{C}{}''^2}} e^{-\frac{1}{2}\frac{(\mu^{(1)}-\mu^{(2)})^2}{\sigma_{C_b}^2+\sigma_{C}{}''^2}} \\
&= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}\sigma_{C}{}' \frac{1}{\sqrt{c}} \frac{1}{\sqrt{\sigma_{C}{}'^2+\frac{1}{c}}} \sigma_{Cb}\sigma_{C}{}'' \frac{1}{\sqrt{\sigma_{C_b}^2+\sigma_{C}{}''^2}} e^{-\frac{1}{2}\frac{(\mu^{(1)}-\mu^{(2)})^2}{\sigma_{C_b}^2+\sigma_{C}{}''^2}} \,. \qquad \text{(B.16)}
\end{aligned}$$

Again, we express this in terms of covariance and variance expressions:

$$\mathbb{E}_{f\sim P(f|\mathbf{Q},\mathbf{q})}\left[H(q \mid \Phi^{(1)}, \Phi^{(2)}, f)\right] = \frac{\sqrt{\pi\ln(2)\sigma_{C}{}^2}\exp\left(-\frac{(\mu^{(1)}-\mu^{(2)})^2}{\pi\ln(2)\sigma_{C}{}^2+2g(\Phi^{(1)},\Phi^{(2)})}\right)}{\sqrt{\pi\ln(2)\sigma_{C}{}^2+2g(\Phi^{(1)},\Phi^{(2)})}} \,. \qquad \text{(B.17)}$$

This completes the derivation.

B.3 Mutual Information Derivation for Section 4.6

We present the derivation of the formula for computing the maximum mutual information query Q^*. Assume at a fixed round i we have made past ranking query observations $\mathcal{D}_R = \{Q^{(i')}, q^{(i')}\}_{i'=1}^{i-1}$, and possibly other types of feedback to have the belief distribution b^{i-1}. The desired query is then

$$Q^* = \arg\max_{Q} I(q; w, \alpha \mid Q, b^{i-1}), \qquad \text{(B.18)}$$

where $I(\cdot;\cdot)$ denotes mutual information and q is the response to the query Q. Equivalently, denoting conditional entropy with $H(\cdot \mid \cdot)$, we note

$$I(q; w, \alpha \mid Q, b^{i-1}) = H(w, \alpha \mid b^{i-1}) - \mathbb{E}_{q'\sim q|Q,b^{i-1}}\left[H(w, \alpha \mid Q, q=q', b^{i-1})\right],$$

which allows us to write the optimization in Equation (B.18) equivalently as

$$Q^* = \arg\min_{Q} \mathbb{E}_{q'\sim q|Q,b^{i-1}}\left[H(w, \alpha \mid Q, q=q', b^{i-1})\right].$$

We further simplify this minimization objective by denoting the joint distribution over q and (w, α) conditioned on Q and b^{i-1} as $P(q, w, \alpha \mid Q, b^{i-1})$ and expanding the entropy term:

$$
\begin{aligned}
Q^* &= \arg\min_Q \mathbb{E}_{q',w',\alpha' \sim q,w,\alpha|Q,b^{i-1}} \log \frac{P(q = q' \mid Q, b^{i-1})}{P(q = q' \mid Q, w = w', \alpha = \alpha']} \\
&= \arg\min_Q \mathbb{E}_{q',w',\alpha' \sim q,w,\alpha|Q,b^{i-1}} \log \frac{\mathbb{E}_{w'',\alpha'' \sim w,\alpha|b^{i-1}} P(q = q' \mid Q, w = w'', \alpha = \alpha'']}{P(q = q' \mid Q, w = w', \alpha = \alpha']} \,. \quad \text{(see 4.28)}
\end{aligned}
$$

Appendix C

Implementation Details

C.1 Metropolis-Hastings for Section 4.6

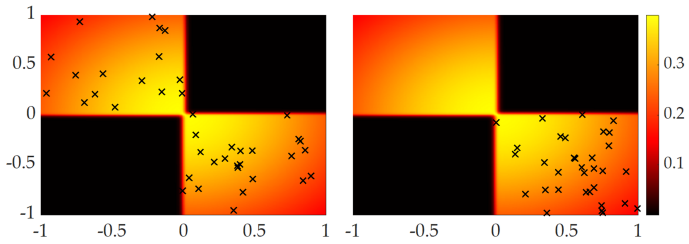


Figure C.1: Multi-chain Metropolis-Hastings sampling **(left)** gives more representative samples from the distribution compared to the single-chain variant **(right)**.

To sample from $P(w, \alpha \mid b^{i-1})$ using Equation (3.38), we use the Metropolis-Hastings algorithm [69], running N_{MH} chains simultaneously for H_{MH} iterations. To avoid autocorrelation between samples, unlike in conventional Metropolis-Hastings we only use the last state in each chain as a sample. In contrast, for conventional Metropolis-Hastings, multiple samples would be drawn from a single chain at set intervals after a short *burn-in* period. As we see in Fig. C.1, for our multimodal Plackett-Luce posteriors, performing multi-chain Metropolis-Hastings yields posterior samples that are far more evenly distributed across different posterior modes. Thus, to achieve well-distributed posterior samples, we set our effective burn-in period to be $H_{\text{MH}} - 1$, taking only the last sample from each chain.

For two states in the chain w, α and w', α', our proposal distribution is then

$$P_{\text{MH}}(w', \alpha' \mid w, \alpha) = \prod_{m=1}^{M} \phi_{\text{MH}}(w_m - w'_m),$$

where ϕ_{MH} is the pdf of the zero-mean Gaussian with the covariance matrix $\sigma^2_{\text{MH}} I$.

The posterior distribution in Figure C.1 is that of a 2-mode Plackett-Luce mixture with fixed uniform mixing coefficients and 1-D weights conditioned on the observations $50 \succ -50$ and $-50 \succ 50$. The single-chain algorithm ran for 2000 steps with a burn-in period of 200 steps after which every 18^{th} sample was selected, while the multi-chain algorithm used 100 chains for 20 iterations each, taking only the last sample from each chain.

C.2 Simulated Annealing for Section 4.6

For our simulated annealing, we run N_{SA} chains in parallel for H_{SA} iterations each, returning the best query Q found across each run. We define the transition proposal distribution $P_{\text{SA}}(Q' \mid Q)$ to be a positive constant if Q' and Q differ by one trajectory and 0 otherwise. We run with a starting temperature of T^0_{SA}, cooling by a factor of γ_{SA} with each subsequent iteration past the first.

C.3 Hyperparameters for Section 4.6

We use the hyperparameters in Table C.1 for the simulated annealing and Metropolis-Hastings algorithms, whose details are provided in Appendix C.1 and Appendix C.2, respectively.

Table C.1: Hyperparameters

Constant	Value
N_{MH}	100
H_{MH}	200
σ_{MH}	0.15
N_{SA}	10
H_{SA}	30
T^0_{SA}	10
γ_{SA}	0.9

C.4 Hyperparameter Tuning for DPPs in Section 4.8.5

We introduced the hyperparameters λ, σ_{DPP} and γ for the DPP-based method. However, using the mode of the DPP distribution as the batch eliminates λ, as it does not affect the results unless trivially $\lambda = 0$. Hence, we need to tune σ_{DPP} and γ only.

As γ is enough in our proposed algorithm to adjust the trade-off between diversity and high volume removal, we use the following heuristic for setting σ_{DPP} to avoid extra computational burden.

We simply set σ_{DPP} 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., $d = \dim(\Phi(\xi))$.

For a human user and a given dynamical system, we cannot try different hyperparameter values, because we need to query the human many times to get the responses under different hyperparameters. Therefore, to perform tuning for and γ, we simulate 100 *synthetic true reward weights* separately for each environment.

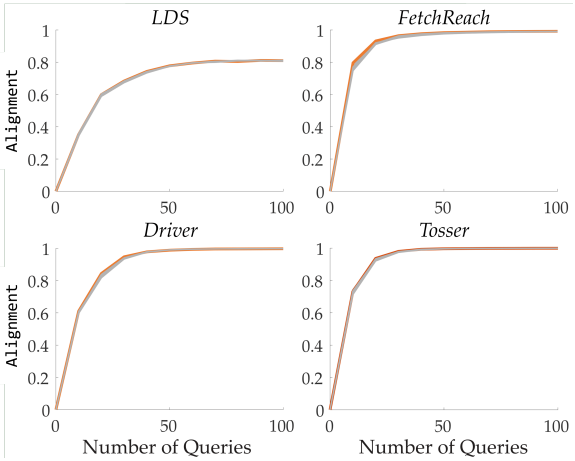


Figure C.2: Tuning results for the DPP-based method for various γ under each environment.

We tuned γ separately for our experiment environments *LDS*, *FetchReach*, *Tosser* and *Driver* where each γ has been experimented with 100 different synthetic true reward functions.

Figure C.2 shows how the `Alignment` value changes with different number of queries for varying γ. We highlighted the selected γ parameters in the plots.

As it can be seen from the results, the effect of γ on performance was slight for the environments we experimented on. It is therefore difficult to select the "best" γ. We qualitatively selected $\gamma = 1$ for *LDS* and *Tosser*, $\gamma = 4$ for *FetchReach*, and $\gamma = 0$ for *Driver* based on their slight advantage in learning rate with respect to the number of queries. While it is possible that other environments have heavier dependencies on γ, our results empirically suggest that the method is robust to the choice of γ, which can ease the use of our DPP-based batch generation algorithm.

Appendix D

Experiment Details

D.1 Environment Features for Sections 4.1.2 and 4.2.4

D.1.1 *FetchReach*

We present the full set of features below. In Section 4.1.2, we used the first three of these features whereas Section 4.2.4 uses all features to make it a more difficult problem.

- The average of $e^{-c_1 d_1}$ over the trajectory where d_1 is the distance between the end effector and the goal object, and $c_1 = 1$.
- The average of $e^{-c_2 d_2}$ over the trajectory where d_2 is the vertical distance between the end effector and the table, and $c_2 = 1$.
- The average of $e^{-c_3 d_3}$ over the trajectory where d_3 is the distance between the end effector and the obstacle, and $c_3 = 1$.
- The average of the end effector speed over the trajectory.

D.1.2 *Driver*

- The average of $e^{-c_4 d_4^2}$ over the trajectory, where d_4 is the shortest distance between the ego car and a lane center, and $c_4 = 30$.
- The average of $(v_1 - 1)^2$ over the trajectory, where v_1 is the speed of the ego car.
- The average of $\cos(\theta_1)$ over the trajectory, where θ_1 is the angle between the directions of ego car and the road.
- The average of $e^{-c_5 d_5^2 - c_6 d_6^2}$ over the trajectory, where d_5 and d_6 are the horizontal and vertical distances between the ego car and the other car, respectively; and $c_5 = 7$, $c_6 = 3$.

Table D.1: Features of the *ExtendedDriver* Environment

	Description	Definition
Φ_1	Lane keeping: mean distance to closest lane center	$\frac{\text{mean}[\exp(-30\cdot\min\{d_8,d_9,d_{10}\})]}{0.15343634}$
Φ_2	Keep speed: mean difference to speed 1	$\frac{\text{mean}[(1-\mathbf{v})^2]}{0.42202643}$
Φ_3	Driving straight: mean heading θ	$\frac{\text{mean}[\theta]}{0.06112367}$
Φ_4	Collision avoidance 1: mean distance to other car	$\frac{\text{mean}[\exp(-7\cdot\Delta\mathbf{x}^2)+3\cdot\Delta\mathbf{y}^2]}{0.15258019}$
Φ_5	Collision avoidance 2: min distance to other car	$\frac{\min[\exp(-7\cdot\Delta\mathbf{x}^2)+3\cdot\Delta\mathbf{y}^2]}{0.10977646}$
Φ_6	Smoothness: mean jerk	$\frac{\text{mean}[\dot{\Delta\mathbf{v}}]}{0.00317041}$
Φ_7	Distance travelled: progress along the road	$\frac{y(T)-y(0)}{1.01818467}$
Φ_8	Final lane L: robot end in the left lane	$\text{int}(\|x(T)-c_8\\| < 0.08)$
Φ_9	Final lane M: robot end in the center lane	$\text{int}(\|x(T)-c_9\\| < 0.08)$
Φ_{10}	Final lane R: robot end in the right lane	$\text{int}(\|x(T)-c_{10}\\|\|\| < 0.08)$

D.1.3 *Tosser*

- The maximum distance the object moved forward from the tosser robot.
- The maximum altitude of the object.
- Number of flips (real number) the object does.
- $e^{-c_7 d_7}$ where d_7 is the final horizontal distance between the object and the center of the closest basket, and $c_7 = 3$.

D.2 Environment Features for Section 4.5.2

Here, we describe the features of the simulation and user study environments we used. These environments are: *ExtendedDriver*, which we used for the simulations in Section 4.5.2, original *Driver*, which was used in Section 4.2.4, and we present the results in Appendix E.2, and finally Fetch robot experiment with drink serving (*FetchDrink*), which we used for the user studies in Section 4.5.2.

D.2.1 *ExtendedDriver*

In Table D.1 we detail the features of the *ExtendedDriver* scenarios. Notation specific to Table D.1: d_8, d_9, d_{10} are the squared distances of the robot car to the center of the left, middle and right lane; $\mathbf{v}$ is the speed profile of the robot trajectory; $\dot{\mathbf{v}}$ the acceleration profile; θ is the heading of the car, $x(t)$ and $y(t)$ are the car's x and y position at a given time $t \in [0, T]$ (x is orthogonal to the road, y is along the road); $\Delta\mathbf{x}$ and $\Delta\mathbf{y}$ are the ordinal distance between the robot car and the other car; and c_8, c_9, c_{10} are the x-coordinates of the lane centers.

D.2.2 Original *Driver*

See Appendix D.1.2.

D.2.3 *FetchDrink*

In the user studies presented in Section 4.5.2 and the simulations presented in Appendix E.2, we used the following eight features for the *FetchDrink* robot experiment:

- Speed of the end-effector $\in \{0, 0.33, 0.67, 1\}$
- Maximum height of the end-effector $\in \{0, 0.33, 0.67, 1\}$
- Selected drink being the orange juice $\in \{0, 1\}$
- Selected drink being the water $\in \{0, 1\}$
- Selected drink being the milk $\in \{0, 1\}$
- Orientation of the pan $\in \{0, 1\}$
- Moving the drink behind or over the pan $\in \{0, 1\}$
- Robot hitting the pan while moving the drink $\in \{0, 1\}$

D.3 Choice of σ_S in the User Studies for Section 4.5.2

In Section 4.5.2, we stated we took $\sigma_S = 0.35$ in the user studies based on pilot trials with different users. We now describe the procedure that yielded this selection of σ_S.

Before all the actual experiments, we recruited 3 participants (3 male, ages 27–40) for a pilot study. In this study, the participants followed the same procedure as in our actual experiments, but responded to only 30 randomly generated queries. These 30 queries were formed by three sets: 10 scale queries, 10 weak comparison queries and another 10 scale queries. We randomized the order of these three sets to avoid any bias.

After we collected these data, we repeated the following procedure for $\sigma_S = 0.05, 0.10, \ldots, 1.00$. We learned a single posterior for each user by using 10 scale and 10 weak comparison query responses under σ_S noise, i.e., the posteriors included both scale and soft choice feedback. We then checked the test set loglikelihood (with the remaining 10 queries) under the learned posterior and the same σ_S.

The σ_S value that yielded the highest test set loglikelihood, $\sigma_S = 0.35$, was then used for all of the actual experiments with real users.

D.4 Baselines for Section 4.6.3

D.4.1 Random

We benchmark against a random agent, wherein at each step the query selected by the agent is a collection of $|Q|$ random items without replacement. We also use the random querying method for comparing the multimodal reward learning with the approaches that assume a unimodal reward (as in Section 4.2), as it does not introduce any bias in the query selection.

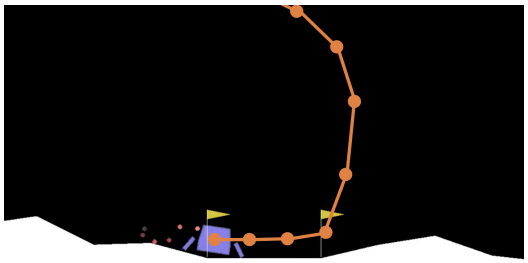
D.4.2 Volume Removal

Volume removal seeks to maximize the difference between the prior distribution over model parameters and the *unnormalized* posterior. Volume removal notably fails to be optimal in domains where there are similar trajectories as we showed in Section 4.1. In these settings, querying sets of trajectories with similar features removes a large amount of volume from the unnormalized posterior (since the robot is highly uncertain about their relative quality), yet yields little information about the model parameters (since the human also has high uncertainty). Mutual information based approaches are better able to generate trajectories to query for which the robot has high uncertainty while the human has enough certainty to yield useful information for the robot.

D.5 Trajectory Generation in Section 4.6.3

D.5.1 *LunarLander* Trajectories

We designed 8 trajectory features based on: absolute heading angle accumulated over trajectory, final distance to the landing pad, total amount of rotation, path length, task completion (or failure) time, final vertical velocity, whether the lander landed on the landing pad without its body touching the ground, and original environment reward from OpenAI Gym [50]. Using these features, we randomly generated 10 distinct reward functions based on the linear reward model and trained a DQN policy [151] for each reward. Finally, we generated 100 trajectories by following each of these 10 policies in the environment to obtain 1000 trajectories in total. We used these trajectories as our dataset for the ranking queries. Figure D.1 presents an example trajectory with extracted, scaled and centered features.



Feature	Value
Mean angle	2.27683634
Total angle	−0.20375356
Distance to goal	5.41860642
Total rotation	0.25948072
Path length	3.71660086
Final vertical velocity	−0.57097337
Crash time	1.11112885
Score	−0.15500268

Figure D.1: Sample *LunarLander* trajectory (left) with extracted features (right).

D.5.2 *FetchBanana* Trajectories

To design our 351 trajectories, we varied the target shelf (3 variations), the movement speed (3), the grasp point on the banana (3) and where in the shelf it is placed (13). We then designed 12 trajectory features based on these varied parameters and appended another binary feature which indicates whether any object dropped from the shelves on that trajectory.

Specifically, for a trajectory ξ, let

$$y_{\text{target},i} = \begin{cases} 1 & i \text{ is the target shelf} \\ 0 & \text{otherwise} \end{cases},$$

$y_{\text{grasp}}, y_{\text{height}}, y_{\text{width}}, y_{\text{speed}}$ specify the grasp position and speed, and y_{success} specifies whether the robot did not drop any objects from the shelves. Our featurization is then

$$\Phi(\xi) = \big(y_{\text{target},1}, y_{\text{target},2}, y_{\text{target},3}, y_{\text{speed}}, y_{\text{speed}}(1-y_{\text{speed}}), y_{\text{grasp}}, y_{\text{grasp}}(1-y_{\text{grasp}}), y_{\text{height}}, y_{\text{height}}(1-y_{\text{height}}), y_{\text{width}}, y_{\text{width}}(1-y_{\text{width}}), 1-(y_{\text{grasp}}-y_{\text{width}})^2, y_{\text{success}}\big).$$

Figure D.2 presents a sample *FetchBanana* trajectory with its featurization.

Feature	Value
$y_{\text{target},1}$	1
$y_{\text{target},2}$	0
$y_{\text{target},3}$	0
y_{speed}	0.5
$y_{\text{speed}}(1-y_{\text{speed}})$	0.25
y_{grasp}	1
$y_{\text{grasp}}(1-y_{\text{grasp}})$	0
y_{height}	0.75
$y_{\text{height}}(1-y_{\text{height}})$	0.1875
y_{width}	0.25
$y_{\text{width}}(1-y_{\text{width}})$	0.1875
$1-(y_{\text{grasp}}-y_{\text{width}})^2$	0.4375
y_{success}	1

Figure D.2: Sample *FetchBanana* trajectory (left) with extracted features (right).

D.6 Metrics in Section 4.6.3

D.6.1 `MSE`

Our metric is

$$\texttt{MSE} = \sum_{m=1}^{M} |w_m^* - \hat{w}_m|_2^2 \tag{D.1}$$

where the learned reward weights of the experts are matched with the true weights using the Hungarian algorithm. When the learning model assumes a unimodal reward function, as in our simulations for Figure 4.27, we compute the `MSE` metric as $\sum_{m=1}^{M} |w_m^* - \hat{w}|_2^2$.

D.6.2 `Log-Likelihood`

Formally, we define the `Log-Likelihood` metric as

$$\texttt{Log-Likelihood} = \mathbb{E}_{Q\sim\mathcal{Q}}\left[\mathbb{E}_{q'\sim q|Q} \log P(q = q' \mid \mathcal{Q}, b^{i-1})\right] \tag{D.2}$$

for $\mathcal{Q}$ the uniform distribution across all possible queries and $P(q \mid Q)$ the distribution over the human's response to query Q (as in Equation (3.37)). We can compute the inner term

$$P(q \mid Q, b^{i-1}) = \mathbb{E}_{w',\alpha'\sim w,\alpha|b^{i-1}}[P(q \mid Q, w = w', \alpha = \alpha')]$$

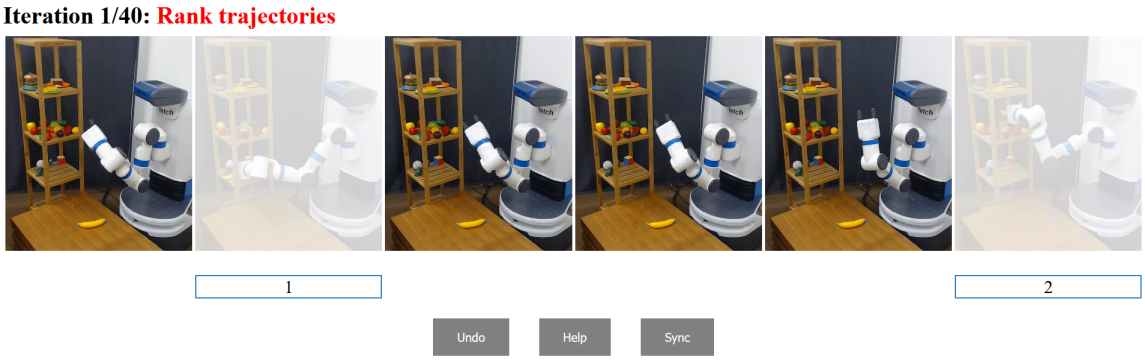


Figure D.3: The user interface for the online studies with the real Fetch robot (*FetchBanana* environment). The user selected the 2^{nd} trajectory as their top choice and the 6^{th} trajectory as the second top.

using Metropolis-Hastings as in Section 4.6.2 to sample from the posterior $P(w, \alpha \mid b^{(i-1)})$ and computing the inner term with Equation (3.37).

D.6.3 `Learned Policy Reward`

Similar to the `MSE` metric, we match the rewards learned via DQN [151] with the true rewards using the Hungarian algorithm.

D.7 Experimental Setup in Section 4.6.3

D.7.1 Shelf Descriptions for *FetchBanana* Environment

A picture of each shelf accompanied the following descriptions.

- The top shelf has some space, but you usually put cooked meals there.
- The middle shelf is for fruits, but it is already full. The robot may accidentally drop other fruits.
- The bottom shelf has a lot of space, but you have been using it for toys.

D.7.2 User Interface

For both environments, subjects were told they need to rank the six trajectories in each query by clicking on the trajectories starting from the most preferred to the least. The web interface (see Figure D.3) equipped them with "Undo" and "Sync" buttons. "Undo" allowed the subjects to undo a selection they make within a query. "Sync" enabled them to restart all videos in the query.

Appendix E

Additional Results

E.1 Additional Simulation Results for Section 4.2.4

E.1.1 Results with User-Specific and Unknown ς

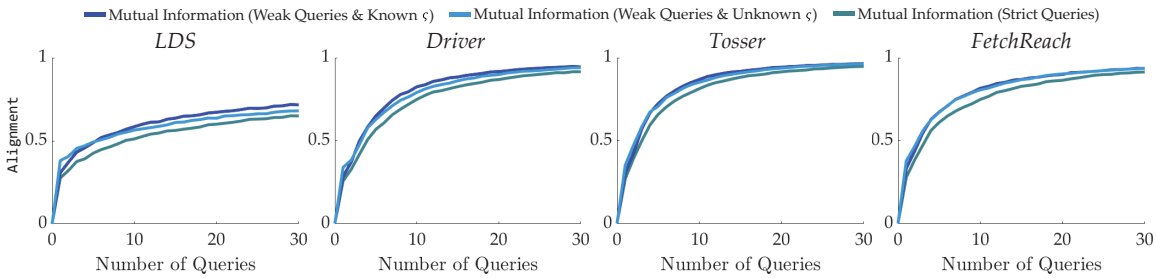


Figure E.1: The simulation results with mutual information formulation for unknown ς. Plots are mean±s.e.

Using the approximate, but computationally faster optimization we introduced in Appendix B.1.1, we performed additional analysis where we compare the performances of strict pairwise comparison queries, weak pairwise comparison queries with known ς and weak pairwise comparison queries without assuming any ς (all with the mutual information formulation). As in the previous simulations, we simulated 100 users with different random reward functions. Each user is simulated to have a true ς, uniformly randomly taken from $[0, 2]$. During the sampling of Ω^+, we did not assume any prior knowledge about ς, except the natural condition that $\varsigma \geq 0$. The comparison results are in Figure E.1. While knowing ς increases the performance as expected, weak pairwise comparison queries are still better than strict queries even when ς is unknown. This supports the advantage of employing weak pairwise comparison queries.

E.1.2 Results without Query Space Discretization

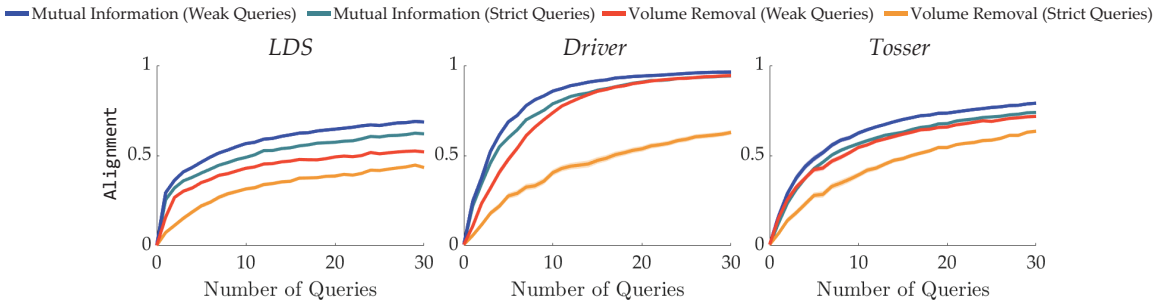


Figure E.2: Alignment values are plotted (mean±s.e.) for the experiments without query space discretization, i.e., with continuous trajectory optimization for active query generation.

We repeated the experiment that supports **H5**, and whose results are shown in Figure 4.6, without query space discretization. By optimizing over the continuous action space of the environments, we tested mutual information and volume removal formulations with both strict and weak pairwise comparison queries in *LDS*, *Driver* and *Tosser* tasks. We excluded *FetchReach* again in order to avoid prohibitive trajectory optimization due to large action space. Figure E.2 shows the results. As it is expected, mutual information formulation outperforms the volume removal with both pairwise comparison query types. And, weak pairwise comparison queries lead to faster learning compared to strict pairwise comparison queries.

E.1.3 Effect of Information from "About Equal" Responses

We have seen that weak pairwise comparison queries consistently decrease wrong answers and improve the performance. However, this improvement is not necessarily merely due to the decrease in wrong answers. It can also be credited to the information we acquire thanks to "About Equal" responses.

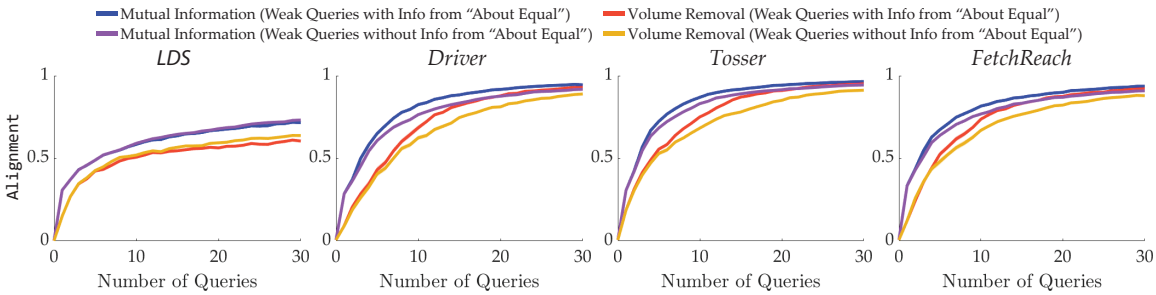


Figure E.3: The results (mean±s.e.) of the simulations with weak pairwise comparison queries where we use the information from "About Equal" responses (blue and red lines) and where we do not use (purple and orange lines).

To investigate the effect of this information, we perform two additional experiments with 100 different simulated human reward functions with weak pairwise comparison queries: First, we use the information by the "About Equal" responses; and second, we ignore such responses and remove the query from the query set to prevent repetition. Figure E.3 shows the results. It can be seen that for both volume removal and mutual information formulations, the information from "About Equal" option improves the learning performance in *Driver*, *Tosser* and *FetchReach* tasks, whereas its effect is very small in *LDS*.

E.1.4 Optimal Stopping under Query-Independent Costs

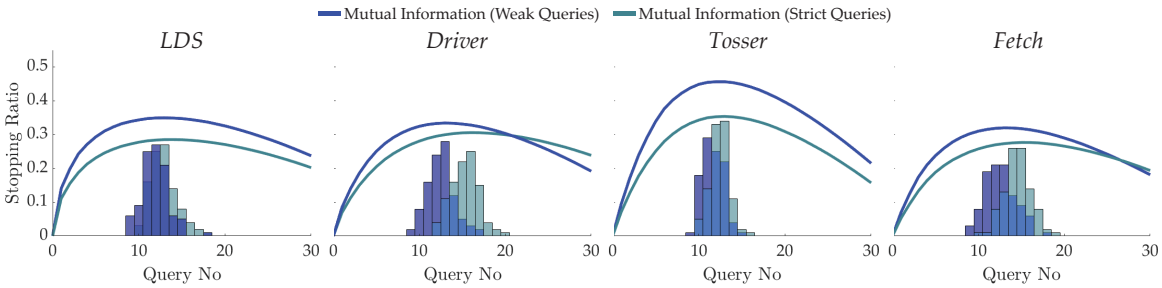


Figure E.4: Simulation results for optimal stopping under query-independent costs. Line plots show cumulative active learning rewards (cumulative difference between the information gain values and the query costs), averaged over 100 test runs and scaled for better appearance. Histograms show when optimal stopping condition is satisfied.

To investigate optimal stopping performance under query-independent costs, we defined the cost function as $c(Q) = \varpi$, which just balances the trade-off between the number of questions and learning performance. Similar to the query-dependent costs case we described in Section 4.2.4, we first simulate 100 random users and tune ϖ accordingly in the same way. We then use this tuned ϖ for our tests with 100 different random users. Figure E.4 shows the results. Optimal stopping rule enables terminating the process with near-optimal cumulative active learning rewards in all environments, which again supports **H9**.

E.2 Additional Simulation Results for Section 4.5.2

We present additional simulation results to compare the proposed scale feedback with weak pairwise comparisons. For the *ExtendedDriver* environment from Section 4.5.2, we additionally show data with higher noise, and show results with the log-likelihood measure used in the user study. Further, we show the same analysis for the original *Driver* experiment, and for the simulated version of the *FetchDrink* experiment from the user study.

For all the simulation results in this Appendix, we simulated 40 different w^*, each with four different $\varrho^* \in \{.25, .5, .75, 1\}$, making 160 runs in total.

E.2.1 *ExtendedDriver*

High Noise. In Section 4.5.2 we showed results for user noise $\sigma_S = 0.1$ in Figure 4.23. In addition, we repeat the same experiment but with $\sigma_S = 0.3$; shown in Figure E.5. Overall, we observe a poorer performance for all approaches compared to $\sigma_S = 0.1$ – higher noise in the user feedback makes learning more difficult. Nevertheless, scale feedback still leads to an improvement on both measures, `Alignment` and `Relative_Reward`.

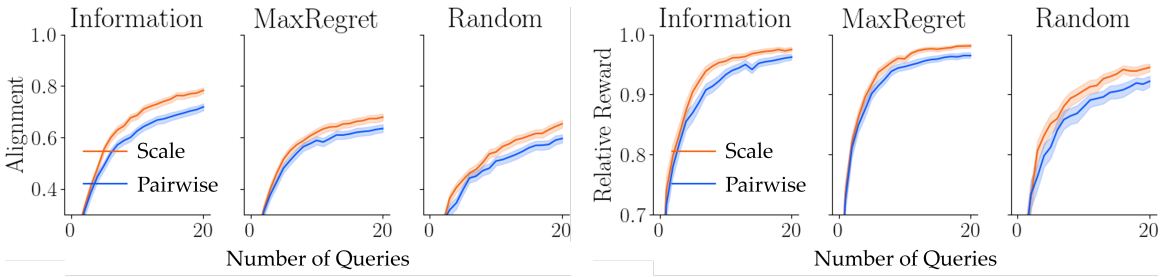
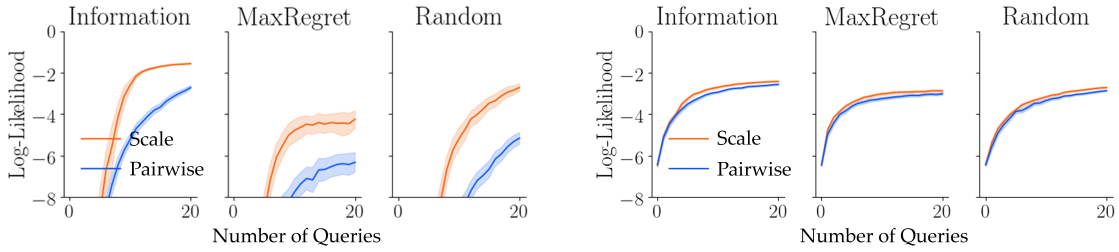


Figure E.5: `Alignment` (left) and `Relative_Reward` (right) for *ExtendedDriver* with $\sigma_S = 0.3$.

Log-Likelihood. Figure E.6 shows the `Log-Likelihood` for the *ExtendedDriver* simulations. When the noise is small, scale feedback significantly outperforms weak pairwise comparisons under all three active querying methods. Further, mutual information based method performs best overall, followed by random. It might be surprising that max regret achieves a lower log-likelihood than random. Max regret greedily tries to find solutions that are close to optimal. Thus, this approach does not gather information about comparably good or bad trajectories (with respect to collected reward). Since the set of test queries is generated randomly, it might contain numerous queries about which the max regret approach is still uncertain since it only focused on finding close to optimal solutions. Mutual information based method, on the other hand, minimizes the uncertainty about weights, regardless of how different the resulting trajectories are. Similarly, random querying is completely unbiased and thus does not focus on a subset of queries as the max regret approach does.

In Figure E.6 (b) we show the log-likelihood for high noise. Here all three active querying methods perform nearly identical, and the difference between scale feedback and weak pairwise comparisons is very small. This is because, when the noise is high, i.e., when the Gaussian over the feedback value has high variance, the log-likelihood measure does not heavily penalize bad predictions, which causes all methods to acquire high log-likelihood values.



(a) $\sigma_S = 0.1$ (b) $\sigma_S = 0.3$

Figure E.6: `Log-Likelihood` for the *ExtendedDriver* simulations.

E.2.2 Original *Driver*

Alignment and Relative Reward. Next, we show results for the original *Driver* experiment. Figure E.7 shows the `Alignment` and `Relative_Reward` for low noise ($\sigma_S = 0.1$), FigureE.8 shows the same measures for high noise ($\sigma_S = 0.3$). While scale feedback still improves `Alignment` and `Relative_Reward` for all querying methods, the gap to weak pairwise comparison feedback is smaller than for the *ExtendedDriver*. However, we observe that all querying methods achieve a substantially stronger performance than in the *ExtendedDriver* model with 10 features, indicating that the original *Driver* environment poses a less difficult learning problem with only 4 features. We notice that the result for weak pairwise comparisons via mutual information optimization achieves a higher `Alignment` after 20 iterations than reported in Section 4.2. There are two reasons for this: First, we use a Gaussian noise instead of the Boltzmann model (also known as MNL or the softmax model). Second, by emulating weak pairwise comparisons using a slider with step size 1, we change the model for when users give a neutral ("About Equal") feedback. Nonetheless, the stronger performance compared to Section 4.2 suggests that these differences do not negatively impact the performance of weak comparison queries with mutual information maximization, and thus that the shown comparisons of scale feedback and weak pairwise comparisons are fair.

Log-Likelihood. We also report the results in the `Log-Likelihood` measure in Figure E.9. The results are very similar to the results of the *ExtendedDriver* environment, except the `Log-Likelihood` values increase faster. This is again because the reward is easier to learn in the original *Driver* environment with the fewer number of features.

E.2.3 *FetchDrink*

We now show simulation results for the experimental setup from the user study, using the Fetch robot: *FetchDrink*. Figure E.10 shows the `Alignment` and `Relative_Reward` for low noise ($\sigma_S = 0.1$), Figure E.11 shows the same measures for high noise ($\sigma_S = 0.3$), and Figure E.12 shows the `Log-Likelihood`. In terms of the comparisons between different feedback types and different active

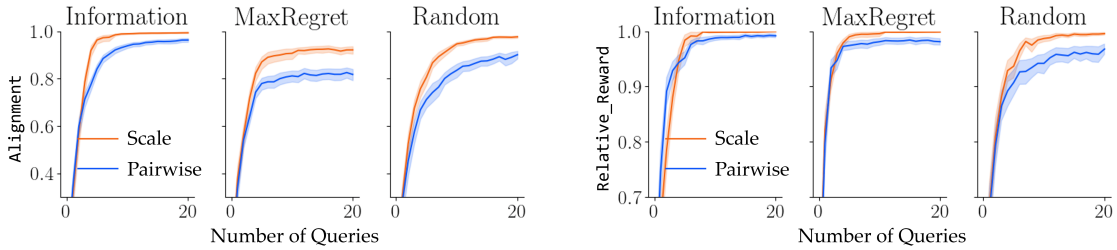


Figure E.7: `Alignment` and `Relative_Reward` for the original *Driver* with $\sigma_S = 0.1$.

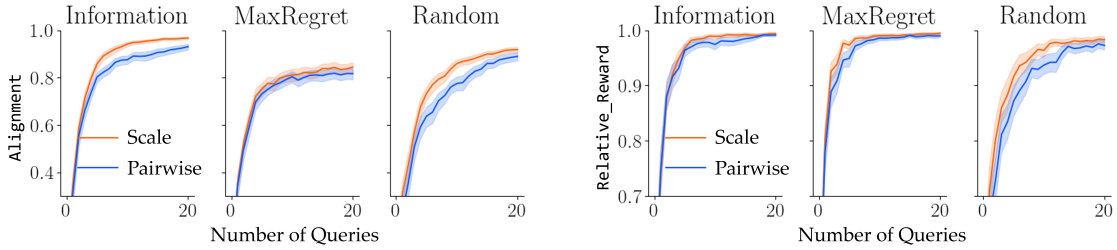
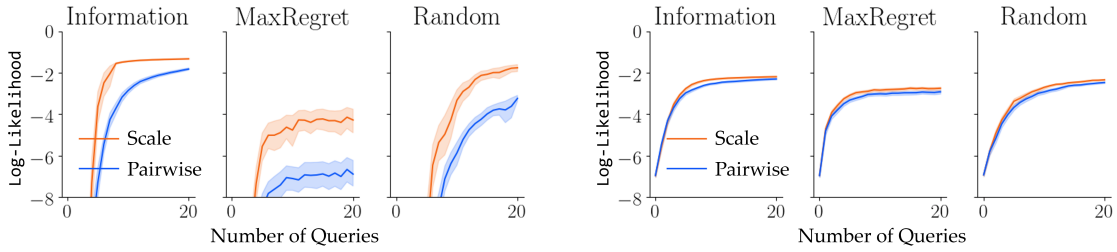


Figure E.8: `Alignment` and `Relative_Reward` for the original *Driver* with $\sigma_S = 0.3$.



(a) $\sigma_S = 0.1$ (b) $\sigma_S = 0.3$

Figure E.9: `Log-Likelihood` for the original *Driver*.

querying methods, the results have the same trend as the *ExtendedDriver* and the original *Driver* environments.

E.3 Results with Test Set with Mixture Data for Section 4.5.2

In both of our user studies, we used a test set that consists of randomly generated scale questions. Given the fact that the subjective user ratings did not point out a significant difference between learning from scale feedback and pairwise comparison queries, one might argue that the superiority of learning from scale feedback in terms of the `Log-Likelihood` metric is simply because the test set also consists of scale feedback. Mathematically, this should not happen, because a good posterior

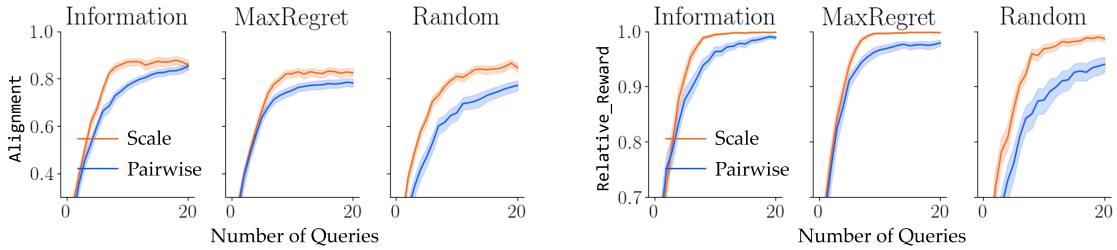


Figure E.10: Fetch robot with drink serving experiment (*FetchDrink*) with $\sigma_S = 0.1$.

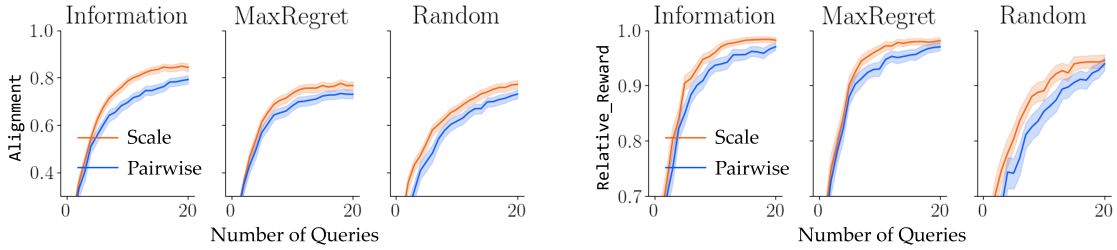
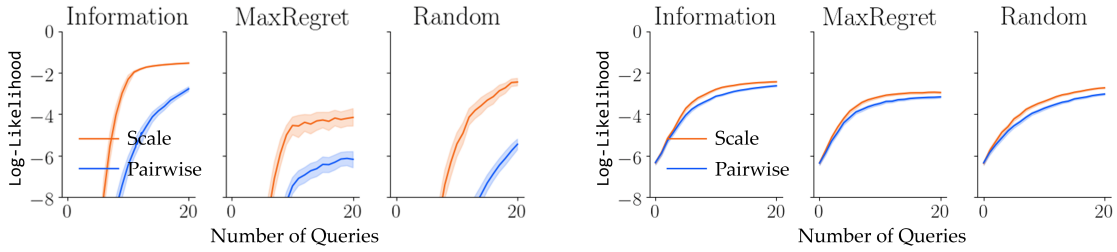


Figure E.11: Fetch robot with drink serving experiment (*FetchDrink*) with $\sigma_S = 0.3$.



(a) $\sigma_S = 0.1$ (b) $\sigma_S = 0.3$

Figure E.12: Log-Likelihood for the Fetch robot with drink serving experiment (*FetchDrink*).

should be able to correctly predict any form of user feedback. However, humans have cognitive biases, which makes it possible that the posterior learned with the scale questions captures the bias caused by the scale questions, whereas the posterior learned with the weak pairwise comparisons cannot do this.

To show this is not the case, we present an additional analysis on the same human data as in our first user study. For this analysis, we take the reward posteriors that have been learned with the first 7 queries (of "Scale - Mutual Information", "Scale - Random", and "Pairwise - Random"). Next, we alter the test set as follows. We take (i) the first 3 scale queries from the original test set, and (ii) the last 3 weak pairwise comparison queries from the original training set of randomly generated weak pairwise comparison queries (and this is why we only take the first 7 posteriors – we

do not mix the training and test data). Finally, we perform the `Log-Likelihood` analysis on this modified test set.

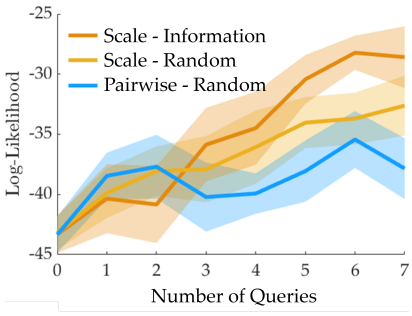


Figure E.13: Additional analysis results are shown (mean±s.e. over 18 subjects).

Results are shown in Figure E.13. It can be seen that even with a test set that consists of mixture data, the results have the same trend as in the original study results. While having smaller test set (6 instead of the 10 in the original study) causes larger standard errors, "Scale - Information" and "Scale - Random" both outperform "Pairwise - Random" with statistical significance ($p < 0.05$ in both comparisons). On the other hand, the comparison between "Scale - Information" and "Scale - Random" gives $p = 0.098$.

This analysis shows the fact that scale feedback outperforms weak pairwise comparisons in terms of `Log-Likelihood` is not because of the data in the test set. Even with a test set that consists of both scale and weak pairwise comparison data, we see the benefits of learning from scale queries.

However, this analysis does not answer the question why user ratings did not have a significant difference between the two feedback types. While the answer to this question requires more analysis and possibly more data collection, we speculate the following reason: the mean user ratings are always around 4, and even higher than 4 when queries are actively generated with mutual information. This means the users are happy with the optimized trajectories, so we can say that 10 queries are enough in this task to find the optimal trajectory. However, while user ratings measure how close the optimal trajectory with respect to the robot's posterior is to the optimal trajectory the user has in mind; `Log-Likelihood` measures the predictive performance of the posterior. Therefore, having a high user rating does not necessarily mean the robot can accurately compare two suboptimal trajectories. On the other hand, a high `Log-Likelihood` value indicates good predictive performance, which is crucial in many robotics applications, such as behavior modeling. Hence, we claim: (i) learning from scale feedback improves the predictive performance over learning from weak pairwise comparisons, and (ii) a more complex task might be needed to show scale feedback leads to more efficient learning than weak pairwise comparisons, which is also suggested by our simulation studies.

E.4 Numerical Results for Section 4.5.2

Here, we present Table E.1 where we report the numerical results of the simulations in Section 4.5.2 at iterations $0, 5, 10, 20$; and Table E.2 where we report the final numerical results of the user studies. Consistent with the section, the numbers are presented as mean $\pm$ standard deviation (simulations) and standard error (user study).

Table E.1: Numerical results of the simulations at selected iterations i

	Mean±Standard Deviation			
Plot	$i = 0$	$i = 5$	$i = 10$	$i = 20$
Fig. 4.23 Scale - Information (Alignment)	$-.01 \pm .33$	$\mathbf{.62 \pm .19}$	$\mathbf{.81 \pm .16}$	$\mathbf{.9 \pm .08}$
Fig. 4.23 Pairwise - Information (Alignment)	$-.02 \pm .31$	$.52 \pm .18$	$.67 \pm .16$	$.79 \pm .15$
Fig. 4.23 Scale - MaxRegret (Alignment)	$.01 \pm .31$	$.57 \pm .19$	$.71 \pm .16$	$.75 \pm .16$
Fig. 4.23 Pairwise - MaxRegret (Alignment)	$-.03 \pm .3$	$.47 \pm .23$	$.59 \pm .17$	$.67 \pm .18$
Fig. 4.23 Scale - Random (Alignment)	$.01 \pm .33$	$.52 \pm .2$	$.67 \pm .17$	$.77 \pm .17$
Fig. 4.23 Pairwise - Random (Alignment)	$.02 \pm .32$	$.4 \pm .21$	$.52 \pm .2$	$.63 \pm .21$
Fig. 4.23 Scale - Information (Rel. Reward)	$.51 \pm .32$	$.92 \pm .12$	$.98 \pm .04$	$\mathbf{1.0 \pm .01}$
Fig. 4.23 Pairwise - Information (Rel. Reward)	$.5 \pm .3$	$.89 \pm .12$	$.95 \pm .07$	$.98 \pm .04$
Fig. 4.23 Scale - MaxRegret (Rel. Reward)	$.52 \pm .31$	$\mathbf{.96 \pm .07}$	$\mathbf{.99 \pm .02}$	$\mathbf{1.0 \pm .01}$
Fig. 4.23 Pairwise - MaxRegret (Rel. Reward)	$.51 \pm .3$	$.91 \pm .12$	$.95 \pm .06$	$.96 \pm .06$
Fig. 4.23 Scale - Random (Rel. Reward)	$.52 \pm .32$	$.89 \pm .14$	$.96 \pm .07$	$.99 \pm .03$
Fig. 4.23 Pairwise - Random (Rel. Reward)	$.52 \pm .32$	$.85 \pm .15$	$.89 \pm .12$	$.93 \pm .12$

E.5 Synthetic Experiment for Section 4.6.3

E.5.1 Testing $M > 2$

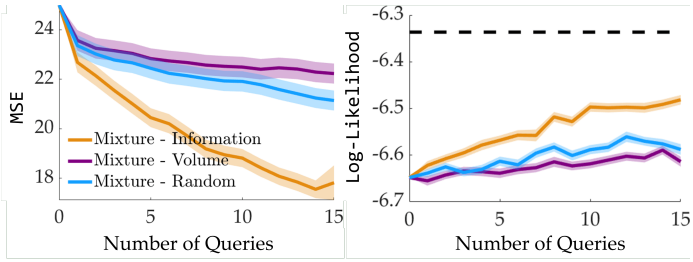


Figure E.14: Different querying methods are compared on a synthetic environment (mean±se over 250 runs).

For our first experiment with synthetic data, we demonstrate effectiveness of our approach for learning mixtures of more than two Plackett-Luce models. In particular, we evaluate our approaches using 250 sets of five randomly simulated reward weights ($M = 5$, $|Q| = 6$), and trajectory features

Table E.2: Final numerical results of the user study

Plot	Mean±Standard Error
Fig. 4.24(a) Scale - Information	-29.7 ± 1.2
Fig. 4.24(a) Scale - Random	-36.2 ± 2.2
Fig. 4.24(a) Pairwise - Random	-51.2 ± 3.5
Fig. 4.24(b) Scale - Information	4.2 ± 0.2
Fig. 4.24(b) Scale - Random	3.6 ± 0.3
Fig. 4.24(b) Pairwise - Random	3.9 ± 0.2
Fig. 4.24(c) Scale (Easiness)	3.8 ± 0.2
Fig. 4.24(c) Pairwise (Easiness)	4.5 ± 0.2
Fig. 4.24(c) Scale (Expressiveness)	3.8 ± 0.3
Fig. 4.24(c) Pairwise (Expressiveness)	4.1 ± 0.2
Fig. 4.25(a) Scale - Information	-28.8 ± 1.3
Fig. 4.25(a) Pairwise - Information	-46.0 ± 3.1
Fig. 4.25(b) Scale - Information	4.5 ± 0.2
Fig. 4.25(b) Pairwise - Information	4.2 ± 0.3
Fig. 4.25(c) Scale (Easiness)	3.6 ± 0.3
Fig. 4.25(c) Pairwise (Easiness)	4.6 ± 0.2
Fig. 4.25(c) Scale (Expressiveness)	4.3 ± 0.2
Fig. 4.25(c) Pairwise (Expressiveness)	4.3 ± 0.2

defined by $\Phi(\xi_{1:10}) \sim \mathcal{N}(0, I)$, $\Phi(\xi_{11:110}) \sim \mathcal{N}(0, 0.1I)$, and $\Phi(\xi_{111:1110}) \sim \mathcal{N}(0, 0.01I)$ where I is the 3×3 identity matrix and $\xi_{i:i'}$ refers to the i^{th} through i'^{th} trajectory in the trajectory dataset for generating queries. This environment models complex multimodal structure in the trajectory feature space, which is common to many robotic settings.

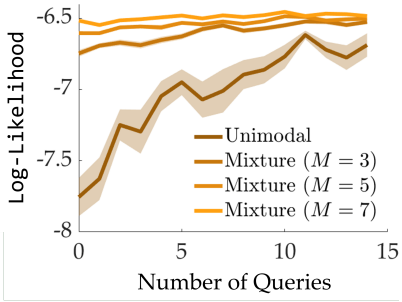


Figure E.15: Different values of M for the mutual information maximization approach are compared (mean±se over 100 runs).

Figure E.14 shows the results of our experiments. We see our approach, Mixture - MI dramatically outperforms the other approaches in both the `MSE` and `Log-Likelihood` metrics.

E.5.2 Testing Robustness to M Parameter

We also test the robustness of our Mixture - MI approach to misspecified M values. We repeat the previous experiment, testing the Mixture - MI approach varying the misspecified value of M between $M = 1$ (Unimodal) and $M = 3, 5, 7$ (see Figure E.15). We use the `Log-Likelihood` metric since `MSE` is not well-defined for methods with $M \neq 5$ because they learn a mixture of a different number of reward functions from the true synthetic mixture.

We see the best performance occurs for $M = 5$ and $M = 7$, with only $M = 1$ performing significantly worse. We conclude that in this experiment our Mixture - MI approach is relatively robust to the value of M, as long as a sufficiently large value > 1 is selected.

E.6 Additional Unimodal Baseline for Section 4.6.3

We test an additional baseline on the random queries made during user studies to show the superiority of our learning approach. The additional baseline represents selecting the unimodal reward with fixed norm that maximizes the reward of the top trajectory of each expert-ranked query. We compare this baseline against a learning method that computes the bimodal MLE of the reward function. Formally, for query responses $\mathcal{D}_R = \left\{Q^{(i')}, q^{(i')}\right\}_{i'=1}^{i}$ with $\xi^{(i')}$ the top trajectory in the ranking $q^{(i')} = (\xi^{(i')}, \ldots)$, we define this baseline to learn the parameters (w, α) where $\alpha = 1$ and

$$\tilde{w} = \sum_{i'=1}^{i} \Phi(\xi^{(i')})$$
$$w = \frac{\tilde{w}}{\|\tilde{w}\|_2}.$$

Note that we do not vary the querying method in this experiment. Rather, we compare two methods of learning reward weights from the 15 random human queries that were performed by the Mixture - Random algorithm on the *FetchBanana* and *LunarLander* during our user studies, and then evaluate these methods on the 10 random evaluation queries presented to the humans at the end of the experiment. We compare the two methods in terms of the `Log-Likelihood` metric. The results are presented in Table E.3, with our method denoted as "Mixture MLE" and the new baseline described above denoted as "Baseline".

Table E.3: Additional User Study Reward Learning Baseline

	LunarLander		*Fetch Robot*	
	Baseline	Mixture MLE	Baseline	Mixture MLE
`Log-Likelihood`	-8.23 ± 0.31	-5.91 ± 0.18	-5.21 ± 0.22	-4.70 ± 0.35
p-value	$6.2 \cdot 10^{-7}$		0.11	

We see the Mixture MLE method outperforms the Baseline method on both environments, with statistical significance ($p < 0.05$) in *LunarLander* when conducting paired t-tests.

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Pieter Abbeel and Andrew Y Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 1–8. ACM, 2005.

[3] Ayush Agrawal, Omar Harib, Ayonga Hereid, Sylvain Finet, Matthieu Masselin, Laurent Praly, Aaron D Ames, Koushil Sreenath, and Jessy W Grizzle. First steps towards translating HZD control of bipedal robots to decentralized control of exoskeletons. *IEEE Access*, 5:9919–9934, 2017.

[4] Nir 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.

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

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

[7] Riad Akrour, Marc Schoenauer, and Michèle 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] Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018.

[9] Aaron D Ames. Human-inspired control of bipedal walking robots. *IEEE Transactions on Automatic Control*, 59(5):1115–1130, 2014.

[10] Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In *Conference on Learning Theory*, pages 103–115, 2016.

[11] Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. 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*, pages 1–12, 2019.

[12] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[13] Cédric Archambeau and François Caron. Plackett-luce regression: a new bayesian model for polychotomous data. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 84–92, 2012.

[14] Brenna Argall, Brett Browning, and Manuela Veloso. Learning by demonstration with critique from a human teacher. In *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 57–64. IEEE, 2007.

[15] Brian S Armour, Elizabeth A Courtney-Long, Michael H Fox, Heidi Fredine, and Anthony Cahill. Prevalence and causes of paralysis—united states, 2013. *American journal of public health*, 106(10):1855–1857, 2016.

[16] Monica Babes, Vukosi N Marivate, Kaushik Subramanian, and Michael L Littman. Apprenticeship learning about multiple intentions. In *ICML*, 2011.

[17] Vivek Bagaria, Govinda Kamath, Vasilis Ntranos, Martin Zhang, and David Tse. Medoids in almost-linear time via multi-armed bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 500–509. PMLR, 2018.

[18] Haoyu Bai, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee. Intention-aware online pomdp planning for autonomous driving in a crowd. In *International Conference on Robotics and Automation (ICRA)*, pages 454–460. IEEE, 2015.

[19] Andrea Bajcsy, Dylan P Losey, Marcia K O'Malley, and Anca D Dragan. Learning robot objectives from physical human interaction. *Proceedings of Machine Learning Research*, 78: 217–226, 2017.

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

[21] Chandrayee Basu, Qian Yang, David Hungerman, Mukesh Sinahal, and Anca D Draqan. Do you want your autonomous car to drive like you? In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 417–425. IEEE, 2017.

[22] Chandrayee Basu, Mukesh Singhal, and Anca D Dragan. Learning from richer human guidance: Augmenting comparison-based learning with feature queries. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 132–140. ACM, 2018.

[23] Chandrayee Basu, Erdem Bıyık, Zhixun He, Mukesh Singhal, and Dorsa Sadigh. Active learning of reward dynamics from hierarchical queries. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2019. doi: 10.1109/IROS40897.2019.8968522.

[24] Christian Bauckhage. Numpy/scipy recipes for data science: k-medoids clustering. *researchgate.net, Feb*, 2015.

[25] Mark Beliaev, Woodrow Z. Wang, Daniel A. Lazar, Erdem Bıyık, Dorsa Sadigh, and Ramtin Pedarsani. Emergent correlated equilibrium through synchronized exploration. In *RSS 2020 Workshop on Emergent Behaviors in Human-Robot Systems*, July 2020.

[26] Mark Beliaev, Erdem Bıyık, Daniel A. Lazar, Woodrow Z. Wang, Dorsa Sadigh, and Ramtin Pedarsani. Incentivizing routing choices for safe and efficient transportation in the face of the covid-19 pandemic. In *12th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, May 2021. doi: 10.1145/3450267.3450546.

[27] Moshe Ben-Akiva and Steven R Lerman. *Discrete choice analysis: theory and application to travel demand*. Transportation Studies, 2018.

[28] Moshe E Ben-Akiva, Steven R Lerman, and Steven R Lerman. *Discrete choice analysis: theory and application to travel demand*, volume 9. MIT press, 1985.

[29] Viktor Bengs, Róbert Busa-Fekete, Adil El Mesaoudi-Paul, and Eyke Hüllermeier. Preference-based online learning with dueling bandits: A survey. *Journal of Machine Learning Research*, 22(7):1–108, 2021.

[30] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with Gaussian processes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[31] Holger Berndt, Jorg Emmert, and Klaus Dietmayer. Continuous driver intention recognition with hidden markov models. In *11th International IEEE Conference on Intelligent Transportation Systems*, pages 1189–1194. IEEE, 2008.

[32] Dimitris Bertsimas, John Tsitsiklis, et al. Simulated annealing. *Statistical science*, 8(1):10–15, 1993.

[33] Erdem Bıyık. Learning from humans for adaptive interaction. In *The 17th Annual Human-Robot Interaction Pioneers Workshop (HRI Pioneers)*, March 2022.

[34] Erdem Bıyık and Dorsa Sadigh. Batch active preference-based learning of reward functions. In *2nd Conference on Robot Learning (CoRL)*, volume 87 of *Proceedings of Machine Learning Research*, pages 519–528. PMLR, October 2018.

[35] Erdem Bıyık, Daniel A Lazar, Ramtin Pedarsani, and Dorsa Sadigh. Altruistic autonomy: Beating congestion on shared roads. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, December 2018. doi: 10.1007/978-3-030-44051-0_51.

[36] Erdem Bıyık, Daniel A. Lazar, Dorsa Sadigh, and Ramtin Pedarsani. The green choice: Learning and influencing human decisions on shared roads. In *Proceedings of the 58th IEEE Conference on Decision and Control (CDC)*, December 2019. doi: 10.1109/CDC40024.2019.9030169.

[37] Erdem Bıyık, Jonathan Margoliash, S Ryan Alimo, and Dorsa Sadigh. Efficient and safe exploration in deterministic markov decision processes with unknown transition models. In *Proceedings of American Control Conference (ACC)*, July 2019. doi: 10.23919/ACC.2019.8815276.

[38] Erdem Bıyık, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. In *3rd Conference on Robot Learning (CoRL)*, October 2019.

[39] Erdem Bıyık, Kenneth Wang, Nima Anari, and Dorsa Sadigh. Batch active learning using determinantal point processes. *arXiv preprint arXiv:1906.07975*, June 2019.

[40] Erdem Bıyık, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. Active preference-based gaussian process regression for reward learning. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2020. doi: 10.15607/rss.2020.xvi.041.

[41] Erdem Bıyık, Daniel A. Lazar, Ramtin Pedarsani, and Dorsa Sadigh. Incentivizing efficient equilibria in traffic networks with mixed autonomy. *IEEE Transactions on Control of Network Systems*, 8(4):1717–1729, 2021. doi: 10.1109/TCNS.2021.3084045.

[42] Erdem Bıyık, Anusha Lalitha, Rajarshi Saha, Andrea Goldsmith, and Dorsa Sadigh. Partner-aware algorithms in decentralized cooperative bandit teams. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, February 2022.

[43] Erdem Bıyık, Dylan P. Losey, Malayandi Palan, Nick Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research (IJRR)*, 41(1):45–67, January 2022. doi: 10.1177/02783649211041652.

[44] Erdem Bıyık, Aditi Talati, and Dorsa Sadigh. Aprel: A library for active preference-based reward learning algorithms. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2022.

[45] Andreea Bobu, Andrea Bajcsy, Jaime F Fisac, and Anca D Dragan. Learning under misspecified objective spaces. In *Conference on Robot Learning*, pages 796–805, 2018.

[46] Alexei Borodin and Eric M Rains. Eynard–mehta theorem, schur process, and their pfaffian analogs. *Journal of statistical physics*, 121(3-4):291–317, 2005.

[47] Allan Borodin, Hyun Chul Lee, and Yuli 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.

[48] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH Sketches*, 10:1278780–1278807, 2007.

[49] Erik Brockbank, Haloliang Wang, Justin Yang, Suvir Mirchandani, Erdem Bıyık, Dorsa Sadigh, and Judith E. Fan. How do people incorporate advice from artificial agents when making physical judgments? In *44th Annual Meeting of the Cognitive Science Society (CogSci)*, July 2022.

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

[51] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, pages 783–792, 2019.

[52] Daniel S Brown and Scott Niekum. Deep bayesian reward learning from preferences. In *Workshop on Safety and Robustness in Decision Making at the 33rd Conference on Neural Information Processing Systems (NeurIPS) 2019*, 2019.

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

[54] Róbert Busa-Fekete and Eyke Hüllermeier. A survey of preference-based online learning with bandit algorithms. In *International Conference on Algorithmic Learning Theory*, pages 18–39. Springer, 2014.

[55] Róbert Busa-Fekete, Eyke Hüllermeier, and Balázs Szörényi. Preference-based rank elicitation using statistical models: The case of mallows. In *International Conference on Machine Learning*, pages 1071–1079. PMLR, 2014.

[56] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[57] Maya Cakmak, Siddhartha S Srinivasa, Min Kyung Lee, Jodi Forlizzi, and Sara Kiesler. Human preferences for robot-human hand-over configurations. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1986–1993. IEEE, 2011.

[58] Zhangjie Cao and Dorsa Sadigh. Learning from imperfect demonstrations from agents with varying dynamics. *IEEE Robotics and Automation Letters (RA-L)*, July 2021. doi: 10.1109/LRA.2021.3068912.

[59] Zhangjie Cao, Erdem Bıyık, Woodrow Z. Wang, Allan Raventos, Adrien Gaidon, Guy Rosman, and Dorsa Sadigh. Reinforcement learning based control of imitative policies for near-accident driving. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2020. doi: 10.15607/rss.2020.xvi.039.

[60] Zhangjie Cao, Erdem Bıyık, Guy Rosman, and Dorsa Sadigh. Leveraging smooth attention prior for multi-agent trajectory prediction. In *International Conference on Robotics and Automation (ICRA)*, May 2022.

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

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

[63] Lawrence Chan, Andrew Critch, and Anca Dragan. Human irrationality: both bad and good for reward inference. *arXiv preprint arXiv:2111.06956*, 2021.

[64] Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, 2020.

[65] Lin Chen, Hamed Hassani, and Amin Karbasi. Near-optimal active learning of halfspaces via query synthesis in the noisy setting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[66] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202, 2013.

[67] Xi Chen, Yuanzhi Li, and Jieming Mao. A nearly instance optimal algorithm for top-k ranking under the multinomial logit model. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2504–2522. SIAM, 2018.

[68] Yuxin Chen and Andreas Krause. Near-optimal batch mode active learning and adaptive submodular optimization. *ICML (1)*, 28:160–168, 2013.

[69] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[70] Flavio Chierichetti, Ravi Kumar, and Andrew Tomkins. Learning a mixture of two multinomial logits. In *International Conference on Machine Learning*, pages 961–969. PMLR, 2018.

[71] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 228–245. Springer, 2018.

[72] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[73] Wei Chu and Zoubin Ghahramani. Preference learning with gaussian processes. In *International Conference on Machine Learning (ICML)*, pages 137–144, 2005.

[74] Wei Chu, Zoubin Ghahramani, and Christopher KI Williams. Gaussian processes for ordinal regression. *Journal of machine learning research*, 6(7):1019–1041, 2005.

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

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

[77] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[78] Yuchen Cui and Scott Niekum. Active reward learning from critiques. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6907–6914. IEEE, 2018.

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

[80] Christian Daniel, Oliver Kroemer, Malte Viering, Jan Metz, and Jan Peters. Active reward learning with a novel acquisition function. *Autonomous Robots*, 39(3):389–405, 2015.

[81] Nathaniel D Daw, John P O'doherty, Peter Dayan, Ben Seymour, and Raymond J Dolan. Cortical substrates for exploratory decisions in humans. *Nature*, 441(7095):876, 2006.

[82] Pierpaolo De Blasi, Lancelot F James, John W Lau, et al. Bayesian nonparametric estimation and consistency of mixed multinomial logit choice models. *Bernoulli*, 16(3):679–704, 2010.

[83] Weishan Dong, Jian Li, Renjie Yao, Changsheng Li, Ting Yuan, and Lanjun Wang. Characterizing driving styles with deep learning. *arXiv preprint arXiv:1607.03611*, 2016.

[84] Anca D Dragan and Siddhartha S Srinivasa. *Formalizing assistive teleoperation*. MIT Press, July, 2012.

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

[86] Cong Fei, Bin Wang, Yuzheng Zhuang, Zongzhang Zhang, Jianye Hao, Hongbo Zhang, Xuewu Ji, and Wulong Liu. Triple-gail: A multi-modal imitation learning framework with generative adversarial nets. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2929–2935, July 2020.

[87] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[88] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pages 357–368. PMLR, 2017.

[89] Jaime F. Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S. Shankar Sastry, and Anca D. Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *International Conference on Robotics and Automation (ICRA)*, May 2019.

[90] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[91] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning and ranking by pairwise comparison. In *Preference learning*, pages 65–82. Springer, 2010.

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

[93] Gerd Gigerenzer and Reinhard Selten. *Bounded rationality: The adaptive toolbox*. MIT press, 2002.

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

[95] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42: 427–486, 2011.

[96] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems*, pages 766–774, 2010.

[97] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D Lawrence. Preferential bayesian optimization. In *International Conference on Machine Learning (ICML)*, pages 1282–1291, 2017.

[98] Nakul Gopalan, Nina Moorman, Manisha Natarajan, and Matthew Gombolay. Negative result for learning from demonstration: Challenges for end-users teaching robots with task and motion planning abstractions. In *Proceedings of Robotics: Science and Systems (RSS)*, June 2022.

[99] Daniel H Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[100] Shengbo Guo and Scott Sanner. Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 289–296, 2010.

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

[102] Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Alexis Duburcq, Ayonga Hereid, Omar Harib, Matthieu Masselin, Jessy Grizzle, and Aaron D Ames. Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons. In *International Conference on Robotics and Automation*, pages 2804–2811. IEEE, 2018.

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

[104] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[105] Joseph Y Halpern, Rafael Pass, and Lior Seeman. Decision theory with resource-bounded agents. *Topics in cognitive science*, 6(2):245–257, 2014.

[106] Omar Harib, Ayonga Hereid, Ayush Agrawal, Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Alexis Duburcq, M Eva Mungai, Mattieu Masselin, Aaron D Ames, et al. Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking. *IEEE Control Systems Magazine*, 38(6):61–87, 2018.

[107] Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *31st Annual Conference on Neural Information Processing Systems (NIPS 2017)*, pages 1236–1246, 2018.

[108] Ayonga Hereid, Christian M Hubicki, Eric A Cousineau, and Aaron D Ames. Dynamic humanoid locomotion: A scalable formulation for HZD gait optimization. *IEEE Transactions on Robotics*, 34(2):370–387, 2018.

[109] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4565–4573, 2016.

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

[111] EA Holmes, MB Bonsall, SA Hales, H Mitchell, F Renner, SE Blackwell, P Watson, GM Goodwin, and M Di Simplicio. Applications of time-series analysis to mood fluctuations in bipolar disorder to promote treatment innovation: a case series. *Translational Psychiatry*, 6(1):e720, 2016.

[112] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[113] Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Jose M Hernández-Lobato. Collaborative gaussian processes for preference learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2096–2104, 2012.

[114] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. In *Advances in neural information processing systems*, pages 8011–8023, 2018.

[115] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization. *Robotics science and systems: online proceedings*, 2015, 2015.

[116] Bjørn Sand Jensen and Jens Brehm Nielsen. Pairwise judgements and absolute ratings with gaussian process priors. *DTU, IMM, Tech. Rep., Nov*, 2011.

[117] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. In *Handbook of the fundamentals of financial decision making: Part I*, pages 99–127. World Scientific, 2013.

[118] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pages 295–304, 2015.

[119] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Active learning with gaussian processes for object categorization. In *International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007.

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

[121] Sydney M Katz, Anne-Claire Le Bihan, and Mykel J Kochenderfer. Learning an urban air mobility encounter model from expert preferences. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, September 2019.

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

[123] Rebecca P Khurshid and Katherine J Kuchenbecker. Data-driven motion mappings improve transparency in teleoperation. *Presence: Teleoperators and Virtual Environments*, 24(2):132–154, 2015.

[124] Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

[125] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3: 71–104, 2014.

[126] KS Krishnan. Incorporating thresholds of indifference in probabilistic choice models. *Management science*, 23(11):1224–1233, 1977.

[127] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

[128] Alex Kulesza and Ben Taskar. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA, 2012. ISBN 1601986289.

[129] Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3075–3084. ACM, 2014.

[130] Dana Kulic and Elizabeth A Croft. Affective state estimation for human–robot interaction. *IEEE Transactions on Robotics*, 23(5):991–1000, 2007.

[131] Minae Kwon, Erdem Bıyık, Aditi Talati, Karan Bhasin, Dylan P. Losey, and Dorsa Sadigh. When humans aren't optimal: Robots that collaborate with risk-aware humans. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2020. doi: 10.1145/3319502.3374832.

[132] Daniel A. Lazar, Erdem Bıyık, Dorsa Sadigh, and Ramtin Pedarsani. Learning how to dynamically route autonomous vehicles on shared roads. *Transportation Research Part C: Emerging Technologies*, 130:103258, September 2021. ISSN 0968-090X. doi: 10.1016/j.trc.2021.103258.

[133] John R Lepird, Michael P Owen, and Mykel J Kochenderfer. Bayesian preference elicitation for multiobjective engineering design optimization. *Journal of Aerospace Information Systems*, 12(10):634–645, 2015.

[134] Chengtao Li, Suvrit Sra, and Stefanie Jegelka. Fast mixing markov chains for strongly rayleigh measures, dpps, and constrained sampling. In *Advances in Neural Information Processing Systems*, pages 4188–4196, 2016.

[135] Kejun Li, Maegan Tucker, Erdem Bıyık, Ellen Novoseller, Joel W. Burdick, Yanan Sui, Dorsa Sadigh, Yisong Yue, and Aaron D. Ames. Roial: Region of interest active learning for characterizing exoskeleton gait preference landscapes. In *International Conference on Robotics and Automation (ICRA)*, May 2021. doi: 10.1109/icra48506.2021.9560840.

[136] Mengxi Li, Alper Canberk, Dylan P Losey, and Dorsa Sadigh. Learning human objectives from sequences of physical corrections. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[137] Stacy Liberatore. Self-driving race car crashes straight into a wall from the starting line during the world's first autonomous race series. *Daily Mail*, 2020. URL `https://www.dailymail.co.uk/sciencetech/article-8899021/Self-driving-race-car-crashes-straight-wall-starting-line-Roborace.html`.

[138] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[139] Allen Liu and Ankur Moitra. Efficiently learning mixtures of mallows models. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 627–638, 2018.

[140] Siyuan Liu, Miguel Araujo, Emma Brunskill, Rosaldo Rossetti, Joao Barros, and Ramayya Krishnan. Understanding sequential decisions via inverse reinforcement learning. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 177–186. IEEE, 2013.

[141] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[142] Dylan P Losey and Marcia K O'Malley. Including uncertainty when learning from human corrections. In *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, pages 123–132. PMLR, 2018.

[143] Alicia Lotz, Klas Ihme, Audrey Charnoz, Pantelis Maroudis, Ivan Dmitriev, and Andreas Wendemuth. Recognizing behavioral factors while driving: A real-world multimodal corpus to monitor the driver's affective state. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

[144] Tyler Lu and Craig Boutilier. Learning mallows models with pairwise preferences. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 145–152, 2011.

[145] Tyler Lu and Craig Boutilier. Effective sampling and learning for mallows models with pairwise-preference data. *Journal of Machine Learning Research*, 15(117):3963–4009, 2014.

[146] Christopher G Lucas, Thomas L Griffiths, Fei Xu, and Christine Fawcett. A rational model of preference learning and choice prediction by children. In *Advances in neural information processing systems*, pages 985–992, 2009.

[147] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

[148] Zelda E Mariet, Suvrit Sra, and Stefanie Jegelka. Exponentiated strongly rayleigh distributions. In *Advances in Neural Information Processing Systems*, pages 4464–4474, 2018.

[149] Albert Maydeu-Olivares. Thurstonian modeling of ranking data via mean and covariance structure analysis. *Psychometrika*, 64(3):325–340, 1999.

[150] Lucas Maystre and Matthias Grossglauser. Fast and accurate inference of plackett-luce models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 172–180, 2015.

[151] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[152] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[153] Jeremy Morton and Mykel J Kochenderfer. Simultaneous policy learning and latent state inference for imitating driver behavior. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017.

[154] Vivek Myers, Erdem Bıyık, Nima Anari, and Dorsa Sadigh. Learning multimodal rewards from rankings. In *5th Conference on Robot Learning (CoRL)*, November 2021.

[155] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[156] Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.

[157] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 189–196. IEEE, 2015.

[158] Stefanos Nikolaidis, David Hsu, and Siddhartha Srinivasa. Human-robot mutual adaptation in collaborative tasks: Models and experiments. *The International Journal of Robotics Research*, 36(5-7):618–634, 2017.

[159] Malayandi Palan, Gleb Shevchuk, Nicholas C. Landolfi, and Dorsa Sadigh. Learning reward functions by integrating human demonstrations and preferences. In *Proceedings of Robotics: Science and Systems (RSS)*, June 2019.

[160] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[161] Li Qian, Jinyang Gao, and HV Jagadish. Learning user preferences by adaptive pairwise comparison. *Proceedings of the VLDB Endowment*, 8(11):1322–1333, 2015.

[162] Ahmed H Qureshi, Jacob J Johnson, Yuzhe Qin, Taylor Henderson, Byron Boots, and Michael C Yip. Composing task-agnostic policies with deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[163] Mattia Racca, Ville Kyrki, and Maya Cakmak. Interactive tuning of robot program parameters via expected divergence maximization. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pages 629–638, 2020.

[164] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[165] Giorgia Ramponi, Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, and Marcello Restelli. Truly batch model-free inverse reinforcement learning about multiple intentions. In *International Conference on Artificial Intelligence and Statistics*, pages 2359–2369. PMLR, 2020.

[166] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. MIT Press, 2005.

[167] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006.

[168] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1765–1772, 2013.

[169] Dorsa Sadigh, S Shankar Sastry, Sanjit A Seshia, and Anca Dragan. Information gathering actions over human internal state. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 66–73. IEEE, 2016.

[170] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, volume 2. Ann Arbor, MI, USA, 2016.

[171] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2017.

[172] Dorsa Sadigh, Nick Landolfi, Shankar S Sastry, Sanjit A Seshia, and Anca 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*, 42(7):1405–1426, 2018.

[173] Dorsa Sadigh, S Shankar Sastry, and Sanjit A Seshia. Verifying robustness of human-aware autonomous cars. *IFAC-PapersOnLine*, 51(34):131–138, 2019.

[174] Jens Schreiter, Duy Nguyen-Tuong, Mona Eberts, Bastian Bischoff, Heiner Markert, and Marc Toussaint. Safe exploration for active learning with Gaussian processes. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 133–149. Springer, 2015.

[175] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[176] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[177] Volkan Sezer, Tirthankar Bandyopadhyay, Daniela Rus, Emilio Frazzoli, and David Hsu. Towards autonomous navigation of unsignalized intersections under uncertainty of human driver intent. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3578–3585. IEEE, 2015.

[178] Ankit Shah, Samir Wadhwania, and Julie Shah. Interactive robot training for non-markov tasks. *arXiv preprint arXiv:2003.02232*, 2020.

[179] David Shinar. Aggressive driving: the contribution of the drivers and the situation. *Transportation Research Part F: traffic psychology and behaviour*, 1(2):137–160, 1998.

[180] Herbert A Simon. Bounded rationality. In *Utility and probability*, pages 15–18. Springer, 1990.

[181] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7461–7472, 2018.

[182] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. In *International Conference on Learning Representations*, 2017.

[183] Elis Stefansson, Jaime Fisac, Dorsa Sadigh, Shankar Sastry, and Karl H. Johansson. Human-robot interaction for truck platooning using hierarchical dynamic games. In *European Control Conference (ECC)*, June 2019.

[184] Hiroaki Sugiyama, Toyomi Meguro, and Yasuhiro Minami. Preference-learning based inverse reinforcement learning for dialog control. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[185] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In *International Conference on Machine Learning (ICML)*, pages 997–1005, 2015.

[186] Yanan Sui, Vincent Zhuang, Joel W Burdick, and Yisong Yue. Multi-dueling bandits with dependent arms. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2017.

[187] Yanan Sui, Joel Burdick, Yisong Yue, et al. Stagewise safe Bayesian optimization with Gaussian processes. In *International Conference on Machine Learning*, pages 4781–4789. PMLR, 2018.

[188] John D. Sutter. Amazon seller lists book at $23,698,655.93 – plus shipping. *CNN*, 2011. URL `http://www.cnn.com/2011/TECH/web/04/25/amazon.price.algorithm/index.html`.

[189] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[190] Nitish Thatte, Helei Duan, and Hartmut Geyer. A method for online optimization of lower limb assistive devices with high dimensional parameter spaces. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[191] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[192] Maegan Tucker, Myra Cheng, Ellen Novoseller, Yisong Yue, Joel Burdick, and Aaron D Ames. Human preference-based learning for high-dimensional optimization of exoskeleton walking gaits. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[193] Maegan Tucker, Ellen Novoseller, Claudia Kann, Yanan Sui, Yisong Yue, Joel Burdick, and Aaron D Ames. Preference-based learning for exoskeleton gait optimization. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[194] Amos Tversky and Daniel Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty*, 5(4):297–323, 1992.

[195] T Velmurugan and T Santhanam. Computational complexity between k-means and k-medoids clustering algorithms for normal and uniform distributions of data points. *Journal of computer science*, 6(3):363, 2010.

[196] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in neural information processing systems*, pages 2352–2360, 2010.

[197] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266, 2018.

[198] Valeria Vitelli, Øystein Sørensen, Marta Crispino, Arnoldo Frigessi Di Rattalma, and Elja Arjas. Probabilistic preference learning with the mallows rank model. *Journal of Machine Learning Research*, 18(158):1–49, 2018.

[199] Woodrow Z. Wang, Mark Beliaev, Erdem Bıyık, Daniel A. Lazar, Ramtin Pedarsani, and Dorsa Sadigh. Emergent prosociality in multi-agent games through gifting. In *30th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2021. doi: 10.24963/ijcai.2021/61.

[200] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, Nando De Freitas, et al. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, pages 1778–1784, 2013.

[201] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2017.

[202] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010. ISBN 1441923225, 9781441923226.

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

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

[205] Nils Wilde, Dana Kulić, and Stephen L Smith. Bayesian active learning for collaborative task specification using equivalence regions. *IEEE Robotics and Automation Letters*, 4(2): 1691–1698, 2019.

[206] Nils Wilde, Alexandru Blidaru, Stephen L Smith, and Dana Kulić. Improving user specifications for robot behavior through active preference learning: Framework and evaluation. *IJRR*, 39(6):651–667, 2020.

[207] Nils Wilde, Dana Kulić, and Stephen L. Smith. Active preference learning using maximum regret. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10952–10959, 2020.

[208] Nils Wilde, Erdem Bıyık, Dorsa Sadigh, and Stephen L. Smith. Learning reward functions from scale feedback. In *5th Conference on Robot Learning (CoRL)*, November 2021.

[209] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press, 2006.

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

[211] Robert C Wilson and Anne GE Collins. Ten simple rules for the computational modeling of behavioral data. *Elife*, 8:e49547, 2019.

[212] Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18 (136):1–46, 2017.

[213] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. Fetch and freight: Standard platforms for service robot applications. In *Workshop on Autonomous Mobile Service Robots*, 2016.

[214] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6818–6827. PMLR, June 2019.

[215] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *Advances in neural information processing systems*, 30, 2017.

[216] Z. Xu, K. Kersting, and T. Joachims. Fast active exploration for link-based preference learning using Gaussian processes. In *European Conference on Machine Learning*, pages 499–514, 2010.

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

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

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

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

[221] Jason Y Zhang and Anca D Dragan. Learning from extrapolated corrections. In *International Conference on Robotics and Automation (ICRA)*, pages 7034–7040, 2019.

[222] Tingru Zhang, Alan HS Chan, and Wei Zhang. Dimensions of driving anger and their relationships with aberrant driving. *Accident Analysis & Prevention*, 81:124–133, 2015.

[223] Zhibing Zhao, Peter Piech, and Lirong Xia. Learning mixtures of plackett-luce models. In *International Conference on Machine Learning*, pages 2906–2914. PMLR, 2016.

[224] Alice X Zheng, Irina Rish, and Alina Beygelzimer. Efficient test selection in active diagnosis via entropy approximation. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 675–682. AUAI Press, 2005.

[225] Zheqing Zhu, Erdem Bıyık, and Dorsa Sadigh. Multi-agent safe planning with gaussian processes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2020. doi: 10.1109/IROS45743.2020.9341169.

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