Risk-Aware Active Inverse Reinforcement Learning

Daniel S. Brown[*], **Yuchen Cui**[*], **and Scott Niekum**
Department of Computer Science
University of Texas at Austin, United States
`dsbrown@cs.utexas.edu, yuchencui@utexas.edu, sniekum@cs.utexas.edu`

Abstract: Active learning from demonstration allows a robot to query a human for specific types of input to achieve efficient learning. Existing work has explored a variety of active query strategies; however, to our knowledge, none of these strategies directly minimize the performance risk of the policy the robot is learning. Utilizing recent advances in performance bounds for inverse reinforcement learning, we propose a risk-aware active inverse reinforcement learning algorithm that focuses active queries on areas of the state space with the potential for large generalization error. We show that risk-aware active learning outperforms standard active IRL approaches on gridworld, simulated driving, and table setting tasks, while also providing a performance-based stopping criterion that allows a robot to know when it has received enough demonstrations to safely perform a task.

Keywords: Inverse Reinforcement Learning, Learning from Demonstrations, Active Learning, Safe Learning

1 Introduction

With recent advancements in robotics, automation is expanding from factories to homes, hotels, and hospitals. Given these less structured workplaces, it is difficult to for engineers to anticipate the wide variety of tasks that a robot may be asked to perform. Learning from demonstration (LfD) [1] algorithms leverage human demonstrations, rather than code, to program robots, providing non-expert end-users with an intuitive way to program robots. Inverse Reinforcement Learning (IRL) [2, 3] is a form of LfD that assumes demonstrations are driven by an underlying reward function that captures the intention of the demonstrator. By recovering this reward function, the robot can perform policy improvement via reinforcement learning [4] and can generalize the task to novel settings.

However, it can be difficult for a human to understand what demonstrations would be most informative to provide to a robot, due to inherent physical, perceptual, and representational differences. To address this issue, *active* IRL algorithms [5, 6, 7] have been proposed that reason about uncertainty and information gain to select queries that are expected to be the most informative under certain criteria. Existing active IRL algorithms aim to minimize uncertainty over policy [5], minimize uncertainty over possible reward functions [7], or maximize expected gain in policy value [6]. To allow robots to be programmed by non-experts, it is crucial that robots can reason about risk and generalization error given limited, ambiguous demonstrations. However, previous active IRL algorithms do not consider the risk of the actual policy learned by the robot. As an example of a demonstration that could lead to high generalization error, consider a human giving a single demonstration placing a vase on the center of a table. There are many plausible motivations for this demonstration: the vase should be in the center of table, the vase can be anywhere on the table, the vase should be above a cup, etc. Our paper seeks to addresses the following question: ***How can a robot that learns from demonstrations actively query for help in states where its learned policy has the potential for high generalization error under the demonstrator's true reward function?***

2 Related Work

There is a growing interest in making AI systems safe and well-behaved [8, 9]. One common way to build safety into a system is to model risk. Risk-aware approaches have been applied to

[*]these authors contributed equally

many different problems including planning [10, 11], reinforcement learning [12, 13], and inverse reinforcement learning [14]. However, risk-aware active learning from demonstrations with high-confidence performance bounds has not been previously addressed.

Leveraging interactive human feedback is an efficient way of learning desired robot behavior and has been explored widely in the literature. The TAMER framework proposed by Knox et al. [15] and the COACH framework proposed by MacGlashan et al. [16] both cast interactive learning as a reinforcement learning (RL) problem. TAMER interprets human feedback as uniform reward signals while COACH argues that human feedback is policy dependent and should be treated as advantage signals. Under an RL paradigm, the policy learning agent is purely exploitative, and does not actively infer the reward function, leading to limited generalization. Recent work on active preference queries[17, 18] approximate a human's true reward function by querying a user's preference between two different policy rollouts.

Most active IRL algorithms use a Bayesian approach to systematically addresses the ambiguity in reward learning [6, 7, 5, 17]. Cohn et al. [6] look at action-wise myopic gain for selecting the active query in order to maximize its potential gain in value, while Cui and Niekum [7] directly compute expected information gain over the distribution of reward functions per state-action pair to maximally reduce uncertainty over the reward distribution. These two methods compute the expected gain in value or information by doing inference over each state-action pair, which is computationally expensive in high dimensional state spaces, and therefore are not practical for real-time robotic active learning tasks. The work of Lopes et al. [5] reasons about state-wise policy entropy over the posterior distribution of policies given demonstrations and asks for demonstrations at states with the highest policy entropies. The state-wise policy entropy is ignorant of the actual policy that will be used for evaluation and does not take the values of states into account. Additionally, Lopes et al. [5] do not provide a semantically meaningful stopping condition for the active learning process. By contrast, our approach to active learning focuses directly on optimizing the objective of learning—the generalization performance of the robot's learned policy.

Existing active IRL approaches do not explicitly use the performance of the robot's learned policy to generate queries. One of the primary reasons for this is that, until recently, practical methods for estimating the performance of a policy learned from demonstrations when the ground-truth reward function is unknown, have not existed. Both Abbeel and Ng [3] and Syed and Schapire [19] derive theoretical Hoeffding bounds on performance based on matching feature counts; however, these bounds are too loose to use in practice [14]. Recently, Brown and Niekum [20, 14] proposed a Value-at-Risk [21] approach based on Bayesian IRL that computes accurate, tight bounds on the performance loss of any policy with respect to the optimal policy under the demonstrator's true, but unknown, reward function, but did not consider the case of active IRL.

3 Background Information

We model our problem as a Markov Decision Process and use a standard Bayesian approach to infer the reward function of the demonstrator.

3.1 Markov Decision Processes

A Markov Decision Process (MDP) is given by the tuple $\langle S, A, T, R, d_0, \gamma \rangle$, where: S is a set of states; A is a set of actions; $T : S \times A \times S \to [0,1]$ is a transition probability function; $R : S \to \mathbb{R}$ is a reward function, with absolute value bounded by R_{max}; d_0 is a starting state distribution and $\gamma \in [0,1)$ is the discount factor. A policy, π maps from states to a probability distribution over actions. The expected value of a policy π under reward function R is the expected return of that policy and is denoted as

$$V_R^\pi = \mathbb{E}_{s_0 \sim d_0}[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi]. \quad (1)$$

Similarly, the expected value of executing policy π starting at any particular state $s \in S$ is defined as

$$V_R^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s, \pi]. \quad (2)$$

The Q-function is defined to describe values of state-action pairs according to some policy:

$$Q^\pi(s,a) = R(s) + \gamma \mathbb{E}_{s' \sim T(s,a,\cdot)}[V^\pi(s')] \quad (3)$$

3.2 Bayesian Inverse Reinforcement Learning

We use Bayesian IRL [22] to find the posterior distribution, $P(R|D) \propto P(D|R)P(R)$, over reward functions R given demonstrations D. Bayesian IRL makes the common assumption that the demonstrator is following a softmax policy, resulting in the following likelihood function:

$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{cQ^*_R(s,a)}}{\sum_{b\in A} e^{cQ^*_R(s,b)}} \tag{4}$$

where c is a parameter representing the degree of confidence we have in the demonstrator's ability to choose the optimal actions [22], and Q^*_R denotes the Q-function of the optimal policy under reward R. We use a uniform prior $P(R)$; however, other priors maybe used to inject domain specific or demonstrator specific knowledge. We use Markov Chain Monte Carlo (MCMC) sampling to obtain a set of likely reward functions given the demonstrations, from which we can extract an estimate of the maximum a posteriori (MAP) reward function R_{MAP} or the mean reward function $\bar{R}$.

4 Methodology

We propose a general framework for risk-aware active queries based on the Value-at-Risk policy loss bounds for LfD proposed by Brown and Niekum [14]. Our approach generates active queries that seek to minimize the policy loss Value-at-Risk of the robot's learned policy.

4.1 Bounding the Performance of a Policy Given Demonstrations

As mentioned in the related work, existing active IRL approaches do not explicitly use performance as a query strategy. By applying and extending the work of Brown and Niekum [14], we demonstrate that it is possible to use performance-based active learning to improve a policy learned purely from demonstration.

Brown and Niekum [14] use samples from $P(R|D)$ to compute the policy loss α-*Value at Risk* (α-quantile worst-case outcome) [21]. Given an MDP\R, a policy to evaluate π_{eval}, and a set of demonstrations D, the policy loss α-VaR provides a high-confidence upper bound on the α-worst-case policy loss incurred by using π_{eval} instead of π^*, where π^* is the optimal policy under the demonstrator's true reward function R. The policy loss of executing π_{eval} under the reward R is given by the Expected Value Difference:

$$\text{EVD}(\pi_{eval}, R) = V_R^{\pi^*} - V_R^{\pi_{eval}}. \tag{5}$$

However, IRL is ill-posed—there are an infinite number of reward functions that explain any optimal policy [2]. Thus, any method that attempts to bound the performance of a policy given only demonstrations should account for the fact that there is never one "true" reward function, but rather a set of reward functions that motivate a demonstration.

To bound the policy loss α-VaR of an evaluation policy π_{eval} given only demonstrations, Brown and Niekum [14] propose using Bayesian IRL to generate samples of likely reward functions, R, from the posterior $P(R|D)$. Each sample reward function produces a sample policy loss, $\text{EVD}(\pi_{eval}, R)$, and these policy losses can then be sorted to obtain a single sided $(1-\delta)$-confidence bound on the policy loss α-VaR [14]. Note that this method uses Markov Chain Monte Carlo simulation to estimate the Value-at-Risk. This allows estimation of the Value-at-Risk over any distribution $P(R|D)$.

4.2 Risk-Aware Active Queries

The method proposed by Brown and Niekum [14] works for any evaluation policy π_{eval}. In an active learning setting, we argue that the most useful policy to evaluate is the robots best guess of the optimal policy under the demonstrators reward. Thus, we use $\pi_{\text{eval}} = \pi_{\text{MAP}}$ where π_{MAP} is the optimal policy corresponding to R_{MAP}, the maximum a posteriori reward given the demonstrations so far; however, our general approach can easily be applied to any policy learned from demonstrations.

Rather than directly using the approach described in Section 4.1 to bound the expected α-Value-at-Risk policy loss of the policy π_{MAP}, we instead generate risk-aware active queries by calculating the

Algorithm 1 Action Query ActiveVaR(Input: MDP\R, D, α, ε; Output: R_{MAP}, π_{MAP})

```
1. Sample a set of reward functions $R$ by running Bayeisan IRL with input $D$ and MDP\R;
2. Extract the MAP estimate $R_{MAP}$ and compute $\pi_{MAP}$;
3. **while** *true*:
   (a) $s_k = \arg\max_{s_i \in S}(\alpha\text{-VaR}(s_i, \pi_{MAP}))$ ;
   (b) Ask for expert demonstration $a_k$ at $s_k$ and add $(s_k, a_k)$ into demonstration set $D$;
   (c) Sample a new set of rewards $R$ by running Bayesian IRL with updated $D$;
   (d) Extract the MAP estimate $R_{MAP}$ and compute $\pi_{MAP}$;
   (e) **break** if $\max_{s_i \in S}(\alpha\text{-VaR}(s_i, \pi_{MAP})) < \varepsilon$;
4. **return** $R_{MAP}$, $\pi_{MAP}$
```

α-VaR for each potential query state s and select the state with the highest policy loss α-VaR as the query. The policy loss of a state s under π_{MAP} given reward R is:

$$Z = \text{EVD}(s, R \mid \pi_{\text{MAP}}) = V_R^{\pi^*}(s) - V_R^{\pi_{\text{MAP}}}(s) \tag{6}$$

where π^* denotes the optimal policy under R. Therefore, the α-VaR for a state can be found using an extension of the method discussed in Section 4.1. We first sample rewards $R \sim P(R|D)$ using Bayesian IRL [22], then for each sampled R and each potential query state s, we compute a sample policy loss Z using Equation 6. These samples for each state can then be used to obtain one-sided confidence bounds on the α-VaR for each state [14]. Note that the state value functions $V_R^{\pi^*}(s)$ can be efficiently computed using the optimal Q-value functions computed during Bayesian IRL.

We consider two types of interactions with the demonstrator: active action queries and active critique queries. An active action query is where the robot proposes a state as its query and a human demonstrator is expected to provide the correct action to take at that state. The active learning process is summarized in Algorithm 1. The objective of our active learning algorithm is to minimize the worst-case generalization error of the learned policy, with as few queries as possible. To do this we compute the α-VaR for each starting state state (sampling states for continuous environments). The state with the highest α-VaR under $P(R \mid D)$ is the state where the robot's policy has the highest α-quantile worst-case policy loss. Thus, selecting this state as the active query is a good approximation for our objective.

In the second type of query, the robot demonstrates a trajectory to the user and asks them to critique the demonstration by segmenting the demonstration into desirable and undesirable segments as proposed by Cui and Niekum [7]. To generate a trajectory for critique, the state with the highest α-VaR policy loss is computed. A trajectory is then generated by executing the MAP policy starting at the selected state. This active learning process can be obtained from Algorithm 1, with the following replacements: (b) Ask for critique of the trajectory of length L under π_{MAP} starting at s_k and (c) Update D with positive and negative segments and sample new set of rewards with Bayesian IRL.

Both of the above active learning algorithms repeat until the α-VaR of the robot's learned policy falls below the desired safety threshold ε. Further discussion of how to choose a meaningful stopping condition can be found in Appendix B. We call our framework Risk-Aware Active IRL and refer to the corresponding active learning algorithms as *ActiveVaR* in future discussion.[2]

4.3 Example

As discussed in 2, given samples of reward functions from running Bayesian IRL, there are many different statistical measurements that can serve as the objective function for active learning [7, 5, 6]. Many existing methods [7, 5] are purely exploratory since they reason only about statistical measures instead of the performance of a specific policy when selecting active queries. Our proposed framework instead uses focused exploration based on policy loss.

[2]Our implementation of ActiveVaR can be found at: https://github.com/Pearl-UTexas/ActiveVaR.git

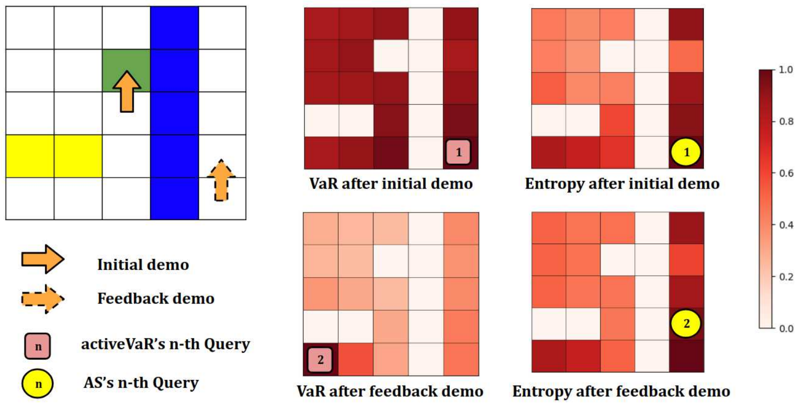


Figure 1: Comparison of active action queries based on performance loss risk or action entropy. The example gridworld has four different unknown features denoted by the yellow, green, white, and blue colors of the cells. White states are legal initial states. AS is the active learning algorithm proposed by Lopes et al. [5] and activeVaR is our proposed active action query algorithm. The first two active queries proposed by each algorithm are annotated on the heatmaps of VaR and entropy values after each iteration. For heatmaps, all values are normalized from 0 to 1.

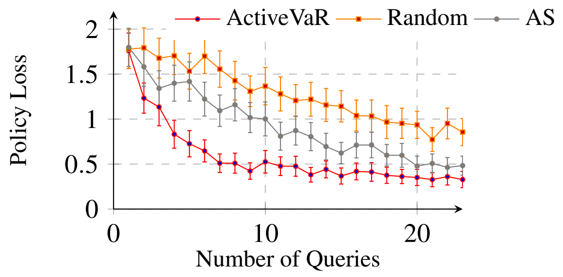


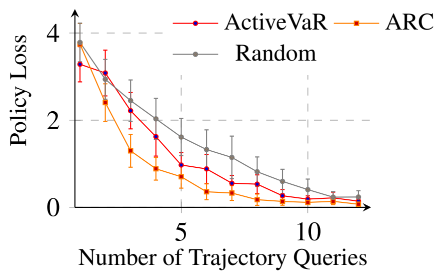
Figure 2: Active action queries: averaged policy losses in 8×8 gridworlds with 48 features.

The example in Figure 1 shows how focusing on Value-at-Risk, rather than minimizing uncertainty over actions as in the Active Sampling (AS) algorithm [5], leads to more intuitively intelligent queries. In this example, there are four indicator features denoted by the white, yellow, green, and blue colors on the cells. Given one initial demonstration going into the green state from a white state, AS and ActiveVaR both pick the bottom right white state as their first query. However, for the second query, AS picks another state next to a blue feature while ActiveVaR picks the state next to a yellow feature. Active queries based on VaR allow the IRL agent to understand that blue feature is unavoidable from all the rightmost states so there is no point asking for more demonstrations from those states while AS only reasons about action entropy.

5 Experiments

5.1 Gridworld Active Action Queries

We first evaluate ActiveVaR on active action queries in which the active learning algorithm selects a state and asks the demonstrator for an action label. The Active Sampling (AS) algorithm proposed by Lopes et al. [5] can only work with action queries. AS is not as computationally efficient as activeVaR, but is much more computationally efficient than methods that require sampling hypothesis probability distributions [7, 6]. We conducted experiments in simulated gridworlds of size 8x8 with 48 random continuous features. The ground-truth feature weights are generated randomly and normalized such that the L1 norm of the weight vector is 1. The expected losses in return comparing to the optimal policy are measured and plotted in Figure 2, where *Random* is a baseline that selects a random state as an active query per iteration. ActiveVaR is able to rapidly reduce the policy loss over iterations since it directly optimizes for performance.



(a) Averaged policy losses

Algorithm	Avg. Time (s)
Random	0.0015
ActiveVaR	0.0101
ARC	865.6993

(b) Timing for one iteration of each algorithm

Figure 3: Active critique queries in 8×8 gridworlds with 48 features.

5.2 Gridworld Active Critique Queries

The previous section demonstrated that action queries using a risk-aware active learning approach based on expected performance loss, or risk, outperforms standard entropy-based queries. We now demonstrate that our risk-aware active learning framework also allows active critique queries [7]. Cui and Niekum [7] proposed a novel active learning algorithm, ARC, where the robot actively chooses sample trajectories to show the human and the human critiques the trajectories by segmenting them into good and bad segments. This type of active query can be more natural for a human than giving an out of context action for a state and only requires the human to be able to recognize, not demonstrate, desirable and undesirable behavior. However, synthesizing trajectories for critique requires costly Bayesian inference over possible segmentations. Additionally, it is purely exploratory since it reasons only about information gain in the reward belief space without reasoning about the performance of current learned policy.

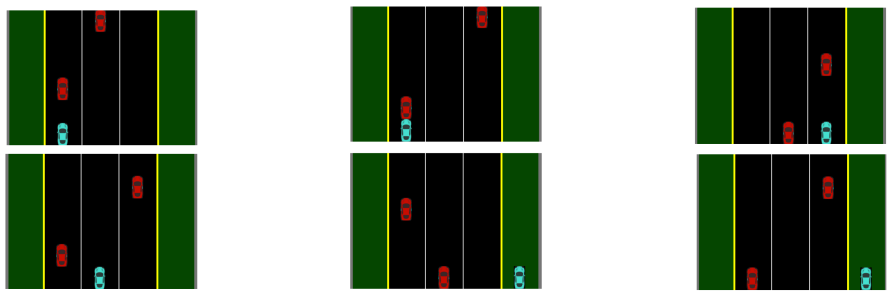
To generate risk-aware performance-based trajectories we examine the robot's policy and evaluate per-state policy loss 0.95-VaR. Because the VaR is calculated based on the value of executing the robot's current policy from that state, we sample a trajectory from the robot's policy starting at that state and ask the demonstrator to critique it. We conducted experiments in 8x8 gridworlds with 48 different continuous features and allow each algorithm to generate a trajectory query of length 8 per critiquing iteration. As baselines, *Random* selects an active query by rolling out the current MAP policy from a randomly selected state. As shown in Figure 3, while ActiveVaR's performance is not as good as that of ARC per number of trajectory queries, the time required to run ActiveVaR is much less than that of ARC. Because ARC's inference algorithm is sequential, not parallelizable, ActiveVaR is more practical for real-time active learning scenarios, while also outperforming random queries. Additional experiments were performed to highlight cases when ActiveVaR outperforms Random by a much larger margin (see Appendix A).

5.3 Active Imitation Learning for a 2D Highway Driving Task

We also evaluated ActiveVaR for learning from demonstration in a 2D driving simulator, in which the transition dynamics are not deterministic and no ground truth reward function is specified. Human demonstrations are provided by controlling the agent (ourselves) via keyboard. In the 3-lane driving simulator, the controlled agent moves forward two times faster than the two other agents on the road and has three available actions at any given time: moving left, move right, or stay. The states are approximated using discrete features including lane positions and distances to other agents. Figure 4 shows the generated active action queries (high-0.95-VaR states) after varying amounts of initial demonstrations are provided. The provided human trajectories demonstrate a safe driving style by avoiding collisions and staying on the road (black lanes). As increasing amounts of initial demonstrations are provided, the active queries start to explore less-frequent states and corner cases that were not addressed in the initial training data.

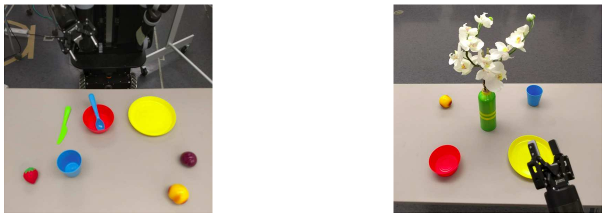
5.4 Robot Table Setting Task

Finally, we consider a robot learning to set a table based on a demonstrator's preferences (see Figure 5). We model the reward function as a linear combination of Gaussian radial basis functions.



(a) 5 Steps of Demos (b) 10 Steps of Demos (c) 20 Steps of Demos

Figure 4: Active action queries in a 2D highway driving task after different numbers of initial human demonstrations. Initial states are randomly sampled and evaluated; high risk states are selected as active action queries.



(a) Place Spoon (b) Place Vase

Figure 5: Setting the table task. (a) Robot actively requests demonstration learning preferences for (a) placing a spoon in the bowl and (b) placing the flower vase in the center of the table.

Given k items on the table, we assume the reward for placement location x is given by

$$R(x) = \sum_{i=1}^{k} w_i \cdot \mathrm{rbf}(x, c_i, \sigma_i^2) \tag{7}$$

where $\mathrm{rbf}(x, c, \sigma^2) = \exp(-\|x - c\|^2/\sigma^2)$.

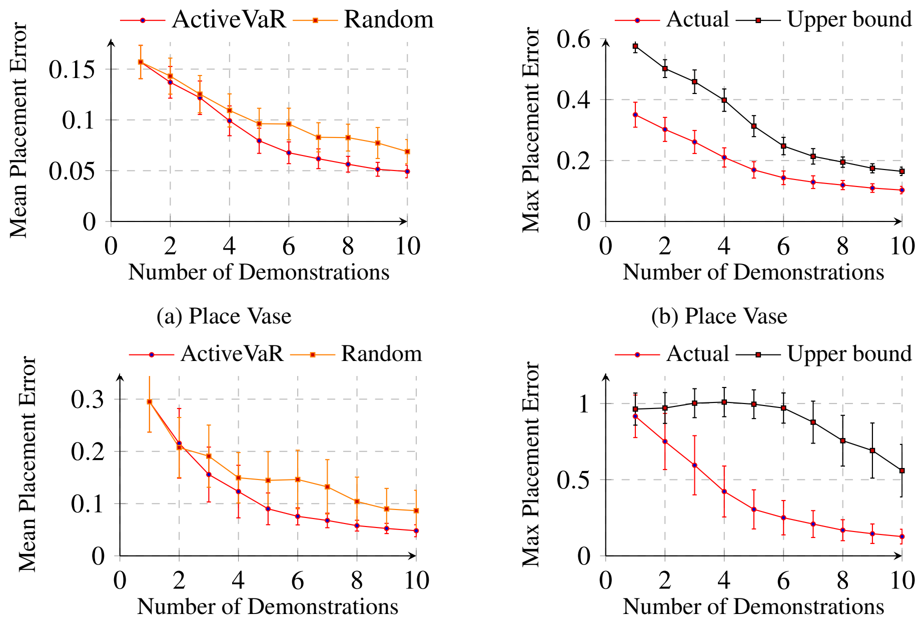
The demonstrator first gives a demonstration from an initial configuration C_0. The robot then needs to infer the correct reward function that matches the demonstrator's intention. We consider an active approach where the robot can generate a novel configuration C_i and ask the demonstrator where it should place the item. The robot hypothesizes multiple configurations C_i and picks the configuration C^* that has maximum 0.95-VaR over its current best guess of the demonstrator's policy.

Given an RBF reward function, the robot needs to estimate an optimal placement position. To calculate the optimal position we use gradient ascent with random restarts and pick the best position. The gradient of the reward with respect to the position x is given by:

$$\nabla_x R(x) = \sum_{i=1}^{k} w_i \cdot \mathrm{rbf}(x, c_i, \sigma_i^2) \left(\frac{-2x + 2c_i}{\sigma_i^2} \right). \tag{8}$$

MCMC is used to estimate the posterior $P(R|D)$ and determine the MAP reward R_{MAP}, which is used as the best guess of the demonstrator's intent. Given the MCMC estimate of the reward posterior, the robot samples random table configurations C_j. For each random configuration, the robot computes the α-VaR by first finding the best placement position x^*_{MAP} given by the MAP reward function, and then evaluating this placement position over the estimated posterior found using MCMC. This is done by calculating the placement loss, $\text{Loss}_i = \|x^i_{R_i} - x^*_{\text{MAP}}\|_2, \ \forall R_i \in P(R|D)$, and then sorting to estimate the α-VaR of that configuration. The robot then picks as the query configuration, $C_{\text{query}} = \arg\max_j \alpha\text{-VaR}(C_j)$, and actively asks for a demonstration in this configuration. Note that in this task the epsilon stopping condition can be defined in terms of placement error (distance between where demonstrator would place object and where robot would place object).

Figure 6 shows the results of two table arrangement experiments. In the first experiment the demonstrator teaches the robot to place a vase of flowers on a table with 4 existing objects. The preference



(a) Place Vase (b) Place Vase

(c) Place Spoon (d) Place Spoon

Figure 6: Results for learning to place a flower vase and learning to place a spoon on a cluttered table. Active queries in (a) and (c) result in lower error than random queries. The 0.95-VaR placement error bound shown in (b) and (d) provides an upper bound on the actual maximum placement error. All results are averaged from 100 trials with 0.5 standard deviation error bars. Placement error is calculated using 200 random test configurations.

is to place the vase in the center of the table, while avoiding placing it on top of other objects. In the second experiment, the demonstrator teaches the robot to place a spoon in a bowl on a table with 6 distractor objects. To allow for expressive reward function hypotheses, we model the reward using RBFs centered on all objects on the table along with 9 fixed RBFs evenly spaced on the table to allow for the possibility of an absolute placement preference. Possible query configurations are randomly generated by changing the position of one of the objects on the table. We generated synthetic ground truth rewards that corresponded to each placement preference and generated synthetic demonstrations using these ground truth rewards. This allows us to rigorously test the active learning process without needing to physically move the objects for each query configuration. We then validated the learned reward function weights on the real robot in a variety of test configurations (video available at `https://youtu.be/yYFae_Obp-0`).

Figures 6(a) and (c) show a comparison of random queries with actively querying the configuration with maximum 0.95-VaR out of 50 randomly generated configurations. Choosing risk-aware queries over random queries results in smaller generalization error. To test whether 0.95-VaR provides a meaningful and accurate upper bound on the true performance of a policy we compared the actual worst-case placement loss under for the robot's current policy after each demonstration with the 0.95-VaR upper bound. Figures 6(b) and (d) demonstrate that the 0.95-VaR upper bound accurately upper bounds the actual worst-case performance and that this bound becomes tighter as more demonstrations are received.

6 Conclusion

We proposed the first active IRL technique that is based on the actual performance of the policy learned from demonstrations. We compared our approach against existing active IRL algorithms and found that our risk-aware approach outperforms entropy based queries in terms of sample complexity and is comparable to active queries based on information gain while requiring three orders of magnitude less computation on the domains we tested. Experiments in simulated 2-d navigation and highway driving domains, as well as robot table placement tasks, demonstrate that risk-aware active queries allow robots to ask for help in areas of the state-space where the robot has the potential for high generalization error. Our approach allows the robot to upper bound its own policy loss and can be used to let a robot to know when it has received enough demonstrations to safely perform a task.

Acknowledgments

This work has taken place in the Personal Autonomous Robotics Lab (PeARL) at The University of Texas at Austin. PeARL research is supported in part by the NSF (IIS-1724157, IIS-1638107, IIS-1617639, IIS-1749204) and ONR (N00014-18-2243).

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[3] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[4] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[5] M. Lopes, F. Melo, and L. Montesano. Active learning for reward estimation in inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 31–46. Springer, 2009.

[6] R. Cohn, E. Durfee, and S. Singh. Comparing action-query strategies in semi-autonomous agents. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1287–1288. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[7] Y. Cui and S. Niekum. Active reward learning from critiques. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[8] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[9] R. Thomas, F. Pega, R. Khosla, A. Verster, T. Hana, and L. Say. Ensuring an inclusive global health agenda for transgender people. *Bulletin of the World Health Organization*, 95(2):154, 2017.

[10] Y. Chow, A. Tamar, S. Mannor, and M. Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1522–1530. 2015.

[11] D. S. Brown, J. Hudack, N. Gemelli, and B. Banerjee. Exact and heuristic algorithms for risk-aware stochastic physical search. *Computational Intelligence*, 33(3):524–553, 2017.

[12] A. Tamar, Y. Glassner, and S. Mannor. Optimizing the cvar via sampling. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2993–2999, 2015.

[13] J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[14] D. S. Brown and S. Niekum. Efficient probabilistic performance bounds for inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[15] W. B. Knox, P. Stone, and C. Breazeal. Teaching agents with human feedback: a demonstration of the tamer framework. In *Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion*, pages 65–66. ACM, 2013.

[16] J. MacGlashan, M. K. Ho, R. Loftin, B. Peng, D. Roberts, M. E. Taylor, and M. L. Littman. Interactive learning from policy-dependent human feedback. *arXiv preprint arXiv:1701.06049*, 2017.

[17] D. Sadigh, A. Dragan, S. S. Sastry, and S. A. Seshia. Active preference-based learning of reward functions. In *Proceedings of Robotics: Science and Systems (RSS)*, page 1, 2017.

[18] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

[19] U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pages 1449–1456, 2008.

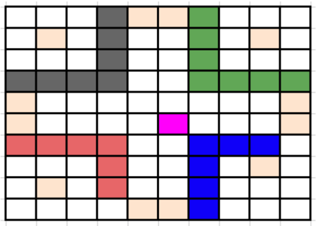
[20] D. S. Brown and S. Niekum. Toward probabilistic safety bounds for robot learning from demonstration. In *AAAI Fall Symposium on AI for HRI*, 2017.

[21] P. Jorion. *Value at risk: the new benchmark for controlling market risk*. Irwin Professional Pub., 1997.

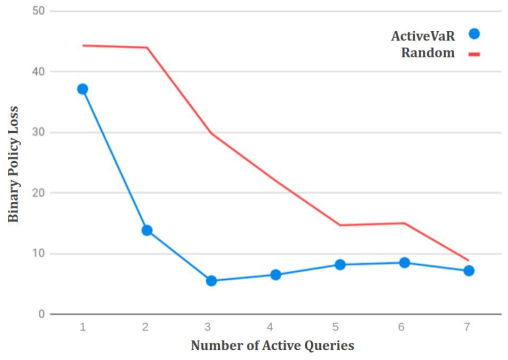
[22] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proc. of 20th International Joint Conference of Artificial Intelligence, 2007*, pages 2586–2591, 2007.

Appendix A Additional Results: Comparing ActiveVaR and Random

In the experiments of navigation in random gridworld (section 5.3), as shown in Figure 3, the improvement of ActiveVaR over Random is not prominent. We believe it is due to the fact that we used dense features and dense rewards that make random queries informative. Therefore, we ran additional experiments where we force the true reward and features to be sparse and the gridworld has only a few informative initial states such that there is a lower chance to sample a trajectory from an informative state.



(a) Gridworld Setup: each color (except light orange) is an unique feature; pink feature is the only feature with positive weight, other colors all have negative weights; states shaded with light orange are designated initial states.



(b) Binary Policy Loss Over Queries

Figure 7: Example Gridworld Navigation Experiment with Sparse Feature and Reward.

Figure 7 shows a selected setup and the policy loss averaged over 10 different runs in the selected environment. As shown in the plot, under this setting, ActiveVaR has a much larger improvement over random.

We also computed the worst-case actual policy loss, as this is what our method seeks to minimize, on a similar barrier domain with all possible states as initial states. In this setting random queries require on average 3.45 times more demonstrations than activeVaR queries to achieve low (< 0.01) worst-case policy loss.

Appendix B Choosing an Intuitive Stopping Condition

The ε stopping criterion in our work is based on an upper bound on performance loss. We know of no other work that has proposed or used such a stopping condition. Without context, an ε stopping condition based on raw policy loss may not be easy to select; however, policy loss can be normalized to obtain a percentage that is more semantically meaningful. The normalization can be computed as

$$\text{normalized EVD}(s, R \mid \pi_{\text{MAP}}) = \frac{V_R^{\pi^*}(s) - V_R^{\pi_{\text{MAP}}}(s)}{V_R^{\pi^*}(s)}. \tag{9}$$

Using the normalized EVD in place of EVD in the above algorithms allows us to calculate an upper bound on the normalized Value-at-Risk.

For example, if the normalized α-VaR is 5% then we can say with high-confidence that the expected return of the learned policy is within 5% of the expected return of the optimal policy under the demonstrators reward. This allows epsilon to be set as a fixed percentage such as 5% of 1% depending on risk aversion.