A Literature Review on the Use of Active Learning for Inverse Reinforcement Learning

Radhika Nayar
Rohan Dutta
Vivek Krishnamurthy

ECE 209AS
Computational Robotics
Professor Ankur Mehta

Table of Contents:

Objective

We try to answer the following questions through our literature review:

1. Why is active learning used in inverse reinforcement learning?
2. What kind of research (old/latest) has been done in the field?
3. What is the current state-of-the-art in this field?
4. What general inferences can be drawn about this field as a whole?
5. Where does the field lack progress and why?

Abstract

In this paper, we provide a comprehensive review of the use of active learning for reward estimation in inverse reinforcement learning (IRL). The major incentives for incorporating active learning in IRL are: 1) it provides an elegant approach to select the most informative states to actively query the expert based on the measure of uncertainty in policy estimation; and 2) it minimizes the number of state-action pair samples required from the expert. We will begin by giving a brief introduction on learning from demonstration (LfD) algorithms, IRL, active learning in general and active learning in IRL framework. Next, we will discuss the background information on Markov Decision Processes (MDP), reinforcement learning (RL) framework, followed by IRL framework, and Bayesian IRL. Next, we will provide a brief timeline of all the different learnings associated with IRL and then provide an in-depth review of the literature involved in active learning IRL framework. Lastly, we will discuss where the field of active learning in IRL is lacking and where future improvements in research can be made. The aim of this paper is to provide an overview of the theoretical background and applications of using active learning for reward estimation in IRL.

Introduction

This literature review addresses the general problem of learning from demonstration (LfD). LfD algorithms leverage human demonstrations and aim to provide an intuitive interface that allows end-users to program robots without the use of code or expert knowledge [2]. In this class of problems, the learner is given a set of samples situation-action pairs by the expert, from which it must recover the overall demonstrated behavior and/or the corresponding task description [13]. We are particularly interested in recovering the task description rather than overall demonstrated behavior.

Inverse Reinforcement Learning (IRL) is a form of LfD that focuses on extracting the reward function that explains the expert's behavior. Using Markov Decision Processes (MDP), the problem can be formalized to a demonstration consisting of a set of state-action pairs and the recovery of task description as the extraction of the reward function. The reward function is generally the more succinct, robust and transferable representation of the task, and completely determines the set of optimal policies [1]. IRL enables for better generalizability to unknown situations since the learner is not just replicating the observed trajectory (imitating the optimal policy) but instead inferring the reason behind such behavior (estimating the reward function).

Active learning from demonstration allows a robot to query an expert for specific types of input to achieve efficient learning. Active learning is a framework in which the learner selects its own input data, leveraging uncertainty and expected information gain. Active learning algorithm works by selecting a small but representative set of samples to be labelled by the expert. The two goals of active learning are: learn a more accurate classifier and minimize the number of state-action samples needed from the expert [2].

Active learning in IRL is used to estimate the reward function while reducing the number of demonstration samples required from the expert. Active IRL measures the uncertainty in policy estimation and then uses this uncertainty information to select best states to query the expert for demonstration. This allows the learner to query the expert for samples at specific states instead of relying only on samples provided by expert at arbitrary states [13]. Assumption made is that the learner has complete access to the state space and can query the expert for a demonstration on any state while the expert is assumed to have perfect state-action knowledge of all states.

The use of active learning in IRL mitigates the need for large number of demonstration samples required from expert and provides the learner with ability to choose “best” states to query the expert for demonstration. Since the choice of which state to query is upto the learner, it is in the learner’s interest to query those states from which maximum information can be gained. The learner tries to find the optimal action by querying the expert at the most informative state. The expert, after receiving the state query, returns to the learner a state-action pair, which the learner uses to update its own estimate of the reward function.

All the algorithms discussed in this literature review are built upon the assumptions made by Bayesian Inverse Reinforcement Learning. All papers discussed define some sort of metric to select states based on a criterion that gives a measure of maximum information gained. In this literature review, we will discuss the different algorithms and metric used for estimating reward function by using active learning in IRL.

Prerequisite Knowledge

Markov Decision Process

MDP’s are discrete stateful systems and is a 6-tuple of (S, A, P, R, H, γ) where,

- S is a finite set of all states of the system,
- A is a finite set of all actions of the system,
- $P(s_{t+1} \mid s_t, a_t)$ is the transition probability that maps a state-action pair at time t onto a distribution of states at time t+1
- $R(a_t, s_t, s_{t+1})$ is the immediate reward received after transitioning from state s_t to state s_{t+1} due to action a_t

- H denotes the number of steps the learner has to accrue reward
- ɤ ∈ [0,1] denotes a discount factor where lower values place more emphasis on the immediate rewards

A Markov Decision Process (MDP) describes a sequential decision making problem in which a learner must choose its action so as to maximize the total discounted reward. The core problem in MDP is to find the optimal policy for the learner that maximizes the cumulative function of random rewards. In MDP, the state is assumed to be fully observed and Markov property holds.

As seen in class we have a couple of equations from MDP's that will be used extensively in the algorithms for Active Learning.

The below equation is the Value function: The optimal value function V* is the expected sum of rewards accumulated when starting from state s and acting most optimally for the horizon of i steps. We can directly define the optimal value function using the Bellman optimality equation.

$$V^\star(s) = R(s) + \gamma \max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} P(s'|s, a)\, V^\star(s')$$

The optimal policy is simply the action that attains this maximum:

$$\pi^\star(s) = \underset{a}{\operatorname{argmax}} \sum_{s'\in\mathcal{S}} P(s'|s, a)\, V^\star(s')$$

The Q Function: the expected value of a state s, and an action a, under the policy

$$Q^*(s,a) = R(s,a) + \gamma E_{s'}[V^* s']$$

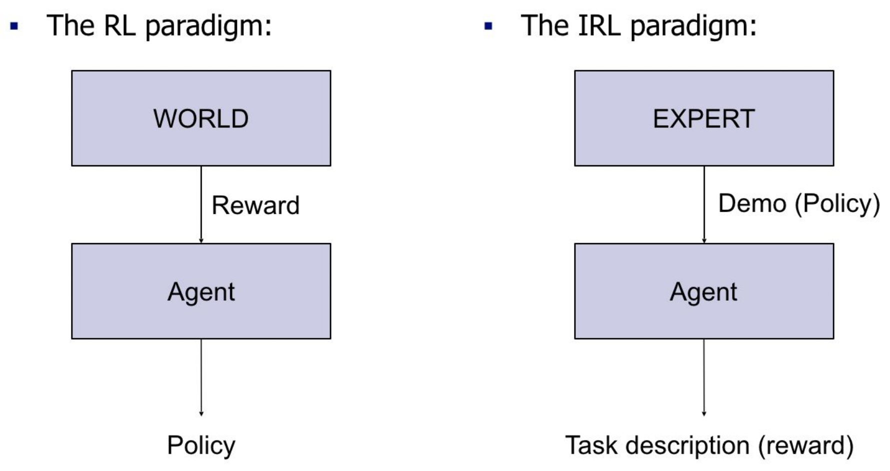


Figure 1: Block diagram of RL framework versus IRL framework

Reinforcement Learning

Reinforcement Learning is the area of machine learning that deals with how the agent should take actions in an environment in order to maximize some reward function. For the scope of this literature review we will define our Reinforcement Learning problems in the form of a Markov Decision Process (MDP). The Active Learning algorithms that we will discuss will make extensive use of MDP equations. There are two main approaches to solving RL problem: methods based on value functions and methods based on policy search.

Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is a paradigm that relies on Markov Decision Processes (MDPs), where the goal of the learner is to extract a reward function from the observed demonstrations of the expert. Reinforcement Learning involves modelling the IRL problem from a Bayesian perspective. The actions of the expert are considered to be evidence which is then used to update a prior distribution on the reward functions. An advantage of this technique is that external information about specific models can be incorporated into the prior. [1]

Bayesian Inverse Reinforcement Learning

The evidence is a series of state-action pairs: $O_X = \{ (s_1,a_1), (s_2,a_2),. . . (s_n, a_n) \}$. Each state-action pair denoted the action a_i taken at $state_i$. We assume that the expert is executing a stationary policy which means that the policy does not change with time. We also assume that each state-action pair is independent of the other state-action pairs.

The probability $P(R \mid O_X)$, which is the probability of the reward given the Observations can then be rewritten using Bayes Rule as follows:

$$P(R|O_x) = \frac{P(O_x|R)P(R)}{P(O_x)}$$

Here $P(O_X \mid R)$ is the likelihood.
$P(R)$ is the prior.
$P (O_X)$ is the posterior.

Block Diagram

Inverse Reinforcement Learning (IRL) [14] was first introduced in the year 2000 and was formulated as a linear programming optimization problem. This was followed by variants to the IRL problem such as Apprenticeship Learning using Bayesian IRL [1] and Maximum Entropy

IRL. In year 2009, we first come across the use of Active Learning for IRL[13] and we consider this as the seminal paper of our literature review. Post year 2010, we observed many Active IRL modifications such as Active Advice Seeking for IRL (2015) [15], Risk Aware (2018) [3] and Risk Sensitive IRL (2019) [5]. We consider the paper on Risk-Aware and Risk-Sensitive IRL as our state-of-the-art papers, due to the novelties introduced by these algorithms which will be discussed in a later section.

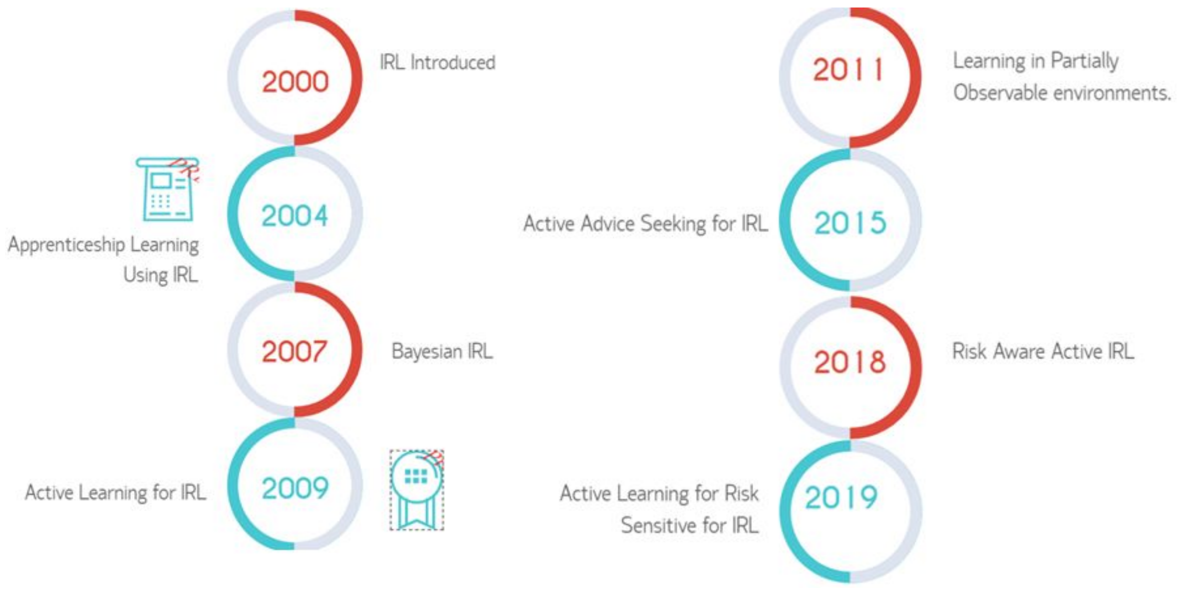

2000
IRL Introduced
2004
Apprenticeship Learning Using IRL
2007
Bayesian IRL
2009
Active Learning for IRL
2011
Learning in Partially Observable environments.
2015
Active Advice Seeking for IRL
2018
Risk Aware Active IRL
2019
Active Learning for Risk Sensitive for IRL

Figure 2: Timeline for IRL and Active IRL

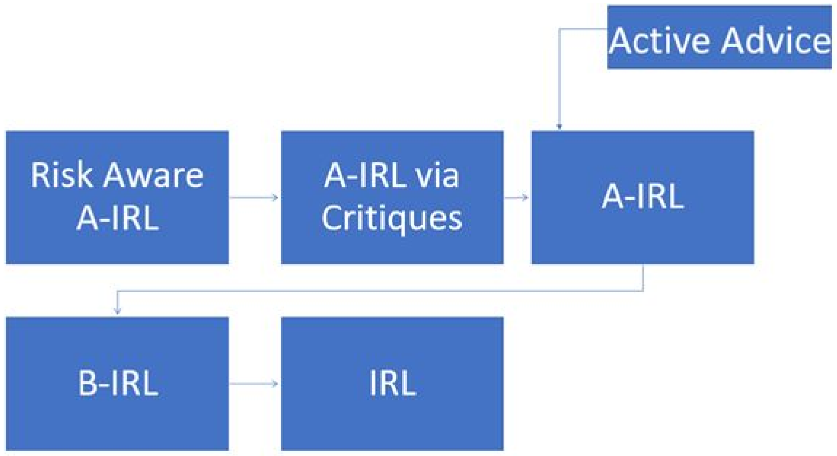


Figure 3: Concept Flow Diagram
Current Literature

Active Learning for Reward Estimation in Inverse Reinforcement Learning

In the year 2009, Lopes et al [13] introduced the first active learning algorithm explicitly designed to estimate reward function from noisy and sampled demonstration of an unknown optimal policy provided by an expert. The idea was built on BIRL which provides access to an estimate of the posterior distribution over the space of possible reward functions, P[r|D], based on a set of demonstrations from the expert. Using this distribution, the learner can select states and query the expert about the corresponding action. The criterion that the learner uses to select states was based on selecting the state with the highest Shannon entropy (the most informative state).

Below are the two algorithms that describe how active sampling strategy can be applied with the IRL methods.

Algorithm 1 General active IRL algorithm.

```
Require: Initial demo $\mathcal{D}$
1: Estimate $\mathbb{P}[r \mid \mathcal{D}]$ using general MC algorithm
2: for all $x \in \mathcal{X}$ do
3:     Compute $\bar{H}(x)$
4: end for
5: Query action for $x^* = \arg\max_x \bar{H}(x)$
6: Add new sample to $\mathcal{D}$
7: Return to 1
```

Algorithm 2 Active gradient-based IRL algorithm.

```
Require: Initial demo $\mathcal{D}$
1: Compute $r^*$ as in (4)
2: Estimate $\mathbb{P}[r \mid \mathcal{D}]$ in a neighborhood of $r^*$
3: for all $x \in \mathcal{X}$ do
4:     Compute $\bar{H}(x)$
5: end for
6: Query action for $x^* = \arg\max_x \bar{H}(x)$
7: Add new sample to $\mathcal{D}$
8: Return to 1
```

Start with initial demonstration D to compute an initial posterior distribution P[r|D]. Then for each sample in the demonstration D, compute the Shannon entropy.
For each state x ∈ X, the mean entropy can be defined as:

$$\bar{H}(x) = \frac{1}{|\mathcal{A}|} \sum_a H(\mu_{xa}) = -\frac{1}{|\mathcal{A}|} \sum_{a,k} \mu_{xa}(k) \log \mu_{xa}(k)$$

At the end of the for loop, select the state with the highest Shannon entropy.

$$x^* = \arg\max_{x \in \mathcal{X}} \bar{H}(x)$$

x* is the most informative state that the learner will use to query the expert about the action to be taken. Add this state-action pair to the demonstration set D and recompute the posterior distribution P[r|D]. This yields the general algorithm summarized in Algorithm 1.

In very large dimensional spaces, Algorithm 2 can be used which replaces step 1 in Algorithm 1 by two steps. The algorithm proceeds by computing the maximum-likelihood estimate r* and then uses Monte-Carlo sampling to approximate P[r|D] in a neighborhood of r*. This significantly reduces the computational requirements of the algorithm 2.

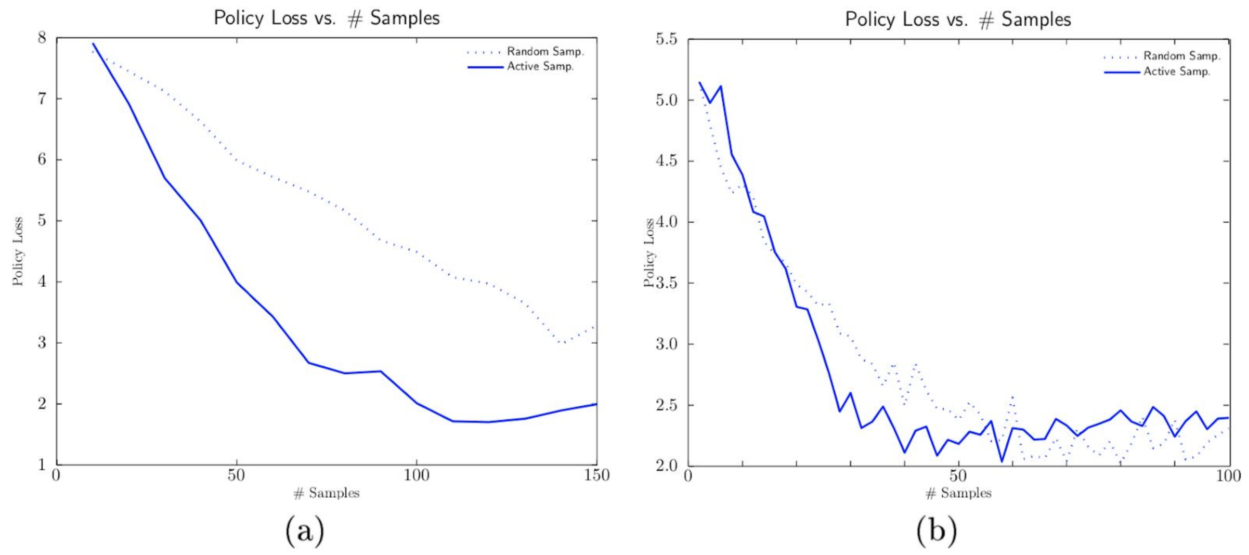


Figure 4: Performance of Algorithm 2 comparing active sampling (solid line) versus random sampling (dash line) as a function of the demonstration size.
(a) Results with parameterized rewards in a 15x15 grid world.
(b) Results with non-parameterized rewards in a 10x10 grid world.

These results illustrate that Figure 4(a) when using parameterized rewards or a prior, the policy in some states restricts the possible policies on other states. Thus, sampling certain states contributes to disambiguate the policy in other states, bringing significant advantage to active sampling approach over uniform sampling approach. On the other hand, Figure 4(b) when using non-parameterized form for a reward function and a prior over possible rewards that is state-wise independent, there is not enough structure in the problem to generalize the observed policy from observed states to non-observed states. Thus, there is no apparent advantage in using active sampling approach over uniform sampling approach.

The algorithm is able to select the potentially most informative state to be queried to the expert by measuring the state-wise entropy over the posterior distribution P[r|D]. The effectiveness of the active learning in IRL is greatly dependent on the prior knowledge about the reward function or the policy. Based on the results of this paper, when considering parameterized policies or priors, the algorithm leads to encouraging results as observed in Figure 4(a). However, in the non-parameterized case or when the prior decorrelates the reward in different states, active learning does not bring a significant advantage as observed in Figure 4(b).

Active Advice Seeking for Inverse RL

An unconventional method was discussed in [15] wherein a human expert is present who specifies reasonable advice. The pieces of advice were given as preferences over states and actions. This paradigm discusses the concept of *active advice-seeking* where the goal is to select areas in the feature space to solicit advice from the expert. Active Advice Seeking is different from traditional active learning in the following ways:

- An active advice seeker can get advice over a larger section of the feature space whereas traditional active learning is constrained to selecting a single example to label.
- Advice can be much more expressive than just a single label, example: grouping a set of classes. In this method a clustering algorithm is used to select like classes.
- Active Advice allows the learner to query similar areas of feature space together potentially reducing the number of advice that the learning algorithm requires this reduces the burden of the expert.

Active Advice Algorithm

The following flowchart describes the framework of the active advice-seeking algorithm:

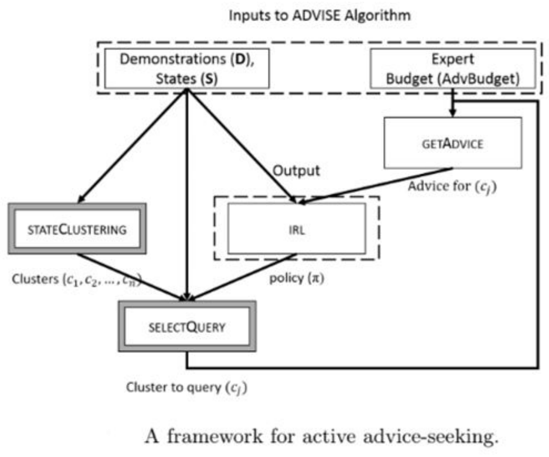

Inputs to ADVISE Algorithm
Demonstrations (D), States (S)
Expert Budget (AdvBudget)
GETADVICE
Output
Advice for (c_j)
STATECLUSTERING
IRL
Clusters ($c_1, c_2, \ldots, c_n$)
policy (π)
SELECTQUERY
Cluster to query (c_j)
A framework for active advice-seeking.

Figure 5: Flowchart for Active Advice algorithm

- The goal of the active ADVIce SEeking agent (ADVISE) is to iteratively accrue advice about different areas of the state space. While accruing advice the agent becomes more and more confident of its current policy.

- It is important to select useful areas to query as well as states that would have similar advice and the expert would give common advice to all these states, and hence the need to cluster these states.
- The above two needs result in the requirement of two functions: *selectQuery* and *stateClustering* as shown in the above framework.
- The goal of the framework is to learn a more robust policy by iteratively soliciting advice from the human expert. The framework allows for advice to be solicited in each step contrary to providing all advice up front.
- Advice is modelled as a set of preferences (a_i) given relative to state c_i and it consists of a set of Preferred Actions (Pr_{ci}) and a set of avoided actions (Av_{ci}).
- This advice is obtained as an output of GetAdvice which is relative to each cluster.

Creating and ordering the clusters

- We first start out by calculating the uncertainty with respect to each state using the following formula:

$$U_s(s_i) = w_p F(s_i) + (1 - w_p) G(s_i)$$

  - S_i: State
  - W_p: corresponding weight
  - $F(S_i)$: uncertainty w.r.t number of demonstrations provided for the state
  - $G(S_i)$: uncertainty w.r.t policy learned for the state

The cluster level uncertainty (U_c) is calculated as:

$$U_c(\mathbf{c}_j) = \frac{\sum_{s_i \in \mathbf{c}_j} U_s(s_i)}{|\mathbf{c}_j|}$$

- The overall goal of the problem is to minimize the total uncertainty of all states of the problem.
- Since the optimal policy is unknown π has to be approximated. The demonstrators action distribution is used as an estimation of π.
- Furthermore K-means is used to cluster states based on the policy of the demonstrator.
- Entropy is used to calculate the the policy uncertainty w.r.t demonstrations

Algorithm 1 ADVISE Algorithm

```
function ADVISE($AdvBudget$,$Expert$,$States$,$D$)
    $aDist$ = PARSEDEM($D$)
    $advice = \emptyset$
    $\pi_0$ = IRL($D$,$adv$)
    $clusters$ = STATECLUSTERING($aDist$)
    for $k = 1$ to $AdvBudget$ do
        $query$ = SELECTQUERY($clusters$,$aDist$,$\pi_{k-1}$)
        $newAdv = Expert(query)$
        $adv = adv \cup newAdv$
        $\pi_k$ = IRL($D$,$adv$)
    end for
    return $\pi_k$
end function
function SELECTQUERY($clusters$,$size$,$aDist$,$\pi$)
    for $i = 1$ to $size$ do
        $U_c(i) = \sum_{s \in clusters(i)}$ UNCERTAINTY($aDist$,$\pi$,$s$)
    end for
    return $\arg\max_i U_c(i)$
end function
function UNCERTAINTY($aDist$,$\pi$,$s$)
    $F(s)$ = entropy of $aDist(s)$
    $G(s)$ = entropy of $\pi(s)$
    return $w_p \cdot F(s) + (1 - w_p) \cdot G(s)$
end function
```

Active Learning from Critiques via Bayesian IRL

The algorithm in this paper[7] proposes a method of automatically generating trajectories by the learner for the expert to evaluate. After the evaluation, the learner will update its prior belief.

Instead of the expert critique the trajectory as a whole, this method allows the expert to segment the trajectory into good and bad segments. The reason behind this approach is that the trajectories are rarely completely optimal or suboptimal. Rather they are a combination of optimal and suboptimal portions. In order to generate trajectories that yield a high information gain, the algorithm builds upon Bayesian Inverse Reinforcement Learning. The Bayesian prior distribution can be used to predict an expected segmentation of any given trajectory, and in turn, predict the expected change in the reward function distribution, and thus the information gain from the expected query.

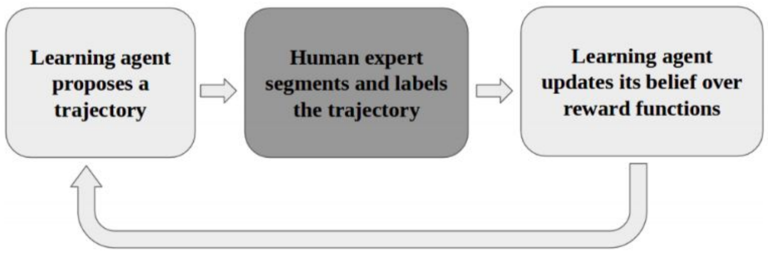

Learning agent proposes a trajectory
Human expert segments and labels the trajectory
Learning agent updates its belief over reward functions

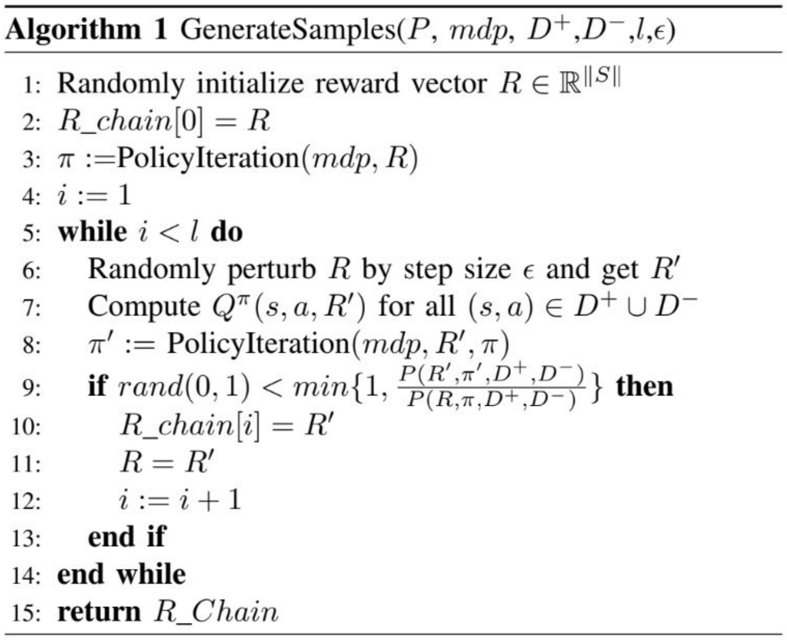
Figure 6: Pictorial representation of Active Learning from Critique via Bayesian IRL

In order to compare different actions in terms of their information gain, we need to find a way to quantify the information gain. The formula used in this paper for this purpose is the Kullback Leibler DIvergence. The mathematical formula for the KL Divergence is given below:

$$D_{KL}(p||q) = \sum_{c=1}^{C} p(c) \log \frac{p(c)}{q(c)}$$

Here p corresponds to the updated state and q corresponds to the previous state. All thought the KL Divergence is not symmetric, its asymmetry helps avoid local optima during the optimization process. Its asymmetric nature also prevents it from imposing any undue bias on states.

Sample trajectories are generated from the optimal policies of the sampled reward function. This is a much efficient way of sampling trajectories than by starting with a state s_0 and then proceeding try every sequence of actions to see which action sequence produces maximum information gain. The pseudo code of the algorithm is below.



Algorithm 1 GenerateSamples(P, mdp, D^+,D^-,l,ϵ)

```
Randomly initialize reward vector $R \in \mathbb{R}^{||S||}$
$R\_chain[0] = R$
$\pi$ :=PolicyIteration($mdp, R$)
$i := 1$
while $i < l$ do
    Randomly perturb $R$ by step size $\epsilon$ and get $R'$
    Compute $Q^\pi(s, a, R')$ for all $(s, a) \in D^+ \cup D^-$
    $\pi'$ := PolicyIteration($mdp, R', \pi$)
    if $rand(0, 1) < min\{1, \frac{P(R',\pi',D^+,D^-)}{P(R,\pi,D^+,D^-)}\}$ then
        $R\_chain[i] = R'$
        $R = R'$
        $i := i + 1$
    end if
end while
return $R\_Chain$
```

Algorithm used to generate trajectories.

Risk-Aware Active IRL [State-Of-The-Art]

In the year 2019, Brown et al [3] proposed the state-of-the-art called risk-aware active IRL algorithm which focuses the queries on areas of the state space with the potentially large generalization error. So far, all the previous active IRL algorithms have aimed to minimize the uncertainty over policy [13] , minimize uncertainty over possible reward functions [7] , or maximize the expected gain in policy value [6]. However, this is the first active IRL algorithm that is based on the actual performance of the policy learned from the demonstrations. It is also the first to provide a performance-based stopping criterion that allowed a robot to know when it has received enough demonstrations to safely perform a task.

The general framework for risk-aware active queries is based on the Value-at-Risk policy loss bounds for learning from demonstration (Lfd) proposed by Brown and Niekum [4]. The approach is to generate queries that seek to minimize the risk (policy loss Value-at-Risk) of the actual policy learned by the robot.

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 $R_{MAP}$, $\pi_{MAP}$
```

where R_{MAP} is the maximum a posteriori reward given the demonstrations so far and
π_{MAP} is the optimal policy corresponding to R_{MAP}

Starting with the initial demonstration D, first sample rewards $R \sim P(R|D)$ using Bayesian IRL and extract the R_{MAP}, maximum a posteriori reward and π_{MAP}, optimal policy based on the demonstrations so far. The objective of the ActiveVaR algorithm is to minimize the worst-case generalization of the learned policy, with as few queries as possible. The learner generates active queries based on the metric α-VaR, which can be thought of as the risk associated with the state. The goal is to calculate the α-VaR (risk) for each potential query state s and select the state with the highest policy loss α-VaR (highest risk) to query the expert. The state with the highest α-VaR under P(R|D) is the state where the policy learned by the robot has the highest α-quantile worst-case policy loss. The learner will use this state, s_k , to query the expert about the action to be taken. Add this state-action pair to the demonstration set D and sample new set of rewards $R \sim P(R|D)$ using Bayesian IRL and recompute the R_{MAP} and π_{MAP}. The algorithm keeps repeating until the α-VaR of the robot's learned policy falls below the user defined desired safety threshold ε.

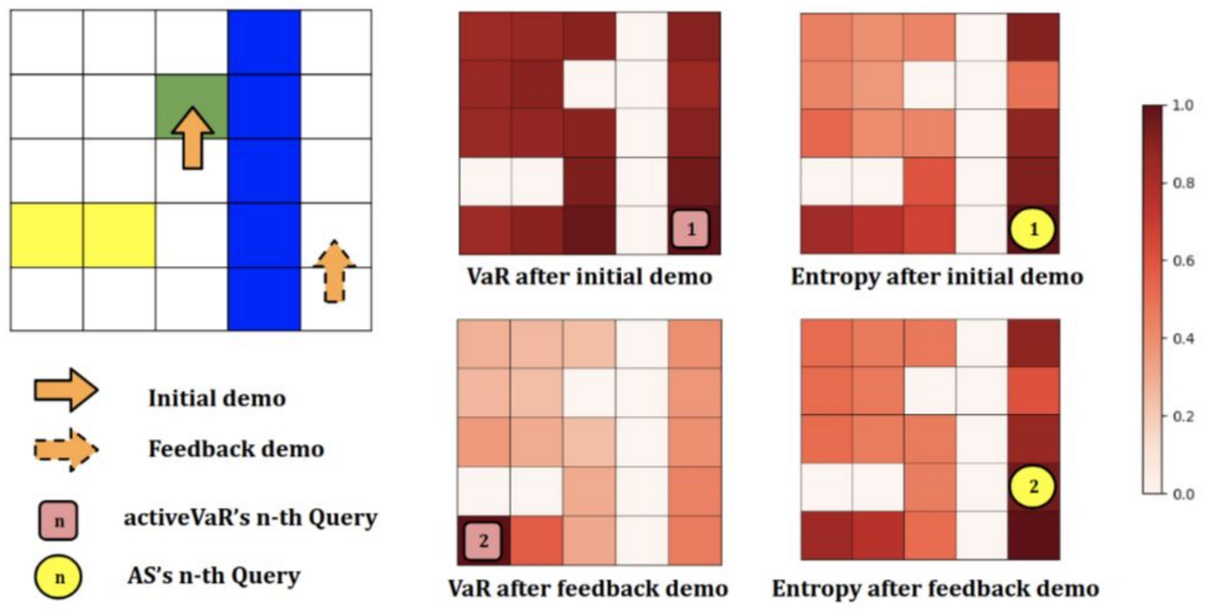


Figure 2: Given one initial demonstration from white state into green state, compare the first-two action queries based on performance loss risk (ActiveVaR) and action entropy (AS). AS is the active learning algorithm proposed by Lopes et al [13] and ActiveVaR is the risk-aware algorithm proposed by Brown et al [3].

The example gridworld above has four indicator features denoted by the yellow, green, white and blue colors of the cells. White states are the legal initial states. Given one initial demonstration from a white state into a green state, both AS and ActiveVaR algorithm pick the bottom right white state as their first query. However, for the second query, AS picks another state next to the blue feature (above the first query) whereas ActiveVaR picks state next to a yellow feature (bottom leftmost). In the second query, the ActiveVaR learner already knows that the blue feature is unavoidable from all the rightmost states so there is no point asking for more demonstrations from the rightmost states while AS learner only reasons about action entropy and keeps querying about the rightmost states. This example shows how focusing on Value-at-Risk (ActiveVaR) rather than minimizing entropy over actions as in the Active Sampling (AS) algorithm, leads to more intuitively intelligent queries.

We consider the Risk Aware Active IRL as our State-Of-The-Art Active IRL algorithm due to its many features that give it a good edge over the other algorithms. The 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.

Conclusions

All the algorithms in Active Learning literature are built on Bayesian Inverse Reinforcement Learning. They all make use of a Bayesian prior which assumes a relationship between the reward and the state-action pairs observed. After a new state-action pair is added to our list of observations, this belief is then updated.

As mentioned earlier, Active Learning gives the learner the freedom to query a state of its choice. We need to use some sort of statistical metric to choose which state would yield maximum information. In all the algorithms we have discussed, there exists some technique to measure the information gain. It could be the Shannon entropy as discussed in [13] or KL Divergence as in [7]. This statistical metric is the most important feature of Active Learning.

The risk aware strategy [3] is the only algorithm we have studied which is concerned with how good the actual policy executed by the learner is and how well it generalizes. The rest of the algorithms are restricted to maximizing some statistical metric and not the actual policy.

Gaps and Drawbacks

One of the problems with Inverse Reinforcement Learning is that it is ill posed, meaning that one reward function can have multiple policies that correspond to it and one policy can have multiple reward functions. This implies that we have to search over a large state space.

In the last decade, AI research has been primarily focused around Computer Vision[12], Speech Recognition[11] and Natural Language Processing. Supervised methods especially Deep Learning have been very successful in such domains. It is very difficult to structure these problems as Inverse Reinforcement Learning problems. The state space is most of these problems is extremely large and hence not suited for IRL. IRL is also ill-suited for Unsupervised problems such as Image generation, where it looses out to algorithms like GAN’s and Variational Autoencoders. [10], [9].

In the current literature, Active Learning has been used for Grid world simulations, simulated 2D navigation tasks and Robot table setting. These are domains that have a very narrow scope and a small state space. In the real world, however, the state and action spaces increase drastically, which again is a problem.

Future Research

All the algorithms discussed in the paper rely on a Bayesian prior. When the prior is decorelated from the reward, Active Learning is not expected to perform well. More research into this field can help us tackle the problem of how to use Active Learning in the event that there is no correlation or very weak correlation between the prior and the reward. This would require a fundamentally different method of approaching the problem as none of the current assumption would be applicable.

In the Advice seeking Active Learning domain, we try to find methods to relate multiple states, so that information gained about one of them can be applicable to all the states in the set. Right now the algorithm used to find related states is the K-Means algorithms which a rather straightforward approach to clustering. We would like to see the use of more sophisticated clustering models like Gaussian Mixture Models or K-Means ++ in this regard.

The current literature also assumed the expert is constantly available and that the expert is always correct. However, what happens when there are multiple experts with conflicting opinions? How do we decide what is the correct action in such scenarios. We must also consider the scenario where the learner does not have complete access to the state space. In such cases what would the optimal query to the query look like? How might we formulate a query with only partial state access?

References

[1] Abbeel P. and Ng A. Y. Apprenticeship learning via inverse reinforcement learning. In ICML, 2004.

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

[3] Brown, D. S., Cui, Y., and Niekum, S. 2018. Risk-aware active inverse reinforcement learning. In *Proceedings of the 2nd Annual Conference on Robot Learning (CoRL)*.

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

[5] Chen, R. and Wenshuo W. 2019. Active learning for risk-sensitive inverse reinforcement learning. In *Proceedings of the 2nd Annual Conference on Robot Learning (CoRL)*.

[6] Cohn R., Durfee E., and Singh S. 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] Cui Y. and Niekum S. Active reward learning from critiques. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[8] Cui, Y., and Niekum, S. 2017. Active learning from critiques via bayesian inverse reinforcement learning. In *Robotics: Science and Systems Workshop on Mathematical Models, Algorithms, and Human-Robot Interaction*.

[9] Diederik P. Kingma and Max Welling. An Introduction to Variational Autoencoders. *arXiv e-prints*, pp. arXiv:1906.02691, 2019.

[10] Goodfellow I. J. , Pouget-Abadie J. , Mirza M. , Xu B. , Warde-Farley D. , Ozair S. , Courville A. C. , and Bengio Y. Generative adversarial nets. In Proceedings of NIPS, pages 2672– 2680, 2014

[11] Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6645–6649. IEEE, 2013.

[12] Krizhevsky A., Sutskever I., and Hinton G. Imagenet classification with deep convolutional neural net- works. In *NIPS*, 2012.

[13] Lopes, F.S. Melo, L. Montesano, Active learning for reward estimation in inverse reinforcement learning, in: European Conference on Machine

[14] Ng A. and Russell S. Algorithms for inverse reinforcement learning. In ICML, 2000.

[15] Odom P and Natarajan S (2015) Active advice seeking for inverse reinforcement learning. In: *Proceedings of AAAI*, pp. 4186–4187

[16] Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. In *Proc. IJCAI*, 2586–2591.