Active Advice Seeking for Inverse Reinforcement Learning

Phillip Odom
Indiana University
Bloomington, Indiana
phodom@indiana.edu

Sriraam Natarajan
Indiana University
Bloomington, Indiana
natarasr@indiana.edu

ABSTRACT

Intelligent systems that interact with humans typically require input in the form of demonstrations and/or advice for optimal decision making. In more traditional systems, such interactions require detailed and tedious effort on the part of the human expert. Alternatively, active learning systems allow for incremental acquisition of the demonstrations from the human expert where the learning system generates the queries. However, active learning allows for only labeled examples as input, significantly restricting the interaction between expert and learning algorithm. Advice-based learning systems increase the expressiveness of the interaction, but typically require all the advice about the domain in advance. By combining active learning and advice-based learning, we consider the problem of actively soliciting human advice. We present the algorithm in an inverse reinforcement learning setting where the utilities are learned from demonstrations. We show empirically the contribution of a more expressive advice over traditional active learning approaches.

Keywords

Inverse Reinforcement Learning; Advice-based Learning; Active Advice Seeking

1. INTRODUCTION

In Inverse Reinforcement Learning (IRL), an autonomous agent tries to explicitly learn the reward function (utilities) of a Markov Decision Process (MDP) by observing demonstrations. The observations include the demonstrator's behavior over time (i.e., actions), measurements of the demonstrator's sensory inputs (i.e., states) and the model of the environment. IRL was introduced [16] as a linear programming problem for finite state spaces and was later extended in different directions including apprenticeship learning [1], Bayesian learning [17], multi-task learning [14], and learning in partially observable environments [4]. While successful, these methods assume *optimality of the trajectories*, which are simply sequences of current state and action pairs.

A recent approach [9] went beyond the simplifying assumption of optimality of trajectories, by considering the presence of a human expert who specifies reasonable advice. The pieces of advice were given as preferences over states and actions and were inspired by the preference elicitation frameworks [2, 24, 18]. A key feature of this approach is that the experts are domain experts and not machine learning experts. As an example, the expert could specify "completely stop at all stop signs - do not make a rolling stop" instead of specifying that "the reward of stopping completely at this specific stop sign is twice that of making a rolling stop" [1]. While capturing more expressive communication between the expert and learning algorithm, this approach required the expert to list all the advice in advance before any learning takes place. The domain experts often lack the machine learning knowledge to understand the most useful advice. They may instead provide the broad trends as advice that would otherwise clearly follow from the demonstrations. This is a disservice not only to the learning algorithm, which may not get informative advice, but also to the expert who has their time squandered.

An ideal human-in-the-loop agent should determine where to solicit advice and when it has accumulated sufficient advice. We propose to employ the concept of *active learning* [20] for this task. Active Learning is a paradigm that has been extensively applied to supervised learning where the learning algorithm works by (actively) selecting a small but representative set of examples to be labeled by the expert. While active learning has been traditionally applied to soliciting example labels, it has not been considered in the realm of seeking advice from an expert which is crucial in reducing the burden of the expert when providing the combination of trajectories and advice.

We introduce *active advice-seeking* where the goal is to select areas in the feature space to solicit advice from the expert. Actively seeking advice has two distinct advantages over active learning. (1) While active learning is constrained to selecting a single example to label, an active advice-seeker can get advice over a larger section of the feature space. This allows the learner to query about similar areas of the feature space together, potentially reducing the number of advice that the learning algorithm requires. (2) Traditional active learning is also limited from the perspective of the communication with the expert as the only information the expert is allowed to provide is a label for a given example. Advice can be much more expressive, for example, by allowing a set of optimal labels instead of a single label.

We make the following key contributions: (1) We present the first algorithm for actively soliciting advice from experts.

[1]The framework is capable of expressing both forms of advice.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, S. Marsella, C. Jonker (eds.), May 9–13, 2016, Singapore.*

(2) We show how the algorithm automatically clusters states and computes uncertainty over these states to solicit this advice. (3) We show how the advice can then be incorporated into the IRL setting. (4) Finally, we demonstrate the efficacy of the proposed approach in standard IRL problems and a real-world drone navigation task. The results clearly show the robustness of the approach as compared to obtaining all the advice in advance.

2. BACKGROUND

2.1 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is a learning framework that learns a reward function based on demonstrations of an expert policy [16]. IRL relies on the expert to act optimally (or near-optimally) and to visit many states in the world. Together, these allow the learner to generate appropriate rewards for the states. IRL is related to Imitation Learning [6, 8, 15] which learns the policy directly from the demonstrations as opposed to learning the reward function. The reward function, used to obtain the optimal policy, mirrors the internal reward function of the demonstrator. IRL is useful in domains where defining the reward function is difficult or tedious, but providing demonstrations of the correct behavior is more practical.

Early work on IRL used linear programs to solve for the reward function [16]. The demonstrations were broken down into constraints on the value of each state. These constraints relied on the vector form of the Bellman equation:

$$\mathbf{v}^\pi = (I - \lambda P_{a_*})^{-1}\mathbf{r}$$

where $\mathbf{v}^\pi$ is a vector that represents the value function, I is the identity matrix, P_a is the probability transition matrix for action a, P_{a_*} is the expert's probability transition matrix, and $\mathbf{r}$ is the reward function. This form of the equation is used to create a set of constraints on the optimal policy ($\pi^*(s) = a_*$) [16] where A is the set of actions.

$$(P_{a_*} - P_a)(I - \lambda P_{a_*})\mathbf{r} \geq 0 \;\forall a \in A \setminus a_*$$

These constraints are used in the resulting linear program along with bounds on the range of possible reward values. These original constraints (shown in red in equation 1) maximize the difference between the value of the demonstrated action and all other actions. Unfortunately, the standard IRL formulation relies on having correct constraints for each state in order to learn an accurate reward function. Noisy demonstrators or rarely visited states that are not visited by near-optimal policies can violate this assumption.

$$\begin{aligned}
&\max_{\mathbf{r},\xi_i,\zeta_j} -\|\mathbf{r}\|_1 + \lambda_t \sum_{i=1}^{n} \xi_i + \lambda_a \sum_{j\in\mathcal{S}_e} \zeta_j \\
&\text{s. t.} \\
&(P^i_{a_*} - P^i_a)\, B\, \mathbf{r} \;\geq\; \xi_i, \quad \forall\, a \in \mathcal{A} \setminus a_*,\; i = 1,\ldots,n, \\
&(P^j_a - P^j_{a'})\, B\, \mathbf{r} \;\geq \zeta_j + \delta_j, \quad \forall\, a \in \mathsf{Pr}_j,\, a' \in \mathsf{Av}_j,\, j \in \mathcal{S}_e, \\
&\zeta_j \geq 0, \quad \forall\, j \in \mathcal{S}_e, \\
&|r_i| \leq r_{\max}, \quad \forall\, i = 1,\ldots,n.
\end{aligned} \tag{1}$$

Kunapuli et al. [9] extended the IRL formulation to include expert advice that is capable of correcting sub-optimal demonstrations. They include additional constraints[2] (shown in blue in equation 1) that allow an expert to specify a set of preferred and avoided actions for a given state. Similar to the original constraints, they enforce that the value of preferred actions be higher than the value of avoided actions. This is captured in the following inequality:

$$\min_{a\in\mathbf{Pr}_j} q^\pi(j,a) - \max_{a'\in\mathbf{Av}_j} q^\pi(j,a') \geq 0$$

where $\mathbf{Pr}_j$ and $\mathbf{Av}_j$ are the set of preferred and avoided actions for state j. These two sets should be mutually exclusive ($\mathbf{Pr}_j \cap \mathbf{Av}_j = \emptyset$). While each set should not be empty, the set of preferred and avoided actions need not be exhaustive ($2 \leq |\mathbf{Pr}_j| + |\mathbf{Av}_j| \leq |A|$). Furthermore, advice can be given for as many (or as few) states as the expert defines.

Their method of advice-giving has two key drawbacks: 1) they assume that the expert must give each advice individually ($\mathbf{Pr}_j$ and $\mathbf{Av}_j$ are only for state j) and 2) the expert is responsible for defining the set of useful advice. While knowledgeable about the domain, an expert may *not be fully aware of what advice the learning algorithm would find most useful.* Furthermore, giving advice *over single states is not practical in large domains.*

To deal with these key issues, our ADVISE framework queries the expert for advice in a particular set of states automatically instead of requiring the advice up-front. This allows for $\mathbf{Pr_j}$ and $\mathbf{Av_j}$ to be given with respect to set $\mathbf{j}$. Furthermore, the responsibility for giving the correct advice is off-loaded from the expert and placed squarely on the learning algorithm.

2.2 Active Learning

In standard machine learning, a labeled dataset is provided. However, in many problems such as information extraction or medical diagnoses, it is not trivial to obtain labeled examples. In many of these problems, there is a wealth of unlabeled examples, such as free text, that can be used for learning. Active Learning [20] relies on the fact that if an algorithm can only solicit the label of a limited number of examples, then not all examples provide the same amount of information. The goal of active learning is to iteratively select the most informative examples to be labeled.

One of the most common ways of selecting an example is uncertainty sampling [10] where the learning algorithm selects examples based on the quality of its prediction (possibly entropy or disagreement among an ensemble of classifiers). After receiving additional labels, the updated predictions result in different uncertainties.

Active learning has been applied to SVMs [19], Bayesian Networks [22, 23], and in sequential decision-making tasks including IRL [13] and imitation learning [3, 8, 5]. As previously shown, active learning has unique challenges in IRL [13]. IRL learns a reward function based on trajectories as an intermediate step. Standard reinforcement learning techniques can then be used to achieve a final policy. Unfortunately *many reward functions may result in similar policies so uncertain reward functions do not imply uncertain policies.* We handle this issue by defining an uncertainty

[2]Their formulation allows for providing advice to state, actions or rewards. We present only the action advice since it is the formulation we consider in this work.

function that considers the multiple levels of potential uncertainty. This is described in detail later.

Active advice-seeking is distinct from standard active learning as it solicits more expressive advice than mere labels. Advice is more general (applying to a set of states vs a single state) as well as more expressive (allowing for label preferences as opposed to a single label). Note that the expert is responsible for providing advice that is most characteristic of a set of states. Therefore, more information can pass between the advice-giver and learning algorithm with each query. The generality of the queries posed to the human expert can depend on their availability. The more available the expert, the more specific the questions. Alternatively, the advice-seeker can be more general when constrained to fewer queries.

3. ACTIVE ADVICE-SEEKING

The goal of the active ADVIce-SEeking (ADVISE) agent is to iteratively accrue advice about different areas of the state space. As the advice accumulates, the learning agent should become more confident about its current policy. While it may be possible to acquire advice about every state, such a strategy is not practical for the advice-giving expert. Therefore, the agent's learning algorithm targets areas of the state space that it believes are uncertain. Conceptually, it is important to select both useful areas to query as well as areas which are likely to have similar advice as the expert will give advice about the entire area jointly. As in active learning, the overall goal of ADVISE is to achieve high performance with few queries to the expert.

There are two key reasons why our advice is more expressive than the examples used by standard active learning. First, the expert is queried over a set of states. This means that the expert must generalize advice over these (possibly significantly different) states. Second, the expert's response is over the set of (possibly partial) label preferences. Thus, there could be multiple reasonable responses that an expert could give. As our advice is more expressive, more information can be received by the learning agent for every interaction with the expert.

For example, consider an agent learning to drive a car. In the active learning framework, the agent might solicit which action to take at a yellow light at the intersection of *Seventh Ave* and *Broadway*. However, any such advice is not broadly applicable as it fits only to the specific situation. Alternatively, in our ADVISE framework, an agent could ask for advice at any yellow light at any (or a set of) intersection(s). Notice that the advice is able to cover significantly more states which all share similar advice. The expert might respond that braking is the preferred action and accelerating is the avoided action. This intentionally leaves out turning the wheel as the driver might be preparing for a right turn. However, current classical active learning methods cannot handle such advice and are limited to specific example labels.

The framework (shown in figure 1) begins by learning a policy from only the expert demonstrations. This allows the algorithm to calculate the best advice query using only the demonstrations. Next, the algorithm seeks advice to correct any deficiencies in the policy by: calculating the uncertainty of the state-clustering, soliciting more advice, and finally updating the current policy.

The algorithm can continue collecting advice as long as necessary to learn a high-quality policy. While we do not explicitly address the problem of discovering how much advice is sufficient to learn a good policy, there is a framework called Knows-What-It-Knows (KWIK) [11] that reasons about the sufficient amount of training data. Such an approach could be extended to apply to our advice-based framework. However, this is beyond the scope of this work.

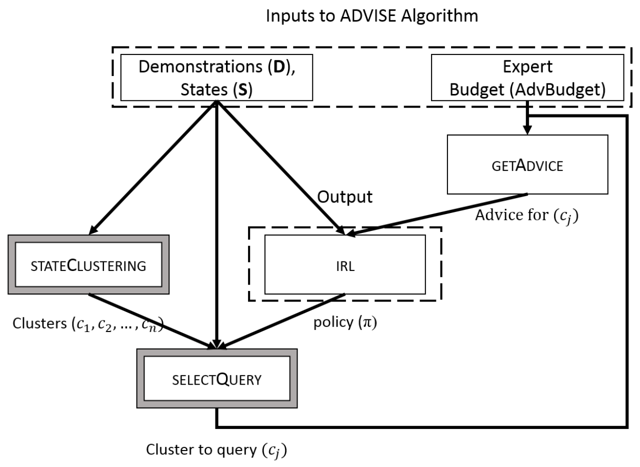
Inputs to ADVISE Algorithm
Demonstrations (D), States (S)
Expert Budget (AdvBudget)
GETADVICE
Output
Advice for (c_j)
STATECLUSTERING
IRL
Clusters ($c_1, c_2, \ldots, c_n$)
policy (π)
SELECTQUERY
Cluster to query (c_j)

Figure 1: A framework for active advice-seeking.

3.1 Problem Formulation

Now, we will formally define the problem and describe it with an intuitive running example.

Goal :	Iteratively solicit advice from the expert to learn a better policy.
Input :	Demonstrations, the set of states/actions, and limited time of expert access.
Output :	A final policy.

The main goal of our framework is to learn a more robust policy by iteratively soliciting advice from the human expert. In previous work [9], experts would be responsible for providing all the relevant advice up-front. This places immense burden on the expert to deeply understand both the domain problem and the learning problem to provide accurate and relevant advice. In many real situations, the domain expert may not be aware of the data available to the learning algorithm. Thus, while the advice will be accurate, it may not be relevant.

We instead formulate the problem of active advice-seeking as an iterative process that aims to solicit relevant advice at each step. Advice received in the previous rounds could effect the advice needed in subsequent rounds. As often in active learning, we assume that there is an advice budget and after that budget is exhausted, the final policy is returned. Our hypothesis, which we verify empirically, is that this policy represents both the best of the advice as well as the best of the data. The framework is shown in figure 1. The key steps of the algorithm are emphasized in grey and will be explained in detail later. We now introduce a running example and use it to explain our algorithm.

Illustrative Example: Consider the example grid nav-

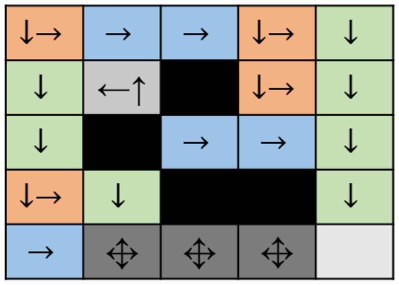
Figure 2: Illustrative example of the ADVISE algorithm for the WumpusWorld domain. The arrows show the actions demonstrated by the human expert. The colors show an ideal clustering of the states. Ideally the algorithm will select the dark grey cluster early to correct the sub-optimal actions.

igation problem in figure 2. The goal is to begin in the upper left state and navigate the grid to reach the lower right state. The arrows inside each cell show the most likely action demonstrated by the expert[3] in that cell (state). Notice how in most of the states the expert gives the correct demonstration. However, the expert makes a mistake on the three middle states in the bottom row (shown in dark grey). The first goal of the algorithm is to cluster the states into sets of similar states. We hypothesize that this will allow for noisy demonstrations to be clustered together. One example clustering is shown through highlighted colors. The second step is to select the cluster with the most uncertainty. Ideally, the dark grey cluster will be selected first and advice will be given to correct the policy in these states yielding a better policy than the one obtained with only the noisy trajectories.

We now define the key concepts of our system using the above example. As we are utilizing an advice-based IRL framework [9] (introduced in the background), the form of our advice is action preferences (sets of preferred and avoided actions). It is important to note that while we share a similar form of advice with Kunapuli et al [9], the goal of this work is to solicit advice from the expert automatically. We now formally define advice using previous notations,

Definition 1. *Advice is a set of preferences ($a_i \in \mathbf{Adv}$). Each piece of advice is given relative to a cluster of states ($\mathbf{c}_i$). It consists of a set of preferred actions ($\mathbf{Pr}_{\mathbf{c}_i}$) and a set of avoided actions ($\mathbf{Av}_{\mathbf{c}_i}$).*

The ADVISE algorithm automatically identifies states of interest ($\boldsymbol{j}$ of the blue constraints in equation 1)[4]. Note that our framework allows $\mathbf{j}$ to be a set of states (as opposed to a single state). The expert then provides the set of preferred ($\mathbf{Pr_j}$) and avoided ($\mathbf{Av_j}$) actions for these states. Note that the states in a cluster could be diverse and the expert is responsible for giving the best representative advice for the entire cluster. Following standard terminology, we refer to a grouping of states as a cluster. A single piece of advice is provided to cover the entire cluster. More formally,

Definition 2. *Cluster $\boldsymbol{c}$ is a set of states, $< s_1, s_2, ..., s_n >$ such that $\pi(s_i) \approx \pi(s_j) \forall s_i, s_j \in \boldsymbol{c}$*

Recall that a policy (π) defines a distribution over the actions for each state [21]. As advice is provided over the entire cluster, it is key that the states in the cluster have similar optimal (or near-optimal) policies. There is a trade-off between the size of the clusters and the accuracy of the clusters. As the size of the clusters are increased, the learning algorithm receives more information. However, as smaller clusters will be more accurate, the learning algorithm receives a higher quality of information.

Example 1. *In our illustrative example (figure 2), advice 1 could be comprised of $\mathbf{c}_1 =< s_{12}, s_{13}, s_{14} >$ where s_{11} is the lower left state (states shown in dark grey). The preferred action should be $\mathbf{Pr}_{\mathbf{c}_1} = \{RIGHT\}$ while the avoided actions should be $\mathbf{Av}_{\mathbf{c}_1} = \{UP, DOWN, LEFT\}$.*

Given a set of clusters, the algorithm must decide how to prioritize (or rank [12]) the clusters so that it is receiving the most useful information at each step. This is important as the advice-giving expert may have limited availability with which questions can be answered. This is similar to the active learning problem discussed earlier [20]. As used in active learning, our proposed approach calculates the uncertainty of each cluster and solicits advice for the most uncertain clusters.

IRL has two sources of uncertainty. (1) The inputs (demonstrations) can be ambiguous in some states about the correct actions due to either too few demonstrations in those states or through mistakes– incorrect actions. (2) The other source of uncertainty is that the output (final policy) may not clearly delineate an action for the agent. Therefore, we define the uncertainty of a state as a combination of these two types of uncertainty.

Definition 3. *State-level uncertainty $U_s(s_i)$ is given by $U_s(s_i) = w_p F(s_i) + (1 - w_p) G(s_i)$ where $F(s_i)$ is the uncertainty w.r.t the number of demonstrations provided for state s_i, w_p is the corresponding weight and $G(s_i)$ is the uncertainty w.r.t the policy learned for state s_i.*

Uncertainty at the cluster level is a function of the uncertainty of each state assigned to that cluster.

Definition 4. *The cluster-level uncertainty U_c is given by $U_c(\boldsymbol{c}_j) = \frac{\sum_{s_i \in \mathbf{c}_j} U_s(s_i)}{|\mathbf{c}_j|}$.*

The two types of uncertainty will be discussed in detail later. The overall goal of active advice seeking is to identify a set of advice out of all possible advice ($\mathbf{a} \subseteq \mathbf{Advice}$), limited by the access to the expert, that minimizes the total uncertainty of all states in the problem. The quality of the advice is also a major concern. Let us denote $\boldsymbol{s_a} = \{\boldsymbol{c}_1 \cup \boldsymbol{c}_2, ..., \cup \boldsymbol{c}_k\}$ as the set of states covered by the k pieces of advice $\boldsymbol{a} = \{a_1, a_2, ..., a_k\}$ and $\boldsymbol{s_b} = \boldsymbol{S} - \boldsymbol{s_a}$. Formally the goal is to find:

$$\arg\min_{\boldsymbol{a}} f(\boldsymbol{a})[U_c(\boldsymbol{s_a}) + U_c(\boldsymbol{s_b})] \qquad (2)$$

where f is a function of the quality of the advice which includes both the size and consistency of the cluster of states

[3]Note that this expert who is generating the demonstrations may be different from the advice-giving expert.
[4]We denote sets as bold.

over which the advice is defined, and U_c is the cluster-level uncertainty.

Unfortunately, this global optimization problem of minimizing cluster-level uncertainty is not tractable. If the size of the example space is n, then the number of possible clusters of examples (and thus the potential advice given) is exponential in n. Furthermore, the advice-quality function is not trivial as it involves estimating the cohesion of the individual clusters (how similar are the states in the cluster in terms of their optimal policy).

Instead, we present an approximate method that involves generating clusters of examples without considering their quality (i.e. uncertainty). After the clusters have been specified, our agent ranks the generated clusters and solicits advice over the clusters in that order. This gives rise to the two important components of active advice-seeking (ADVISE is shown in algorithm 1): 1) stateClustering - Clustering the states over which to ask for advice, 2) selectQuery - Deciding what order to query the clusters for advice. We will discuss each component respectively.

3.2 Creating the State-Clustering

One key difficulty for ADVISE is deciding the parts of the state/action space where advice would be most beneficial. Assuming that there are a set of n states $(s_1, s_2, ..., s_n)$, the goal of group discovery is to find m clusters $(\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_m)$ that are likely to have similar optimal actions (ie $\forall\ s_i, s_j \in \boldsymbol{c}_k, \pi(i) \approx \pi(j)$). Since the optimal policy is unknown, π has to be approximated.

Our approach is to use the demonstrator's action distribution as an estimation of π. Furthermore, we assume that experts do not always choose sub-optimal actions randomly, but instead they make systematic errors in certain kinds of states. For instance, if a driver rolls through a stop-sign at one intersection, he is likely to roll through most intersections. Note that it is trivial to add any structural information about the states that exist in the world. Such information could have significant utility in identifying similar states. However, as this information does not exist in all domains, we use the same information that is used in IRL to learn the reward functions - the state id. We use *k-means* [7] to cluster states based on the policy of the demonstrator. However, any clustering algorithm could be used to replace k-means. While this will not produce optimal state-clusters, the goal is only to create a reasonable clustering. Our empirical evaluation will show that this approach can create reasonable clusters that are useful in learning.

It must be mentioned clearly that the goal of this work is not to propose an interesting clustering algorithm but motivate the need for clustering the states so that they could be queried for advice. As experiments show, a basic algorithm such as k-means can possibly suffice in many domains. Investigating specialized clustering techniques that can better cluster the states is an interesting direction for future research.

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
```

3.3 Ordering the Clusters

Now that the state-clusters have been created, the algorithm must decide the order in which to query the clusters. A common method in traditional active learning is called uncertainty sampling [20] that selects examples based on the entropy of the predicted labels (or another uncertainty measure). In IRL, we calculate the uncertainty w.r.t. each state in the feature space. The idea is to ask for more information about states for which the learner is less likely to know the correct action. While the first step of IRL is to learn a reward function, the final output of interest is still a policy, i.e., a mapping from each state to a distribution over the actions. The uncertainty with respect to the reward function may not be equivalent to the uncertainty with respect to the policy as there are many reward functions that result in the same policy [13]. To counter this issue, our approach seeks to combine two potential sources of uncertainty in IRL. The uncertainty might be either due to the paucity of demonstrations or due to the final policy learned from the reward function.

Uncertainty w.r.t. the demonstrations $(F(s_i))$ can be calculated as the entropy $[F(s) = -\sum_{a \in A} P_s(a) log(P_s(a))]$ of the distribution of actions a taken by the expert in state s. As not all states may be visited, Laplace smoothing can be used to ensure there is a valid action distribution in each state. We expect that as the number of actions taken in a particular state by the demonstrator increases, there will be less uncertainty about the reward learned in that particular state.

Uncertainty w.r.t. the policy is more complicated as the final policy is only optimal w.r.t. the learned reward function [21] which is calculated from the set of demonstrations and advice. As the advice changes, new reward functions must be induced leading to different uncertainty over the policy space unlike the uncertainty over the demonstrations which remains constant. There are also various policy strategies once the value function (or q-values) have been learned including ϵ-greedy or Boltzmann distribution. We use the Boltzmann policy:

$$p(s_i, a) = \frac{e^{Q(s_i,a)}}{\sum_{b \in A} e^{Q(s_i,b)}} \tag{3}$$

Then, entropy is used to calculate policy uncertainty.

It is important to mention that the states chosen to solicit advice are the best states to get advice w.r.t. the most

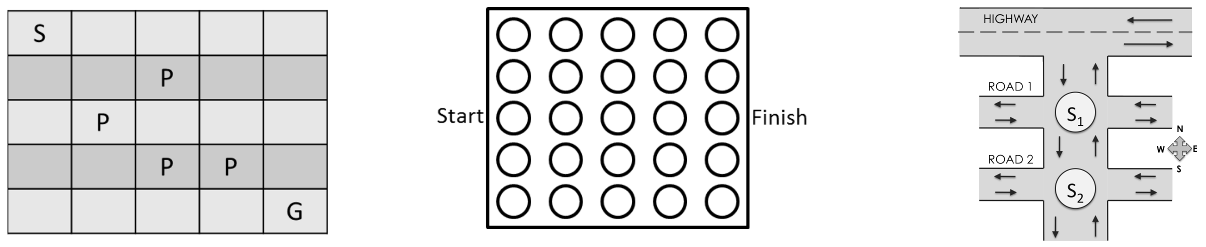

Figure 3: Experimental Domains: WumpusWorld (**LEFT**), Sailing (**CENTER**), Traffic (**RIGHT**)

current reward function. After getting advice, the learner should be able to learn an improved reward function that will in turn provide a potentially different set of uncertain states from which to collect additional advice.

As we will show in several experiments, the ADVISE algorithm is simple, yet effective, in reducing the burden on the expert in two ways. First, while our advice is propositional, we seek general (applying to multiple states) advice as opposed to querying about each state individually. When the state-clusters are large, this reduces the number of queries posed to the expert. The size of these groupings can be controlled based on the query budget. Second, we select the most uncertain queries first. Our hypothesis, which we verify empirically, is that we get sufficient advice even with very few queries.

4. EXPERIMENTS

To evaluate our ADVISE system, we aim to answer the following questions:

General

Q1: Is cluster-level advice effective?

Q2: Does our algorithm correct sub-optimal behaviors?

Q3: Is our algorithm effective in large, realistic problems?

Clustering/Uncertainty Measure

Q4: How effective is the STATECLUSTERING in identifying states with similar optimal actions?

Q5: Does the uncertainty function correctly order clusters?

Q6: How sensitive is the algorithm to sub-optimal state-clusterings?

We demonstrate the performance of ADVISE in three standard domains and one real-world application. In contrast with earlier work [9] focusing on random noise, the demonstrations in our experiments were systematically noisy in certain states. Thus the demonstrator would choose sub-optimal actions in those states. Random actions are also selected with ∼10% chance to ensure that they cover the state space. We use GLPK [5] LP solver to learn the rewards and perform value iteration [21]. We consider our expert to be always correct for the purposes of the experiments and use parameter settings $\lambda_t = 1000, \lambda_a = 500$.

WumpusWorld (figure 3.**LEFT**) is an adaptation [9] of a classic domain [21] containing a 5 × 5 grid with several obstacles. The goal is to navigate the grid, avoid the obstacles, and reach the goal state. If the agent enters a state with an obstacle, it loses. The agent can move in four directions. Noise is introduced in specific states by performing sub-optimally. As the goal of this domain is to reach the goal state, performance is measured by the number of times the learned policy is able to reach this state. The results in this domain are from 100 independent trials for learning the reward function.

The **sailing** domain (figure 3.**CENTER**) is a navigation domain where the goal is to sail from one side of a lake to the other. Running aground on the lake causes the agent to crash. The domain is stochastic as wind causes an uncertain transition function, pushing the agent off course depending on the direction of the wind. Our sailing domain consists of a 5 × 5 grid with 2 configurations of the wind. The agent may choose to sail across or laterally for 3 possible actions. As the goal is to reach the dock on the other side, performance is measured as the number of times the agent reaches this state. The main experiment is over 100 independent trials while cluster quality experiments are run 250 times [6].

The **traffic signal** domain (figure 3.**RIGHT**) controls the signals at two intersections. Each stoplight has four possible configurations to let cars through the light by letting cars coming from a single cardinal direction through at once. The different roads around these intersections have different levels of traffic throughput resulting in 256 possible states. The goal of this domain is to reduce the wait time of the cars waiting at the intersection. All the results in this domain are over 100 independent trials.

Finally, we consider a real-world **drone flying** domain where the goal is to fly a quad-copter in a particular path. The position of the drone is collected using a built-in GPS sensor. In our experiments, we consider moving in the four possible cardinal directions. We considered only a constant altitude. The goal of the domain is to navigate the drone through a sequential set of waypoints on the way to the final position. We consider 3 waypoints (in each corner of a room). The optimal path is shown in figure 4 along with an image of the drone as it is navigating. Note that performance is measured in the percentage of time the agent reaches the waypoints in order and flies to the final destination. The agent has a finite number of opportunities to reach the final state before the battery resource is depleted. This domain is difficult for the demonstrator as the drone controls can be imprecise. Therefore, the demonstrations are noisy 40% of the time.

Methods considered: We compare the performance of our ADVISE system (called *Clustered Advice*) to several

[5]http://www.gnu.org/software/glpk

[6]Note that similar results can be shown with 100 runs.

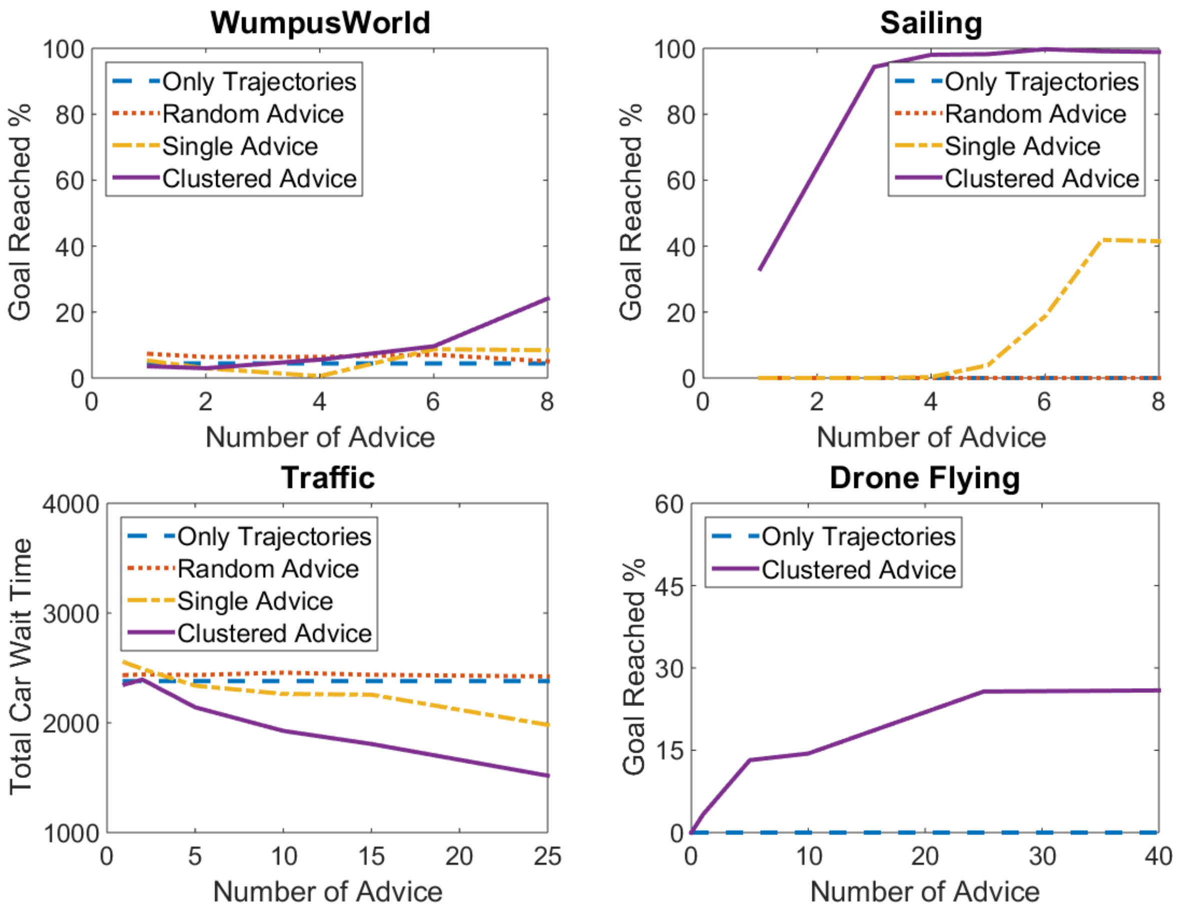

Figure 5: Results for varying the amount of advice in the WumpusWorld (**UPPERLEFT**), Sailing (**UPPERRIGHT**), Traffic (**LOWERLEFT**), and Drone (**LOWERRIGHT**) domains. Performance is measured in domain-specific performance measures. In all cases, our clustered advice outperforms the baselines.

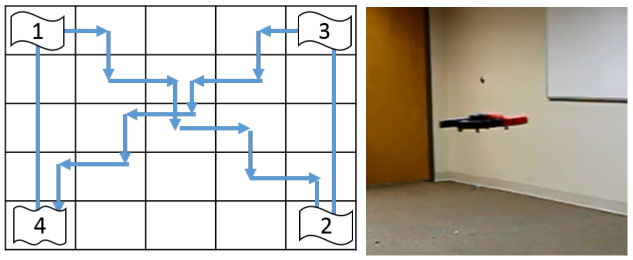

Figure 4: The LEFT image shows an example of the optimal path that the drone should follow. It must visit the waypoints (flags) in order before returning to the starting position. The RIGHT image shows a screenshot of the drone in action.

baselines including only (1) learning from the demonstrations (called *Only Trajectories*), (2) learning from advice over single states instead of advice over the entire cluster (called *Single Advice*), and (3) learning with advice on randomly selected states (called *Random Advice*). When querying about a single state, the expert returns the advice about that particular state. When querying about a group of states, the human expert is responsible for giving advice that is most relevant to the cluster. One possible metric that we use in our experiments is providing preferred actions that appear as optimal actions in the most number of states in the cluster. While this is one strategy, human experts may identify more important states in a cluster. Exploration of these strategies is a area for future work.

Results: We will first discuss results from WumpusWorld, Sailing and Traffic before explaining our progress in the Drone Flying domain. Following the prior work on advice giving, the domains we consider have systematic noise in them. The hypothesis is that ADVISE will correct these behaviors in the learned reward function and therefore in the final policy. Please note that advice is provided only on solicitation by the algorithm and not apriori. Consequently, we evaluate our ADVISE algorithm in different ways: to test whether the advice contributes to the policy, we compare against only using the trajectories; to test whether the STATECLUSTERING generates good advice, we compare against soliciting single pieces of advice; to test the uncertainty measure, we compare against random advice.

The results are shown in figure 5. Please note that in WumpusWorld and Sailing, a higher goal reached % means higher performance, while in Traffic, a lower wait time means higher performance. There are several general trends across all the domains. As the amount of advice increases, the

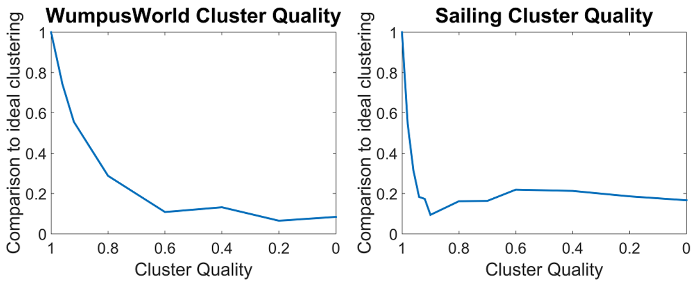

Figure 6: Results for varying the quality of the cluster in the WumpusWorld (**LEFT**) and Sailing (**RIGHT**) domains. Performance is measured relative to the ideal clustering.

performance of the advice methods (other than random) increase. It is clear that advice significantly contributes to the learned policy in all domains (**Q2**). Furthermore, clustered advice yields significantly higher gains much more quickly than soliciting a single piece of advice (**Q1**). Not only is performance higher with clustered advice, it also decreases the number of queries an expert has to answer. This results in significant time savings for the human expert.

For further analysis of these results, we computed the quality of the learned clusters. We defined the quality of the clusters to be the fraction of states which are in a cluster with similar states (states with similar optimal actions). Note we consider the plurality of the cluster to be similar and all other states to be dissimilar. For reference, the minimum possible value for this metric is 25% (WumpusWorld), 33.3% (Sailing), 6.3% (Traffic). For these experiments (100 independent trials), the quality of the clusters is the following: 94.2% (WumpusWorld), 91.3% (Sailing), 69.58% (Traffic). Each of these domains has a fairly high value suggesting that our algorithm is able to identify similar states (**Q4**). There is also room for improvement suggesting that our algorithm could perform better if the clustering algorithm could be improved.

Ideally, the algorithm should target the sub-optimal regions of the state space when requesting advice. Unfortunately, this is very difficult to capture correctly. However, as seen in the experiments, with only a few clustered pieces of advice the algorithm significantly outperforms all of the baselines. This is especially true in the Sailing and Traffic domains. This suggests that our algorithm is able to target these regions, thus answering **Q5** affirmatively.

Varying Cluster Quality: As our method is dependent on the quality of the clusters generated, we investigate how varying the cluster quality impacts the performance. We replace the automatic clustering (StateClustering function in algorithm 1) with a static clustering defined by experts. These clusterings will be optimal in the sense that states appearing in the same cluster will require the same advice. We then measure the performance of the algorithm as we perturb the static clustering. For example when cluster quality is 50%, half of the states will be displaced into incorrect clusters (and thus receive incorrect advice). We compare the performance (goal reached % used in previous experiments) of each sub-optimal clustering to the ideal clustering. Therefore, a performance of 1.0 means that the performance is equal to using an ideal clustering.

We present results for two domains: WumpusWorld (figure 6.**LEFT**) and Sailing (figure 6.**RIGHT**). As expected, in both domains, as the cluster quality decreases, the performance decreases. This confirms the suspicion that the clustering technique is vital to taking the most advantage of expert advice. This answers question **Q6** that the algorithm is reasonably sensitive to sub-optimal state-clusterings. However, as seen in the previous experiments, our algorithm is still able to improve performance by using reasonable approximations of the clusterings.

Real-World Application: In order to show the applicability of our approach on realistic problems and not just on standard benchmarks, we focus on a drone flying domain where the goal is to control a drone as it flies to particular waypoints. This is an especially difficult problem as the drone must visit the waypoints in a particular order to successfully complete the task. We compare only our clustered advise and using only trajectories as this domain represents a proof of concept (figure 5.**LOWERRIGHT**). The difficulty of the problem is clear as noisy trajectories are not sufficient to learn a reasonable policy. However, our clustered advice is capable of guiding the drone more reliably. This domain suggests that our algorithm is effective in large, realistic domains (**Q3**).

5. CONCLUSION

We have presented a new formalism for active advice-seeking. Like traditional active learning, the learning algorithm is responsible for querying the expert about areas of uncertainty. However, unlike active learning, advice is solicited over an area of the state space (multiple states) to further reduce the queries directed to the expert. We demonstrate our algorithm through an advice-taking for inverse reinforcement learning framework and develop an uncertainty function for learning a policy through demonstrations. We demonstrate the validity of our approach in three standard domains and show its applicability to a real-world task. Improving both the clustering technique and the uncertainty measure are clear areas for future improvement. Also, our technique could be improved through relational techniques that would increase the expressiveness of our advice. Our clustering of the ground states is a way of approximating the underlying structure that relational techniques could make use of more faithfully.

6. ACKNOWLEDGMENTS

The authors thank the Army Research Office (ARO) grant number W911NF-13-1-0432 under the Young Investigator Program and the CwC Program Contract W911NF-15-1-0461 with the US Defense Advanced Research Projects Agency (DARPA) and the ARO. Any opinions, findings and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, ARO or the US government. The authors also acknowledge the support of PARC XEROX through their faculty award. The authors thank Juan Zapata Valencia for his work with the drone.

REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.
[2] C. Boutilier. A POMDP formulation of preference elicitation problems. In *AAAI*, pages 239–246, 2002.

[3] M. Cakmak and A. Thomaz. Active learning with mixed query types in learning from demonstration. In *ICML Workshop on New Developments in Imitation Learning*, 2011.

[4] J. Choi and K.-E. Kim. Inverse reinforcement learning in partially observable environments. *JMLR*, 12:691–730, 2011.

[5] M. Hamidi, P. Tadepalli, R. Goetschalckx, and A. Fern. Active imitation learning of hierarchical policies. In *IJCAI*, 2015.

[6] H. He, H. Daume III, and J. Eisner. Imitation learning by coaching. In *NIPS*, 2012.

[7] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall Inc., Upper Saddle River, NJ, 1988.

[8] K. Judah, A. Fern, P. Tadepalli, and R. Goetschalckx. Imitation learning with demonstrations and shaping rewards. In *AAAI*, 2014.

[9] G. Kunapuli, P. Odom, J. Shavlik, and S. Natarajan. Guiding autonomous agents to better behaviors through human advice. In *ICDM*, 2013.

[10] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR*, 1994.

[11] L. Li, M. Littman, T. Walsh, and A. Stregl. Knows what it knows: a framework for self-aware learning. *Machine Learning*, 82(3):399–443, 2011.

[12] P. Li, C. Burges, and Q. Wu. Mcrank: Learning to rank using mulitple classification and gradient boosting. In *NIPS*, 2007.

[13] M. Lopes, F. Melo, and L. Montesano. Active learning for reward estimation in inverse reinforcement learning. In *ECML*, 2009.

[14] S. Natarajan, G. Kunapuli, K. Judah, P. Tadepalli, K. Kersting, and J. Shavlik. Multi-agent inverse reinforcement learning. In *ICMLA*, 2010.

[15] S. Natarajan, P. Odom, S. Joshi, T. Khot, K. Kersting, and P. Tadepalli. Accelerating imitation learning in relational domains via transfer by initialization. In *ILP*, 2013.

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

[17] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *IJCAI*, 2007.

[18] C. Rothkopf and C. Dimitrakakis. Preference elicitation and inverse reinforcement learning. In *ECML-PKDD*, 2011.

[19] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *ICML*, 2000.

[20] B. Settles. *Active Learning*. Morgan & Claypool, 2012.

[21] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, Cambridge, MA, 1998.

[22] S. Tong and D. Koller. Active learning for parameter estimation in bayesian networks. In *NIPS*, 2000.

[23] S. Tong and D. Koller. Active learning for structure in bayesian networks. In *IJCAI*, 2001.

[24] T. Walsh. Uncertainty in preference elicitation and aggregation. In *AAAI*, 2007.