Apprenticeship Learning About Multiple Intentions

Monica Babeş-Vroman BABES@CS.RUTGERS.EDU
Vukosi Marivate VUKOSI@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, 110 Frelinghuysen Rd, Piscataway, NJ 08854 USA

Kaushik Subramanian KAUSUBBU@GATECH.EDU
College of Computing, Georgia Institute of Technology, 801 Atlantic Dr., Atlanta, GA 30332 USA

Michael Littman MLITTMAN@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, 110 Frelinghuysen Rd, Piscataway, NJ 08854 USA

Abstract

In this paper, we apply tools from inverse reinforcement learning (IRL) to the problem of learning from (unlabeled) demonstration trajectories of behavior generated by varying "intentions" or objectives. We derive an EM approach that clusters observed trajectories by inferring the objectives for each cluster using any of several possible IRL methods, and then uses the constructed clusters to quickly identify the intent of a trajectory. We show that a natural approach to IRL—a gradient ascent method that modifies reward parameters to maximize the likelihood of the observed trajectories—is successful at quickly identifying unknown reward functions. We demonstrate these ideas in the context of apprenticeship learning by acquiring the preferences of a human driver in a simple highway car simulator.

1. Introduction

Apprenticeship learning (Abbeel & Ng, 2004), or AL, addresses the task of learning a policy from expert demonstrations. In one well studied formulation, the expert is assumed to be acting to maximize a reward function, but the reward function is unknown to the apprentice. The only information available concerning the expert's intent is a set of trajectories from the expert's interaction with the environment. From this information, the apprentice strives to derive a policy that performs well with respect to this unknown reward function. A basic assumption is that the expert's intent can be expressed as a reward function that is a linear combination of a known set of features. If the apprentice's goal is also to learn an explicit representation of the expert's reward function, the problem is often called inverse reinforcement learning (IRL) or inverse optimal control.

In many natural scenarios, the apprentice observes the expert acting with different intents at different times. For example, a driver might be trying to get to the store safely one day or rushing to work for a meeting on another. If trajectories are labeled by the expert to identify their underlying objectives, the problem can be decomposed into a set of separate IRL problems. However, more often than not, the apprentice is left to infer the expert's intention for each trajectory.

In this paper, we formalize the problem of apprenticeship learning about multiple intentions. We adopt a clustering approach in which observed trajectories are grouped so their inferred reward functions are consistent with observed behavior. We report results using seven IRL/AL approaches including a simple but effective novel approach that chooses rewards to maximize the likelihoods of the observed trajectories under a (near) optimal policy.

2. Background and Definitions

In this section, we define apprenticeship learning (AL) and the closely related problem of inverse reinforcement learning (IRL). Algorithms for these problems take as input a Markov decision process (MDP) without a reward function and the observed behavior of the expert in the form of a sequence of state–action pairs. This behavior is assumed to be (nearly) optimal in the

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

MDP with respect to an unknown reward function. The goal in IRL is to find a proxy for the expert's reward function. The goal in AL is to find a policy that performs well with respect to the expert's reward function. As is common in earlier work, we focus on IRL as a means to solving AL problems. IRL is finding application in a broad range of problems from inferring people's moral values (Moore et al., 2009) to interpreting verbal instructions (Branavan et al., 2009).

We use the following notation: MDP\$r$ (or MDP) is a tuple (S, A, T, γ), where S is the state space, A is the action space, the transition function $T : S \times A \times S \rightarrow [0, 1]$ gives the transition probabilities between states when actions are taken, and $\gamma \in [0, 1)$ is a discount factor that weights the outcome of future actions versus present actions. We will assume the availability of a set of trajectories coming from expert agents taking actions in the MDP in the form $D = \{\xi_1, ..., \xi_N\}$. A trajectory consists of a sequence of state-action pairs $\xi_i = \{(s_1, a_1), \ldots\}$.

Reward functions are parameterized by a vector of reward weights θ applied to a feature vector for each state-action pair $\phi(s, a)$. Thus, a reward function is written $r_\theta(s, a) = \theta^T \phi(s, a)$. If the expert's reward function is given by θ_E, the apprentice's objective is to behave in a way that maximizes the discounted sum of expected future rewards with respect to r_{θ_E}. However, the apprentice does not know θ_E and must use information from the observed trajectories to decide how to behave. It can, for example, hypothesize its own reward weights θ_A and behave accordingly.

IRL algorithms differ not just in their algorithmic approach but also in the objective function they seek to optimize (Neu & Szepesvári, 2009). In this work, we examined several existing algorithms for IRL/AL. In Projection (Abbeel & Ng, 2004), the objective is to make the features encountered by the apprentice's policy match those of the expert. LPAL and MWAL (Syed et al., 2008) behave in such a way that they outperform the expert according to θ_A. Policy matching (Neu & Szepesvári, 2007) tries to make the actions taken by its policy as close as possible to those observed from the expert. Maximum Entropy IRL (Ziebart et al., 2008) defines a probability distribution over complete trajectories as a function of θ_A and produces the θ_A that maximizes the likelihood of the observed trajectories.

It is worth noting several approaches that we were not able to include in our comparisons. Bayesian IRL (Ramachandran & Amir, 2007) is a framework for estimating posterior probabilities over possible reward functions given the observed trajectories. It assumes that randomness is introduced into each decision made by the expert. In Active Learning (Lopes et al., 2009), transitions are provided dynamically. The apprentice queries the expert for additional examples in states where needed.

We devised two new IRL algorithms for our comparisons. The linear program that constitutes the optimization core of LPAL (Linear Programming Apprenticeship Learning) is a modified version of the standard LP dual for solving MDPs (Puterman, 1994). It has as its variables the "policy flow" and a minimum per-feature reward component. We note that taking the dual of this LP results in a modified version of the standard LP primal for solving MDPs. It has as its variables the value function and θ_A. Because it produces explicit reward weights instead of just behavior, we call this algorithm Linear Programming Inverse Reinforcement Learning (LPIRL). Because its behavior is defined indirectly by θ_A, it can produce slightly different answers from LPAL. Our second algorithm seeks to maximize the likelihood of the observed trajectories, as described in the next section.

3. Maximum Likelihood Inverse Reinforcement Learning (MLIRL)

We present a simple IRL algorithm we call Maximum Likelihood Inverse Reinforcement Learning (MLIRL). Like Bayesian IRL, it adopts a probability model that uses θ_A to create a value function and then assumes the expert randomizes at the level of individual action choices. Like Maximum Entropy IRL, it seeks a maximum likelihood model. Like Policy matching, it uses a gradient method to find optimal behavior. The resulting algorithm is quite simple and natural, but we have not seen it described explicitly.

To define the algorithm more formally, we start by detailing the process by which a hypothesized θ_A induces a probability distribution over action choices and thereby assigns a likelihood to the trajectories in D. First, θ_A provides the rewards from which discounted expected values are derived:

$$Q_{\theta_A}(s, a) = \theta_A^T \phi(s, a) + \gamma \sum_{s'} T(s, a, s') \bigotimes_{a'} Q_{\theta_A}(s', a').$$

Here, the "max" in the standard Bellman equation is replaced with an operator that blends values via Boltzmann exploration (John, 1994): $\bigotimes_a Q(s, a) = \sum_a Q(s, a) e^{\beta Q(s,a)} / \sum_{a'} e^{\beta Q(s,a')}$. This approach makes the likelihood (infinitely) differentiable, although, in practice, other mappings could be used. In our work, we calculate these values via 100 iterations of value iteration and use $\beta = 0.5$.

Algorithm 1 Maximum Likelihood IRL

Input: MDP\r, features ϕ, trajectories $\{\xi_1, \ldots, \xi_N\}$, trajectory weights $\{w_1, \ldots, w_N\}$, number of iterations M, step size for each iteration (t) α_t, $1 \leq t < M$.
Initialize: Choose random set of reward weights θ_1.
for $t = 1$ **to** M **do**
  Compute Q_{θ_t}, π_{θ_t}.
  $L = \sum_i w_i \sum_{(s,a)\in\xi} \log(\pi_{\theta_t}(s,a))$.
  $\theta_{t+1} \leftarrow \theta_t + \alpha_t \nabla L$.
end for
Output: Return $\theta_A = \theta_M$.

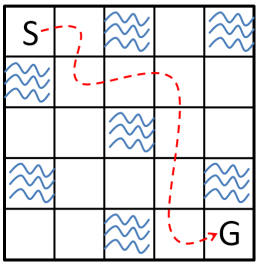

Figure 1. A single trajectory from start to goal.

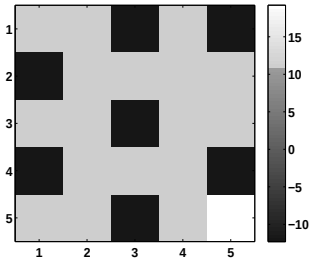

Figure 2. Reward function computed using MLIRL.

The Boltzmann exploration policy is $\pi_{\theta_A}(s,a) = e^{\beta Q_{\theta_A}(s,a)} / \sum_{a'} e^{\beta Q_{\theta_A}(s,a')}$. Under this policy, the log likelihood of the trajectories in D is $L(D|\theta) =$

$$\log \prod_{i=1}^{N} \prod_{(s,a)\in\xi_i} \pi_\theta(s,a)^{w_i} = \sum_{i=1}^{N} \sum_{(s,a)\in\xi_i} w_i \log \pi_\theta(s,a). \tag{1}$$

Here, w_i is a trajectory-specific weight encoding the frequency of trajectory i. MLIRL seeks $\theta_A = \text{argmax}_\theta L(D|\theta)$—the maximum likelihood solution. In our work, we optimized this function via gradient ascent (although we experimented with several other optimization approaches). These pieces come together in Algorithm 1.

It is open whether infinite-horizon value iteration with the Boltzmann operator will converge. In our finite-horizon setting, it is well-behaved and produces a well-defined answer, as illustrated later in this section and in our experiments (Section 5).

We illustrate the functioning of the MLIRL algorithm using the example shown in Figure 1. It depicts a 5×5 grid with puddles (indicated by wavy lines), a start state (S) and an absorbing goal state (G). The dashed line shows the path taken by an expert from S to G. The algorithm is now faced with the task of inferring the parameters of the expert's reward function θ_E using this trajectory. It appears that the expert is trying to reach the goal by taking the shortest path and at the same time avoid any intermediate puddles. The assignment of reward weights to the three features—ground, puddle, and goal—that makes this trajectory maximally likely is one that assigns the highest reward to the goal. (Otherwise, the expert would have preferred to travel somewhere else in the grid.) The probability of the observed path is further enhanced by assigning lower reward weights to puddles than to ground. Thus, although one explanation for the path is that it is one of a large number of possible shortest paths to the goal, the trajectory's probability is maximized by assuming the expert intentionally missed the puddles. The MLIRL-computed reward function is shown in Figure 2, which assigns high likelihood (0.1662) to the single demonstration trajectory.

One of the challenges of IRL is that, given an expert policy, there are an infinite number of reward functions for which that policy is optimal in the given MDP. Like several other IRL approaches, MLIRL addresses this issue by searching for a solution that not only explains why the observed behavior is optimal, but also by explaining why the other possible behaviors are suboptimal. In particular, by striving to assign high probability to the observed behavior, it implicitly assigns low probability to unobserved behavior.

4. Apprenticeship Learning about Multiple Intentions

The motivation for our work comes from settings like surveillance, in which observed actors are classified as "normal" or "threatening" depending on their behavior. We contend that a parsimonious classifier results by adopting a generative model of behavior—assume actors select actions that reflect their intentions and then categorize them based on their inferred intentions. For example, the behavior of people in a train station might differ according to their individual goals: some have the goal of traveling causing them to buy tickets and then go to their trains, while others may be picking up passengers causing them to wait in a visible area. We adopt the approach of using unsupervised clustering to identify the space of common intentions from a collection of examples, then mapping later examples to this set using Bayes rule.

Similar scenarios include decision making by automatic doors that infer when people intend to go through them, a home climate control system that sets temperature controls appropriately by reasoning

about the home owner's likely destinations when driving. A common theme in these applications is that unlabeled data—observations of experts with varying intentions—are much easier to come by than trajectories labeled with their underlying goal. We define our formal problem accordingly.

In the problem of apprenticeship learning about multiple intentions, we assume there exists a finite set of K or fewer intentions each represented by reward weights θ_k. The apprentice is provided with a set of $N > K$ trajectories $D = \{\xi_1, ..., \xi_N\}$. Each intention is represented by at least one element in this set and each trajectory is generated by an expert with one of the intentions. An additional trajectory ξ_E is the test trajectory—the apprentice's objective is to produce behavior π_A that obtains high reward with respect to θ_E, the reward weights that generated ξ_E. Many possible clustering algorithms could be applied to attack this problem. We show that Expectation-Maximization (EM) is a viable approach.

4.1. A Clustering Algorithm for Intentions

We adopt EM (Dempster et al., 1977) as a straightforward approach to computing a maximum likelihood model in a probabilistic setting in the face of missing data. The missing data in this case are the cluster labels—the mapping from trajectories to one of the intentions. We next derive an EM algorithm.

Define z_{ij} to be the probability that trajectory i belongs in cluster j. Let θ_j be the estimate of the reward weights for cluster j, and ρ_j to be the estimate for the prior probability of cluster j. Following the development in Bilmes (1997), we define $\Theta = (\rho_1, \ldots, \rho_K, \theta_1, \ldots, \theta_K)$ as the parameter vector we are searching for and Θ^t as the parameter vector at iteration t. Let $y_i = j$ if trajectory i came from following intention j and $y = (y_1, \ldots, y_N)$. We write $z_{ij}^t = \Pr(\xi_i|\theta_j^t)$, the probability, according to the parameters at iteration t, that trajectory i was generated by intention j.

The E step of EM simply computes

$$z_{ij}^t = \prod_{(s,a)\in\xi_i} \pi_{\theta_j^t}(s,a)\rho_j^t/Z, \tag{2}$$

where Z is the normalization factor.

To carry out the M step, we define the EM Q function (distinct from the MDP Q function):

$$\begin{aligned}
&Q(\Theta, \Theta^t) \\
&= \sum_y L(\Theta|D, y)) \Pr(y|D, \Theta^t) \\
&= \sum_y \sum_{i=1}^N \log(\rho_{y_i} \Pr(\xi_i|\theta_{y_i})) \prod_{i'=1}^N \Pr(y_{i'}|\xi_{i'}, \Theta^t) \\
&= \sum_{y_1} \cdots \sum_{y_N} \sum_{i=1}^N \sum_{l=1}^K \delta_{l=y_i} \log(\rho_l \Pr(\xi_i|\theta_l)) \\
&\quad \times \prod_{i'=1}^N \Pr(y_{i'}|\xi_{i'}, \Theta^t) \\
&= \sum_{l=1}^K \sum_{i=1}^N \log(\rho_l \Pr(\xi_i|\theta_l)) \sum_{y_1} \cdots \sum_{y_N} \delta_{l=y_i} \\
&\quad \times \prod_{i'=1}^N \Pr(y_{i'}|\xi_{i'}, \Theta^t) \\
&= \sum_{l=1}^K \sum_{i=1}^N \log(\rho_l \Pr(\xi_i|\theta_l)) z_{il}^t \\
&= \sum_{l=1}^K \sum_{i=1}^N \log(\rho_l) z_{il}^t + \sum_{l=1}^K \sum_{i=1}^N \log(\Pr(\xi_i|\theta_l)) z_{il}^t.
\end{aligned}$$

Algorithm 2 EM Trajectory Clustering

Input: Trajectories $\{\xi_1, ..., \xi_N\}$ (with varying intentions), number of clusters K.
Initialize: $\rho_1, \ldots, \rho_K, \theta_1, \ldots, \theta_K$ randomly.
repeat
  E Step: Compute $z_{ij} = \prod_{(s,a)\in\xi_i} \pi_{\theta_j}(s,a)\rho_j/Z$, where Z is the normalization factor.
  M step: For all l, $\rho_l = \sum_i z_{il}/N$. Compute θ_l via MLIRL on D with weight z_{ij} on trajectory ξ_i.
until target number of iterations completed.

In the M step, we need to pick Θ (ρ_l and θ_l) to maximize Equation 3. Since they are not interdependent, we can optimize them separately. Thus, we can set $\rho_l^{t+1} = \sum_i z_{il}^t/N$ and $\theta_l^{t+1} = \text{argmax}_\theta \sum_{i=1}^N z_{il}^t \log(\Pr(\xi_i|\theta_l))$. The key observation is that this second quantity is precisely the IRL log likelihood, as seen in Equation 1. That is, the M step demands that we find reward weights that make the observed data as likely as possible, which is precisely what MLIRL seeks to do. As a result, EM for learning about multiple intentions alternates between calculating probabilities via the E step (Equation 2) and performing IRL on the current clusters. Algorithm 2 pulls these pieces together. This EM approach is a fairly direct interpretation of the clustering problem we defined. It differs from much of the published work on learning from multiple experts, however, which starts with the assumption that all the experts have the same intentions (same reward function), but perhaps differ in their reliabil-

ity (Argall et al. 2009, Richardson & Domingos 2003).

4.2. Using Clusters for AL

The input of the EM method of the previous section is a set of trajectories D and a number of clusters K. The output is a set of K clusters. Associated with each cluster i are the reward weights θ_i, which induce a reward function r_{θ_i}, and a cluster prior ρ_i. Next, we consider how to carry out AL on a new trajectory ξ_E under the assumption that it comes from the same population as the trajectories in D.

By Bayes rule, $\Pr(\theta_i|\xi_E) = \Pr(\xi_E|\theta_i)\Pr(\theta_i)/\Pr(\xi_E)$. Here, $\Pr(\theta_i) = \rho_i$ and $\Pr(\xi_E|\theta_i)$ is easily computable (z in Section 4.1). The quantity $\Pr(\xi)$ is a simple normalization factor. Thus, the apprentice can derive a probability distribution over reward functions given a trajectory (Ziebart et al., 2008). How should it behave? Let $f^\pi(s,a)$ be the (weighted) fraction of the time policy π spends taking action a in state s. Then, with respect to reward function r, the value of policy π can be written $\sum_{s,a} f^\pi(s,a)r(s,a)$. We should choose the policy with the highest expected reward:

$$\begin{aligned}
&\operatorname*{argmax}_\pi \sum_i \Pr(\theta_i|\xi_E) \sum_{s,a} f^\pi(s,a) r_{\theta_i}(s,a) \\
&= \operatorname*{argmax}_\pi \sum_{s,a} f^\pi(s,a) \sum_i \Pr(\theta_i|\xi_E) r_{\theta_i}(s,a) \\
&= \operatorname*{argmax}_\pi \sum_{s,a} f^\pi(s,a) r'(s,a),
\end{aligned}$$

where $r'(s,a) = \sum_i \Pr(\theta_i|\xi_E) r_{\theta_i}(s,a)$. That is, the optimal policy for the apprentice is the one that maximizes the sum of the reward functions for the possible intentions, weighted by their likelihoods. This problem can be solved by computing the optimal policy of the MDP with this averaged reward function. Thus, to figure out how to act given an initial trajectory and collection of example trajectories, our approach is to cluster the examples, use Bayes rule to figure out the probability that the current trajectory belongs in each cluster, create a merged reward function by combining the cluster reward functions using the derived probabilities, and finally compute a policy for the merged reward function to decide how to behave.

5. Experiments

Our experiments were designed to compare the performance of the MLIRL (Section 3) and LPIRL (Section 2) algorithms with five existing IRL/AL approaches summarized in Section 2. We compare these seven approaches in several ways to assess (a) how well they perform apprenticeship learning and (b) how well they function in the setting of learning about multiple intentions. We first look at their performance in a grid world with a single expert (single intention), a domain where a few existing approaches (Abbeel & Ng 2004, Syed et al. 2008) have already been tested. Our second experiment, in a grid world with puddles, demonstrates the MLIRL algorithm as part of our EM approach (Section 4) to cluster trajectories from multiple intentions—each corresponding to a different reward function. Thirdly, we compare the performance of all the IRL/AL algorithms as part of the EM clustering approach in the simulated Highway Car domain (Abbeel & Ng 2004, Syed et al. 2008), an infinite-horizon domain with stochastic transitions.

Our experiments used implementations of the MLIRL, LPIRL, Maximum Entropy IRL, LPAL, MWAL, Projection, and Policy Matching algorithms. We obtained implementations from the original authors wherever possible.

5.1. Learning from a Single Expert

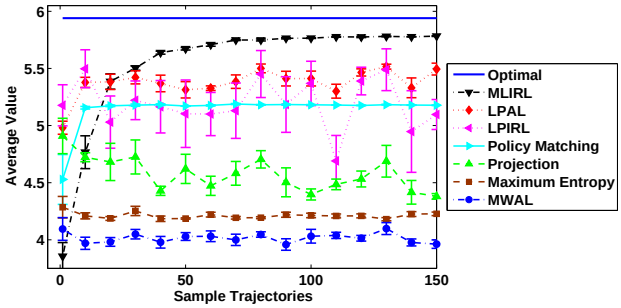

Figure 3. A plot of the average reward computed with increasing number of sample trajectories.

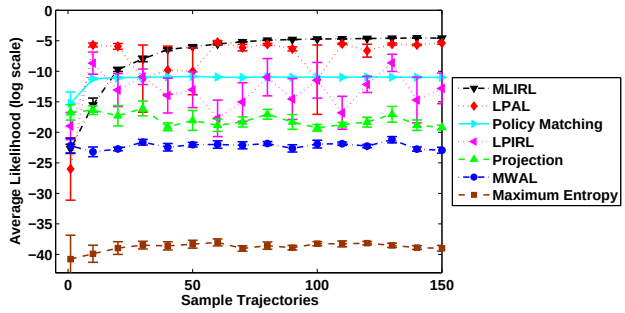

Figure 4. A plot of the average trajectory likelihood computed with increasing number of sample trajectories.

In this experiment, we tested the performance of each IRL/AL algorithm in a grid world environ-

ment similar to one used by Abbeel & Ng (2004) and Syed et al. (2008). We use a grid of size 16×16. Movement of the agent is possible in the four compass directions with each action having a 30% chance of causing a random transition. The grid is further subdivided into non-overlapping square regions, each of size 4×4. Using the same terminology as Abbeel & Ng (2004), we refer to the square regions as "macrocells". The partitioning of the grid results in a total of 16 macrocells. Every cell in the gridworld is characterized by a 16-dimensional feature vector ϕ indicating, using a 0 or 1, which macrocell it belongs to. A random weight vector is chosen such that the true reward function just encodes that some macrocells are more desirable than others. The optimal policy π^* is computed for the true reward function and the single expert trajectories are acquired by sampling π^*. To maintain consistency across the algorithms, the start state is drawn from a fixed distribution and the lengths of the trajectories are truncated to 60 steps.

Of particular interest is the ability of the seven IRL/AL algorithms to learn from a small amount of data. Thus, we illustrate the performance of the algorithms by varying the number of sample trajectories available for learning. Results are averaged over 5 repetitions and standard error bars are given. Note that in this and the following experiments, we use Boltzmann exploration polices to transform the reward functions computed by the IRL algorithms into policies when required.

Figure 3 shows the average reward accumulated by the policy computed by each algorithm as more trajectories are available for training. With 30 or more trajectories, MLIRL outperforms the other six. LPAL and LPIRL also perform well. An advantage of LPIRL over LPAL is that it returns a reward function, which makes it able to generalize over states that the expert has not visited during the demonstration trajectories. However, we observed that designing a policy indirectly through the reward function was less stable than optimizing the policy directly. It is interesting to note that MaxEnt lags behind in this setting. MaxEnt appears best suited for settings with very long demonstration trajectories, as opposed to the relatively short trajectories we used in this experiment.

Figure 4 shows that for the most part, in this dataset, the better an algorithm does at assigning high probability to the observed trajectories, the more likely it is to obtain higher rewards.

5.2. Learning about Multiple Intentions—Grid World with Puddles

In our second experiment, we test the ability of our proposed EM approach, described in Section 4, to accurately cluster trajectories associated with multiple intentions.

We make use of a 5×5 discrete grid world shown in Figure 5 (Left). The world contains a start state, a goal state and patches in the middle indicating puddles. Furthermore, the world is characterized by three feature vectors, one for the goal, one for the puddles and another for the remaining states. For added expressive power, we also included the negations of the features in the set thereby doubling the number of features to six.

We imagine data comes from two experts with different intentions. Expert 1 goes to the goal avoiding the puddles at all times and Expert 2 goes to the goal completely ignoring the puddles. Sample trajectories from these experts are shown in Figure 5 (Left). Trajectory T1 was generated by Expert 1, T2 and T3, by Expert 2. This experiment used a total of $N = 12$ sample trajectories of varying lengths, 5 from Expert 1, 7 from Expert 2. We initiated the EM algorithm by setting the value of K, the number of clusters, to 5 to allow some flexibility in clustering. We ran the clustering, then hand-identified the two experts. Figure 5 (Right) shows the algorithm's estimates that the three trajectories, T1, T2 and T3, belong to Expert 1. The EM approach was able to successfully cluster all of the 12 trajectories in the manner described above: the unambiguous trajectories were accurately assigned to their clusters and the ambiguous ones were "properly" assigned to multiple clusters. Since we set the value of $K = 5$, EM produced 5 clusters. On analyzing these clusters, we found that the algorithm produced 2 unique policies along with 3 copies. Thus, EM correctly extracted the preferences of the experts using the input sample trajectories.

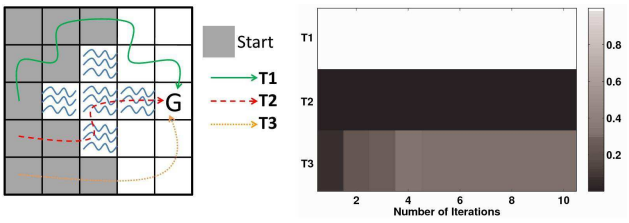

Figure 5. **Left:** Grid world showing the start states (grey), goal state (G), puddles and three sample trajectories. **Right:** Posterior probabilities of the three trajectories belonging to Expert 1.

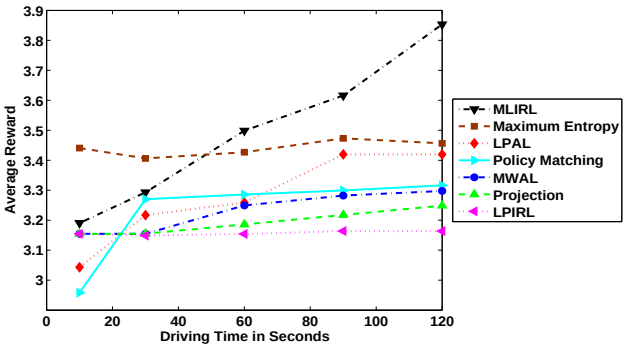

Figure 6. Average reward for **Student** trajectory for EM approach with varying IRL/AL components.

The probability values were computed at intermediate steps during the 10 iterations of the EM algorithm. After the 1st iteration, EM estimated that T1 belongs to Expert 1 with high probability and T2 belongs to Expert 1 with very low probability (implying that it therefore belongs to Expert 2). It is interesting to note here that EM estimated that trajectory T3 belongs to Expert 1 with probability 0.3. The uncertainty indicates that T3 could belong to either Expert 1 or Expert 2.

5.3. Learning about Multiple Intentions—Highway Car Domain

In our third experiment, we instantiated the EM algorithm in an infinite horizon domain with stochastic transitions, the simulated Highway Car domain (Abbeel & Ng 2004, Syed et al. 2008). This domain consists of a three-lane highway with an extra off-road lane on either side, a car driving at constant speed and a set of oncoming cars. Figure 7 shows a snapshot of the simulated highway car domain. The task is for the car to navigate through the busy highway using three actions: left, right and stay. The domain consists of three features: speed, number of collisions, number of off-road visits. Our experiment uses these three features along with their negations, making a total of six features. The transition dynamics are stochastic. Four different experts were used for this experiment: **Safe**: Avoids collisions and avoids going off-road. **Student**: Avoid collisions and does not mind going off-road. **Demolition**: Collides with every car and avoids going off-road. **Nasty**: Collides with every car and does not mind going off-road. Sample trajectories were collected from between ten seconds and two minutes of driving time from a human subject emulating each of the four experts. Using these sample trajectories, the EM approach performed clustering ($K = 6$) for 10 iterations. The trajectory used for evaluation ξ_E was generated by **Student**. The actions selected by the approach outlined in the previous section were evaluated according to the reward function from **Student** and plotted in Figure 6. Although our MLIRL algorithm is best suited to carry out the M step in the EM algorithm, any IRL can be used to approximately optimize the likelihood. Indeed, even AL algorithms can be used in the EM framework where a probabilistic policy takes the place of the reward weights as the hidden parameters. Thus, we instantiated each of the 7 AL/IRL approaches within the EM algorithm. It is interesting to note that maximum likelihood algorithms (MLIRL and MaxEnt) are the most effective for this taks. This time, MaxEnt was provided with longer trajectories, leading to an improvement in its performance compared to Section 5.1 and Figure 3.

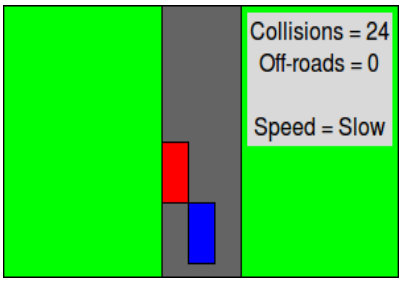

Figure 7. Simulated Highway Car.

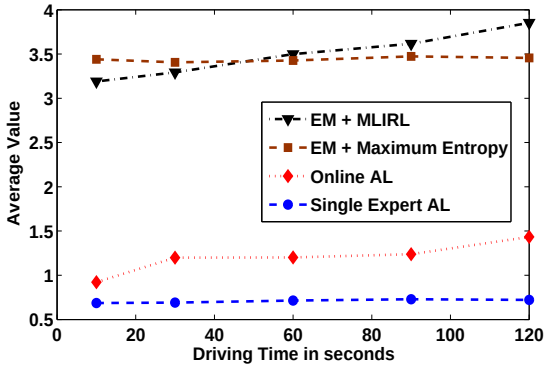

Figure 8. Value of the computed policy as a function of length of driving trajectories for three approaches to learning about multiple intentions.

Other approaches to learning about multiple intentions are possible. We compared the EM approach to AL (Section 4.2) with two other possibilities: (1) an AL learner that ignores all previous data in D and only learns from the current trajectory ξ_E (online AL), and (2) all the prior data D and current trajectory ξ_E are treated as a single input to the AL learner , commingling data generated from different

intentions (single expert AL). Figure 8 shows that the EM approach (with either MLIRL or Maximum Entropy) makes much better use of the available data and mixing data from multiple experts is undesirable.

6. Conclusion and Future Work

We defined an extension of inverse reinforcement learning and apprenticeship learning in which the learner is provided with unlabeled example trajectories generated from a number of possible reward functions. Using these examples as a kind of background knowledge, a learner can more quickly infer and optimize reward functions for novel trajectories.

Having shown that an EM clustering approach can successfully infer individual intentions from a collection of unlabeled trajectories, we next intend to pursue using these learned intentions to predict the behavior of and better interact with other agents in multiagent environments.

References

Abbeel, Pieter and Ng, Andrew Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2004.

Argall, Brenna, Browning, Brett, and Veloso, Manuela M. Automatic weight learning for multiple data sources when learning from demonstration. In *Proceedings of the International Conference on Robotics and Automation*, pp. 226–231, 2009.

Bilmes, Jeff A. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Computer Science Institute, 1997.

Branavan, S. R. K., Chen, Harr, Zettlemoyer, Luke S., and Barzilay, Regina. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 82–90, 2009.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

John, George H. When the best move isn't optimal: Q-learning with exploration. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 1464, Seattle, WA, 1994.

Lopes, Manuel, Melo, Francisco S., and Montesano, Luis. Active learning for reward estimation in inverse reinforcement learning. In *ECML/PKDD*, pp. 31–46, 2009.

Moore, Adam B., Todd, Michael T., and Conway, Andrew R. A. A computational model of moral judgment. Poster at Psychonomics Society Meeting, 2009.

Neu, Gergely and Szepesvári, Csaba. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the Conference of Uncertainty in Artificial Intelligence*, 2007.

Neu, Gergely and Szepesvári, Csaba. Training parsers by inverse reinforcement learning. *Machine Learning*, 77(2–3):303–337, 2009.

Puterman, Martin L. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

Ramachandran, Deepak and Amir, Eyal. Bayesian inverse reinforcement learning. In *Proceedings of IJCAI*, pp. 2586–2591, 2007.

Richardson, Matthew and Domingos, Pedro. Learning with knowledge from multiple experts. In *Proceedings of the International Conference on Machine Learning*, pp. 624–631, 2003.

Syed, Umar, Bowling, Michael, and Schapire, Robert E. Apprenticeship learning using linear programming. In *Proceedings of the International Conference on Machine Learning*, pp. 1032–1039, 2008.

Ziebart, Brian D., Maas, Andrew, Bagnell, J. Andrew, and Dey, Anind K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, pp. 1433–1438, 2008.