Model-Free IRL Using Maximum Likelihood Estimation

Vinamra Jain,[1] **Prashant Doshi,**[1] **Bikramjit Banerjee**[2]
[1]THINC Lab, Dept. of Computer Science; University of Georgia; Athens, GA 30602
[2]School of Computing Sciences & Computer Engineering; University of Southern Mississippi; Hattiesburg, MS 39406
pdoshi@cs.uga.edu, Bikramjit.Banerjee@usm.edu

Abstract

The problem of learning an expert's unknown reward function using a limited number of demonstrations recorded from the expert's behavior is investigated in the area of inverse reinforcement learning (IRL). To gain traction in this challenging and underconstrained problem, IRL methods predominantly represent the reward function of the expert as a linear combination of known features. Most of the existing IRL algorithms either assume the availability of a transition function or provide a complex and inefficient approach to learn it. In this paper, we present a model-free approach to IRL, which casts IRL in the maximum likelihood framework. We present modifications of the model-free Q-learning that replace its maximization to allow computing the gradient of the Q-function. We use gradient ascent to update the feature weights to maximize the likelihood of expert's trajectories. We demonstrate on two problem domains that our approach improves the likelihood compared to previous methods.

1 Introduction

Inverse reinforcement learning (IRL) (Russell 1998) is the problem of ascertaining an agent's preferences from observations of its behavior on a task. It inverts RL with its focus on learning the reward function given information about optimal action trajectories. IRL lends itself naturally to a robot learning the task from demonstrations by a human teacher (often called the expert) in controlled environments, and therefore finds application in robot learning from demonstration (Argall et al. 2009), imitation learning (Osa et al. 2018), and forming ad hoc collaborations (Trivedi and Doshi 2018).

Key model assumptions of popular IRL methods are that the expert's stochastic transition function is fully known to the learner as in IRL for apprenticeship learning (Abbeel and Ng 2004) and in Bayesian IRL (Ramachandran 2007). Alternately, the transition function is effectively deterministic and thus is easily approximated from the observed trajectories as in entropy maximization (Ziebart et al. 2008) with the assumption that transition randomness has a limited effect on the final behavior. The prior knowledge requirement is often difficult to satisfy in practice, for example, in scenarios where environmental noise is not random and it influences transitions significantly. An example of this is learning the driving styles of cars in the merging lane of a congested freeway; cars adjacent to the merging lane add noise. On the other hand, assuming that any transition error is inconsequential is a strong imposition in the context of robots.

In this paper, we introduce novel model-free algorithms to generalize IRL to problem domains where the transition function is not available. A previous method for model-free IRL (Boularias, Kober, and Peters 2011) casts the problem as one of relative entropy optimization while using importance sampling to avoid using the model. A more recent method (Uchibe 2018) casts IRL as a problem of estimating the density ratio between expert state transitions and a baseline one, and shows how this is related to the reward function in linearly solvable MDPs where the state transition probability is optimized. Both of these methods assume the availability of a secondary set of (non-expert) trajectories. In stark contrast, our approach does not require this secondary set of trajectories. We build on the model-based maximum likelihood IRL (MLIRL) (VRoman 2014) to remove its dependency on the transition function. Our method replaces the traditional Bellman update in MLIRL with two ways of performing model-free Q-learning that is modified to be differentiable. The first modification is to replace the *max* operator in the update with an averaging operator. The second modification replaces the max operator with a Boltzmann-weighted mean. As we show, the search for the most likely reward function can now be guided by the gradient of the likelihood function for the current feature weights.

We evaluate both our methods on two problem domains and compare their performances with the existing model-based MLIRL as well as the relative-entropy IRL technique. The first domain is the standard grid world problem that is commonly used to compare IRL methods. Our second problem facilitates real-world applications in the domain of autonomous vehicles. While self-driving cars have made tremendous progress in the last few years, specific road situations continue to challenge them. One of these is autonomously merging into a congested freeway. We evaluate our model-free IRL methods on real-world trajectories of cars driving in the merging lane of a congested freeway in order to learn the preferences of a class of drivers. IRL's relevance toward learning the driving styles of humans on a highway has been explored before for autonomous cars (Kuderer, Gulati, and Burgard 2015), but not in the context of freeway merging. In all of these evaluations, our methods meet and often exceed the performance of MLIRL, while easily improving on the performance of relative-entropy IRL. Between the two modifications, the averaging estimation is significantly better. These results make the maximum-likelihood IRL with the Q-averaging estimation as the new frontline method for model-free IRL.

2 Background

In this section, we briefly review the fundamentals of IRL and the maximum likelihood estimation approach to IRL.

2.1 Inverse Reinforcement Learning

Informally, IRL refers to both the problem and method by which an agent learns preferences of another agent that explain the latter's observed behavior (Russell 1998). Usually considered an "expert" in the task that it is performing, the observed agent, say I, is modeled as executing the optimal policy of a standard MDP defined as $\langle S_I, A_I, T_I, R_I \rangle$. The learning agent is assumed to perfectly know the parameters of the MDP except the reward function. Consequently, the learner's task may be viewed as finding a reward function under which the expert's observed behavior is optimal.

This problem in general is ill-posed because for any given behavior there are infinitely-many reward functions which align with the behavior. Abbeel and Ng (2004) present an algorithm that allows the expert I to provide task demonstrations instead of its policy. The reward function is modeled as a linear combination of K binary features, ϕ: $S_I \times A_I \rightarrow \{0, 1\}$, each of which maps a state from the set of states S_I and an action from the set of I's actions A_I to either a 0 or 1. Note that non-binary feature functions can always be converted into binary feature functions although there will be more of them. Throughout this paper, we assume that these features are known to, or selected by, the learner. The reward function for expert I is then defined as $R_I(s, a) = \sum_{k=1}^{K} \theta_k \cdot \phi_k(s, a)$, where θ_k are the *weights*. The learner's task is reduced to finding a vector of weights that complete the reward function, and subsequently the MDP such that the demonstrated behavior is optimal.

To assist in finding the weights, feature expectations are calculated for the expert's demonstration and compared to those of possible trajectories (Ziebart et al. 2008). A demonstration is provided as one or more *trajectories*, which are a sequence of length-T state-action pairs, $(\langle s, a \rangle^1, \langle s, a \rangle^2, \ldots \langle s, a \rangle^T)$, corresponding to an observation of the expert's behavior across T time steps. Feature expectations of the expert are averages over all observed trajectories, $\hat{\phi}_k = \frac{1}{|X|} \sum_{x \in X} \sum_{\langle s,a \rangle \in x} \phi_k(s, a)$, where x is a trajectory in the set of all observed trajectories, X.

Given a set of reward weights, the expert's MDP is completed and solved optimally to produce π_I^*. The difference $\hat{\phi} - \phi^{\pi_I^*}$ provides a gradient with respect to the reward weights for a numerical solver. To resolve the degeneracy of this problem, Abbeel and Ng (2004) maximize the margin between the value of the optimal policy and the next best policy. The resulting program may be solved with a quadratic program solver such as a support vector machine.

2.2 MLE for Model-based IRL

Babes-VRoman et al. (2014) showed how IRL could be formulated as a maximum likelihood estimation (MLE) problem.

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \; L(\boldsymbol{\theta})$$

where $L(\boldsymbol{\theta})$ is the log-likelihood of the demonstration trajectories in X. In other words,

$$L(\boldsymbol{\theta}) = \log Pr(X; \boldsymbol{\theta}) \tag{1}$$

As the trajectories in X are conditionally independent of each other given $\boldsymbol{\theta}$, we can decompose $Pr(X; \boldsymbol{\theta})$ as:

$$P(X; \boldsymbol{\theta}) = \prod_{x \in X} P(x; \boldsymbol{\theta}) \tag{2}$$

Because x denotes a trajectory as shown in Section 2.1, we may further decompose Eq. 2 as:

$$Pr(X; \boldsymbol{\theta}) = \prod_{x \in X} \Bigg(Pr(s^1)\; Pr(a^1 | s^1; \boldsymbol{\theta}) \prod_{i=1}^{T-1} Pr(s^{i+1} | s^i, a^i) \times\; Pr(a^{i+1} | s^{i+1}; \boldsymbol{\theta}) \Bigg). \tag{3}$$

The term $Pr(a^{i+1} | s^{i+1}; \boldsymbol{\theta})$ is given by the agent's policy parameterized by $\boldsymbol{\theta}$, and $Pr(s^{i+1} | s^i, a^i)$ is given by the transition function. Therefore,

$$Pr(X; \boldsymbol{\theta}) = \prod_{x \in X} \Bigg(Pr(s^1)\; \pi_{\boldsymbol{\theta}}(s^1, a^1) \prod_{i=1}^{T-1} T(s^i, a^i, s^{i+1}) \times\; \pi_{\boldsymbol{\theta}}(s^{i+1}, a^{i+1}) \Bigg).$$

Taking log on both sides we get the log likelihood as,

$$L(\boldsymbol{\theta}) = \sum_{x \in X} \Bigg(log\; Pr(s^1) + \sum_{i=0}^{T-1} log\; \pi_{\boldsymbol{\theta}}(s^{i+1}, a^{i+1}) + \sum_{i=1}^{T-1} log\; T(s^i, a^i, s^{i+1}) \Bigg). \tag{4}$$

The transition function in the log likelihood above is usually given. Therefore, it is common practice to exclude it from the computation while comparing the performance of various methods (Ramachandran 2007; VRoman 2014).

$$L(\boldsymbol{\theta}) = \sum_{x \in X} \Bigg(log\; Pr(s^1) + \sum_{i=0}^{T-1} log\; \pi_{\boldsymbol{\theta}}(s^{i+1}, a^{i+1}) \Bigg). \tag{5}$$

The policy is commonly modeled using the parameterized Boltzmann exploration (Ramachandran 2007; VRoman 2014) as follows,

$$\pi_{\boldsymbol{\theta}}(s, a) = \frac{e^{\beta\; Q_{\boldsymbol{\theta}}(s,a)}}{\sum_{a' \in A_I} e^{\beta\; Q_{\boldsymbol{\theta}}(s,a')}} = \frac{e^{\beta\; Q_{\boldsymbol{\theta}}(s,a)}}{Z_{\boldsymbol{\theta}}(s)}. \tag{6}$$

As β approaches infinity, the Boltzmann exploration assigns a probability approaching 1 to the action with the largest Q-value. Given this exploration policy, the action-value (Q-) function becomes,

$$Q_{\boldsymbol{\theta}}(s,a) = R_{\boldsymbol{\theta}}(s,a) + \gamma \sum_{s' \in S_I} T(s,a,s') \times \sum_{a' \in A_I} Q_{\boldsymbol{\theta}}(s',a')\, \pi_{\boldsymbol{\theta}}(s',a') \quad (7)$$

where $\pi_{\boldsymbol{\theta}}(s',a')$ is as defined in Eq. 6.

To obtain the MLE, we may partially differentiate the log likelihood of Eq. 5:

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{x \in X} \sum_{i=1}^{T} \frac{1}{\pi_{\boldsymbol{\theta}}(s^i,a^i)} \frac{\partial \pi_{\boldsymbol{\theta}}(s^i,a^i)}{\partial \boldsymbol{\theta}} \quad (8)$$

In the context of Eq. 6, differentiating the policy above involves differentiating the Q-function w.r.t. $\boldsymbol{\theta}$. Although the derivative of the Q-function requires differentiating the policy in turn, Babes-VRoman et al. (VRoman 2014) show how this recursive differentiation may be performed and the feature function weights are updated using the standard gradient descent until the weights all converge. Notice that the computation of the Q-function involves the transition function of the expert I. *Consequently, the approach given by Babes-VRoman et al. is a model-based MLE for IRL.*

3 Model-Free MLE for IRL

We are motivated to study model-free IRL because it relaxes a debilitating prior knowledge requirement by most IRL methods. Specifically, they assume that the learner knows the dynamics of the expert as modeled by its stochastic transition function as in apprenticeship learning (Abbeel and Ng 2004) and in Bayesian IRL (Ramachandran 2007). Alternately, the transition function is assumed to be effectively deterministic and thus is easily approximated from the observed trajectories as in entropy maximization (Ziebart et al. 2008) with the assumption that transition randomness has a limited effect on the final behavior. The prior knowledge requirement is often difficult to satisfy in practice, for example, in scenarios that are not cooperative. Alternately, the supposed impotency of transition errors is a strong assumption in the context of robots possessing noisy actuators.

Two observations enable us to adapt the MLE of Section 2.2 for model-free IRL. First, notice that the right-hand side of the gradient of the log likelihood in Eq. 5 does not involve the transition function. Second, if we replace the Q-function computation in Eq. 7 with a way of computing it that does not involve the transition function, then the gradient can be entirely computed without knowledge of the transition function.

We begin by presenting a method to estimate the Q-value without knowledge of the transition function.

3.1 Differentiable learning

A straightforward model-free estimation of the expert's $Q_{\boldsymbol{\theta}}(s,a)$ is to use Watkin's off-policy Q-learning (Watkins and Dayan 1992).

$$Q_{\boldsymbol{\theta}}(s,a) \leftarrow (1-\alpha)\, Q_{\boldsymbol{\theta}}(s,a) + \alpha\big(R_{\boldsymbol{\theta}}(s,a) + \gamma \max_{a' \in A_I} Q_{\boldsymbol{\theta}}(s',a')\big)$$

where α is the learning schedule, γ is the discount factor, s' is the next state in the trajectory x, and $R_{\boldsymbol{\theta}}(s,a)$ is the reward for taking action a in state s. Recall that the reward function is defined as $R_{\boldsymbol{\theta}}(s,a) = \sum_{k=1}^{K} \theta_k\ \phi_k(s,a)$. *However, the presence of the* max *operator in the estimation above makes the Q-function discontinuous and therefore not differentiable (Asadi and Littman 2017).*

Consequently, as a first step, we seek a simple modification of the Q-learning equation to make it differentiable. We propose two estimations of the Q-function, first of which replaces the max operator with an averaging operator.

$$Q_{\boldsymbol{\theta}}(s,a) \leftarrow (1-\alpha)\, Q_{\boldsymbol{\theta}}(s,a) + \alpha\big(R_{\boldsymbol{\theta}}(s,a) + \gamma \frac{\sum_{a' \in A_I} Q_{\boldsymbol{\theta}}(s',a')}{|A_I|}\big) \quad (9)$$

Notice that the averaging operation simply replaces the maximal Q-value for the next state s' with the Q-value averaged over all expert's actions for the next state. We refer to this version of Q-learning as Q-averaging. This use of the averaging operation for IRL is novel to the best of our knowledge.

The second estimation replaces the max operator with the Boltzmann-weighted mean as given below:

$$Q_{\boldsymbol{\theta}}(s,a) \leftarrow (1-\alpha)\, Q_{\boldsymbol{\theta}}(s,a) + \alpha\big(R_{\boldsymbol{\theta}}(s,a) + \gamma \sum_{a' \in A_I} \pi_{\boldsymbol{\theta}}(s',a') Q_{\boldsymbol{\theta}}(s',a')\big) \quad (10)$$

Here, $\pi_{\boldsymbol{\theta}}(s',a')$ is the Boltzmann softmax as defined in Eq. 6. We refer to this version of Q-learning as Q-softmax.

For both these Q-learning methods, we will continue to use the Boltzmann policy for the action distribution. This introduces a discrepancy between how the Q-value is estimated in Q-averaging – essentially selecting one of the expert actions at s' at random – and the action distribution. However, not all is lost with this approximation. In addition to being easily differentiable, the Q-averaging update is a non-expansion given fixed $\boldsymbol{\theta}$ ensuring convergence to a unique fixed point (Asadi and Littman 2017). As such, it serves as a useful method. In comparison, Q-softmax is differentiable and approximates the traditional "max" operator for selecting an action at s' as $\beta \to \infty$. However, it lacks the non-expansion property and therefore may not always converge. [1]

[1] Recently, Asadi and Littman (2017) pointed out the convergence limitation of using the Boltzmann policy for exploration in the context of SARSA. It's likely that Q-learning also suffers from a similar fixed-point issue, and the alternative mellowmax may be used instead, but this needs further investigation.

3.2 Gradient Computation

Our approach is to obtain the gradient vector of the log likelihood, which is defined as:

$$\nabla L(\boldsymbol{\theta}) = \left\langle \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_1}, \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_2}, \ldots, \frac{\partial L(\boldsymbol{\theta})}{\partial \theta_k} \right\rangle$$

We may then use the appropriate component of the gradient vector to update the parameters $\boldsymbol{\theta}$ as a step of gradient ascent.

$$\forall i, \quad \theta_k^{t+1} = \theta_k^t + \alpha_t \nabla L_k(\boldsymbol{\theta})$$

where, α_t is the dynamic step size in the t^{th} iteration and $\nabla L_k(\boldsymbol{\theta})$ denotes the component of the gradient corresponding to the differentiation by θ_k.

The derivative of the log likelihood in Eq. 8 requires computing the gradient of the policy. This is obtained as,

$$\frac{\partial \pi_{\boldsymbol{\theta}}(s^i, a^i)}{\partial \boldsymbol{\theta}} = \frac{1}{Z^2_{\boldsymbol{\theta}}(s)} \left[Z_{\boldsymbol{\theta}}(s)\, \beta\, e^{\beta\, Q_{\boldsymbol{\theta}}(s^i, a^i)}\, \frac{\partial Q_{\boldsymbol{\theta}}(s^i, a^i)}{\partial \boldsymbol{\theta}} \right. \\ \left. -\, e^{\beta\, Q_{\boldsymbol{\theta}}(s^i, a^i)}\, \frac{\partial Z_{\boldsymbol{\theta}}(s)}{\partial \boldsymbol{\theta}} \right]. \tag{11}$$

Q-averaging The above requires the gradient of the Q-function, whose computation proceeds as follows for Q-averaging. Equation 9 is iterated until (near-)convergence. For the t^{th} iteration, we may write it as:

$$Q^t_{\boldsymbol{\theta}}(s,a) = (1-\alpha)\, Q^{t-1}_{\boldsymbol{\theta}}(s,a) + \alpha\big(R_{\boldsymbol{\theta}}(s,a) + \\ \frac{\gamma}{|A_I|} \sum_{a'} Q^{t-1}_{\boldsymbol{\theta}}(s',a')\big) \tag{12}$$

and its gradient becomes,

$$\frac{\partial}{\partial \theta_k} Q^t_{\boldsymbol{\theta}}(s,a) = (1-\alpha)\, \frac{\partial}{\partial \theta_k} Q^{t-1}_{\boldsymbol{\theta}}(s,a) + \alpha\big(\phi_k(s,a) + \\ \frac{\gamma}{|A_I|} \sum_{a'} \frac{\partial}{\partial \theta_k} Q^{t-1}_{\boldsymbol{\theta}}(s',a')\big) \tag{13}$$

As the Q-function at the first time step is the reward function, we get,

$$\frac{\partial}{\partial \theta_k} Q^1_{\boldsymbol{\theta}}(s,a) = \frac{\partial}{\partial \theta_k} R_{\boldsymbol{\theta}}(s,a) = \phi_k(s,a) \tag{14}$$

Recall that Q-averaging is a non-expansion given $\boldsymbol{\theta}$ and differentiable. Consequently, iterating over Eq. 13 with Eq. 14 as the base case will cause the gradient to converge to a fixed point. As such, the gradient of the model-free Q-function is available to us.

Next, we need the gradient of $Z_{\boldsymbol{\theta}}$ w.r.t. θ_k, which uses the converged gradient of the Q-function obtained before.

$$\frac{\partial Z_{\boldsymbol{\theta}}(s)}{\partial \theta_k} = \beta \sum_{a \in A_I} e^{\beta\, Q_{\boldsymbol{\theta}}(s,a)}\, \frac{\partial\, Q_{\boldsymbol{\theta}}(s,a)}{\partial \theta_k} \\ = \beta \sum_{a \in A_I} e^{\beta\, Q_{\boldsymbol{\theta}}(s,a)} \Bigg((1-\alpha)\, \frac{\partial Q_{\boldsymbol{\theta}}(s,a)}{\partial \theta_k} + \\ \alpha\big(\phi_k(s,a) + \frac{\gamma}{|A_I|} \sum_{a'} \frac{\partial Q_{\boldsymbol{\theta}}(s',a')}{\partial \theta_k}\big) \Bigg)$$

Algorithm 1 Model-free ML IRL using Q-averaging

```
Require: MDP$\backslash\{R,T\}$: $\langle S, A_I, \gamma \rangle$, Features $\Phi$ : $\{\phi_1, \phi_2, ..., \phi_K\}$, Trajectories $X$: $(x_1, x_2, \ldots, x_N)$, $\alpha$, $\alpha_n$, $\epsilon$
1: Initialize randomly $\boldsymbol{\theta}$ : $\langle \theta_1, \theta_2, \ldots, \theta_k \rangle$
2: Initialize local variables $L' \leftarrow 0$, $n \leftarrow 0$, $t \leftarrow 0$
3: Initialize $Q_{\boldsymbol{\theta}}(s,a) \leftarrow 0$ for all $s \in S$, $a \in A_I$
4: Initialize stochastic policy using Eq. 6
5: repeat
6:     $n \leftarrow n + 1$
7:     $L \leftarrow L'$
8:     $R_{\boldsymbol{\theta}}(s,a) = \sum_{k=1}^{K} \theta_k\ \phi_k(s,a)$
9:     repeat
10:        $t \leftarrow t + 1$
11:        Update $Q^t_{\boldsymbol{\theta}}(s,a)$ (Eq. 12) for all $(s,a)$ using Boltzmann exploration
12:        Update $\frac{\partial Q^t_{\boldsymbol{\theta}}(s,a)}{\partial \theta_k}$ using Eq. 13 for all $(s,a)$ and $\theta_k$
13:        Use Eq. 6 and updated Q-values to update policy
14:    until $|Q^t_{\boldsymbol{\theta}}(s,a) - Q^{t-1}_{\boldsymbol{\theta}}(s,a)| < \epsilon(1-\gamma)/\gamma$
15:    $Q^*_{\boldsymbol{\theta}}(s,a) \leftarrow Q^t_{\boldsymbol{\theta}}(s,a)$ for all $(s,a)$
16:    $\pi_{\boldsymbol{\theta}}(s,a) \leftarrow \frac{e^{\beta Q^*_{\boldsymbol{\theta}}(s,a)}}{\sum_{a'} e^{\beta Q^*_{\boldsymbol{\theta}}(s,a')}}$ for all $(s,a)$
17:    Evaluate likelihood $L(\boldsymbol{\theta})$ using Eq. 5
18:    Obtain $\nabla L_k(\boldsymbol{\theta})$ (Eq. 8) using $\frac{\partial Q^t_{\boldsymbol{\theta}}}{\partial \theta_k}$ in Eq. 11
19:    $L' \leftarrow L(\boldsymbol{\theta})$
20:    for all $\theta_k \in \boldsymbol{\theta}$ do
21:        $\theta_k \leftarrow \theta_k + \alpha_n \nabla L_k(\boldsymbol{\theta})$
22:    $\delta = |L' - L|$
23: until $\delta < \epsilon(1-\gamma)/\gamma$
24: return $\boldsymbol{\theta}$
```

We summarize the method for model-free IRL using Q-averaging in Algorithm 1. We initialize the feature weight vector and the Q-table with random values and zeroes respectively, and compute the initial policy (lines 1-4). For the current $\boldsymbol{\theta}$, we obtain the nearly converged Q-values and its gradient in an iterative manner. The model-free Q-averaging uses the Boltzmann policy (kept updated) for exploration (lines 9-14). These are then utilized to obtain the current log likelihood for the expert's trajectories and its gradient. Lines 20-22 perform gradient ascent to update the feature weights, and these steps are repeated until approximate convergence.

Q-softmax Let us proceed in a similar manner as in Q-averaging, and write out the iterative version of the Q-function.

$$Q^t_{\boldsymbol{\theta}}(s,a) \leftarrow (1-\alpha)\, Q^{t-1}_{\boldsymbol{\theta}}(s,a) + \alpha\big(R_{\boldsymbol{\theta}}(s,a) + \\ \gamma \sum_{a' \in A_I} \pi^{t-1}_{\boldsymbol{\theta}}(s',a')\, Q^{t-1}_{\boldsymbol{\theta}}(s',a')\big) \tag{15}$$

where for any iteration t the probability $\pi^t_{\boldsymbol{\theta}}(s,a)$ is obtained from the t^{th} iteration of the Q-function, $\pi^t_{\boldsymbol{\theta}}(s,a) = \frac{e^{\beta\ Q^t_{\boldsymbol{\theta}}(s,a)}}{\sum_{a \in A_I} e^{\beta\ Q^t_{\boldsymbol{\theta}}(s,a)}} = \frac{e^{\beta\ Q^t_{\boldsymbol{\theta}}(s,a)}}{Z^t_{\boldsymbol{\theta}}(s)}$. Thus, convergence of Q-softmax additionally requires iterations over the Boltzmann policy.

Differentiating Eq. 15 w.r.t. θ_k we get,

$$\frac{\partial}{\partial\theta_k} Q^t_{\boldsymbol{\theta}}(s,a) = (1-\alpha)\,\frac{\partial}{\partial\theta_k} Q^{t-1}_{\boldsymbol{\theta}}(s,a) + \alpha\Bigg(\phi_k(s,a) + \gamma\sum_{a'} \pi^{t-1}_{\boldsymbol{\theta}}(s',a')\frac{\partial}{\partial\theta_k} Q^{t-1}_{\boldsymbol{\theta}}(s',a') + Q^{t-1}_{\boldsymbol{\theta}}(s',a') \times \frac{\partial}{\partial\theta_k}\pi^{t-1}_{\boldsymbol{\theta}}(s',a')\Bigg) \tag{16}$$

Notice that Eq. 16 also involves the gradient of the policy in contrast to the gradient of Q-averaging. This is obtained analogously to Eq. 11 with the difference that $Q_{\boldsymbol{\theta}}(s,a)$ and its derivative is replaced with $Q^{t-1}_{\boldsymbol{\theta}}(s,a)$ and its derivative, respectively, both of which are available from the previous iteration. With Q-softmax not guaranteed to be a non-expansion, its gradient may not be a non-expansion either in some cases. Finally, we obtain the gradient of $Z^t_{\boldsymbol{\theta}}$ as,

$$\frac{\partial Z^t_{\boldsymbol{\theta}}(s)}{\partial\theta_k} = \beta \sum_{a\in A_I} e^{\beta\, Q^t_{\boldsymbol{\theta}}(s,a)}\, \frac{\partial\, Q^t_{\boldsymbol{\theta}}(s,a)}{\partial\theta_k}$$

where $\frac{\partial\, Q^t_{\boldsymbol{\theta}}(s,a)}{\partial\theta_k}$ is computed as given in Eq. 16.

The algorithm for model-free IRL with MLE using Q-softmax is analogous to Algorithm 1 with just a few changes. In particular, the Q-value in line 11 is updated using Eq. 15 and its gradient in the next line is computed using Eq. 16. The remaining steps of the algorithm including the likelihood computation and the iterative update of feature weights remain unchanged.

4 Experiments

We evaluated our model-free IRL algorithm using two application domains. Multiple experiments were performed on two domains: the grid world domain, a small toy problem, and the freeway merging domain, a real-world problem with a considerably larger state space than the first. We also compared our results with the model-based MLIRL approach and another model-free approach, relative entropy IRL (REIRL). We used Monica Babes-VRoman's code for MLIRL, and REIRL code from `https://github.com/aravindsiv/irl-lab`. Our code for both presented methods is also publicly available at `https://github.com/RAILUSM/Model-free-IRL`.

4.1 Gridworld Domain

Our first domain is a simple grid of size 5×5. An agent can navigate this domain using 4 directional movements. Figure 1 depicts the graphical user interface for the grid world environment. The gray colored circle is the agent. The five different color grids signify the unique location features.

The MDP model for the grid world environment includes 25 states, 4 actions, and the discount factor of 0.99. The task is to reach the nearest corner (goal) cell from any non-blue starting location, while avoiding the blue cells. We used two variants of this domain—one with no transition noise, and another where the agent has a 10% chance of slipping laterally relative to the intended direction of motion. We used the Boltzmann temperature (β) as 0.01 for all methods.

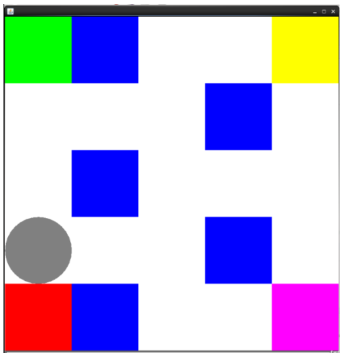

Figure 1: The graphical user interface for grid world environment. The gray circle is the agent exploring the 5×5 grid. Each different color in the grid represents a distinct (Boolean) feature of the state. The agent tries to learn the cost associated with each color using the expert's trajectories.

We recorded 20 trajectories in each variant, by moving the agent across the grid following an optimal policy for the above tasks. We then removed any repeated trajectories, leading to a set of 12 distinct expert trajectories in 0-noise variant, and 14 distinct expert trajectories in 10%-noise variant. These trajectories were then used by the agent to learn the reward weights associated with each grid-color (feature). The weights were initialized to random values in $[-1, 1]$ for all methods. Accurate transition matrices–with or without noise–were additionally available to the model based method (MLIRL). While MLIRL and our methods are based on the log-likelihood of data, REIRL does not use this metric, and instead runs for a given number of iterations. We allowed it 5×10^5 iterations in the no-noise variant, and 10^6 iterations in the 10% noise variant, to ensure convergence of its objective value, then used the learned feature weights to construct a reward function, and optimally solved the resulting MDP to get the final log-likelihood. REIRL also requires two other adjustments to the data: (1) it assumes all trajectories are of the same length, for which we prolonged all shorter trajectories by copying the last (s, a) pair to match the length of the longest trajectory; (2) it needs an additional set of baseline trajectories that are generated by a random policy. We conducted 30 independent trials for each setting.

Tables 1 and 2 show the mean with standard deviations of learned feature weights and corresponding final log-likelihood values achieved using model-free and model-based IRL methods in the two grid world variants. It is interesting that while MLIRL (correctly) learns negative weights for the blue feature avoided in the expert trajectories due to the availability of the true transition matrix, REIRL learns negative weights for this feature due to the availability of a set of baseline trajectories where the blue feature is visited by a random policy, and it optimizes entropy of expert trajectories relative to these baseline trajectories. By comparison, our methods have access to neither the true transition matrix nor a set of baseline trajectories. Consequently they learn to associate the lowest weight to this feature in

Method	Learned feature weights (feature 5 is blue)					Final log-likelihood
	θ_1	θ_2	θ_3	θ_4	θ_5	
MLIRL	43.04 ± 14.83	33.31 ± 11.35	54.23 ± 18.23	53.93 ± 18.00	-49.09 ± 16.80	-36.50 ± 3.12
Q-Avg	152.33 ± 60.81	92.06 ± 36.91	1526.06 ± 608.96	151.51 ± 60.49	0.03 ± 0.56	$-\mathbf{29.25 \pm 6.59}$
Q-SM	17.92 ± 14.09	11.96 ± 9.15	109.88 ± 84.96	17.87 ± 14.07	0.02 ± 0.60	-40.39 ± 4.15
REIRL	0.87 ± 0.00	28.27 ± 0.60	40.43 ± 0.58	44.62 ± 0.55	-3.10 ± 0.05	-43.58 ± 0.03

Table 1: Comparison of learned feature weights and corresponding final log-likelihood values of trajectories for grid world domain with no transition noise, from various algorithms.

Method	Learned feature weights (feature 5 is blue)					Final log-likelihood
	θ_1	θ_2	θ_3	θ_4	θ_5	
MLIRL	94.00 ± 31.88	86.91 ± 29.56	93.26 ± 31.67	95.35 ± 32.11	-71.57 ± 23.78	-45.05 ± 9.07
Q-Avg	227.72 ± 1.41	89.83 ± 0.68	157.59 ± 1.18	3368.95 ± 0.24	151.43 ± 0.67	$\mathbf{-40.49 \pm 0.04}$
Q-SM	30.37 ± 5.22	10.98 ± 1.69	18.40 ± 2.91	225.20 ± 0.21	11.33 ± 0.80	-57.35 ± 0.30
REIRL	0.58 ± 0.09	0.13 ± 0.05	1.96 ± 0.46	0.31 ± 0.06	-0.50 ± 0.40	-72.11 ± 0.01

Table 2: Comparison of learned feature weights and corresponding final log-likelihood values of trajectories for grid world domain with 10% transition noise, from various algorithms.

the no-noise variant (Table 1). With transition noise, however, the blue feature sometimes appears in the expert trajectories due to slippage, but without a true transition matrix or baseline trajectories our methods have no way to realize that this feature is undesirable to the expert. Hence we see higher weights associated with this feature for both Q-averaging and Q-softmax in Table 2. Notwithstanding this, the final log-likelihoods of our methods are competitive, in particular, Q-averaging achieves the highest log-likelihood in both cases. The relatively large variances for some indicate that different trials converged to different local optima. This is a consequence of the termination criterion (line 23 of Algorithm 1) which was also applied to MLIRL, and this may explain why despite being model-based it failed to find solutions with higher log-likelihoods on the average.

A Wilcoxon signed rank test of significance between the sets of final log-likelihoods of different methods compared to MLIRL found that for the no-noise case, the Q-averaging method achieves significantly higher log-likelihoods than MLIRL at the 99% confidence level (p-value=0.00194), while MLIRL achieves significantly higher log-likelihoods compared to Q-softmax and REIRL (p-value 0 in both cases). In the 10% noise gridworld, the same relative advantages hold with p-values 0, 0, and 0 respectively.

Instead of a comparison of precise run times (because the methods were implemented in different programming languages), we note that all methods ran in a comparable amount of time, each under 15 minutes per trial on a Linux workstation with Intel core i7 CPU each core is 3.6GHz and 16GB of main memory.

4.2 Freeway Merging Domain

The freeway merging domain is a real-world problem faced by autonomous vehicles in making decisions about when to merge, keeping in consideration the stochastic behavior of human drivers on the freeway. Solving the freeway merging problem requires modeling the relevant traffic. Here, we model this problem using a sufficient A-B-C model as shown in Fig. 2. Vehicle in role B is an autonomous vehicle that is about to merge onto the freeway. A is the vehicle on rightmost lane of the freeway but relatively behind B. C is also the vehicle on rightmost lane of the freeway but relatively ahead of B. The problem is that vehicle B must merge between A and C but the preferences of A's driver about allowing B to merge ahead are not known. The objective of IRL in this context is to model the preferences of A's driving model as it detects B.

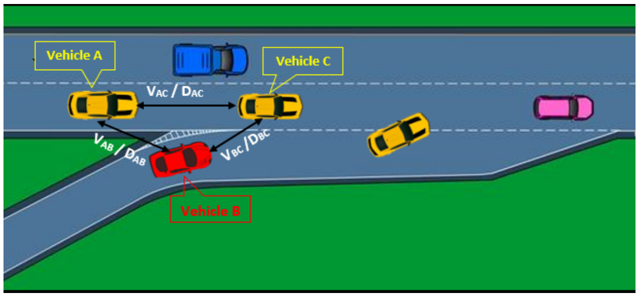

Figure 2: Detailed A-B-C model representing the freeway merging problem. B is an autonomous vehicle about to merge onto the freeway. Relative variables like velocity and distance between any two vehicles play crucial roles in defining the state of vehicle in role A.

The Next Generation SIMulation (NGSIM) (Alexiadis, Colyar, and Halkias 2007) program was launched by United States Department of Transportation (US DOT) Federal Highway Administration (FHWA)'s Traffic Analysis Tools Program to develop algorithms in support of traffic simulation, with a primary focus on microscopic modeling. The researchers for the NGSIM program collected detailed vehicle trajectory data on eastbound I-80 in the San Francisco Bay area in Emeryville, CA. Seven synchronized digital video cameras were mounted on the top of a 30-story building adjacent to the freeway to record vehicle passing through over approximately 500 meters (1,640 feet) in length. The study included all 6 freeway lanes and an additional on-ramp merging onto the freeway. The collected dataset is well documented with necessary meta-data. Out of the 260 trajectories in the full dataset that encompasses a wide variety of driving preferences, we isolated 12 trajectories where A exhibited risky behavior, accelerating frequently. This experiment is based on these 12 trajectories.

We define the state space using the following 5 state variables:

1. d_{AC}: Distance between vehicles A and C (in $ft.$)
2. d_{AB}: Distance between vehicles A and B (in $ft.$)
3. v_{AC}: Velocity of A relative to C (in $ft./sec$)
4. v_{AB}: Velocity of A relative to B (in $ft./sec$)
5. Vehicle type of vehicle B (truck or not).

We discretized the first 4 state variables into 5 intervals and the fifth variable is binary in nature. This yields 1,250 states of vehicle A.

The instantaneous acceleration values are modeled as actions of the driver. We discretized the acceleration (in $ft./sec^2$.) into five intervals and named them as the following actions:

1. High Brake: $-11.20 \leq acc \leq -4.80$
2. Low Brake: $-4.79 \leq acc \leq -0.60$
3. Zero Acceleration: $-0.59 \leq acc \leq 0.59$
4. Low Acceleration: $0.60 \leq acc \leq 4.79$
5. High Acceleration: $4.80 \leq acc \leq 11.2$

We accounted for 3 binary features in the reward function. Any feature is considered active with the value 1 and inactive when the value is 0.

Safe (ϕ_1): This feature is inactive only when $d_{AC} < 35\ ft.$ and $acc \geq 0.6\ ft./sec^2$, i.e. distance from the preceding vehicle is less than 35 ft and the vehicle is accelerating. This feature signifies the preference of being safe when active.

Time to travel (ϕ_2): This feature is active when $acc > -0.6\ ft./sec^2$, i.e. either the vehicle is accelerating or moving with a constant speed. This feature signifies the importance of time to reach the destination.

Ahead of Truck (ϕ_3): This feature is active when the vehicle B is a truck and $acc \geq 0.6\, ft./sec^2$, i.e., A accelerates when B is a truck to get ahead of it.

Below is the summarized description of the experimental setup.

- MDP : $\langle S, A, \gamma \rangle$ = $\langle 1250, 5, 0.99 \rangle$;
- Features $\Phi = \{\phi_1, \phi_2, \phi_3\}$;
- $\mathcal{T}$ = $\{\zeta_1, \zeta_2, ..., \zeta_{12}\}$, i.e. set of 12 expert's trajectory exhibiting risky behavior from the NGSIM dataset;
- Boltzmann temperature, $\beta = 0.01$;
- Learning rate for Q-averaging and Q-softmax, $\alpha = 0.01$.

Table 3 shows the learned feature weights and corresponding final log-likelihood values of trajectories using our model-free IRL approach as well as the existing baselines MLIRL and REIRL, based on 10 independent trials for each setting. MLIRL requires the transition model, which we estimated by relative frequency of transition counts from the trajectories. For states not seen in the trajectories, we applied the mid-point of action ranges to the midpoint of state feature ranges to estimate the next states using standard dynamical equations. Although this is a reasonable model, the result is unlikely to be an accurate transition model for a domain as complex as NGSIM, and this highlights a major limitation of model-based approaches. For REIRL, we prolonged shorter trajectories by assuming constant velocity motion for A, B, and C beyond the last frame. We also selected a subset of trajectories randomly from the full set (of 260 trajectories) to serve as the baseline trajectories for REIRL. Once again, we see from Table 3 that Q-Averaging achieves the best log-likelihood, while this time MLIRL achieves the worst, perhaps because the estimated transition model is a poor approximation in this domain. In particular, MLIRL terminated early, in less than 25 iterations in all trials, confirming the unsuitability of a model-based approach for a complex domain where just the trajectory data is available. The log-likelihoods in Table 3 indicate performance improvement in the order MLIRL $<$ REIRL $<$ Q-softmax $<$ Q-averaging, with the Wilcoxon signed rank tests being significant at the 99% level (p-values across successive pairs being 0.00694, 0.00512, 0.00512). While the selection criterion for the 12 trajectories, viz., vehicle A speeding often, may indicate that the weight of the second feature (i.e., θ_2) should be the highest, that is not the case for Q-averaging and REIRL. However, the intuition is essentially correct, because the trials where θ_2 ended up higher for Q-averaging also had a slightly higher log-likelihood (≈ -1890). Unfortunately, only 40% of the trials converged to this solution. Once again the methods had comparable runtimes (except MLIRL which exited early with poor solutions) of about one hour per trial.

The overall results in Table 3 indicate a few interesting conclusions about the trajectories themselves. Even though the trajctories were selected based on what appeared to be risky driving behavior, the methods reach the conclusion that vehicle A still prefers to maintain safety. Also, contrary to the other methods, REIRL came to the conclusion that vehicle A strongly prefers to get ahead of B when the latter is a truck. However, given that both our methods achieve higher log-likelihoods with low values of θ_3, we must conclude that the trajectories display insufficient evidence of this behavior.

5 Conclusion and Future work

We have proposed a novel model-free approach to inverse reinforcement learning that relaxes constraining assumptions of existing methods, that the transition model is known, or that an additional set of non-expert trajectories is available. We have formulated our approach within the frame-

Method	Learned feature weights			Final log-likelihood
	θ_1	θ_2	θ_3	
MLIRL	0.28 ± 0.72	0.42 ± 0.48	0.45 ± 0.76	-1927.09 ± 1.1
Q-Avg	2.01 ± 0.23	1.86 ± 0.37	0.15 ± 0.55	$\mathbf{-1891.34 \pm 0.79}$
Q-SM	0.41 ± 0.30	0.80 ± 0.53	-0.12 ± 0.67	-1914.56 ± 1.38
REIRL	0.08 ± 0.18	0.01 ± 0.14	15.89 ± 7.62	-1920.48 ± 3.24

Table 3: Comparison of learned feature weights and corresponding final log-likelihood values of NGSIM trajectories, from various algorithms.

work of maximum likelihood IRL, and proposed two algorithms – Q-averaging and Q-softmax. Experiments in two domains show that Q-averaging achieves higher log-likelihood compared to both an existing model-based and a model-free method. Furthermore, the experiment on a real-world complex problem involving freeway merging, highlights the considerable limitation of model-based MLIRL. Thus our experimental findings position Q-averaging as the new state-of-the-art for model-free IRL.

It may appear surprising that Q-averaging is able to perform so well despite involving no optimization within the Q-update rule itself. Note however, that it simply gives a plausible value estimate of the policy in the current iteration, while the real search is in the space of feature weights, where gradient ascent provides the needed optimization in IRL. During algorithm design, we expected Q-softmax to perform better since it is closer to the Bellman operator for high values of β, and it is instructive to inspect why it did not. We found that high values of β leads to exponential blow-up in the gradient values (due to the presence of exponential terms in equation 11) and this forced us to use smaller β values in the experiments. Note that this problem appears with the MLIRL method as well. Smaller β actually makes the Q-values of Q-averaging and Q-softmax (i.e., equations 12 and 15) to be rather similar; however their gradients (equations 13 and 16) are quite different. The net effect on the learning outcome seems to be that the Q-softmax gradients tend to keep the feature weights ($\boldsymbol{\theta}$) closer to each other, compared to $\boldsymbol{\theta}$ of Q-averaging which can be vastly different (see Tables 1– 3). It appears to effectively become a kind of constraint for Q-softmax, which might explain why it was unable to reach better log-likelihoods.

Future work may include implementing our approach with other optimization techniques which do not require differentiating the likelihood function. This will allow the use of conventional Q-learning with the "max" operator. Our current approach is limited to a single expert, therefore another avenue would be to address multiple experts. Modeling of a multi-agent environment and their interaction with other experts in the environment might lead to better understanding of their behavior.

6 Acknowledgment

We thank Tomoki Nishi of Toyota Research Institute of North America and TCRDL for many helpful discussions and insights into the freeway merge domain. We thank Monica Babes-VRoman for her MLIRL implementation which was used to generate baseline results in this paper. We also thank the anonymous reviewers for constructive comments and suggestions. This work was supported in part by a research contract with the Toyota Research Institute of North America (TRI-NA), and by National Science Foundation grants IIS-1830421 and IIS-1526813.

References

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

Alexiadis, V.; Colyar, J.; and Halkias, J. 2007. A model endeavor. *Public Roads* 70(4).

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Asadi, K., and Littman, M. 2017. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Boularias, A.; Kober, J.; and Peters, J. 2011. Relative entropy inverse reinforcement learning. In *AISTATS*, 182–189.

Kuderer, M.; Gulati, S.; and Burgard, W. 2015. Learning driving styles for autonomous vehicles from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2641–2646.

Osa, T.; Pajarinen, J.; Neumann, G.; Bagnell, J. A.; Abbeel, P.; and Peters, J. 2018. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics* 7(1-2):1–179.

Ramachandran, D. 2007. Bayesian inverse reinforcement learning. In *IJCAI*, 2586–2591.

Russell, S. 1998. Learning agents for uncertain environments (extended abstract). In *Eleventh Annual Conference on Computational Learning Theory*, 101–103.

Trivedi, M., and Doshi, P. 2018. Inverse learning of robot behavior for collaborative planning. In *IROS*, 6.

Uchibe, E. 2018. Model-free deep inverse reinforcement learning by logistic regression. *Neural Processing Letters* 47(3):891–905.

VRoman, M. C. 2014. *Maximum Likelihood Inverse Reinforcement Learning*. Ph.D. Dissertation, Rutgers University.

Watkins, C., and Dayan, P. 1992. Q-learning. *Machine Learning Journal* 8(3/4).

Ziebart, B. D.; Maas, A.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*, 1433–1438.