Infinite Time Horizon Maximum Causal Entropy Inverse Reinforcement Learning

Michael Bloem and Nicholas Bambos

***Abstract*— We extend the maximum causal entropy framework for inverse reinforcement learning to the infinite time horizon discounted reward setting. To do so, we maximize discounted future contributions to causal entropy subject to a discounted feature expectation matching constraint. A parameterized class of stochastic policies that solve this problem are referred to as soft Bellman policies because they can be specified in terms of values that satisfy an equation identical to the Bellman equation but with a $\mathrm{softmax}$ (the log of a sum of exponentials) instead of a $\max$. Under some assumptions, algorithms that repeatedly solve for a soft Bellman policy, evaluate the policy, and then perform a gradient update on the parameters will find the optimal soft Bellman policy. For the first step, we extend techniques from dynamic programming and reinforcement learning so that they derive soft Bellman policies. For the second step, we can use policy evaluation techniques from dynamic programming or perform Monte Carlo simulations. We compare three algorithms of this type by applying them to a problem instance involving demonstration data from a simple controlled queuing network model inspired by problems in air traffic management.**

I. Introduction

Inverse reinforcement learning (IRL) attempts to use demonstrations of "expert" decision making in a Markov decision process (MDP) to infer a reward function and corresponding policy that are in some sense consistent with the expert demonstrations [1]. Several IRL approaches suffer from at least one of some common weaknesses [2]. One such weakness is the imposition of assumptions about expert behavior in order to arrive at a well-posed problem. A second is that when the demonstration data is sub-optimal, it may be assigned probability zero by the inferred stochastic policy. A third is that, for certain assumed stochastic policy distributions, the most likely policy is not the one that achieves the largest objective value. A fourth is requiring the solution of a non-convex optimization problem. An approach that is free from these weaknesses is maximum causal entropy (MCE) IRL, proposed by Ziebart et al. [2]–[4]. Given the ill-posed nature of the problem, MCE IRL proposes to infer the most uncertain stochastic policy that satisfies some statistic-matching constraints that capture the "structured, purposeful qualities" in the demonstration data [2]. Furthermore, the MCE IRL approach can be interpreted as maximizing the causal likelihood of the demonstration data when a certain stochastic policy distribution is assumed, and it provides a worst-case prediction log-loss guarantee [2]. However, as far as we know, MCE IRL has not been explored in the popular infinite discounted reward setting. MDPs with an infinite discounted reward objective have been utilized to model a range of problems in which there is no terminal state, including problems from finance and manufacturing [5]. Discounting future rewards captures a preference for present rewards over future rewards, which arises naturally in many contexts, such as when one is concerned with the net present value of cash flows. For IRL there is an additional motivation for using a discounted objective: expert decision makers may be guided by such an objective because they find it easier to determine the impact of present actions on present rewards than on future rewards. A final limitation of many previous IRL algorithms, including the algorithms for MCE IRL proposed by Ziebart et al., is that they require knowledge of the state transition probabilities.

In this paper, we extend MCE IRL to the popular infinite time horizon discounted reward setting by maximizing discounted future contributions to causal entropy subject to a discounted feature expectation matching constraint. This leads to the maximum discounted causal entropy (MDCE) IRL problem. Like MCE IRL, MDCE IRL is free from the four weaknesses that plague several other approaches to IRL, and it provides a worst-case prediction discounted log-loss guarantee. By extending an existing algorithm for the finite time MCE IRL problem and techniques from dynamic programming and reinforcement learning, we specify three algorithms. One is an online algorithm that, unlike most IRL algorithms, does *not* require state transition probabilities. We demonstrate the behavior of these algorithms on an MDCE IRL problem instance based on a simple queuing network model inspired by problems in air traffic management.

The remainder of this paper is structured as follows. We establish some preliminary definitions and assumptions in Section II. Then, we specify and describe the MDCE IRL problem in Section III. In Section IV, we derive parameterized soft Bellman policies that solve the MDCE IRL problem. We describe and prove characteristics of algorithms for the problem in Section V. Section VI contains a description and results of a computational experiment we performed to study and compare some algorithms for this problem. In Section VII, we finish the paper with conclusions.

II. Preliminary Definitions and Assumptions

A. Markov Decision Processes

A Markov decision process (MDP) is specified by a six-tuple: $(\mathcal{S}, \mathcal{A}, P_0, P, \gamma, R)$. The finite set of states is $\mathcal{S}$ and the finite set of actions available at each state is $\mathcal{A}$. The

M. Bloem is in the Aviation Systems Division, NASA Ames Research Center, Moffett Field, CA 94035, USA michael.bloem@nasa.gov

N. Bambos is with the Departments of Electrical Engineering and Management Science & Engineering, Stanford University, Stanford, CA 94305, USA bambos@stanford.edu

distribution of the initial state s_0 is specified by P_0. The state transition probabilities are captured by P: $P(s_{t+1}|s_t, a_t)$ is the probability of transitioning to state s_{t+1} when in state s_t and selecting action a_t. The reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$ specifies that $R(s, a)$ is the reward achieved for selecting action a when in state s. A policy π specifies how to select actions in an MDP. Stationary policies do not depend on the time step. A stationary stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ specifies a conditional distribution over actions. Sequences of states and actions in an MDP can also be interpreted as two interacting random processes: $S_{0:t_\mathrm{f}} \triangleq \{S_t\}_{t=0}^{t_\mathrm{f}}$ and $A_{0:t_\mathrm{f}} \triangleq \{A_t\}_{t=0}^{t_\mathrm{f}}$. At each t, the random variable S_t takes a value $s_t \in \mathcal{S}$ and the the random variable A_t takes a value $a_t \in \mathcal{A}$. The distributions of the random variables making up these random processes (and their interactions) are described by the distribution P_0 and the conditional distributions P and π. The discount factor $\gamma \in (0, 1)$ helps define the expected infinite discounted reward objective $\mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t)\right]$. In IRL problems, R is not given but rather inferred from demonstration data. An MDP but with unknown reward function is an MDP$\backslash R$ [6].

B. Demonstration Data

The specification of an IRL problem includes demonstrations of expert decision making that can be used to infer a reward function and corresponding policy. We are given a data set $\mathcal{D} = \left\{(s_{d,t}, a_{d,t})_{t=0}^{t_\mathrm{f}}\right\}_{d=1}^{D}$, where $a_{d,t}$ was generated by $\pi^{\mathrm{E}}(s_{d,t})$, the expert's policy evaluated at $s_{d,t}$.

C. Feature Expectation Matching

One way to quantify the "structured, purposeful qualities" of expert actions is to study the expected value of a *reward feature vector* $f : \mathcal{S} \times \mathcal{A} \to \mathbf{R}^K$ [2]. For a given MDP, a *feature expectation vector* (FEV) $\bar{f}_\pi^\gamma \in \mathbf{R}^K$ for a policy π and a discount factor γ is defined as $\bar{f}_\pi^\gamma \triangleq \mathbf{E}_{P_0,P,\pi}\left[\sum_{t=0}^{\infty} \gamma^t f(S_t, A_t)\right]$. We can use the demonstration data $\mathcal{D}$ to compute an estimate $\hat{\bar{f}}_{\pi^{\mathrm{E}}}^\gamma$ of $\bar{f}_{\pi^{\mathrm{E}}}^\gamma$: $\hat{\bar{f}}_{\pi^{\mathrm{E}}}^\gamma = \frac{1}{D}\sum_{d=1}^{D}\sum_{t=0}^{t_\mathrm{f}} \gamma^t f(s_{d,t}, a_{d,t})$. Assuming that the reward feature vector components measure quantities that direct expert decisions, we can find a π that shares the "purposeful qualities" of π^{E} captured in the demonstration data by enforcing the *feature expectation matching* (FEM) constraint: $\bar{f}_\pi^\gamma = \hat{\bar{f}}_{\pi^{\mathrm{E}}}^\gamma$ [2].

D. Maximizing Entropy

One way to select from among probability distributions that satisfy some constraints is to pick the one that maximizes entropy. This distribution assigns probability to outcomes as evenly as possible, making it the "least committed" or "most uncertain" option [2]. When given side information and selecting from among *conditional* probability distributions that satisfy constraints, the corresponding objective is to find the distribution that maximizes *conditional entropy*. In the finite time horizon IRL context, the side information is a sequence of visited states $s_{0:t_\mathrm{f}} \triangleq \{s_t\}_{t=0}^{t_\mathrm{f}}$, and the distribution to be inferred is a stochastic policy $\pi(a_t|s_{0:t}, a_{0:t-1}) = \mathbf{Prob}(a_t|s_{0:t}, a_{0:t-1})$, which can be simplified to $\pi(a_t|s_t) = \mathbf{Prob}(a_t|s_t)$ for an MDP. The conditional entropy is defined as $H(A_{0:t_\mathrm{f}}|S_{0:t_\mathrm{f}}) = \mathbf{E}\left[-\log \mathbf{Prob}(A_{0:t_\mathrm{f}}|S_{0:t_\mathrm{f}})\right]$, where $\mathbf{Prob}(a_{0:t_\mathrm{f}}|s_{0:t_\mathrm{f}})$ is the *conditional probability*. The conditional probability conditions on the *entire* set of side information $(s_{0:t_\mathrm{f}})$: $\mathbf{Prob}(a_{0:t_\mathrm{f}}|s_{0:t_\mathrm{f}}) = \prod_{t=0}^{t_\mathrm{f}} \mathbf{Prob}(a_t|s_{0:t_\mathrm{f}}, a_{0:t-1})$. This is problematic because it leads to non-causal policies. Therefore, it is more appropriate to find a conditional distribution (stochastic policy) that maximizes *causal entropy* $H(A_{0:t_\mathrm{f}}||S_{0:t_\mathrm{f}}) = \mathbf{E}\left[-\log \mathbf{Prob}(A_{0:t_\mathrm{f}}||S_{0:t_\mathrm{f}})\right]$, where $\mathbf{Prob}(a_{0:t_\mathrm{f}}||s_{0:t_\mathrm{f}})$ is the *causally-conditioned probability* $\mathbf{Prob}(a_{0:t_\mathrm{f}}||s_{0:t_\mathrm{f}}) = \prod_{t=0}^{t_\mathrm{f}} \mathbf{Prob}(a_t|s_{0:t}, a_{0:t-1})$ [7].

E. Discounted Causal Entropy

An issue with extending the notion of causal entropy to an infinite time horizon context is that causal entropy can be infinite. A related but finite quantity is discounted contributions to future causal entropy, or just *discounted causal entropy*, for some discount factor $\beta \in (0, 1)$, which is defined as:

$$H^\beta(A_{0:\infty}||S_{0:\infty}) \triangleq \mathop{\mathbf{E}}_{P_0,P,\pi}\left[\sum_{t=0}^{\infty} -\beta^t \log \pi_t(A_t|S_{0:t}, A_{0:t-1})\right].$$

Discounted entropy has been employed in other contexts (*e.g.*, control that is robust to model mis-specification [8]).

III. Maximum Discounted Causal Entropy Inverse Reinforcement Learning Problem

The maximum discounted causal entropy (MDCE) IRL problem is

$$\begin{aligned}
\mathop{\text{maximize}}_{\pi} \quad & H^\beta(A_{0:\infty}||S_{0:\infty}) && (1)\\
\text{subject to} \quad & \bar{f}_\pi^\gamma = \hat{\bar{f}}_{\pi^{\mathrm{E}}}^\gamma && (2)\\
& \pi_t(a_t|s_t) \geq 0 \quad \forall a_t, s_t, t \geq 0 && (3)\\
& \sum_{a_t \in \mathcal{A}} \pi_t(a_t|s_t) = 1 \quad \forall s_t, t \geq 0. && (4)
\end{aligned}$$

The decision variables that make up the potentially non-stationary policy π are $\pi_t(a_t|s_t)$ $\forall s_t \in \mathcal{S}, a_t \in \mathcal{A}, t \geq 0$. The objective (1) is to maximize discounted causal entropy. Constraint (2) is the FEM constraint. Constraints (3) and (4) require that π be a valid stochastic policy. No constraint explicitly specifies that π be causally-conditioned (not dependent on future states). However, a causally-conditioned policy must factor as $\pi(a_{0:\infty}||s_{0:\infty}) = \prod_{t=0}^{\infty} \pi_t(a_t|s_t)$, so by using the factors $\pi_t(a_t|s_t)$ as the decision variables, we force π to be causally-conditioned (see [2], Remark 5.7).

While the MDCE IRL problem as specified in (1)–(4) is non-convex, a mathematically-equivalent specification in which decision variables specify $\mathbf{Prob}(a_{0:\infty}||s_{0:\infty})$ for each possible sequence of state-action pairs $(s_t, a_t)_{t=0}^{\infty}$ is convex (see [2], Definition 5.6 and Theorem 5.8). That convex problem has an infinite number of variables and constraints, but its dual is tractable when we enforce the causal-conditioning constraints by using the factors $\pi_t(a_t|s_t)$ as the decision variables. Furthermore, when strong duality holds, a solution to the dual is also a solution to the primal, a property

that will enable us to derive a tractable recursive solution specification. The sharp version of Slater's condition, which in this context is met when there is a non-deterministic policy that meets the FEM constraint, is sufficient for strong duality [9]. Slater's condition is not restrictive because by definition there is a policy (π^{E}) that satisfies the FEM constraint (assuming $\hat{\bar{f}}^{\gamma}_{\pi_{\mathrm{E}}}$ is a sufficiently accurate estimate of $\bar{f}^{\gamma}_{\pi_{\mathrm{E}}}$) [3]. If needed, we can "loosen" the FEM constraint to ensure that there is a non-deterministic feasible policy.

A solution to the MDCE IRL problem achieves a worst-case guarantee when predicting future expert actions.

Theorem 1: A maximum discounted causal entropy policy minimizes the worst-case discounted log-loss of predictions of the actions taken by any other policy $\tilde{\pi}$, $\sup_{\tilde{\pi}} \mathbf{E}_{P_0,P,\tilde{\pi}} \left[\sum_{t=0}^{\infty} -\beta^t \log \pi(A_t|S_t)\right]$, given that $\tilde{\pi}$ achieves the FEM constraint (2).

Proof: This follows from Theorem 5.10 of [2]. ■

IV. Soft Bellman Policies

By investigating the dual problem mentioned in Section III, we can derive a parameterized and recursively-defined stochastic policy that solves the MDCE IRL problem.

Theorem 2: If the sharp version of Slater's condition holds, then a *soft Bellman policy* solves the MDCE IRL problem in eqs. (1)–(4). A soft Bellman policy has parameters $\theta \in \mathbf{R}^K$ and is defined recursively as:

$$\pi^{\mathrm{soft}}_{t,\theta}(a_t|s_t) = \exp\left(Q^{\mathrm{soft}}_{t,\theta}(s_t,a_t) - V^{\mathrm{soft}}_{t,\theta}(s_t)\right), \qquad (5)$$

where

$$Q^{\mathrm{soft}}_{t,\theta}(s,a) = \left(\frac{\gamma}{\beta}\right)^t \theta^\top f(s,a) + \beta \sum_{s' \in \mathcal{S}} P(s'|s,a) V^{\mathrm{soft}}_{t+1,\theta}(s') \qquad (6)$$

and

$$V^{\mathrm{soft}}_{t,\theta}(s) = \operatorname*{softmax}_{a \in \mathcal{A}} Q^{\mathrm{soft}}_{t,\theta}(s,a) \triangleq \log\left(\sum_{a \in \mathcal{A}} \exp(Q^{\mathrm{soft}}_{t,\theta}(s,a))\right). \qquad (7)$$

Proof: As discussed in Section III, there is a mathematically equivalent and convex specification of the MDCE IRL problem in which decision variables specify $\mathbf{Prob}(a_{0:\infty}||s_{0:\infty})$ for each possible sequence of state-action pairs $(s_t,a_t)_{t=0}^{\infty}$. Since we assume the sharp version of Slater's condition, strong duality holds. Therefore, if we write the Lagrangian of this specification of the problem with the factors $\pi_t(a_t|s_t)$ as variables, differentiate it, and set it equal to zero, we arrive at a general θ-parameterized form for $\pi_t(a_t|s_t)$: $\pi_{t,\theta}(a_t|s_t) = \frac{Z_{a_t|s_t,\theta}}{Z_{s_t,\theta}}$, where $Z_{a_t|s_t,\theta} = \exp\left\{\left(\frac{\gamma}{\beta}\right)^t \theta^\top f(s,a) + \beta \sum_{s' \in \mathcal{S}} P(s'|s,a) \log Z_{s_{t+1},\theta}\right\}$ and $Z_{s_t,\theta} = \sum_{a_t \in \mathcal{A}} Z_{a_t|s_t,\theta}$. By defining $Q^{\mathrm{soft}}_{t,\theta}(s,a) \triangleq \log Z_{a_t|s_t,\theta}$ and $V^{\mathrm{soft}}_{t,\theta}(s) \triangleq \log Z_{s_t,\theta}$, we see that soft Bellman policies are of this general form. ■

This proof is very similar to that of Theorem 1 in [3]. The θ parameters arise as dual variables corresponding to the FEM constraint (2). We use $\hat{\theta}$ to denote the values that specify a soft Bellman policy that solves the MDCE IRL problem.

In general, soft Bellman policies are non-stationary. Deriving non-stationary policies is generally intractable for infinite time horizon problems, so we will assume that $\beta = \gamma$ when specifying algorithms in Section V. This assumption dictates that future contributions to causal entropy, of concern to the analyst, and future reward feature values, of concern to the expert, be discounted identically. We do not suspect that there is motivation for this assumption in many contexts, and so requiring this assumption is a weakness of this work. When $\beta = \gamma$, we will denote the resulting stationary soft Bellman policies, soft state-action values, and soft state values as $\pi^{\mathrm{soft}}_{\theta}$, $Q^{\mathrm{soft}}_{\theta}$, and $V^{\mathrm{soft}}_{\theta}$, respectively. We conjecture that maximizing per-time step average contributions to causal entropy while matching per-time step average feature vectors would lead to stationary soft Bellman policies that could be derived using adaptations of average cost dynamic programming techniques.

Finally, we know that for the reward $R(s,a) = \hat{\theta}^\top f(s,a)$, the soft Bellman policy corresponding to $\hat{\theta}$ achieves the largest objective value that can be achieved by any soft Bellman policy (see [2], Theorem 6.11).

V. Algorithms

When strong duality holds for the MDCE IRL problem, we can solve the dual mentioned in the proof of Theorem 2 to find a $\hat{\theta}$ that specifies a policy $\pi^{\mathrm{soft}}_{\hat{\theta}}$ that solves the MDCE IRL problem. In fact, this corresponds to finding maximum discounted causal likelihood estimates of the θ parameters when given $\mathcal{D}$ and assuming the soft Bellman policy form (see [2], Theorem 6.4). Furthermore, the dual problem is convex, so we can use gradient ascent techniques to solve it. The gradient is $\hat{\bar{f}}^{\gamma}_{\pi^{\mathrm{E}}} - \bar{f}^{\gamma}_{\pi^{\mathrm{soft}}_{\theta}}$ (see [3], Theorem 2).

Therefore, a class of algorithms for the MDCE IRL problem is specified by:

Require: MDP$\backslash R$, f, $\mathcal{D}$ {MDCE IRL problem instance}
Require: $\theta_0 \in \mathbf{R}^K$ {initial guess for $\hat{\theta}$}
Require: $\alpha : \mathbf{Z}_+ \to \mathbf{R}_+$ {$\alpha(n)$ is the gradient step size}
Require: N {maximum number of iterations}
Require: $\epsilon \in \mathbf{R}_+$ {stopping criterion}
1: Compute $\hat{\bar{f}}^{\gamma}_{\pi^{\mathrm{E}}}$
2: **for** $n = 0$ to N **do**
3: Derive $\pi^{\mathrm{soft}}_{\theta_n}$
4: Compute $\bar{f}^{\gamma}_{\pi^{\mathrm{soft}}_{\theta_n}}$
5: **if** $\delta^{\mathrm{FEM}}\left(\bar{f}^{\gamma}_{\pi^{\mathrm{soft}}_{\theta_n}}, \hat{\bar{f}}^{\gamma}_{\pi^{\mathrm{E}}}\right) \leq \epsilon$ **then**
6: **return** θ_n and $\pi^{\mathrm{soft}}_{\theta_n}$
7: **end if**
8: $\theta_{n+1} = \theta_n + \alpha(n)\left(\hat{\bar{f}}^{\gamma}_{\pi^{\mathrm{E}}} - \bar{f}^{\gamma}_{\pi^{\mathrm{soft}}_{\theta_n}}\right)$
9: **end for**
10: **return** θ_N and $\pi^{\mathrm{soft}}_{\theta_N}$

Here steps 5 and 6 allow the algorithm to terminate when the FEM constraint (2) is approximately met, since we define $\delta^{\mathrm{FEM}} : \mathbf{R}^K \times \mathbf{R}^K \to \mathbf{R}_+$ as: $\delta^{\mathrm{FEM}}(f,f') \triangleq \max_k \frac{|f_k - f'_k|}{|f'_k|}$.

This class of algorithms is an infinite time horizon generalization of algorithms specified by Ziebart in Chapter 9 of [2]. In the remainder of this Section, we will describe three

ways to derive π_θ^{soft} for a given θ (step 3) and three ways to then evaluate a policy by computing $\bar{f}^{\gamma}_{\pi_\theta^{\text{soft}}}$ (step 4).

A. Soft Bellman Policy Derivation

We propose three approaches for step 3 by extending techniques from dynamic programming and reinforcement learning.

1) Soft Value Iteration: When P is known and $|\mathcal{S}|$ and $|\mathcal{A}|$ are not too large to prohibit repeated sums over states and actions, soft value iteration can be used to derive the soft Bellman policy. The soft Bellman operator corresponding to θ is defined as

$$T_\theta^{\text{soft}}(V)(s) \triangleq \underset{a\in\mathcal{A}}{\text{softmax}} \left(\theta^\top f(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)V(s') \right).$$

Soft value iteration involves repeatedly applying T_θ^{soft} to an initial soft value vector $V_0 \in \mathbf{R}^{|\mathcal{S}|}$. We will show that T_θ^{soft} is a contraction mapping so that, for any V_0, when we repeatedly apply T_θ^{soft}, we will converge to the unique fixed point $V^\star \in \mathbf{R}^{|\mathcal{S}|}$ satisfying $V^\star = T_\theta^{\text{soft}} V^\star$. This $V^\star$ is exactly the V_θ^{soft} we need to specify π_θ^{soft} using eqs. (5)–(7).

Fact 1: The softmax function is *monotone*: for $y, y' \in \mathbf{R}^N$, if $y_n \le y'_n$ for all $n = 1, \ldots, N$, then $\text{softmax}(y) \le \text{softmax}(y')$.

Fact 2: For $y \in \mathbf{R}^N$ and $M \in \mathbf{R}$, $\text{softmax}(y + M\mathbf{1}^\top) = \text{softmax}(y) + M$ (where $\mathbf{1}^\top$ is an $N \times 1$ vector of ones).

Lemma 1: T_θ^{soft} is *monotone*: for $V, V' \in \mathbf{R}^{|\mathcal{S}|}$, if $V \le V'$, then $T_\theta^{\text{soft}}(V) \le T_\theta^{\text{soft}}(V')$.

Proof: This follows directly from the monotonicity of the softmax function. ∎

Theorem 3: T_θ^{soft} is a contraction mapping with contraction factor γ.

Proof: Consider $V, V' \in \mathbf{R}^{|\mathcal{S}|}$. There exists $0 \le M \le \infty$ such that $\|V - V'\|_\infty = M$. Therefore, $-M \le V(s) - V'(s) \le M$ for all $s \in \mathcal{S}$. Since T_θ^{soft} is monotone, $T_\theta^{\text{soft}}(V)(s) \le \text{softmax}_{a\in\mathcal{A}} \left(\theta^\top f(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) \left[V'(s) + M \right] \right)$ holds for all $s \in \mathcal{S}$ and a symmetric inequality can bound $T_\theta^{\text{soft}}(V')$ above for all $s \in \mathcal{S}$. By Fact 2 and since $\sum_{s'\in\mathcal{S}} P(s'|s,a)M = M$, we also know that $T_\theta^{\text{soft}}(V)(s) \le T_\theta^{\text{soft}}(V')(s) + \gamma M$ and $T_\theta^{\text{soft}}(V')(s) \le T_\theta^{\text{soft}}(V)(s) + \gamma M$ for all $s \in \mathcal{S}$. Together, these two inequalities imply that $\|T_\theta^{\text{soft}}(V) - T_\theta^{\text{soft}}(V')\|_\infty \le \gamma \|V - V'\|_\infty$. ∎

2) Soft Values via Convex Optimization: We can find V_θ^{soft} by solving a convex optimization problem:

$$\underset{V}{\text{minimize}} \;\; c^\top V \tag{8}$$

$$\text{subject to } V \ge T_\theta^{\text{soft}} V, \tag{9}$$

for any $c \in \mathbf{R}^{|\mathcal{S}|}_{++}$.

Theorem 4: V_θ^{soft} is the unique solution to the convex optimization problem in eqs. (8)–(9).

Proof: Since T_θ^{soft} is monotone and also a contraction mapping that converges to V_θ^{soft}, we know that for any feasible V (satisfying constraint (9)), $V \ge T_\theta^{\text{soft}} V \ge (T_\theta^{\text{soft}})^2 V \ge (T_\theta^{\text{soft}})^3 V \ge \cdots \ge V_\theta^{\text{soft}}$. This implies that $V \ge V_\theta^{\text{soft}}$ for all feasible V, which in turn implies that $c^\top V \ge c^\top V_\theta^{\text{soft}}$. Since V_θ^{soft} is also a feasible solution and the unique fixed point $V_\theta^{\text{soft}} = T_\theta^{\text{soft}} V_\theta^{\text{soft}}$, it is the unique optimal solution. ∎

As was the case for soft value iteration, this approach requires knowledge of P. This problem has $|\mathcal{S}|$ variables and $|\mathcal{S}|$ constraints, so it may require long computation times. Computation times could be reduced by solving for linear approximations of the soft value function while utilizing constraint sampling.

3) Soft Q-Learning: When P is not known, we can learn Q_θ^{soft} while simulating the system by extending the well-known Q-learning reinforcement algorithm to this context (see [5], sub-section 6.4). The update equation for Q^{soft} is

$$Q^{\text{soft}}(s_t, a_t) \leftarrow Q^{\text{soft}}(s_t, a_t) + \eta(t) \Big[\theta^\top f(s_t, a_t) + \gamma \underset{a_{t+1}\in\mathcal{A}}{\text{softmax}} Q^{\text{soft}}(s_{t+1}, a_{t+1}) - Q^{\text{soft}}(s_t, a_t) \Big],$$

where $\eta : \mathbf{Z}_+ \to \mathbf{R}_+$ is a step size parameter function.

Theorem 5: If each $Q^{\text{soft}}(s,a)$ is updated infinitely often, $\sum_{t=0}^\infty \eta(t) = \infty$, and $\sum_{t=0}^\infty \eta(t)^2 < \infty$, then soft Q-learning converges to Q_θ^{soft} w.p. 1.

Proof: This result can be proven using the same approach used by Melo to prove the convergence of Q-learning in [10]. This proof depends on the fact that the soft Bellman operator for soft state-action values is a contraction mapping, which can be shown using the same approach as was used in the proof of Theorem 3. ∎

B. Policy Evaluation

When evaluating a soft Bellman policy π_θ^{soft} in step 4, we are interested in computing its feature expectation vector $\bar{f}^{\gamma}_{\pi_\theta^{\text{soft}}}$. Two of the approaches described here are based on the observation that we can compute the k^{th} component of $\bar{f}^{\gamma}_{\pi_\theta^{\text{soft}}}$ by evaluating π_θ^{soft} in an MDP with reward function $f_k(s,a)$. The third approach uses Monte Carlo simulations.

1) Dynamic Programming Operator: As proposed by Ziebart et al. for the finite time horizon context, we can repeatedly apply the regular (non-soft) dynamic programming operator (T) defined for reward function $f_k(s,a)$ and policy π_θ^{soft} to compute the expected infinite discounted sum of reward feature k assuming we start in any given state [3]. The inner product of this vector and P_0 yields the k^{th} component of $\bar{f}^{\gamma}_{\pi_\theta^{\text{soft}}}$. This approach requires knowledge of P.

2) Matrix Computations: We can also compute the expected infinite discounted sum of reward feature k, assuming we start in any given state and follow π_θ^{soft}, by solving a set of linear equations. Again, the inner product of the resulting vector and P_0 yields the k^{th} component of $\bar{f}^{\gamma}_{\pi_\theta^{\text{soft}}}$. Although accurate, this approach requires knowledge of P and can require long computation times when $|\mathcal{S}|$ is large.

3) Monte Carlo Simulations: Finally, we can use Monte Carlo simulations to estimate $\bar{f}^{\gamma}_{\pi_\theta^{\text{soft}}}$. This involves repeatedly initializing the MDP$\backslash R$ and then selecting actions as prescribed by π_θ^{soft}. Finally, the resulting data set of state-action pairs can be used to estimate $\bar{f}^{\gamma}_{\pi_\theta^{\text{soft}}}$. This approach does not require knowledge of P.

TABLE I

ALGORITHM PARAMETERS

Parameter	Value
$\alpha(n)$	$1000/(1000+\sqrt{n+2})$
N	1000
ϵ	0.025
θ_0	$[-1.0,-1.0,-1.0,-1.0,-1.0]^\top$

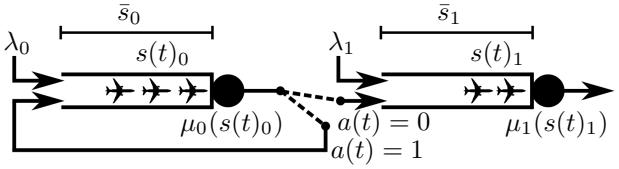

Fig. 1. Controlled queuing network

VI. COMPUTATIONAL EXPERIMENT

A. Algorithms

We selected three algorithms such that each policy derivation approach and each evaluation approach is used at least once, with the exception of the matrix computation approach, which is instead used in post-processing to illustrate properties of the other two policy evaluation approaches. The first algorithm (`SoftVI-T`) uses soft value iteration to derive soft Bellman policies and then the dynamic programming operator to evaluate them. The second (`ConvOpt-T`) uses convex optimization to derive soft Bellman policies and then the dynamic programming operator to evaluate them. The third (`SoftQL-Sim`) uses soft Q-learning to derive soft Bellman policies and then Monte Carlo simulations to evaluate them. Unlike the other two algorithms, `SoftQL-Sim` requires the ability to simulate the system but not the specification of P_0 and P. All of these algorithms were implemented by extending the PyMDPToolbox Python package[1] [11]. The solutions to convex optimization problems required by `ConvOpt-T` were found using the CVXPY and CVXOPT Python packages [12], [13]. Finally, relevant algorithm parameter values are specified in Table I.

B. Problem Instance

We evaluate these algorithms on a problem instance based on an MDP for a simple controlled queuing network. The MDP system model is inspired by controlled queuing network models of air traffic management problems [14]–[16], but it would need additional size and complexity to meaningfully represent a real-world problem. For such models we could approximate soft Bellman policies by extending techniques from approximate dynamic programming.

The network, depicted in Fig. 1, consists of a sequence of two servers and corresponding buffers, and the control input determines whether a flight completing service in the first server transitions to the second buffer or is instead recirculated back into the first buffer. Recirculation represents actions taken to delay aircraft. Both buffers have finite capacities, and an arrival to a full buffer exits the network, an event that could model a flight being canceled, diverted, or rerouted.

The system state at time step t is $s(t) = [s(t)_0, s(t)_1, s(t)_2, s(t)_3]^\top$ and it describes the number of flights in each buffer at the start of the time step ($s(t)_0$ and $s(t)_1$), as well as the number of flights attempting to enter each buffer during the previous time step that had to exit the system ($s(t)_2$ and $s(t)_3$ for flights encountering a full buffer 0 and buffer 1, respectively). The buffer capacities are $\bar{s}_0$ and $\bar{s}_1$. The distribution P_0 is uniform.

In each time step, a flight arrives at and joins the first buffer with probability λ_0 and at the second with probability λ_1. Similarly, a flight completes service in the first server with probability $\mu_0(s(t)_0)$ and the second with probability $\mu_1(s(t)_1)$. These probabilities are monotonically non-decreasing functions of the number of flights in the corresponding buffer. For many air traffic management problems, flights modeled as in each buffer are in fact traversing a region of airspace, and service completion represents that a flight is ready to move to the next region of airspace. When more flights are in a region, there is a greater probability that one of them will be ready in a time step. The state-dependent service completion probabilities are meant to capture this phenomenon [14]. In this problem instance, $\mu_0(s(t)_0)$ increases linearly from $\underline{\mu}_0$ when $s(t)_0 = 1$ to $\bar{\mu}_0$ when $s(t)_0 = \bar{s}_0$ and similarly $\mu_0(s(t)_0)$ increases linearly from $\underline{\mu}_1$ when $s(t)_1 = 1$ to $\bar{\mu}_1$ when $s(t)_1 = \bar{s}_1$. All of these probabilities are independent from each other and across time steps. The action $a(t) = 0$ indicates that a flight completing service at the first server will be routed onward, while $a(t) = 1$ indicates that it will be recirculated.

We define an MDP based on this system by specifying a discount factor γ and reward function $R(s,a) = \theta^\top f(s,a)$ for $f(s,a) = [s_0^{\sigma_0}, s_1^{\sigma_1}, s_2, s_3, a]^\top / f_{\max}$, where $f_{\max}$ is a 5×1 vector containing the maximum attainable value for each element in the vector in the numerator. We generate an MDCE IRL instance by deriving an expert policy and then simulating it to provide a demonstration data set. The expert policy is a Boltzmann policy for the MDP: $\pi_\tau^{\text{Boltzmann}}(a|s) \propto Q^\star(s,a)/\tau$, where τ is the *temperature* parameter and $Q^\star$ is the state-action value function that is optimal with respect to $R(s,a) = \theta^\top f(s,a)$. We use the expert policy to generate a test data set of the same size as the training data set.

Table II shows the parameters we used for the instance. The Python code was run on a MacPro workstation with two 6-core Intel Xeon 2.66 GHz processors and 24 GB of RAM.

C. Results

Table III shows the results of the three algorithms for the MDCE IRL problem instance. The δ^{FEM} "estimate" column is based on $\bar{f}^{\gamma}_{\pi^{\text{soft}}_\theta}$ computed by each algorithm (using the dynamic programming operator or simulation), while the δ^{FEM} "actual" column is based on $\bar{f}^{\gamma}_{\pi^{\text{soft}}_\theta}$ computed by the

[1]NASA plans to release implementations of these algorithms in a patch for the PyMDPToolbox package. Contact the authors for details.

TABLE II

MDCE IRL PROBLEM INSTANCE PARAMETERS

Parameter	Value	Parameter	Value
$\bar{s}_0$	10	$\bar{s}_1$	2
λ_0	0.8	λ_1	0.4
$\underline{\mu}_0$	0.2	$\bar{\mu}_0$	0.99
$\underline{\mu}_1$	0.3	$\bar{\mu}_1$	0.8
γ	0.8	θ	$[-1.0, -1.0, -2.0, -8.0, -0.2]^\top$
σ_0	1.0	σ_1	1.5
D	2000	t_f	50
τ	0.25		

TABLE III

ALGORITHM PERFORMANCE

	δ^{FEM}		Discounted Log-Loss		Average Log-Loss		Time
Algorithm	Estimate	Actual	Train	Test	Train	Test	[sec]
`SoftVI-T`	0.0250	0.0250	3.23	3.22	0.570	0.569	82.7
`ConvOpt-T`	0.0250	0.0250	3.23	3.22	0.570	0.569	18000.
`SoftQL-Sim`	0.0178	0.0426	3.48	3.46	0.633	0.633	486.

accurate matrix computation approach. While `SoftVI-T` and `ConvOpt-T` estimate $\bar{f}^{\gamma}_{\pi^{\mathrm{soft}}_{\theta}}$ accurately, `SoftQL-Sim` terminates prematurely because it fails to do so. The discounted and average log-loss values achieved by `SoftVI-T` and `ConvOpt-T` are identical and lower than those achieved by `SoftQL-Sim`. Furthermore, the discounted and average log-loss values are nearly equal on the training and testing data sets, indicating that over-fitting is not an issue. Finally, `SoftVI-T` requires the shortest time to execute, followed by `SoftQL-Sim` and then `ConvOpt-T`. In fact, `ConvOpt-T` requires more than 200 times longer than `SoftVI-T`.

VII. CONCLUSIONS

The maximum discounted causal entropy inverse reinforcement learning problem extends maximum causal entropy inverse reinforcement learning to the infinite time horizon discounted reward context. Many of the desirable theoretical properties of finite time horizon maximum causal entropy inverse reinforcement learning still hold after this extension. Soft Bellman policies specify a parameterized form for stochastic policies that solve the maximum discounted causal entropy inverse reinforcement learning problem. Assuming that the same discount factor is used to discount contributions to future causal entropy and future reward feature values, and also assuming some mild conditions that ensure strong duality, a class of algorithms based on gradient ascent will find optimal values of the soft Bellman policy parameters. One step in this class of algorithms requires that a soft Bellman policy be derived for an estimate of the policy parameters. Extensions of three algorithms from dynamic programming and reinforcement learning will find this unique soft Bellman policy. Another step in this class of algorithms requires that a reward feature expectation vector be computed for a soft Bellman policy. This step can be performed by applying policy evaluation techniques or via Monte Carlo simulations.

We perform a computational experiment to illustrate properties of three algorithms in this class of algorithms. We define a problem instance based on a simple controlled queuing network model that is similar to models used in air traffic management research and then deploy the three algorithms on the instance. Two of these algorithms generate policies that achieve identical discounted and average log-losses when predicting expert actions. However, the algorithm that derives policies with an extension of value iteration executes more than 200 times faster than the algorithm that derives policies by solving a convex optimization problem. The third algorithm derives soft Bellman policies with an extension of Q-learning and evaluates them using Monte Carlo simulations, so it requires the ability to simulate the system but not a full specification of state transition probabilities. It terminates prematurely due to an inaccurate feature expectation vector estimate, and the resulting policy achieves higher discounted and average log-losses.

REFERENCES

[1] S. Zhifei and E. M. Joo, "A review of inverse reinforcement learning theory and recent advances," in *IEEE World Congress on Computational Intelligence*, Brisbane, Australia, June 2012.

[2] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Carnegie Mellon University, 2010.

[3] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," in *Proc. of International Conference on Machine Learning*, Haifa, Israel, 2010.

[4] ——, "The principle of maximum causal entropy for estimating interacting processes," *IEEE Trans. on Information Theory*, vol. 59, no. 4, pp. 1966–1980, April 2013.

[5] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Nashua, NH: Athena Scientific, 2005, vol. 1.

[6] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of International Conference on Machine Learning*, Banff, Canada, 2004.

[7] G. Kramer, "Directed information for channels with feedback," PhD Dissertation, Swiss Federal Institute of Technology, Zurich, 1998.

[8] L. P. Hansen, T. J. Sargent, G. Turmuhambetova, and N. Williams, "Robust control and model misspecification," *Journal of Economic Theory*, vol. 128, pp. 45–90, 2006.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[10] F. S. Melo, "Convergence of Q-learning: A simple proof," `http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/ProofQlearning.pdf`, February 2007.

[11] S. Cordwell, "PyMDPToolbox: Markov decision process (MDP) toolbox for Python," `https://code.google.com/p/pymdptoolbox/`, 2013.

[12] T. T. D. Rubira, "CVXPY: A Python package for modeling convex optimization problems," `https://code.google.com/p/cvxpy/`, 2013.

[13] M. Andersen, J. Dahl, and L. Vandenberghe, "CVXOPT: Python software for convex optimization," `http://cvxopt.org/`, 2013.

[14] J. L. Ny and H. Balakrishnan, "Feedback control of the National Airspace System," *AIAA Journal of Guidance, Control, and Dynamics*, December 2010.

[15] Y. Wan, C. Taylor, S. Roy, C. Wanke, and Y. Zhou, "Dynamic queuing network model for flow contingency management," *IEEE Trans. on Intelligent Transportation Systems*, 2013.

[16] M. Bloem and N. Bambos, "Ground Delay Program analytics with behavioral cloning and inverse reinforcement learning," in *AIAA Aviation Technology, Integration, and Operations Conference*, Atlanta, GA, August 2014.