Bootstrapping Apprenticeship Learning

Abdeslam Boularias and Brahim Chaib-Draa

Laval University, Québec, Canada

1 Overview

- We consider the problem of imitation learning where the examples, given by an expert, cover only a small part of a large state space.
- Inverse Reinforcement Learning (IRL) provides an efficient tool for generalizing the partial demonstration, based on the assumption that the expert is maximizing an unknown utility function.
- IRL consists in learning a reward function that explains the expert's behavior, and that is a linear combination of state-action features.
- Previous IRL algorithms use the empirical averages to estimate the expected feature counts under the expert's policy.
- We introduce a new technique for bootstrapping the examples. The proposed technique consists in iteratively learning a reward function. At each iteration,
  - the examples are replaced by a complete optimal policy given the current reward function,
  - the feature counts are calculated by using the current optimal policy and the dynamics model.

2 Background

2.1 Markov Decision Process (MDP)

A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, T, R, \alpha, H, \gamma)$, where $\mathcal{S}$ is a set of states and $\mathcal{A}$ is a set of actions. T is a transition function with $T(s,a,s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ for $s, s' \in \mathcal{S}, a \in A$, and R is a reward function where $R(s,a)$ is the reward given for executing action a in state s. The initial state distribution is denoted by α, γ is a discount factor, and H is a planning horizon.

2.2 Policies

A policy π is a function that maps every state into an action. We assume that the expert's policy used in the demonstration is deterministic.
The value of a policy π is the expected sum of the rewards received by following this policy,

$$V(\pi) = E[\sum_{t=0}^{H-1} \gamma^t R(s_t, a_t) | \alpha, \pi].$$

2.3 Apprenticeship Learning via Inverse Reinforcement Learning

- Handcrafting a reward function for matching a complex behavior is a hard problem.
- It is often easier to demonstrate examples of the desired behavior [1].
- Apprenticeship learning via inverse reinforcement learning consists in learning a reward function that explains an observed behavior.
- The reward is assumed to be a linear function of k state-action features ϕ_i,

$$R(s,a) = \sum_{i=0}^{k-1} w_i \phi_i(s,a), \qquad R = w^T \Phi,$$

where Φ is $k \times |\mathcal{S}||\mathcal{A}|$ feature matrix, defined as $\Phi[i,(s,a)] = \phi_i(s,a)$.

- The value of a policy π can be rewritten as $V(\pi) = w^T \Phi \mu_\pi$, where $\mu_\pi(s,a)$ is the expected frequency of visiting (s,a), defined as

$$\mu_\pi(s,a) = \sum_{t=0}^{H-1} \gamma^t Pr(s_t = s, a_t = a | \alpha, \pi).$$

- Given examples of an expert's policy π^*, find a reward weight vector w that satisfies

$$\forall \pi \in \mathcal{A}^{|\mathcal{S}|}: \quad \underbrace{w^T \Phi \mu_{\pi^*}}_{\text{value of the expert's policy}} \geq \underbrace{w^T \Phi \mu_\pi}_{\text{value of any other policy}}.$$

- Use the reward weight vector w to recover the expert's policy π^*.

3 Bootstrapping Maximum Margin Planning

3.1 Maximum Margin Planning (MMP) [2]

Find the reward weight vector w by solving the following optimization problem

$$\min_w \Big(\max_{\mu_\pi \in \mathcal{G}} (w^T \Phi + l) \underbrace{\mu_\pi}_{\checkmark\text{exact counts}} - w^T \Phi \underbrace{\mu_{\pi^*}}_{\times\text{empirical counts}} \Big) + \frac{\lambda}{2} \| w \|^2,$$

where $\mathcal{G}$ denotes the set of state-action average counts μ_π, satisfying Bellman flow constraints

$$\mu_\pi(s) = \alpha(s) + \gamma \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_\pi(s',a) T(s',a,s), \qquad \sum_{a \in \mathcal{A}} \mu_\pi(s,a) = \mu_\pi(s), \qquad \mu_\pi(s,a) \geq 0.$$

The loss vector l represents the cost of deviating from the expert's examples. It can be defined as $l(s,a) = 1$ if $a \neq \pi^*(s)$, and $l(s,a) = 0$ if $a = \pi^*(s)$.

3.2 Empirical Feature Counts

The feature counts $\Phi \mu_{\pi^*}$ can be analytically calculated (using Bellman flow equation) only if π^* is completely known. However, the examples cover only a part of the state space and the feature counts are approximated by using the empirical counts. Two main problems are related to this approximation:

- The empirical averages suffer from a large variance when the transition function is highly stochastic.
- The empirical averages do not take into account the known transition probabilities.

3.3 Bootstrapping Maximum Margin Planning

Assuming that the expert's policy π^* is optimal and deterministic, the term $w^T \Phi \mu_{\pi^*}$ can be replaced by $\max_{\mu \in \mathcal{G}_{\pi^*}} w^T \Phi \mu$, the value of the optimal policy according to the intermediate reward weight w (in a gradient descent algorithm) that selects the same actions as the expert in all the states that occurred in the demonstration. *Bootstrapped* Maximum Margin Planning consists in solving the following optimization problem

$$\min_w \Big(\max_{\mu_\pi \in \mathcal{G}} (w^T \Phi + l) \underbrace{\mu_\pi}_{\text{exact counts}} - \max_{\mu_{\pi'} \in \mathcal{G}_{\pi^*}} w^T \Phi \underbrace{\mu_{\pi'}}_{\text{exact counts}} \Big) + \frac{\lambda}{2} \| w \|^2,$$

where $\mathcal{G}_{\pi^*}$ is the set of vectors μ_π subject to the following modified Bellman flow constraints

$$\mu_\pi(s) = \alpha(s) + \gamma \sum_{s' \in \mathcal{S}_e} \mu_\pi(s') T(s', \pi^*(s'), s) + \gamma \sum_{s' \in \mathcal{S} \setminus \mathcal{S}_e} \sum_{a \in \mathcal{A}} \mu_\pi(s',a) T(s',a,s),$$

$$\sum_{a \in \mathcal{A}} \mu_\pi(s,a) = \mu_\pi(s), \mu_\pi(s,a) \geq 0$$

$\mathcal{S}_e$ is the set of states encountered in the examples, where the expert's deterministic policy is known.

3.4 Theoretical Results

Theorem 1 *The objective function of the bootstrapped MMP algorithm has at most* $\frac{|\mathcal{A}|^{|\mathcal{S}|}}{|\mathcal{A}|^{|\mathcal{S}_e|}}$ *different local minima.*

Theorem 2 *If there exists a reward weight vector* $w^* \in \mathbb{R}^k$*, such that the expert's policy* π^* *is the only optimal policy with* w^**, i.e.* $\arg\max_{\mu \in \mathcal{G}} w^{*T} \Phi \mu = \{\mu_{\pi^*}\}$*, then there exists* $\alpha > 0$ *such that: (i), the expert's policy* π^* *is the only optimal policy with* αw^**, and (ii), the objective function of the bootstrapped MMP algorithm has a local minimum at* αw^*.

Theorem 3 *If* Φ *is an identity matrix and* $\lambda = 0$*, then all the local minima of the objective function of the bootstrapped MMP algorithm are equal to* 0.

4 Bootstrapping LPAL

4.1 Linear Programming Apprenticeship Learning (LPAL) [3]

LPAL is based on the following observation, if the reward weights are positive and sum to 1 then $V(\pi) \geq V(\pi^*) + v$, where

$$v = \min_{\phi_i} [\underbrace{\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_\pi(s,a) \phi_i(s,a)}_{\text{exact counts}} - \underbrace{\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{\pi^*}(s,a) \phi_i(s,a)}_{\text{empirical counts}}].$$

LPAL consists in finding a policy π that maximizes the margin v. The maximal margin is found by solving a linear program where the variables are the margin v and the expected visitation frequencies $\mu_\pi(s,a)$, subject to Bellman flow constraints.

4.2 Bootstrapping Linear Programming Apprenticeship Learning

As with MMP, the feature frequencies in LPAL can be analytically calculated only when a complete policy π^* of the expert is provided. The same bound $V(\pi) \geq V(\pi^*) + v$ can be guaranteed by putting

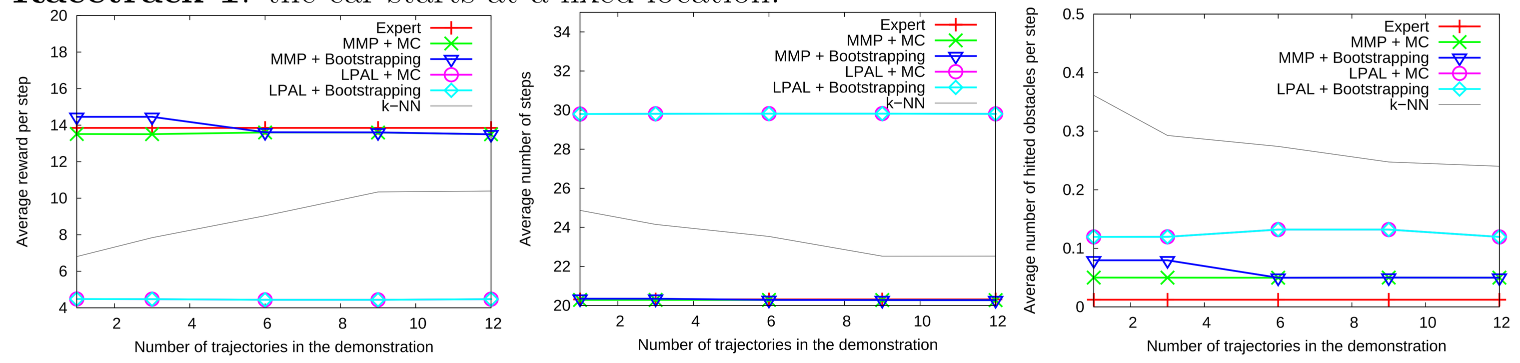
$$v = \min_{\phi_i} [\underbrace{\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_\pi(s,a) \phi_i(s,a)}_{\text{exact counts}} - \max_{\pi_i' \in \Pi^*} \underbrace{\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu_{\pi_i'}(s,a) \phi_i(s,a)}_{\text{exact counts}}],$$

where Π^* denotes the set of all the policies that select the same actions as the expert in all the states that occurred in the demonstration.
The maximal margin v is found by solving a linear program where the variables correspond to $\mu_\pi(s,a)$.
The policies π_i' are found by solving k separate optimization problems (k is the number of reward features). For each feature ϕ_i, π_i' is the policy that selects the same action as the expert in the known states, and is optimal under the reward weights w defined as $w_i = 1$, and $w_j = 0, \forall j \neq i$.

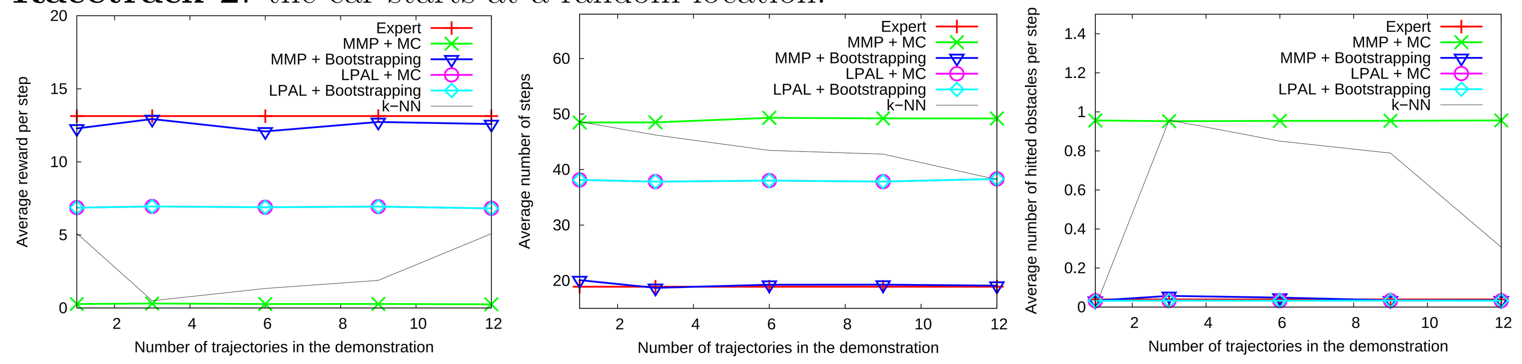
5 Experimental Results

The experiments were performed on a car race simulator, where a positive reward is given for reaching the finish line and a negative reward is given for hitting obstacles.

Racetrack 1: the car starts at a fixed location.



Racetrack 2: the car starts at a random location.



Notice that MMP using empirical feature counts (MMP+MC) achieved good performances only when the car starts at a fixed position. When the car starts at a random position, the performance of MMP is significantly improved by using the bootstrapping technique.

References

[1] Pieter Abbeel and Andrew Ng. Apprenticeship Learning via Inverse Reinforcement Learning. ICML 2004, pp. 1–8.

[2] Nathan Ratliff, J. Andrew Bagnell and Martin Zinkevich. Maximum Margin Planning. ICML 2006, pp. 729–736.

[3] Umar Syed, Michael Bowling and Robert Schapire. Apprenticeship Learning using Linear Programming. ICML 2008, pp. 1032–1039.

NIPS 2010, Vancouver, December 2010