Apprenticeship Learning via Inverse Reinforcement Learning

Yi-Jen (Rena) Ju
Department of Computer Science
National Tsing Hua University
yi-jen.ju@gapp.nthu.edu.tw

1 Introduction

The authors of this paper consider learning in a Markov decision process where the reward function is not explicitly given. In complex control tasks such as highway driving, the reward function is difficult to specify manually. When driving, humans typically trade off many different goals, such as maintaining safe following distance, keeping away from the curb, staying far from pedestrians, maintaining a reasonable speed, not changing lanes too often, etc. One way to address the problem of multiple goals is perhaps assigning weights for their relative importance to each goal. This can be eluding, however, since the relative importance of different goals might change under different circumstances. A more natural, or at least more similar to human, way of learning a complex task such as driving, is through imitating an expert. This is called *apprenticeship learning*, and it is much like a human apprentice who learns through following demonstration by a more experienced person. A number of approaches have been proposed for apprenticeship learning. Most of these methods try to mimic the demonstrator by applying a supervised learning algorithm to learn a direct mapping from the states to actions. Meanwhile, the authors of this paper consider an approach to apprenticeship learning whereby the reward function is learned. The problem of deriving a reward function from observed behavior is called inverse reinforcement learning. Specifically, the authors use a particular class of inverse reinforcement algorithm called the max-margin method, and propose an algorithm that finds a policy that performs as well as the expert, where the expert is trying to maximize a reward function that is expressible as a linear combination of known features.

The main contributions of this paper is 1) the novel inverse reinforcement learning approach to apprenticeship learning, 2) the demonstration that the algorithm terminates in a small number of iterations, and 3) that the policy output by the algorithm will attain performance close to that of the expert.

The algorithm proposed in this paper is reminiscent of the idea of support vector machines (SVM), in which we try to find a function that maximally separates classes of data. Indeed, this paper also alludes to SVM as a solver for the maximization step in the proposed algorithm. In the context of this paper, what we try to maximize is the distance between the expert performance and that of all of the policies we have found so far. As far as this author is aware, this paper is the earliest proposal of using the max-margin method to solve inverse reinforcement learning problems. Such algorithm is intuitive, and this author will elaborate in the conclusion after explanation of the details of the algorithm.

2 Problem Formulation

2.1 Preliminaries and Notations

- MDP :=

$$(S, A, T, \gamma, D, R) \tag{1}$$

- MDP\R (MDP without a reward function) :=

$$(S, A < T < \gamma, D) \tag{2}$$

- Vector of features:

$$\phi : S \longrightarrow [0,1]^k \tag{3}$$

- True reward function linearly expressed by feature vector:

$$R^*(s) = \omega^* \cdot \phi(s),\ where\, \omega^* \in \mathbb{R}^k \tag{4}$$

- Demonstration by expert i.e. optimal policy under the true reward function ($R^* = \omega^{*T}\phi$):

$$\pi_E \tag{5}$$

- Value of a policy π:

$$\begin{aligned} E_{s_0 \sim D}[V^\pi(s_0)] &= E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] \\ &= E[\sum_{t=0}^{\infty} \gamma^t \omega \cdot \phi(s_t)|\pi] \\ &= \omega \cdot E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \end{aligned} \tag{6}$$

- Expected discounted accumulated feature value vector:

$$\mu(\pi) = E_{s_0 \sim D}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \tag{7}$$

- Estimate of $\mu_E = \mu(\pi_E)$:

$$\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)}) \tag{8}$$

2.2 The optimization problem of interest

Given the MDP\R, a feature mapping ϕ and the expert's feature expectations μ_E, find a policy whose performance is close to that of the expert's on the unknown reward function $R^* = \omega^{*T}\phi$. We will find a policy $\tilde{\pi}$ such that $||\mu(\tilde{\pi}) - \mu_E||_2 \leq \epsilon$.

2.3 Technical assumptions

- The reward function is expressible as a linear combination of known features.
- $||\omega^*||_2 \leq ||\omega^*||_1 \leq 1$

2.4 Algorithms

2.4.1 Max-margin algorithm

Inverse reinforcement learning as we try to guess the reward function being optimized by the expert. Recall from Section **2.2** that we are trying to find a policy $\tilde{\pi}$ such that $||\mu(\tilde{\pi}) - \mu_E||_2 \leq \epsilon$. In order to do so we would have that for

1. Randomly pick some policy $\pi^{(0)}$, compute or approximate via Monte Carlo $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.
2. Compute $t^{(i)} = max_{\omega:||\omega||_2 \leq 1} min_{j \in \{0 \ldots (i-1)\}} \omega^T(\mu_E - \mu^{(j)})$, and let $\omega^{(i)}$ be the value of ω that attains this maximum, i.e. $\omega^{(i)} = argmax_{\omega:||\omega||_2 \leq 1} min_{j \in \{0 \ldots (i-1)\}} \omega^T(\mu_E - \mu^{(j)})$.
3. If $t^{(i)} \leq \epsilon$, terminate.

4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (\omega^{(i)})^T \phi$.
5. Compute or estimate $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set $i = i + 1$, and return to step 2.

The optimization in step 2 can be equivalently written

$$max_{t,\omega}\ t\ s.t.\ \omega^T \mu_E \geq \omega^{\mu(j)} + t, j = 0, ..., i-1, \tag{9}$$

which can be viewed as the inverse reinforcement learning step in which the algorithm is trying to find a reward function $R = \omega^{(i)} \cdot \phi$ such that $E_{s_0 \sim D}[V^{\pi_E}(S_0)] \geq E_{s_0 \sim D}[V^{\pi^{(i)}}(S_0)] + t$, i.e. a reward on which the expert does better by a margin of t than any of the i policies found in previous iterations.

, i.e. $\pi^{(i+1)} = argmax_\pi(\hat{\mu}_E - \bar{\mu}^{(i)}) \cdot \mu(\pi)$.

solved by quadratic programming problem solver

2.4.2 Projection algorithm

1. Randomly pick some policy $\pi^{(0)}$, compute or approximate via Monte Carlo $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$, $\omega^{(1)} = \mu_E - \mu^{(0)}$, and $\bar{\mu}^{(0)} = \mu^{(0)}$.
2.
   - Set $\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$
     (The orthogonal projection of μ_E onto the line through $\bar{\mu}^{(i-2)}$, $\mu^{(i-1)}$)
   - Set $\omega^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$
   - Set $t^{(i)} = ||\mu_E - \bar{\mu}^{(i-1)}||_2$
3. If $t^{(i)} \leq \epsilon$, terminate.
4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (\omega^{(i)})^T \phi$.
5. Compute or estimate $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set $i = i + 1$, and return to step 2.

3 Theoretical Analysis

3.1 Theoretical Results

3.1.1 Lemma

Lemma 3. *Let there be given an $MDP\backslash R$, features $\phi : S \mapsto [0,1]^k$, and a set of policies Π, $\bar{\mu}^{(i)} \in M$. Let $\pi^{(i+1)}$ be the optimal policy for the $MDP\backslash R$ augmented with reward $R(s) = (\hat{\mu}_E - \bar{\mu}^{(i)}) \cdot \phi(s)$, i.e. $\pi^{(i+1)} = argmax_\pi(\hat{\mu}_E - \bar{\mu}^{(i)}) \cdot \mu(\pi)$, and let $\tilde{\mu}^{(i+1)} = \frac{(\hat{\mu}_E - \bar{\mu}^{(i)}) \cdot (\mu^{(i+1)} - \bar{\mu}^{(i)})}{||\mu^{(i+1)} - \bar{\mu}^{(i)}||_2^2} (\mu^{(i+1)} - \bar{\mu}^{(i)}) + \bar{\mu}^{(i)}$, i.e. the projection of $\hat{\mu}_E$ onto the line through $\mu^{(i+1)} = \mu(\pi^{(i+1)})$, $\bar{\mu}^{(i)}$. Then $\frac{||\hat{\mu}_E - \tilde{\mu}^{(i+1)}||_2}{||\hat{\mu}_E - \bar{\mu}^{(i)}||_2} \leq \frac{k}{\sqrt{k^2 + (1-\gamma)^2 ||\hat{\mu}_E - \bar{\mu}^{(i)}||_2^2}}$.*

3.1.2 Theorems

Theorem 1. *Let an $MDP\backslash R$, features $\phi : S \mapsto [0,1]^k$, and any $\epsilon > 0$ be given. Then the apprenticeship learning algorithm (both max-margin and projection versions) will terminate with $t^{(i)} \leq \epsilon$ after at most $n = O(\frac{k}{(1-\gamma)^2 \epsilon^2} log \frac{k}{(1-\gamma)\epsilon})$ iterations.*

Theorem 2. *Let an $MDP\backslash R$, features $\phi : S \mapsto [0,1]^k$, and any $\epsilon > 0$ be given. Suppose the apprenticeship learning algorithm (either max-margin or projection version) is run using an estimate $\hat{\mu}_E$ for μ_E obtained by m Monte Carlo samples. In order to ensure that with probability at least $1 - \delta$ the algorithm terminates after at most a number of iterations n given by Eq. (14), and outputs a policy $\tilde{\pi}$ so that for any true reward $R^*(S) = \omega^{*T} \phi(s)$ ($||\omega^*||_1 \leq 1$) we have $E[\sum_{t=0}^{\infty} \gamma^t R^*(S_t) | \tilde{\pi}] \geq E[\sum_{t=0}^{\infty} \gamma^t R^*(S_t) | \pi_E] - \epsilon$, it suffices that $m \geq \frac{2k}{(\epsilon(1-\gamma))^2} log \frac{2k}{\delta}$.*

3.2 Errors in Theoretical Results

3.2.1 Partial Proof of Theorem 2.

Recall $\phi \in [0,1]^k$ so $\mu \in [0, \frac{1}{1-\gamma}]^k$. Let μ_i denote the i'th component of μ. Then applying the Hoeffding inequality on the m-sample estimate $(1-\gamma)\hat{\mu}_i$ of $(1-\gamma)\mu_i \in [0,1]$ gives

$$P((1-\gamma)|\mu_i - \hat{\mu}_i| > \tau) \leq 2 \cdot exp(-2\tau^2 m). \tag{10}$$

Using 10 for all i and the union bound gives

$$P(\exists i \in \{1...k\}.(1-\gamma)|\mu_i - \hat{\mu}_i| > \tau) \leq 2k \cdot exp(-2\tau^2 m) \tag{11}$$

which can be rewritten as

$$P((1-\gamma)||\mu_E - \hat{\mu}_E||_\infty \leq \tau) \geq 1 - 2k \cdot exp(-2\tau^2 m). \tag{12}$$

Substituting $\tau = (1-\gamma)\epsilon/(2\sqrt{k})$ gives

$$P(||\mu_E - \hat{\mu}_E||_\infty \leq \frac{\epsilon}{s\sqrt{k}}) \geq 1 - 2k \cdot exp(-2(\frac{\epsilon(1-\gamma)}{2\sqrt{k}})^2 m), \tag{13}$$

where k is the dimension of the feature vectors ϕ and feature expectations μ, and m is the number of sample trajectories used for the estimate $\hat{\mu}_E$. So if we take $m \geq \frac{2k}{(\epsilon(1-\gamma))^2} log \frac{2k}{\delta}$ then with probability at least $(1-\delta)$ we have that $||\mu_E - \hat{\mu}_E||_\infty \leq \frac{\epsilon}{2\sqrt{k}}$. Using $||\cdot||_2 \leq \sqrt{k}||\cdot||_\infty$ for k-dimensional spaces, we get

$$||\mu_E - \hat{\mu}_E||_2 \leq \frac{\epsilon}{2} \; w.p. \; 1-\delta \; for \; m \geq \frac{2k}{(\epsilon(1-\gamma))^2} log \frac{2k}{\delta} \tag{14}$$

This marks the end of the partial proof of Theorem 2. Using this partial result and the result of Theorem 1, the authors show that

$$\begin{aligned} |E[\sum_{t=0}^{\infty} \gamma^t R^*(s^{(t)})|\tilde{\pi}] - E[\sum_{t=0}^{\infty} \gamma^t R*(s^{(t)})|\pi_E]| \\ = |(\omega^*)^T(\tilde{\mu} - \mu_E)| \\ \leq \epsilon, \end{aligned} \tag{15}$$

proving the convergence property of Theorem 2, which states that in order for the expectation of total discounted rewards under the estimate $\tilde{\pi}$ and that under the optimal policy π_E to differ by a constant ϵ, it suffices that $m \geq \frac{2k}{(\epsilon(1-\gamma))^2} log \frac{2k}{\delta}$. However, the derivation for the presupposed value of m erred. The complete proof for this particular section is shown below:
Using 11 and subtract both sides from 1, we have

$$\begin{aligned} P(\neg\exists i \in \{1...k\}.(1-\gamma)|\mu_i - \hat{\mu}_i| > \tau) \\ = P((1-\gamma)||\mu_E - \hat{\mu}_E||_\infty \leq \tau) \\ \geq 1 - 2k \cdot exp(-2\tau^2 m) \end{aligned} \tag{16}$$

Substituting $\tau = (1-\gamma)\epsilon/(2\sqrt{k})$ gives

$$\begin{aligned} P(||\mu_E - \hat{\mu}_E||_\infty \leq \frac{\epsilon}{s\sqrt{k}}) \\ \geq 1 - 2k \cdot exp(-2(\frac{\epsilon(1-\gamma)}{2\sqrt{k}})^2 m) \\ = 1 - 2k \cdot exp(-(\frac{(\epsilon(1-\gamma))^2}{2\sqrt{k}})m) \end{aligned} \tag{17}$$

3.2.2 Correction

$$1 - 2k \cdot exp(-(\frac{(\epsilon(1-\gamma))^2}{2\sqrt{k}})m) = 1 - \frac{4k^2}{\delta}, \; where \; m \geq \frac{2k}{(\epsilon(1-\gamma))^2} log \frac{2k}{\delta} \tag{18}$$

4 Conclusion

Since the idea of inverse reinforcement learning was first proposed by Stuart Russell in 1998, a myriad of algorithms for inverse reinforcement learning have been proposed. There are broadly three explicit categories of algorithms: max-margin, max-entropy, and Bayesian. The max-margin algorithms attempt to estimate the rewards that maximize that margin between the optimal policy and all other policies. The intuition lies in that in each iteration of maximization, the algorithm guesses a reward function with respect to which the demonstrator outperforms all previously found policies. Since we suppose that the demonstrator is an "expert" in the learned task, it should outperform all other policies found by the algorithm. The algorithm then "tunes" its policy performance closer and closer to that of the demonstrator. Since we also assume there is a "true" reward function with respect to which the demonstrator is optimized, the algorithm should converge to the performance of the demonstrator by a small enough margin.

However, the assumption that the reward function is expressible as a linear combination of features can be limiting. More expressive reward functions would perhaps require representation by neural networks, but whether learning such functions enjoy the same convergence properties remains to be seen.

References