Apprenticeship Learning via Inverse Reinforcement Learning

Daniel, Richard, and Omar

March 2019

What is the Gap?

- In MDPs, we usually know the reward function
- For some tasks, it is hard to formulate the reward function
- We can use behavioural cloning, but the biggest weakness is under-performance in new situations

Example

Teaching your kid how to drive

Summary

- Algorithm which generates reward function to explain expert trajectories
- Reward function is compatible with arbitrary RL algorithms
- Algorithm terminates in finite time with exact information about the MDP

$$n = O\left(\frac{k}{(1-\gamma)^2\epsilon^2}\log\frac{k}{(1-\gamma)\epsilon}\right)$$

- Performance guaranteed to be within a margin of the expert's demonstrations

Variables of Interest

Inputs

- State Set S
- Action Set A
- Discount Factor $\gamma \in [0, 1)$
- Initial-state distribution D
- Feature vector $\phi : S \to [0, 1]^k$, where k is the number of features

Assumptions

- Reward is a linear combination of the ϕ and is bounded by ± 1
- Sum of absolute values of w_i must satisfy $\|w^*\|_1 \leq 1$

Unknowns

- Weight of each feature $w_i \in \mathbb{R}$
- True Reward Function $R^*(s) = w^{*\intercal} \cdot \phi(s)$

$$R^* = \underbrace{\begin{bmatrix} W_{\text{Collided Car}} \\ W_{\text{Middle Lane}} \\ \vdots \\ W_{\text{Desire K}} \end{bmatrix}^{\intercal}}_{\text{w*}} \cdot \underbrace{\begin{bmatrix} X_{\text{Collided Car}} \\ X_{\text{Middle Lane}} \\ \vdots \\ X_{\text{Desire K}} \end{bmatrix}}_{\phi\text{(s)}}$$

Feature Expectations

$$
\begin{aligned}
\mathbb{E}_{s_0 \sim D}[V^{\pi}(s_0)] &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t) | \pi\right] \\
&= w \cdot \ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi\right]
\end{aligned}
$$

We define the expected discounted accumulated feature value vector as:

$$
\mu(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi\right] \in \mathbb{R}^k
$$

Goal

Find a policy $\tilde{\pi}$ whose performance is within ϵ of the expert's, given feature vector ϕ, and we know the expert's feature expectation μ_E

Algorithm

- Recall it is assumed that $||R(x)||$ is bounded by 1, $\phi : S \rightarrow [0,1]^k$, and $||w||_1 \leq 1$
- Find policy $\tilde{\pi}$ whose performance is close to that of the expert's, on the unknown reward $R^* = {w^*}^T \phi$
- Note that $\left|x^T y\right| \leq ||x||_2 \, ||y||_2$, and $||w||_2 \leq ||w||_1 \leq 1$

$$
\begin{aligned}
&|E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi_E] - E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\tilde{\pi}]| \\
&= |w^T \mu(\tilde{\pi}) - w^T \mu_E| \\
&\leq ||w||_2 ||\mu(\tilde{\pi}) - \mu_E||_2 \\
&\leq 1 \cdot \epsilon = \epsilon
\end{aligned}
$$

Algorithm 1

Randomly pick some policy $\pi^{(0)}$, compute (or approximate via Monte Carlo) $\mu^{(0)} = \mu(\pi^{(0)})$, set $i = 1$

for $i = 1$ to N **do**

$$\begin{aligned} & max_{t,w}\, t \\ & s.t. \quad w^T \mu_E \geq w^T \mu^{(j)} + t, j = 0, ..., i-1 \\ & ||w||_2 \leq 1 \end{aligned}$$

  If $t^{(i)} \leq \epsilon$, then terminate

  Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (w^{(i)})^T \phi$.

  Compute (or estimate) $\mu^{(i)} = \mu(\pi^{(i)})$

end for

Algorithm 1: Max Margin Method

Algorithm 2

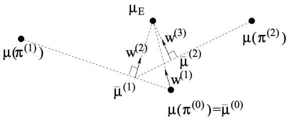

Figure: Three iterations of the projection algorithm [1].

$w^{(0)} \sim \mathbb{W}$

$\mu^{(0)} = \mathrm{RL}(w^{(0)})$

$w^{(1)} = \mu_E - \mu^{(0)}$

$\bar{\mu}^{(0)} = \mu^{(0)}$

for $i = 1$ to N **do**

$\quad \mu^{(i)} = \mathrm{RL}(w^{(i)})$

$\quad \bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)}-\bar{\mu}^{(i-2)})^T(\mu_E-\bar{\mu}^{(i-2)})}{(\mu^{(i-1)}-\bar{\mu}^{(i-2)})^T(\mu^{(i-1)}-\bar{\mu}^{(i-2)})}(\mu^{(i-1)} - \bar{\mu}^{(i-2)})$

$\quad w^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$

$\quad t^{(i)} = \left\|\mu_E - \bar{\mu}^{(i-1)}\right\|_2$

end for

Algorithm 2: Projection Method - Replace step 2 of Max Margin

Termination Point for Known μ_E

- If the expert's feature expectations vector is known, the algorithm will terminate with $t^{(i)} \leq \epsilon$ after at most:

$$n = O\left(\frac{k}{(1-\gamma)^2\epsilon^2} \log \frac{k}{(1-\gamma)\epsilon}\right)$$

When μ_E is Unknown

- We only have access to expert's policy π_E. Can we estimate μ_E?
- Given m trajectories generated by the expert or using Monte Carlo, we define the estimated feature expectation of the expert as:

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

- For the algorithm to terminate in the same number of time-steps as in the previous slide with probability $1 - \delta$, the number of trajectories needs to be:

$$m \geq \frac{2k}{(\epsilon(1-\gamma))^2} \log \frac{2k}{\delta}$$

OpenAI Gym Cartpole Example

- **Observation Space**
  - Cart Position
  - Cart Velocity
  - Pole Angle
  - Pole Velocity at Tip
- **Action Space**
  - Push Cart To Left
  - Push Cart to Right
- **Episode Termination**
  - Pole Angle is more than 12
  - center of the cart reaches the edge of the display
  - Episode length is greater than 200
- **Reward:** 1 for each step Taken

Tabular Q-Learning

- Used traditional Q-learning with tabular form
- Discretized observations and encoded into states
- Non-linear mapping of observation into features
- Training example: `https://youtu.be/Wd1xfNNo9kc`
- Google Colab Code: `https://colab.research.google.com/drive/1Tmc5fPHP9J0s-vQukLDzRywe47BNni37`

State	Action: 0	Action: 1
0001	0.3824	0.7245
0002	0	0
⋮		
9998	3.4252	0.2341
9999	-0.1234	0.2452

Figure: Q-Table

Tabular Q-Learning - Performance

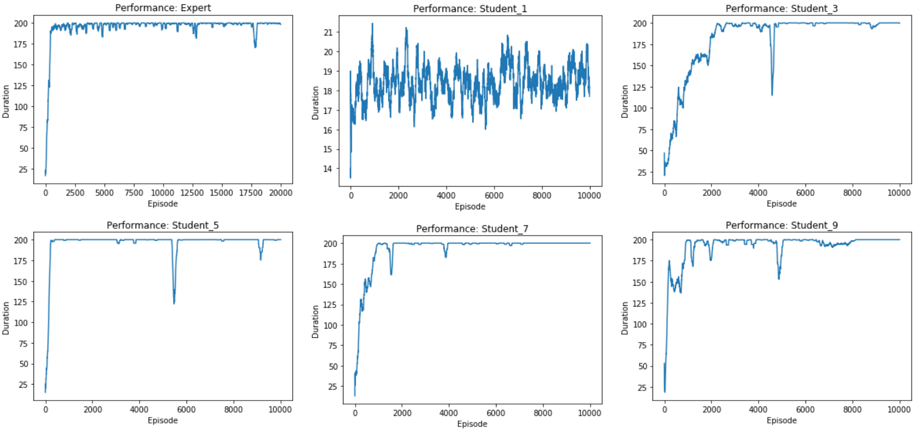

Figure: Performance of tabular Q-learning on cart-pole task

Tabular Q-Learning

- Reach similar performance to expert in small number of iterations
- Each student performs consistently through the episodes
- Consistent performance from students to students
- Difficult to extend to high dimension observation space

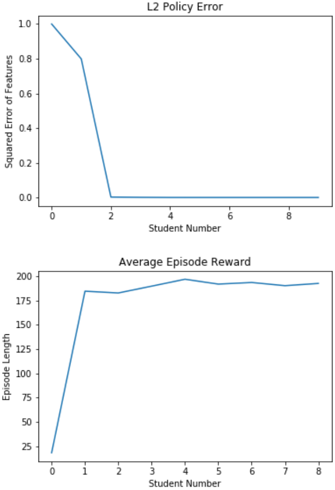

Figure: Policy progress of tabular

Double Deep Q Network

- Multi Layer Perceptron to approximate Q function
- Double Q formulation to reduce Q function bias
- Determine if method is still viable for noisy approximators
- Tuned reward function in order to aid convergence
- Training example: https://youtu.be/COAyi4-VlEw

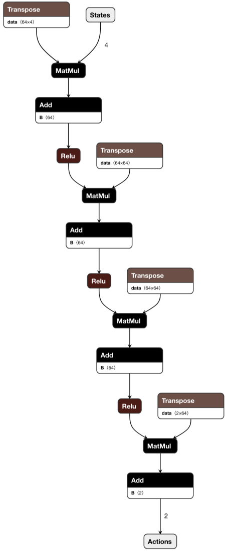

Figure: Network architecture

Double Deep Q Network - Performance

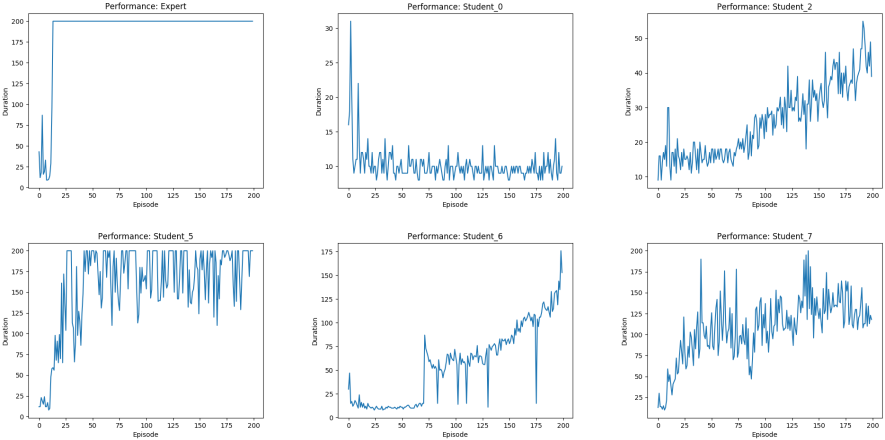

Figure: Performance of DDQN on cart-pole task

Double Deep Q Network

- MLP able to reach similar performance to expert
- Imperative to follow authors suggestion to examine the real performance of students
- Steady convergence of the models to the expert does not mean steady performance gains

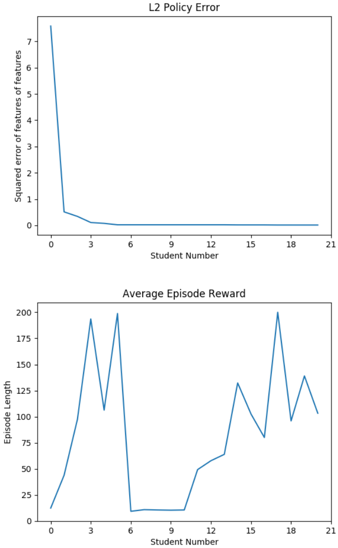

Figure: Policy progress of DDQN

Contributions

- Ability to define reward function from examples
- Worst case number of iterations, n, of algorithm with known μ_E

$$n = O\left(\frac{k}{(1-\gamma)^2\epsilon^2}\log\frac{k}{(1-\gamma)\epsilon}\right)$$

- Number of expert samples, m for $\hat{\mu}_E$ in order to gain convergence after n iterations with probability p

$$m \geq \frac{2k}{(\epsilon(1-\gamma))^2}\log\frac{2k}{\delta}$$

Contributions

- Strong empirical results show usability and robustness
- Excellent starting point to extend existing imitation learning algorithms

Figure: Using simulator defined in [2], can extend end to end training with learned reward function

Criticisms

- Cost of expert samples can be high for tasks requiring experts
- Feature mean requires full distribution support
  - Difficult for high dimensional input
  - Random starts not feasible in real world
- Suggested policy optimization over convex hull not practical
- Linear reward function is very restrictive
  - No clear indication on extension to non-linear case
  - Average feature results no longer applicable in non-linear cases
- Alternative algorithms take into account many of these issues.

Criticisms

- Feature selection is difficult
  - Convolutional Networks are not clearly applicable
  - Tuning basis functions / feature functions is expensive
- Overall algorithm very expensive bounds indicate how many times an RL agent must be fully retrained

w

Conclusion

- Method of apprenticeship learning based on inverse RL
- Algorithm converges in a small number of iterations
- Policy found will have comparable to the expert performance within a margin
- A couple of practical examples we developed based on the algorithm

References I

P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 1–. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015430

M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316