Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction

Wen Sun[1], Arun Venkatraman[1], Geoffrey J. Gordon[1], Byron Boots[2], J. Andrew Bagnell[1]

[1]School of Computer Science, Carnegie Mellon University, USA
[2]College of Computing, Georgia Institute of Technology, USA

ICML 17'/ Presenter: Ji Gao

Outline

Motivation

- For some task, there are oracle policy could be utilized. (For example, human expert)
- Imitation learning: Supervised learning on the oracle
- AggreVaTeD: Differentiable version of AggreVaTe (Aggregate Values to Imitate (Ross & Bagnell, 2014))

MDP

Defintion: Markov Decision Process

A MDP is defined as (S, A, P, C, ρ_0, H).
S : Set of states
A : Set of Actions
$P(s_{t+1}|s_t, a_t)$: Transition probablity
$C(\cdot|s_t, a_t)$: A distribution of cost (negative reward). $\bar{c}(s_t, a_t)$: Expected cost.
ρ_0 : initial distribution
H : Max Length of the MDP
Define a policy $\pi(\cdot|s)$ as a probability distribution on A.

The final distribution of the trajectories $\tau = (s_1, a_1, .., a_{H-1}, s_H)$ is determined by *pi* and the MDP, as:

$$\rho_\pi(\tau) = \rho_0(s_1) \prod_{t=2}^{H} \pi(a_{t-1}|s_{t-1}) P_{t-1}(s_t|s_{t-1}, a_{t-1})$$

Expert

- Value function:

$$Q_t^{\pi}(s_t, a_t) = \bar{c}_t(s_t, a_t) + \mathbb{E}_{s \sim P_t(\cdot|s_t, a_t), a \sim \pi(\cdot|s)} Q_{t+1}^{\pi}(s, a)$$

- Define expert policy π^* and expert oracle value $Q_t^*(s, a)$.
- Assume $Q_t^*(s, a)$ is known or can be estimated without bias.
- Idea: Approximate the export policy using an RNN.

Imitation Learning by AggreVaTe

- Use an online learner to update policies using the loss function at episode n:
  $l_n(\pi) = \frac{1}{H} \sum_{t=1}^{H} \mathbb{E}_{s_t}[\mathbb{E}_{a \sim \pi}[Q_t^*(s_t, a)]]$
- Specifically, the algorithm use Follow-the-Leader to update polices:
  $\pi_{n+1} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{n} l_n(\pi)$
  Π is a predefined convex set.
- After N iterations, the algorithm can find a policy with:
  $\mu(\hat{\pi}) \leq \mu(\pi^*) - \epsilon_N + O(\ln(N)/N)$
  Where $\epsilon_N = [\sum_{n=1}^{N} l_n(\pi^*) - \min_\pi \sum_{n=1}^{N} l_n(pi)]/N$
- Can outperform the original π^* when π^* is not optimal in the loss.

Gradient of the policy

- Suppose the policy π is parametrized by θ
- If actions are discrete, the gradient of $l_n(\pi_\theta)$ is:

$$\nabla_\theta l_n(\theta) = \frac{1}{H}\sum_{t=1}^{H}\mathbb{E}_{\pi_{\theta_n}}\sum_a \nabla_\theta \pi(a|s_t;\theta)Q_t(s_t,a)$$

- If the actions are continuous, the score function must be changed to

$$l_n(\pi_\theta) = \frac{1}{H}\mathbb{E}_{\tau\sim\rho_{\pi_{\theta_n}}}\sum_{t=1}^{H}\frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_n)}Q_t^*(s_t,a_t)$$

  In this form, the gradient is

$$\nabla_\theta l_n(\theta) = \frac{1}{H}\mathbb{E}_{\tau\sim\rho_{\pi_{\theta_n}}}\sum_{t=1}^{H}\nabla_\theta \ln(\pi(a|s_t;\theta_n))Q_t(s_t,a_t)$$

- Then the θ could be efficiently updated via gradient descent.

Natural Gradient

- If the parameter space is not an Euclidean space, gradient might be suboptimal.
- Natural Gradient: The steepest direction of change of a function whose manifold is on a Riemannian space.
- Euclidean space with orthonormal $|dw|^2 = \sum_i dw_i^2$
  Riemannian space: $|dw|^2 = \sum_{i,j} g_{ij} w_i w_j$, where $G = g_{ij}$ is the Riemannian metric tensor.
- In the case of MDP, the trajectory is a variable in Riemannian space. The Fisher Information matrix is:
  $I(\theta_n) = \frac{1}{H^2} \mathbb{E}_{\tau \sim \rho_{\pi_{\theta_n}}} \nabla_{\theta_n} \log(\rho_{\pi_{\theta_n}}(\tau)) \nabla_{\theta_n} \log(\rho_{\pi_{\theta_n}}(\tau))^T$
- Natural gradient update:

$$\theta_{n+1} = \theta_n - \eta_n I(\theta_n)^{-1} \nabla_\theta l_n(\theta)$$

Practical way

- Use sampling to approximate gradient:

$$\tilde{\nabla}_\theta l_n(\theta) = \frac{1}{HK} \sum_{t=1}^{H} \sum_{i=1}^{K} \sum_{a} \nabla_\theta \pi(a|s_t^i; \theta) Q_t(s_t^i, a)$$

Algorithm

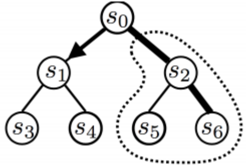
Use an annealing way to train:

Algorithm 1 AggreVaTeD (Differentiable AggreVaTe)

```
Input: The given MDP and expert π*. Learning rate {η_n}. Schedule rate {α_i}, α_n → 0, n → ∞.
Initialize policy π_{θ_1} (either random or supervised learning).
for n = 1 to N do
    Mixing policies: π̂_n = α_n π* + (1 − α_n) π_{θ_n}.
    Starting from ρ_0, roll in by executing π̂_n on the given MDP to generate K trajectories {τ_i^n}.
    Using Q* and {τ_i^n}_i, compute the descent direction δ_{θ_n} (Eq. 10, Eq. 11, Eq. 12, Eq. 13, or CG).
    Update: θ_{n+1} = θ_n − η_n δ_{θ_n}.
end for
Return: the best hypothesis π̂ ∈ {π_n}_n on validation.
```

Compare IL and RL

- Suppose an MDP is a tree with $S = 2^K - 1$ states, and only leaf have a cost, random sampled from a given distribution.



- RL have the regret $E[R_N] \geq \Omega(\sqrt{SN})$.
- However, IL have the regret $R_N \leq O(\ln S)$ with the optimal Q^*, because it can directly know which way to go.
- In the case that the query of Q^* is noisy, it is proved that AggreVaTeD can achieve the regret bound for the tree MDP with at least $1 - \delta$ probablity:

$$R_N \leq O(\ln(S)(\sqrt{\ln(S)N}) + \sqrt{\ln(2/\delta)N})$$

Near Optimality

- In the general case, with access to an unbiased estimates of Q^*, the algorithm achives the regret upper bound:

$$R_N \leq O(HQ^e_{max}\sqrt{|S|\ln(|A|)N})$$

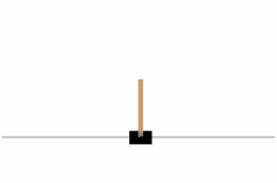
Q^e_{max} is the largest cost-to-go value of the expert.

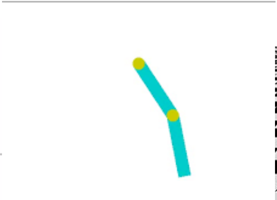
- Also, it is proved that there exists an MDP(H=1) that with acccess to the unbiased estimates of Q^*, any imitation learning algorithm have:

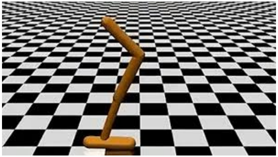
$$E[R_N] \geq \Omega(\sqrt{|S|\ln(|A|)N})$$

Experiment1 - Simulations of robots using OpenAI Gym

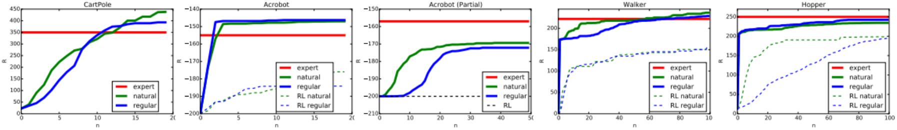
- Simulations of robots using OpenAI Gym
- Tasks:
  - Cartpole
  - Acrobot
  - Hopper
  - Walker







Result



(a) Cartpole (b) Acrobot (c) Acrobot (POMDP) (d) Walker (e) Hopper

Figure 2. Performance (cumulative reward R on y-axis) versus number of episodes (n on x-axis) of AggreVaTeD (blue and green), experts (red), and RL algorithms (dotted) on different robotics simulators.

Experiment2 - Handwritten Algebra parsing

- Parse handwritten algebra from raw image
- RNN policy from (Sutskever et al., 2014) paper

Arc-Eager	AggreVaTeD (LSTMs)	AggreVaTeD (NN)	SL-RL (LSTMs)	SL-RL(NN)	RL (LSTMs)	RL (NN)	DAgger	SL (LSTMs)	SL (NN)	Random
Regular	**0.924**±0.10	0.851±0.10	0.826± 0.09	0.386±0.1	0.256±0.07	0.227±0.06	0.832±0.02	0.813±0.1	0.325±0.2	∼0.150
Natural	0.915±0.10	0.800±0.10	0.824±0.10	0.345±0.1	0.237±0.07	0.241±0.07				

Table 1. Performance (UAS) of different approaches on handwritten algebra dependency parsing. *SL* stands for supervised learning using expert's samples: maximizing the likelihood of expert's actions under the sequences generated by expert itself. *SL-RL* means RL with initialization using SL. *Random* stands for the initial performances of random policies (LSTMs and NN). The performance of DAgger with Kernel SVM is from (Duyck & Gordon, 2015).