Deeply AggreVaTeD: Differentiable Imitation Learning for Sequential Prediction

Wen Sun\textsuperscript{1}, Arun Venkatraman\textsuperscript{1}, Geoffrey J. Gordon\textsuperscript{1}, Byron Boots\textsuperscript{2}, J. Andrew Bagnell\textsuperscript{1}

\textsuperscript{1}School of Computer Science, Carnegie Mellon University, USA
\textsuperscript{2}College of Computing, Georgia Institute of Technology, USA

ICML 17'/ Presenter: Ji Gao
Outline

1 Motivation
2 Background
3 Method
4 Analysis
5 Experiments
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))
Definition: Markov Decision Process

A MDP is defined as \((S, A, P, C, \rho_0, H)\).

\(S\): Set of states
\(A\): Set of Actions

\(P(s_{t+1}|s_t, a_t)\): Transition probability

\(C(\cdot|s_t, a_t)\): A distribution of cost (negative reward). \(\bar{c}(s_t, a_t)\): Expected cost.

\(\rho_0\): 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 \(\tau = (s_1, a_1, \ldots, a_{H-1}, s_H)\) is determined by \(\rho\) and the MDP, as:

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

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

- Define expert policy \( \pi^* \) 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.
Use an online learner to update policies using the loss function at episode n:
\[ l_n(\pi) = \frac{1}{H} \sum_{t=1}^{H} \mathbb{E}_{s_t} \left[ \mathbb{E}_{a \sim \pi} \left[ Q^*_t(s_t, a) \right] \right] \]

Specifically, the algorithm use Follow-the-Leader to update polices:
\[ \pi_{n+1} = \arg \min_{\pi \in \Pi} \sum_{i=1}^{n} l_n(\pi) \]
\( \Pi \) 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 \( \epsilon_N = \left[ \sum_{n=1}^{N} l_n(\pi^*) - \min_{\pi} \sum_{n=1}^{N} l_n(\pi) \right] / N \)

Can outperform the original \( \pi^* \) when \( \pi^* \) is not optimal in the loss.
Gradient of the policy

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

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

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

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

In this form, the gradient is

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

- Then the $\theta$ 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:

$$\theta_{n+1} = \theta_n - \eta_n I(\theta_n)^{-1} \nabla_{\theta} I_n(\theta)$$
Use sampling to approximate gradient:

\[
\tilde{\nabla}_\theta l_n(\theta) = \frac{1}{HK} \sum_{t=1}^{H} \sum_{i=1}^{K} \sum_a \nabla_\theta \pi(a|s^i_t; \theta) Q_t(s^i_t, a)
\]
Algorithm

Use an annealing way to train:

**Algorithm 1** AggreVaTeD (Differentiable AggreVaTe)

1: **Input:** The given MDP and expert $\pi^*$. Learning rate $\{\eta_n\}$. Schedule rate $\{\alpha_i\}$, $\alpha_n \to 0, n \to \infty$.
2: Initialize policy $\pi_{\theta_1}$ (either random or supervised learning).
3: **for** $n = 1$ to $N$ **do**
4: Mixing policies: $\hat{\pi}_n = \alpha_n \pi^* + (1 - \alpha_n) \pi_{\theta_n}$.
5: Starting from $\rho_0$, roll in by executing $\hat{\pi}_n$ on the given MDP to generate $K$ trajectories $\{\tau^n_i\}$.
6: Using $Q^*$ and $\{\tau^n_i\}_i$, compute the descent direction $\delta_{\theta_n}$ (Eq. 10, Eq. 11, Eq. 12, Eq. 13, or CG).
7: Update: $\theta_{n+1} = \theta_n - \eta_n \delta_{\theta_n}$.
8: **end for**
9: **Return:** the best hypothesis $\hat{\pi} \in \{\pi_n\}_n$ 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 \( 1 - \delta \) probability:

\[
R_N \leq O(\ln(S)(\sqrt{\ln(S)N}) + \sqrt{\ln(2/\delta)N})
\]
Near Optimality

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

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

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

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

$$E[R_N] \geq \Omega(\sqrt{|S| \ln(|A|)N})$$
Experiment 1 - Simulations of robots using OpenAI Gym

- Simulations of robots using OpenAI Gym
- Tasks:
  - Cartpole
  - Acrobot
  - Hopper
  - Walker
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.
Parse handwritten algebra from raw image

RNN policy from (Sutskever et al., 2014) paper

<table>
<thead>
<tr>
<th>Are-Enger</th>
<th>AggreVaTeD (LSTMs)</th>
<th>AggreVaTeD (NN)</th>
<th>SL-RL (LSTMs)</th>
<th>SL-RL (NN)</th>
<th>RL (LSTMs)</th>
<th>RL (NN)</th>
<th>DAgger</th>
<th>SL (LSTMs)</th>
<th>SL (NN)</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>0.924±0.10</td>
<td>0.851±0.10</td>
<td>0.826±0.09</td>
<td>0.386±0.1</td>
<td>0.256±0.07</td>
<td>0.227±0.06</td>
<td>0.832±0.02</td>
<td>0.813±0.1</td>
<td>0.325±0.2</td>
<td>~0.150</td>
</tr>
<tr>
<td>Natural</td>
<td>0.915±0.10</td>
<td>0.800±0.10</td>
<td>0.824±0.10</td>
<td>0.345±0.1</td>
<td>0.237±0.07</td>
<td>0.241±0.07</td>
<td>0.832±0.02</td>
<td>0.813±0.1</td>
<td>0.325±0.2</td>
<td>~0.150</td>
</tr>
</tbody>
</table>

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).