Off-Policy Correction and Batch Learning

Deep Reinforcement Seminar FS 2020, 03.03.2020

Xiang Li
On-policy algorithm := algorithm requiring $\mu = \pi$.

Why do we want an off-policy algorithm?

- Can choose a better $\mu$
- Sample efficient
Actor-Critic Algorithm

- **Goal**: find policy \( \pi : S \times A \rightarrow [0,1] \) such that \( V^{\pi} \) is large
- **Algorithm**: 
  - Repeat for \( t = 1, \ldots \)
    - Sample trajectories \( \{(s_i, a_i, r_i, s_i') : i \in I \} \subseteq S \times A \times \mathbb{R} \times S \), where \( \forall i \in I : a_i \sim \mu(s_i) \)
    - Improve the policy \( \pi \)
    - Estimate the value \( V^{\pi} \)
Actor-Critic Algorithm: Details

„Policy Improvement“
Improve the policy $\pi_w$
Find $w$ such that $V^{\pi_w}(s)$ is large
• Using gradient ascent:
  • $w \leftarrow w + \eta \nabla_w V^{\pi_w}_\theta(s)$

„Policy Evaluation“
Estimate $V^{\pi_w}$
Find $\theta$ such that $V^\pi \approx V^\pi_\theta$
• i.e. $\min_\theta [V^\pi_\theta(s) - V^\pi(s)]^2$
• Using gradient descent:
  • $\theta \leftarrow \theta + \eta' [V^\pi_\theta(s) - y] \nabla_\theta V^{\pi}_\theta(s)$
  • $y$ is an estimate of $V^\pi(s)$,
    e.g. $y = r + \gamma V_\theta(s_{next})$
Policy Evaluation: How to estimate $V^\pi(s_0)$?

• **Given**: $s_0, a_0, r_0, s_1, ..., a_{n-1}, r_{n-1}, s_n; a_i \sim \mu(s_i)$

• Approach 1: $y := r_0 + \gamma V(s_1)$ (Abbreviate $V := V^\pi_\theta$)

• Approach 2: $y := r_0 + \gamma r_1 + \cdots + \gamma^{n-1}r_{n-1} + \gamma^n V(s_{n+1})$

  \[= V(s_0) + \sum_{k=0}^{n-1} \gamma^k (r_k + \gamma V(s_{k+1}) - V(s_k))\]

• Approach 3: $y := V(s_0) + \sum_{k=0}^{n-1} \gamma^k (r_k + \gamma V(s_{k+1}) - V(s_k)) \prod_{j=0}^{k} \min \left(1, \frac{\pi(s_j,a_j)}{\mu(s_j,a_j)} \right)$

Weights
Policy Improvement: How to estimate $\nabla_w V^{\pi_w}(s_0)$?

• $\nabla_w V^{\pi_w}(s_0) = E_{\pi_w}[Q^{\pi_w}(s, a) \nabla_w \ln(\pi_w(s, a))]$

  $= E_\mu \left[ Q^{\pi_w}(s, a) \nabla_w \ln(\pi_w(s, a)) \frac{\pi_w(s, a)}{\mu(s, a)} \right]$
Effect of Off-Policy Correction

Performance on 5 DeepMind Lab tasks

<table>
<thead>
<tr>
<th></th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Replay</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V-trace</td>
<td>46.8</td>
<td>32.9</td>
<td>31.3</td>
<td>229.2</td>
<td>43.8</td>
</tr>
<tr>
<td>No-correction</td>
<td>40.3</td>
<td>29.1</td>
<td>5.0</td>
<td>94.9</td>
<td>16.1</td>
</tr>
<tr>
<td>With Replay</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V-trace</td>
<td>47.1</td>
<td>35.8</td>
<td>34.5</td>
<td>250.8</td>
<td>46.9</td>
</tr>
<tr>
<td>No-correction</td>
<td>35.0</td>
<td>21.1</td>
<td>2.8</td>
<td>85.0</td>
<td>11.2</td>
</tr>
</tbody>
</table>
Performance of Impala
**Figure 4. Top Row:** Single task training on 5 DeepMind Lab tasks. Each curve is the mean of the best 3 runs based on final return. IMPALA achieves better performance than A3C. **Bottom Row:** Stability across hyperparameter combinations sorted by the final performance across different hyperparameter combinations. IMPALA is consistently more stable than A3C.
On/Off-policy & Offline/Online learning

• Task: find a target policy $\pi$ using data $D$ generated by behavioural policy $\mu$
  • $D \equiv \{(s_i, a_i, r_i, s'_i) : i \in I\} \subset S \times A \times \mathbb{R} \times S$, where $\forall \; i \in I : a_i \sim \mu(s_i)$

• On-policy Algorithm := an algorithm working well only for $\mu = \pi$
• Off-policy Algorithm := an algorithm working well for all $\mu$

• Online learning := able to choose a behavioural policy and interact with the environment
• Offline/batch learning := no interaction possible. $\mu$ generally not known.
The End