Improving Stability in Deep Reinforcement Learning with Weight Averaging

Evgenii Nikishin$^1$, Pavel Izmailov$^2$, Ben Athiwaratkun$^2$, Dmitri Podoprikhin$^{1,3}$
Timur Garipov$^4$, Pavel Shvechikov$^1$, Dmitry Vetrov$^{1,3}$, Andrew Gordon Wilson$^2$

$^1$National Research University Higher School of Economics, $^2$Cornell University
$^3$Samsung-HSE Laboratory, $^4$Samsung AI Center in Moscow

Outline

• Deep reinforcement learning (RL) methods are notoriously unstable during training
• Stochastic weight averaging (SWA) is a technique based on averaging the weights collected during training with an SGD-like method
• We propose to apply SWA, in order to reduce the effect of noise on training
• We show that SWA stabilizes solutions and improves the average rewards

Background

• Advantage Actor-Critic (A2C) is a standard RL algorithm, often applied to problems with discrete action spaces.
• Deep Deterministic Policy Gradient (DDPG) is another standard RL algorithm, but suitable for continuous action spaces.

Stochastic weight averaging

- Use learning rate schedule that doesn’t decay to zero, e.g. cyclical or high constant at the end of training
- Average weights at the end of each of the last $K$ epochs or at the end of each cycle

SWA for RL

SWA was shown to find solutions with better generalization in both supervised and semi-supervised learning. We introduce several modifications for RL:

\[
\begin{align*}
\bar{w}_{n+1}^{\text{SWA}} & = \frac{n w_{n+1}^{\text{SWA}} + w_n}{n+1} \\
ns_{n+1}^{\text{SWA}} & = ns_n^{\text{SWA}} + 1
\end{align*}
\]

- Use constant learning rate
- Use adaptive optimizers (Adam, RMSProp)
- Collect weights once every $c$ training steps after the initial pre-training stage

Results

Average cumulative rewards of A2C for CartPole

- Even on simple tasks A2C forgets optimal policy
- SWA is able to stabilize performance

Discussion

- In SWA averaging does not affect the training procedure; using SWA averages during training could stabilize convergence and accelerate training
- Theoretical justification for averaging in RL context

A2C on Atari environments

<table>
<thead>
<tr>
<th>ENV NAME</th>
<th>A2C</th>
<th>A2C + SWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakout</td>
<td>522 ± 34</td>
<td>703 ± 60</td>
</tr>
<tr>
<td>Qbert</td>
<td>18777 ± 778</td>
<td>21272 ± 655</td>
</tr>
<tr>
<td>SpaceInvaders</td>
<td>7727 ± 1121</td>
<td>21676 ± 8897</td>
</tr>
<tr>
<td>Seaguest</td>
<td>1779 ± 4</td>
<td>1795 ± 4</td>
</tr>
<tr>
<td>CrazyClimber</td>
<td>147030 ± 10239</td>
<td>139752 ± 11618</td>
</tr>
<tr>
<td>BeamRider</td>
<td>9999 ± 402</td>
<td>11321 ± 1065</td>
</tr>
</tbody>
</table>

DDPG on MuJoCo environments

<table>
<thead>
<tr>
<th>ENV NAME</th>
<th>DDPG</th>
<th>DDPG + SWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hopper</td>
<td>613 ± 683</td>
<td>1615 ± 1143</td>
</tr>
<tr>
<td>Walker2d</td>
<td>1803 ± 96</td>
<td>2457 ± 241</td>
</tr>
<tr>
<td>Half-Cheetah</td>
<td>3825 ± 1187</td>
<td>4228 ± 1117</td>
</tr>
<tr>
<td>Ant</td>
<td>865 ± 899</td>
<td>1051 ± 696</td>
</tr>
</tbody>
</table>

- We use OpenAI baselines’ implementations of A2C and DDPG with default hyperparameters
- SWA achieves consistent improvement with both methods