Improving Stability in Deep Reinforcement Learning with Weight Averaging

Evgenii Nikishin^1 **Pavel Izmailov**^{2} **Ben Athiwaratkun**^{2} **Dmitrii 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

Abstract

Deep reinforcement learning (RL) methods are notoriously unstable during training. In this paper, we focus on model-free RL algorithms where we observe that the average reward is unstable throughout the learning process and does not increase monotonically given more training steps. Furthermore, a highly rewarded policy, once learned, is often forgotten by an agent, leading to performance deterioration. These problems are partly caused by fundamental presence of noise in gradient estimators in RL. In order to reduce the effect of noise on training, we propose to apply stochastic weight averaging (SWA), a recent method that averages weights along the optimization trajectory. We show that SWA stabilizes the model solutions, alleviates the problem of forgetting the highly rewarded policy during training, and improves the average rewards on several Atari and MuJoCo environments.

1 INTRODUCTION

Deep reinforcement learning (RL) methods have made significant progress over the last several years. However, the training stability still remains an important issue for deep RL. In Figure 1, we show the cumulative rewards as a function of the number of interactions with the environment for A2C method [Barto et al., 1983, Mnih et al., 2016] for CartPole environment. As we can see, while the agent reaches near-optimal average cumulative reward of approximately 200, the trajectory of average reward is highly non-monotonic. In particular, the agent often forgets the highly rewarded policy, and the average cumulative reward decreases significantly. The dependence of learning data distribution on agent's decisions may further exacerbate the consequences of noise.

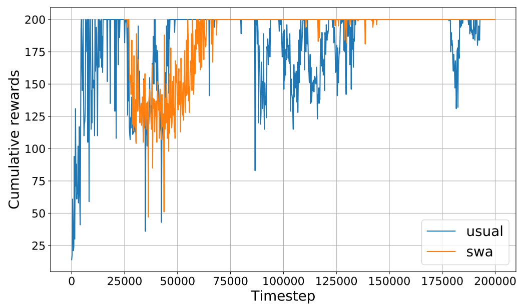

Figure 1: An example of A2C reward trajectory on CartPole environment with and without SWA

Fundamental presence of noise in gradient estimators is one of the most important reasons for learning curve's non-monotonicity and forgetting. This noise may be due to different factors ranging from stochastic transitions to delayed rewards and exploratory actions. Methods reducing the impact of such noise (either directly or indirectly), include the following: adding regularization to objective function [Farahmand et al., 2009]; augmenting an objective function with auxiliary losses [Jaderberg et al., 2016]; constraining gradient updates of policy parameters [Schulman et al., 2015a, 2017, Bhatia et al., 2017]; replacing an objective with its more robust counterpart [Gilbert and Weng, 2016]. In contrast to aforementioned approaches, our work investigates stabilization benefits unrelated to objective function modification and gradient update constraints. More specifically, we propose to apply the stochastic weight averaging (SWA) [Izmailov et al., 2018] in a RL context. SWA is a training technique based on averaging weights of the models collected during training, which was shown to improve generalization for both supervised and semi-supervised [Athiwaratkun et al., 2018] learning.

We show that applying SWA to A2C and DDPG it is possible to improve the solutions found by these methods and in particular alleviate the issue of forgetting the highly rewarded policy (see Figure 1). Also, SWA can substantially improve the average cumulative reward obtained by an agent trained on several Atari and MuJoCo environments.

2 BACKGROUND

In this section, we briefly describe stochastic weight averaging (SWA), advantage actor-critic (A2C) and deep deterministic policy gradient (DDPG) algorithms.

2.1 STOCHASTIC WEIGHT AVERAGING

SWA [Izmailov et al., 2018] is a recently proposed training technique that allows finding solutions with better generalization in supervised and semi-supervised learning. SWA is based on averaging the weights collected during training with an SGD-like method. In supervised learning, the weights are collected at the end of each training epoch. Izmailov et al. [2018] use a constant or cyclical learning rate schedule to ensure that the optimization does not converge to a single solution and instead continues to explore the region of high-performing networks.

2.2 ADVANTAGE ACTOR-CRITIC AND DEEP DETERMINISTIC POLICY GRADIENT

We consider a standard reinforcement learning setting where an agent aims to maximize expected discounted cumulative reward

$$J(\pi) \;=\; \mathbb{E}_\pi\left[R_0\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{k+1}\right], \qquad (1)$$

while starting in some initial state $s_0 \sim p(s_0)$, sampling actions $a_t \sim \pi(a_t|s_t)$, receiving rewards $r_{t+1} \sim p(r_{t+1}|s_t, a_t)$ and observing new states $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. Action-value function $q_\pi(s_t, a_t)$ measures the value of state-action pair (s_t, a_t) following policy π, while value function $v_\pi(s_t)$ measures the value of state pair s_t following policy π:

$$q_\pi(s_t, a_t) = \mathbb{E}_\pi\left[R_t|s_t, a_t\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t, a_t\right],$$

$$v_\pi(s_t) = \mathbb{E}_\pi\left[R_t|s_t\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t\right].$$

Advantage actor-critic (A2C) [Barto et al., 1983, Mnih et al., 2016] algorithm parametrizes policy by a neural network (actor) and approximates the advantage function, the difference between action-value and value functions, by another neural network (critic). The critic is then used in an estimator of the gradient of the objective (1) with respect to policy parameters. A2C is a model-free algorithm, and is commonly used in environments with discrete action spaces.

Deep deterministic policy gradient (DDPG) [Lillicrap et al., 2015], similarly to A2C algorithm, uses actor and critic networks, but is based on deterministic policy gradient [Silver et al., 2014] which is applicable to environments with continuous action spaces.

We choose A2C and DDPG to show that SWA stabilizes solutions and increases agent's overall performance for both discrete and continuous control tasks.

Algorithm 1 SWA

```
Require:
  Initial weights $\hat{w}$, SWA update frequency $c$, number of
  training steps $n$
Ensure: $w_{\text{SWA}}$ {SWA model}
  $w \leftarrow \hat{w}$ {Initialize weights with $\hat{w}$}
  $w_{\text{SWA}} \leftarrow w$
  $n_{\text{SWA}} \leftarrow 1$ {Number of weights in SWA average}
  for $i \leftarrow 1, 2, \ldots, n$ do
    $w \leftarrow \text{Upd}(w)$ {Perform optimization update}
    if $\text{mod}(i, c) = 0$ then
      $w_{\text{SWA}} \leftarrow \frac{n_{\text{SWA}} \cdot w_{\text{SWA}} + w}{n_{\text{SWA}} + 1}$
      $n_{\text{SWA}} \leftarrow n_{\text{SWA}} + 1$
    end if
  end for
```

3 SWA FOR A2C AND DDPG

In order to apply SWA to A2C and DDPG algorithms, we introduce frequency c of updating the SWA weights. This is in contrast to supervised learning where the SWA weights are usually updated in the end of every epoch. We summarize the SWA procedure in algorithm 1. To initialize the weights $\hat{w}$ we use the weights of the model that was trained for a fixed amount of steps with conventional training. Then we apply SWA for the weights of both actor and critic networks. Note that SWA model has a separate set of weights which does not influence the optimization process.

We demonstrate that SWA reduces the learning instabilities caused by noisy gradient updates, resists forgetting of highly rewarded policies and also increases the overall agent's performance.

The intuition behind applying SWA is that by averaging

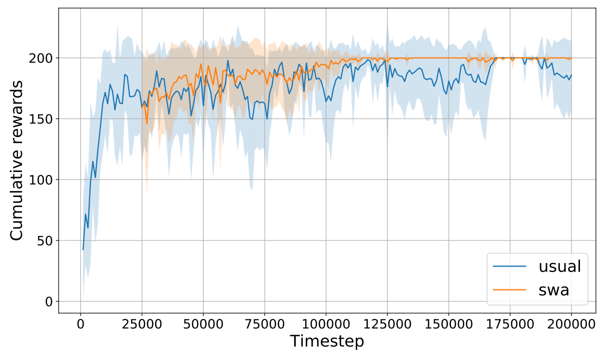

Figure 2: Average cumulative reward trajectories of A2C for CartPole environment with and without SWA. Plots are averaged over 5 runs, shaded region represents the standard deviation

multiple samples from optimization trajectory we reduce the effect of noise in the gradients, since noise in individual samples cancels out. In convex stochastic optimization, similar averaging techniques for SGD are known to deliver optimal convergence rates [Polyak and Juditsky, 1992, Rakhlin et al., 2012, Bach and Moulines, 2013]. Lakshminarayanan and Szepesvari [2018] show how averaging could be applied to a linear value-function evaluation problem, yielding the optimal convergence rate for TD(0) learning. This result appears promising for establishing the connection between the stochastic optimization theory and our findings, especially in the light of policy gradient algorithms being the approximate versions of policy iteration, which, in turn, consists of interleaved policy evaluation and policy improvement. We view a theoretical exploration as a promising direction for future work.

4 EXPERIMENTS

We evaluate SWA on the CartPole environment, 6 Atari games and 4 MuJoCo environments. For CartPole, we have implemented A2C with Generalized Advantage Estimation [Schulman et al., 2015b]. For Atari games, we use OpenAI baselines' [Dhariwal et al., 2017] implementation of A2C with default hyperparameters. For MuJoCo environments, we also use OpenAI baselines implementation of DDPG with default hyperparameters.

Figure 2 shows the average cumulative reward versus the number of training steps for the usual training solution and SWA. The usual A2C agent reaches the highest possible average reward of 200 during training, but the average reward deteriorates because of instability in training (notice the high variance of the average reward over multiple runs). On the other hand, the SWA solution gradually increases average rewards and maintains the optimal average reward of 200 once it reaches it. For CartPole, we update SWA weights after every episode.

To further analyze the effect of SWA on A2C we apply it to several Atari benchmark games. We first pretrain the methods for 33M timesteps for Breakout and Qbert and for 55M timesteps for other Atari games. We use smaller number of training steps for Breakout and Qbert because these environments are simpler. Then we continue training for the same amount of timesteps with and without SWA. Here we set c to 100 updates, which corresponds to 8000 timesteps for default hyperparameters in the implementation of A2C. We report the results in Table 1.

Table 1: Average final cumulative reward for 6 games for A2C and A2C + SWA solutions. The experiments are repeated 3 times to estimate standard deviation.

ENV NAME	**A2C**	**A2C + SWA**
Breakout	522 ± 34	$\mathbf{703 \pm 60}$
Qbert	18777 ± 778	$\mathbf{21272 \pm 655}$
SpaceInvaders	7727 ± 1121	$\mathbf{21676 \pm 8897}$
Seaquest	1779 ± 4	$\mathbf{1795 \pm 4}$
CrazyClimber	$\mathbf{147030 \pm 10239}$	139752 ± 11618
BeamRider	9999 ± 402	$\mathbf{11321 \pm 1065}$

The performance increase due to SWA also holds for DDPG applied to continous control tasks. For DDPG, we set $c = 2000$ timesteps, which corresponds to 1 epoch of training for default hyperparameters. Then, we pretrain all agents for 1.4M steps and continue training with and without SWA for 0.4M time steps. The results are reported in Table 2.

Table 2: Average final cumulative reward for 4 MuJoCo environments for DDPG and DDPG + SWA solutions. The experiments are repeated 3 times to estimate the standard deviation.

ENV NAME	**DDPG**	**DDPG + SWA**
Hopper	613 ± 683	$\mathbf{1615 \pm 1143}$
Walker2d	1803 ± 96	$\mathbf{2457 \pm 241}$
Half-Cheetah	3825 ± 1187	$\mathbf{4228 \pm 1117}$
Ant	865 ± 899	$\mathbf{1051 \pm 696}$

Note that for all tested environments except CrazyClimber SWA solution achieves higher average cumulative reward.

5 DISCUSSION AND FUTURE WORK

In this paper, we have proposed to apply SWA to reinforcement learning problems. Our results on A2C and DDPG suggest that the SWA solutions alleviate the learning instabilities caused by presence of noise in gradient estimators and attain higher average cumulative reward on a range of problems.

Inspired by the results obtained, we plan to explore the effect of SWA on other model-free RL algorithms such as A3C, DQN, PPO and model-based algorithms. In addition, our work so far involves an offline use of SWA; that is, the averaging does not affect the training procedure. Modification of the training procedure based on weight averaging can potentially help in stabilization and training acceleration. For example, we can use the SWA solutions to influence action selection or to calculate the value function.

We see theoretical justification of weight averaging in RL context as another promising research direction. In image classification scenarios, Athiwaratkun et al. [2018] found that applying SWA helps to improve the performance due to the approximate convexity of the error surface and proposed averaging the weights corresponding to higher learning rates of a cyclical schedule. Similarly to this finding, an analysis of the RL loss surface can reveal new approaches to weight averaging for improving stability in reinforcement learning settings.

Acknowledgements

This work was partly supported by Samsung Research, Samsung Electronics, Sberbank AI Lab and the Russian Science Foundation grant 17-11-01027.

References

B. Athiwaratkun, M. Finzi, P. Izmailov, and A. G. Wilson. Improving Consistency-Based Semi-Supervised Learning with Weight Averaging. *ArXiv e-prints*, June 2018.

Francis Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate o (1/n). In *Advances in neural information processing systems*, pages 773–781, 2013.

Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

Abhishek Bhatia, Jaan Altosaar, and Shixiang Gu. Proximity-constrained reinforcement learning. 2017.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. `https://github.com/openai/baselines`, 2017.

Amir M Farahmand, Mohammad Ghavamzadeh, Shie Mannor, and Csaba Szepesvári. Regularized policy iteration. In *Advances in Neural Information Processing Systems*, pages 441–448, 2009.

Hugo Gilbert and Paul Weng. Quantile reinforcement learning. *arXiv preprint arXiv:1611.00862*, 2016.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Chandrashekar Lakshminarayanan and Csaba Szepesvari. Linear stochastic approximation: How far does constant step-size and iterate averaging go? In *International Conference on Artificial Intelligence and Statistics*, pages 1347–1355, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Alexander Rakhlin, Ohad Shamir, Karthik Sridharan, et al. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*. Citeseer, 2012.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.