Idea: Better gradient steps for deep on-policy reinforcement learning

Ryan Pégoud
EPF Engineering school
Montpellier, France
ryan.pegoud@epfedu.fr

Thibault Lahire
Dassault Aviation
Saint Cloud, France
thibault.lahire@dassault-aviation.com

Abstract

In deep on-policy reinforcement learning, algorithms collect transitions by executing the current policy in the environment. These transitions are then used during a gradient ascent step, aiming at updating the neural network(s) parameter. This article studies how the collected transitions can be prioritized to speed up the gradient ascent process toward a favorable policy. To do so, we weigh the transitions in the update gradient ascent equation with their per-sample gradient norms, which is a measure of the margin of change which can occur in the neural network.

1 Introduction

In deep Reinforcement Learning (Sutton and Barto, 2018, RL), neural network policies and value functions can be learnt thanks to gradient descent algorithms on transitions collected by interacting with the environment. This is particularly the case for model-free, policy-based algorithms, such as REINFORCE (Williams, 1992), TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017). These algorithms execute the current policy on the environment, collect trajectories, and update their neural network(s) with one gradient step thanks to the backpropagation of the statistics of the trajectories. As the collected trajectories are replaced each learning step by new ones, the RL algorithms mentioned before boil down to a sequence of supervised learning problems. Even though each supervised learning problem is not fully solved (as only one gradient step is taken), one would like to take better gradient steps each time learning is performed, as this implies a faster convergence.

The policy gradient theorem suggests gathering experiences by executing the current policy in the environment, and updating the policy network through backpropagation of each experience (Sutton et al., 1999). Each experience has equal importance in the gradient step. However, at a given time of the training, one experience could benefit more than another to the learning process. Indeed, some parts of the interactions with the environment might be well-handled by the neural network policy, whereas it is taking sub-optimal actions in some (less encountered) states. In model-free, value-based algorithms, such as Deep Q-Networks (Mnih et al., 2015, DQN), Prioritized Experience Replay (Schaul et al., 2016, PER) samples mini-batches thanks to a temporal difference error (TD error) associated to each collected experience in the replay buffer. This can be cast as importance sampling (Rubinstein and Kroese, 2016) and has brought a speed-up in convergence compared to vanilla DQN,

38th Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2024).
This work highlights some mathematical issues at stake in deep reinforcement learning and is to be pursued in the AI illustrations carried out in the EDF*2021-101103669-EICACS (European Initiative Collaborative Air Combat Standardisation) project.

This publication was co-funded by the European Union. Its contents are the sole responsibility of the author and do not necessarily reflect the views of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.
* EDF stands for European Defence Fund.

since sampling experiences with high TD error has been shown to reduce variance of the stochastic gradient estimate (Lahire et al., 2022).

To the best of our knowledge, PER remains a method associated to the existence of a replay buffer, which complicates its use in policy-based algorithms, such as REINFORCE or PPO. Moreover, contrarily to DQN, where a mini-batch is sampled from the replay buffer, the collected experiences in policy-based methods are all used for backpropagation, and no sampling is performed. In this work, we propose a connection between importance sampling schemes, as can be found in PER, and policy-based algorithms which are not depending on a replay buffer.

In particular, we weigh the collected experiences in policy-based algorithms, so that the gradient used for the update has less variance. This allows a better use of the trajectories: the parts on which the neural network has the most to learn drive preferentially the optimization. Our method has a computational cost equivalent to that of the base algorithm it is implemented on. In this work, we take REINFORCE as base algorithm to derive the theory for pedagogical purposes, since it is the simplest policy-based method. Our goal for future works will be to explain how the theory applies to elaborated algorithms such as TRPO and PPO. The experiments assessing our theory remains to be done.

Hence, this work has to be seen as an idea paper only, and is structured as follows. Section 2 introduces the main concepts in deep, policy-based, reinforcement learning on which we will present our contributions. Section 3 covers related work in prioritization for RL and importance sampling of training sets. Then section 4 grounds theoretically how some experiences can improve the learning process, and introduces how we can make the best use of these experiences. Section 5 discusses experiments which remains to be done and concludes.

2 Background

In the standard RL framework, one searches for the optimal control policy when interacting with a discrete-time system behaving as a Markov Decision Process (Puterman, 2014). At time step t, the system is in state $s_t \in S$, and upon applying action $a_t \in A$, it transitions to a new state s_{t+1}, while receiving reward r_t. Starting from an initial state (or from a probability distribution over initial states), this process (also called episode) stops when a terminal condition is met. For pedagogical reason, we assume in this work that the episodes stop after H interactions with the environment. A policy π is a function mapping states to distributions over actions. In deep, model-free and policy-based RL, the policy is a neural network parameterized by θ. Its performance can be assessed through the function $J(\theta) = \mathbb{E}[R|\pi_\theta]$ where $R = \sum_{t=0}^{H-1} r_t$ is the return. The goal of RL is to find the parameter θ^* associated with the policy having the largest possible J.

Note that we intentionally omitted the discount factor γ, which can be found in most RL research articles, to simplify our explanations. Although this is legitimized by the fact that the episode ends after H interactions, ensuring that the sum of reward is finite, our work also applies to the case of an infinite horizon, provided we replace R by the discounted return $R = \sum_{t=0}^{H-1} \gamma^t r_t$, with the discount factor $\gamma \in]0; 1[$. Note also that we used the shortcut equation $J(\theta) = \mathbb{E}[R|\pi_\theta]$ to express that the expectation $\mathbb{E}$ depends on the interactions with the environment through π_θ.

The gradient of J with respect to θ can be written $\nabla_\theta J(\theta) = \mathbb{E}\left[R \nabla_\theta \log \pi_{\theta_t}(a_t|s_t)|\pi_\theta\right]$ where a_t is the action sampled by the policy at time t. Starting from a random initial parameter θ_0, the REINFORCE algorithm builds a sequence $(\theta_m)_{m\geq 0}$ of parameters by performing gradient steps in the direction maximizing the function J. In theory, the update equation is $\theta_{m+1} = \theta_m + \eta \nabla_\theta J(\theta_m)$, with η a learning rate. However, this cannot be done in practice, as computing $\nabla_\theta J(\theta_m)$ is often computationally intractable. In practice, the expectation behind $\nabla_\theta J(\theta_m)$ is approximated with a mean estimator (also called Monte-Carlo estimator), which consists in drawing K trajectories executed with policy π_{θ_m}. The update equation used in practice is:

$$\theta_{m+1} = \theta_m + \eta \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{H-1} R^{(k)} \nabla_\theta \log \pi_{\theta_m}(a_t^{(k)}|s_t^{(k)})$$

where $s_t^{(k)}$, $a_t^{(k)}$ and $R^{(k)}$ are respectively the state, the selected action at time t, and the sum of collected rewards within trajectory k. The collected trajectories can be considered a dataset of size $N = KH$ composed of transition tuples $\left\{(s_t^{(k)}, a_t^{(k)}, R^{(k)}), 1 \leq k \leq K, 0 \leq t \leq H-1\right\}$. Re-

indexing the dataset with $i \in [1; N]$,it yields the dataset $\{(s_i, a_i, R_i), 1 \leq i \leq N\}$, and the gradient step can be written:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{N} \sum_{i=1}^{N} (-R_i) \nabla_\theta \log \pi_{\theta_t}(a_i|s_i). \tag{1}$$

This form is reminiscent of the minimization of an empirical risk in supervised learning, where the per-sample loss would be equal to $(-R_i) \log \pi_{\theta_t}(a_i|s_i)$. The objective would be to find $\theta^* \in \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} (-R_i) \nabla_\theta \log \pi_{\theta_t}(a_i|s_i)$ by gradient descent. However, contrarily to risk minimization in supervised learning, policy gradient methods such as REINFORCE only take a single gradient descent step before collecting a new dataset. In this work, we show that the collected dataset at each learning step can be used in a more relevant way, without adding any computational cost. Our contribution focuses on lowering the variance of the gradient estimate to perform better gradient steps.

3 Related work

Our work focuses on reweighing the per-sample gradients according to a prioritization metric. We consider the collected transitions as a dataset over which stochastic gradient descent (Robbins and Monro, 1951, SGD) is performed to take a step towards the optimum θ^* this dataset defines. According to the theory behind SGD, a small subset of items is uniformly drawn with replacement from the dataset to compute per-sample gradients and apply the update. However, the uniform distribution has shown not to be the most effective distribution to converge fast towards the optimum (Needell et al., 2014; Wang et al., 2017). The most effective distribution, proportional to the per-sample gradient norm of the losses, requires to be computed over the all dataset, which makes it practically intractable.

This has led to a whole research field, where tractable approximations of the most effective sampling scheme have been proposed. For instance, Loshchilov and Hutter (2016) proposed a sampling proportional to a loss ranking. Katharopoulos and Fleuret (2018) proposed an upper-bound on the per-sample gradient norm which is fast to compute and can be used as a surrogate of the optimal sampling scheme. Alain et al. (2016) deploy heavy computational resources, using clusters of GPUs, to be as close as possible to the optimal sampling scheme.

In RL, the well-known Priotized Experience Replay (Schaul et al., 2016, PER) has been introduced for the DQN algorithm as a heuristic accelerating the learning process, drawing inspiration from Prioritized Sweeping (Moore and Atkeson, 1993). Each transition in the replay buffer is assigned a priority that is proportional to the temporal difference (TD) error, and gradients are estimated by sampling according to these priorities. Later, PER was combined with other DQN improvements, including distributional RL (Bellemare et al., 2017; Hessel et al., 2018), as well as applied to actor-critic methods (Wang and Ross, 2019).

Other methods to select a mini-batch have been proposed. To emphasize recent experience and draw a mini-batch of size B, Zhang and Sutton (2017) sample uniformly $B - 1$ transitions from the replay buffer and add the last experience tuple collected along the trajectory. Similarly, Wang and Ross (2019) have observed a faster convergence of the soft actor-critic algorithm (Haarnoja et al., 2018, SAC) by sampling more frequently the recent experiences collected, which can be seen as a smooth version of the sampling proposed by Zhang and Sutton (2017). However, to the best of our knowledge, our work is the first attempt at reweighing the transitions in the gradient descent update step in deep, policy-base, reinforcement learning.

In this work, we re-use the theory developed in the supervised learning literature, showing the most effective sampling scheme is the one proportional to the per-sample gradient norms. We recall it at the beginning of the next section, and we then move to our contribution, based on the fact that all per-sample gradients are computed in deep, policy-based, RL algorithms. Instead of sampling the collected dataset, either with a uniform or a more effective sampling, one could simply weigh the collected items according to the per-sample gradient norms (that are computed anyway), and obtain the speed-up theoretically grounded in the supervised learning literature.

4 Contribution

Let u be the uniform discrete distribution over the transitions in the dataset constituted of the collected trajectories, such that $\forall i \in [1; N],\ u_i = 1/N$. The gradient can be written as an expectation over the transitions (or experiences):

$$\frac{1}{N}\sum_{i=1}^{N}(-R_i)\nabla_\theta \log \pi_{\theta_t}(a_i|s_i) = \sum_{i=1}^{N} u_i(-R_i)\nabla_\theta \log \pi_{\theta_t}(a_i|s_i) = \mathbb{E}_{i\sim u}[(-R_i)\nabla_\theta \log \pi_{\theta_t}(a_i|s_i)].$$

Let p be any probability distribution over the transitions of the dataset such that $p_i \neq 0, \forall i$. Importance sampling can be used to express the expectation with respect to the distribution p:

$$\mathbb{E}_{i\sim u}\left[R_i\nabla_\theta \log \pi_{\theta_t}(a_i|s_i)\right] = \mathbb{E}_{i\sim p}\left[R_i\nabla_\theta \log \pi_{\theta_t}(a_i|s_i)\frac{u_i}{p_i}\right] = \frac{1}{N}\mathbb{E}_{i\sim p}\left[\frac{R_i}{p_i}\nabla_\theta \log \pi_{\theta_t}(a_i|s_i)\right]$$

We introduce $G_i^{(m)} = \frac{-R_i}{Np_i}\nabla_\theta \log \pi_{\theta_m}(a_i|s_i)$ for any sampling scheme p such that $p_i > 0,\ \forall i \in [1; N]$. Setting η as a constant learning rate, a gradient descent update has the form: $\theta_{m+1} = \theta_m - \eta\mathbb{E}_{i\sim p}[G_i^{(m)}]$. This can be surprising at first sight since it seems to depend on p, but it is not the case. Indeed: $\mathbb{E}_{i\sim p}[G_i^{(m)}] = \sum_{i=1}^{N} p_i G_i^{(m)} = \frac{1}{N}\sum_{i=1}^{N} -R_i\nabla_\theta \log \pi_{\theta_m}(a_i|s_i)$, and this notation gives consistency with the gradient descent equation 1.

Adapting the ideas of Katharopoulos and Fleuret (2018) and Wang et al. (2017) to our setting, we define the convergence speed S of gradient descent under a sampling scheme p as $S(p) = -\mathbb{E}_{i\sim p}\left[\|\theta_{m+1} - \theta^*\|_2^2 - \|\theta_m - \theta^*\|_2^2\right]$. Using $\theta_{m+1} = \theta_m - \eta G_i^{(m)}$, the convergence speed can be written: $S(p) = 2\eta(\theta_m - \theta^*)^T\mathbb{E}_{i\sim p}[G_i^{(m)}] - \eta^2\mathbb{E}_{i\sim p}[{G_i^{(m)}}^T G_i^{(m)}]$. This derivation can be found in Appendix A. Our goal is to find p that maximizes the convergence speed, which is equivalent to minimizing $\mathbb{E}_{i\sim p}[{G_i^{(m)}}^T G_i^{(m)}]$. This minimization is a constrained optimization problem:

$$\min_p \mathbb{E}_{i\sim p}[{G_i^{(m)}}^T G_i^{(m)}] = \min_p \sum_{i=1}^{N} p_i\|G_i^{(m)}\|_2^2 \quad \text{such that } \sum_{i=1}^{N} p_i = 1 \text{ and } \forall i \in [1, N],\ p_i > 0.$$

The optimal sampling is $p_i^{GN} = \|R_i\nabla_\theta \log \pi_{\theta_m}(s_i|a_i)\|_2 / \sum_{j=1}^{N}\|R_j\nabla_\theta \log \pi_{\theta_m}(s_j|a_j)\|_2$, it is the weighing proportional to the per-sample gradient norms (hence the superscript GN). The proof is reported in Appendix A.

As a consequence, instead of performing a standard gradient descent step as proposed in Equation 1, this work proposes to study a different gradient step, namely:

$$\theta_{m+1} = \theta_m - \eta\frac{1}{N}\sum_{i=1}^{N}\frac{-R_i}{Np_i^{GN}}\nabla_\theta \log \pi_{\theta_m}(a_i|s_i) \tag{2}$$

This update equation weighs each item of the collected trajectories according to its potential of change in the neural network parameter. The higher the per-sample gradient norm associated to one transition, the most changes in the neural network parameter this transition produces.

5 Future work

This article is a work-in-progress, and focuses on deep, policy-based, reinforcement learning. It proposes an update equation for the neural network, which is different from the one usually performed in such framework. This new update equation gives more importance to transitions yielding a higher change in the neural network parameter, and is theoretically backed by the literature on importance sampling applied to supervised learning.

For pedagogical purposes, we restricted our study to the simplest form of deep, policy-based, reinforcement learning, namely the REINFORCE algorithm. We plan to extend our research to more used in practice (and complex) algorithms, such as TRPO and PPO. Weighing collected transitions in these algorithms will require adaptations to the base case we presented. Indeed, the loss of PPO is made of three terms, hence the per-sample gradient norms will have to be properly defined to

design an efficient algorithm. Hence, we believe this work-in-progress will benefit from the ARLET workshop in identifying potential shortcomings which can happen when the theory is applied to in-production algorithms such as PPO.

Finally, no experiment has been run yet to test the presented ideas. We plan to run experiments in environments with discrete and continuous action spaces to provide a diversity of control tasks. Once again, attending the ARLET workshop will give us insights on how to conduct statistically relevant experiments, and to identify a class of environments on which our reweighing scheme particularly works well.

6 Aknowledgements

This publication was co-funded by the European Union. Its contents are the sole responsibility of the author and do not necessarily reflect the views of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

References

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. (2016). Variance reduction in sgd by distributed importance sampling. In *International Conference on Learning Representations*.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Katharopoulos, A. and Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. In *International Conference on Machine Learning*, pages 2525–2534.

Lahire, T., Geist, M., and Rachelson, E. (2022). Large batch experience replay. In *International Conference on Machine Learning*, pages 11790–11813. PMLR.

Loshchilov, I. and Hutter, F. (2016). Online batch selection for faster training of neural networks. In *International Conference on Learning Representations*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Moore, A. W. and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13(1):103–130.

Needell, D., Ward, R., and Srebro, N. (2014). Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in neural information processing systems*, pages 1017–1025.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Rubinstein, R. Y. and Kroese, D. P. (2016). *Simulation and the Monte Carlo method*. John Wiley & Sons.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *ICLR (Poster)*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Wang, C. and Ross, K. (2019). Boosting soft actor-critic: Emphasizing recent experience without forgetting the past. *arXiv preprint arXiv:1906.04009*.

Wang, L., Yang, Y., Min, R., and Chakradhar, S. (2017). Accelerating deep neural network training with inconsistent stochastic gradient descent. *Neural Networks*, 93:219–229.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

Zhang, S. and Sutton, R. S. (2017). A deeper look at experience replay. In *NeurIPS 2017 Deep Reinforcement Learning Symposium*.

A Appendix

This appendix is dedicated to the derivation of the optimal sampling scheme, the one with the fastest convergence speed. In the main paper, we reuse this result, but we intentionally omit the sampling part, and instead use the result as a reweighing. We define the convergence speed S for a sampling scheme p as $S(p) = -\mathbb{E}_{i\sim p}\left[\|\theta_{m+1} - \theta^*\|_2^2 - \|\theta_m - \theta^*\|_2^2\right]$. We recall that a stochastic gradient descent update has the form $\theta_{m+1} = \theta_m - \eta G_i^{(m)}$, see Section 4 for more details. The following derivations from (Wang et al., 2017) help us rewrite the convergence speed:

$$\begin{aligned}
S(p) &= -\mathbb{E}_{i\sim p}\left[\|\theta_{m+1} - \theta^*\|_2^2 - \|\theta_m - \theta^*\|_2^2\right] \\
&= -\mathbb{E}_{i\sim p}\left[\theta_{m+1}^T\theta_{m+1} - 2\theta_{m+1}^T\theta^* - \theta_m^T\theta_m + 2\theta_m^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[(\theta_m - \eta G_i^{(m)})^T(\theta_m - \eta G_i^{(m)}) + 2\eta {G_i^{(m)}}^T\theta^* - \theta_m^T\theta_m\right] \\
&= -\mathbb{E}_{i\sim p}\left[-2\eta(\theta_m - \theta^*)^T G_i^{(m)} + \eta^2 {G_i^{(m)}}^T G_i^{(m)}\right] \\
&= 2\eta(\theta_m - \theta^*)^T\mathbb{E}_{i\sim p}[G_i^{(m)}] - \eta^2\mathbb{E}_{i\sim p}[{G_i^{(m)}}^T G_i^{(m)}]
\end{aligned}$$

It is possible to gain a speed-up by sampling from the distribution that minimizes $\mathbb{E}_{i\sim p}[{G_i^{(m)}}^T G_i^{(m)}]$. This yields the constrained optimization problem:

$$\min_p \mathbb{E}_{i\sim p}[{G_i^{(m)}}^T G_i^{(m)}] = \min_p \sum_{i=1}^N p_i \|G_i^{(m)}\|_2^2 \quad \text{such that } \sum_{i=1}^N p_i = 1 \text{ and } p_i \geq 0$$

Recall that $G_i^{(m)} = \frac{-R_i}{Np_i}\nabla_\theta \log \pi_{\theta_m}(a_i|s_i)$. Let $g_i = \|R_i\nabla_\theta \log \pi_{\theta_m}(a_i|s_i)\|_2$. The problem boils down to:

$$\min_p \frac{1}{N^2}\sum_{i=1}^N \frac{1}{p_i} g_i^2, \quad \text{such that } \sum_{i=1}^N p_i = 1 \text{ and } p_i \geq 0.$$

Lemma Optimal sampling distribution

The optimal sampling distribution p^{GN} verifies $p_i^{GN} \propto \|R_i\nabla_\theta \log \pi_{\theta_m}(a_i|s_i)\|_2$, the per-sample gradient norm.

Proof

We note $\mu \in \mathbb{R}$ the Lagrange multiplier associated to the equality constraint, $\nu \in \mathbb{R}_+^N$ the Lagrange multipliers associated to the inequality constraints. Hence:

$$\text{Lag}(p, \mu, \nu) = \sum_{i=1}^N \frac{1}{p_i} g_i^2 + \mu\left(\sum_{i=1}^N p_i - 1\right) - \sum_{i=1}^N \nu_i p_i$$

Setting the derivatives of the Lagrangian with respect to the primal variables yields:

$$\forall\, i \in [1, N],\ -\frac{g_i^2}{p_i^2} + \mu - \nu_i = 0$$

Multiplying the above equation by p_i and using $\forall\, i,\ p_i\nu_i = 0$ (complementary slackness), we have: $p_i = g_i/\sqrt{\mu}$, which yields the result.