Variance Reduction for Deep Q-Learning Using Stochastic Recursive Gradient

Haonan Jia[1], Xiao Zhang[2,3], Jun Xu[2,3](✉), Wei Zeng[4], Hao Jiang[5], and Xiaohui Yan[5]

[1] School of Information, Renmin University of China, Beijing, China
jiahaonan00@ruc.edu.cn
[2] Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
{zhangx89,junxu}@ruc.edu.cn
[3] Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China
[4] Baidu Inc., Beijing, China
zengwei04@baidu.com
[5] Huawei Technologies, Shenzhen, China
{jianghao66,yanxiaohui2}@huawei.com

Abstract. Deep Q-learning often suffers from poor gradient estimations with an excessive variance, resulting in unstable training and poor sampling efficiency. Stochastic variance-reduced gradient methods such as SVRG have been applied to reduce the estimation variance. However, due to the online instance generation nature of reinforcement learning, directly applying SVRG to deep Q-learning is facing the problem of the inaccurate estimation of the anchor points, which dramatically limits the potentials of SVRG. To address this issue and inspired by the recursive gradient variance reduction algorithm SARAH, this paper proposes to introduce the recursive framework for updating the stochastic gradient estimates in deep Q-learning, achieving a novel algorithm called SRG-DQN. Unlike the SVRG-based algorithms, SRG-DQN designs a recursive update of the stochastic gradient estimate. The parameter update is along an accumulated direction using the past stochastic gradient information, and therefore can get rid of the estimation of the full gradients as the anchors. Additionally, SRG-DQN involves the Adam process for further accelerating the training process. Theoretical analysis and the experimental results on well-known reinforcement learning tasks demonstrate the efficiency and effectiveness of the proposed SRG-DQN algorithm.

Keywords: Deep Q-learning · Variance reduction

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-981-99-1639-9_53.

M. Tanveer et al. (Eds.): ICONIP 2022, CCIS 1791, pp. 634–646, 2023.
https://doi.org/10.1007/978-981-99-1639-9_53

1 Introduction

Recent years have witnessed the dramatic progress of deep reinforcement learning (RL) in a variety of challenging tasks including computer games, robotics, natural language process, and information retrieval. Amongst the RL algorithms, deep Q-learning is a simple yet quite powerful algorithm for solving sequential decision problems [8,9]. Roughly speaking, deep Q-learning makes use of a neural network (Q-network) to approximate the Q-value function in traditional Q-learning models. The system state is given as the input and the Q-values of all possible actions are generated as the output. The learning of the parameters in Q-network amounts to sequences of optimization problems on the basis of the stored agent's experiences. Stochastic gradient descent (SGD) is often employed to solve these optimization problems. That is, at each iteration of the optimization, to calculate the parameter gradients, the agent samples an action according to the current Q-network, issues the action to the environment, gathers the reward, and moves to the next state. The reward is used as the supervision information for calculating the gradient for updating the Q-network parameters. The gradient points in the direction of maximum increase the possibility of getting high accumulative future rewards.

In real-world applications, SGD method in deep Q-learning often suffers from the inaccurate estimation of the gradients. The high variance gradient inevitably hurts the efficiency and effectiveness of the deep Q-learning algorithms. How to reduce the variance has become one of the key problems in deep RL. Research efforts have been undertaken to solve this problem. For example, Averaged-DQN [2] extends the traditional DQN algorithm by averaging the previously learned Q-values estimates, achieving a variance reduced gradient estimation with an approximation error guarantee. More recently, SVR-DQN [18] proposed an optimization strategy by combining the stochastic variance reduced gradient (SVRG) [5] technique and the deep Q-learning, called SVR-DQN. It has been shown that reducing the variance leads to a more stable and accurate training procedure. The Adam [6] optimization algorithm is an extension to stochastic gradient descent. Though it was mainly designed for optimizing neural networks, Adam can also be directly applied to improve the training process of deep Q-learning. More methods on variance reduction for deep Q-learning please refer to [12,14]. SVRG has also been applied to policy gradient methods in RL as an effective variance-reduced technique for stochastic optimization, such as off-line control [17], policy evaluation [4], and on-policy control [11]. The convergence rate of the variance-reduced policy gradient has been proved showing its advantages over vanilla methods [15]. [16] applied the recursive variance reduction techniques to policy gradient algorithms and proved the state-of-the-art convergence rate of policy gradient methods.

Though preliminary successes have been achieved, current methods are far from optimal because they ignored the essential differences between RL and traditional machine learning. The SVRG-based methods need to pre-calculate full gradients as the anchors. The anchors are crucial for finding more accurate gradient direction estimations in the down-stream parameter update. When being executed on a fixed training set, SVRG-based methods can easily estimate the anchors by scanning all of the training instances. In deep Q-learning, however, the *anchors cannot be accurately estimated anymore* because the learning is conducted in an online manner: (1) In deep Q-learning, the training instances (i.e., the sampled transitions) are gradually generated with the training goes on, via issuing actions to the environment at each iteration. The algorithm cannot access the instances that will be generated in the future iterations; (2) In deep Q-learning, the selection of the actions is guided by the DQN with current parameters. Therefore, the generated instances at different iterations cannot be identically distributed, as the DQN parameters have been updated. The phenomenon makes the problem of inaccurate estimation of the anchors more severe. Empirical analyses also have shown that the inaccurate estimation of the anchors greatly impacted the performances of the SVRG-based methods.

In this paper, to address the issue and inspired by the variance reduction algorithm SARAH [10], we propose to adopt the recursive gradient estimation mechanism in SARAH into the training iterations of deep Q-learning, achieving a novel deep Q-learning algorithm called SRG-DQN. Specifically, SRG-DQN contains an outer loop which samples N training instances (i.e., N transitions including the state, action, reward, and next-state) based on the current Q-network and from the experience replay, and an inner loop which first estimates the stochastic gradients recursively and then updates the Q-network parameters. Besides, the Adam process is executed at the end of the outer loop for further improving the efficiency of the training. Theoretical and experimental analyses demonstrate that the recursive gradient estimation mechanism successfully addresses the problem of inaccurately anchors in SVRG-based methods. It also heritages the advantages from SARAH including the fast convergence rate, and the stable and reliable training. We conduct experiments on RL benchmark tasks to evaluate the proposed SRG-DQN algorithm. Experimental results show that SRG-DQN outperforms the state-of-the-art baselines including SGD-based and SVRG-based deep Q-learning algorithms, in terms of reward scores, convergence rates, and training time. Empirical analyses also show that SRG-DQN dramatically reduces the variance of the estimated gradients, discovering how and why SRG-DQN can improve the performance of the baseline algorithms.

2 Related Work

2.1 SGD for Deep Q-Learning

In Q-learning, it is assumed that the agent will perform the sequence of actions that will eventually generate the maximum total reward (return). The return is also called the Q-value and the strategy is formalized as:

$$Q(s,a) = r(s,a) + \gamma \max_{a' \in \mathcal{A}(s)} Q(s',a'), \tag{1}$$

where $\gamma \in [0,1]$ is a discount factor which controls the contribution of rewards in the future, s is the current state, $\mathcal{A}(s)$ contains all of the candidate actions under state s, a is the selected action, $r(s,a)$ is the received reward after issuing a at s, and s' is the next state that the system moves to after issuing a. The equation states that the Q-value yielded from being at state s and performing action a is the immediate reward $r(s,a)$ plus the highest Q-value possible from the next state s'. It is easy to see that $Q(s,a)$ helps the agent to figure out exactly which action to perform.

Traditionally, Q-value is defined as a table with which the agent figures out exactly which action to perform at which state. However, the time and space complexities become huge when facing large state and action spaces. Deep neural networks have been used to approximate the Q-value function, called deep Q-learning. The learning of the parameters in the Q-network amounts to a serious of optimization problems. Specifically, assuming that at time step t, the system is at state S_t and the agent issues an action A_t. After that at the time step $t+1$, it receives a reward R_{t+1} and transits to state S_{t+1}. Therefore, we collect a transition tuple $(S_t, A_t, R_{t+1}, S_{t+1})$. The loss function, therefore, is defined as the mean squared error of the Q-value predicted by the Q-network and the target Q-value, where the target Q-value is derived from the Bellman equation

$$y(S_t, A_t) = R_{t+1} + \gamma \max_{a \in \mathcal{A}(S_t)} Q(S_{t+1}, a; \theta), \tag{2}$$

where $Q(S_{t+1}, a; \theta)$ is the Q-network with parameters θ that predicts the Q-value for the next state with the same Q-network. Stochastic gradient descent has been employed for conducting the optimization. Given a sampled transition (S, A, R, S'), the stochastic gradient can be estimated as:

$$\mathbf{g} = \nabla(y(S,A) - Q(S,A;\theta))^2 = 2(y(S,A) - Q(S,A;\theta))\nabla Q(S,A;\theta), \tag{3}$$

where $\nabla Q(S,A;\theta)$ calculates the gradient of Q w.r.t. the parameter θ.

2.2 Variance Reduced Deep Q-Learning

The original stochastic gradient descent based on a single transition often hurts from the problem of high gradient estimation variances. There are many ways to accelerate SGD convergence from the perspective of reducing variance, such as SAG [13], SAGA [3], and SVRG [5]. Researchers have combined the variance reduction techniques proposed in traditional machine learning with deep Q-learning. For example, Zhao et al. [18] proposed an algorithm called Stochastic Variance Reduction for Deep Q-learning (SVR-DQN) which combines SVRG with utilizes the optimization strategy of SVRG during the learning. Specifically, at each outer iteration $s = 1, 2, \cdots, S$, the algorithm samples a batch of N training transitions $\mathcal{D} = \{(S_i, A_i, R_{i+1}, S_{i+1})\}_{i=1}^{N}$, and calculates a full gradient according to Eq. (3) on $\mathcal{D}$ as the anchor:

$$\tilde{\mathbf{g}} = \frac{1}{N}\sum_{i=1}^{N} 2(y_i - Q(S_i, A_i; \theta_0^s))\nabla Q(S_i, A_i; \theta_0^s), \tag{4}$$

where $y_i = R_{i+1} + \gamma \max_{a' \in \mathcal{A}(S_{i+1})} Q(S_{i+1}, a'; \theta_0^s)$, and θ_0^s is the network parameter at the beginning of the s-th outer iteration. In its inner iteration indexed by m, for each sampled transition $(S, A, R, S') \in \mathcal{D}$, the stochastic gradients w.r.t. up to date parameters are calculated:

$$\mathbf{g}_m^s = 2(y_m - Q(S, A; \theta_m^s))\nabla Q(S, A; \theta_m^s), \tag{5}$$

where $y_m = R + \gamma \max_{a' \in \mathcal{A}(S')} Q(S', a'; \theta_m^s)$. Similarly, the stochastic gradients w.r.t. 'old parameters' are also calculated:

$$\mathbf{g}_0^s = 2(y_0 - Q(S, A; \theta_0^s))\nabla Q(S, A; \theta_0^s), \tag{6}$$

where$y_0 = R + \gamma \max_{a' \in \mathcal{A}(S')} Q(S', a'; \theta_0^s)$. Finally based on Eq. (4), (5) and (6), the variance reduced gradient is calculated as

$$\Delta = \mathbf{g}_m^s - \mathbf{g}_0^s + \tilde{\mathbf{g}}. \tag{7}$$

Besides, at each outer iteration, SVR-DQN obtains a more accurate estimation of the gradient using Adam process, which can accelerate the training of deep Q-learning and improve the performances [18].

3 Our Approach: SRG-DQN

In this section, we analyze the limitations of the variance reduction mechanism in SVR-DQN and propose a novel deep Q-learning algorithm called SRG-DQN.

3.1 Problem Analysis

In SVR-DQN, the accurate estimations of the anchors $\tilde{\mathbf{g}}$ are crucial they provide stable baselines to adjust the stochastic gradient. In traditional machine learning, the SVRG anchors can be accurately estimated based on the whole train data (i.e., full gradients). In deep Q-learning, however, the training instances are not fixed in advance but we need to collect them at each parameter change. Therefore, the estimated anchors are only based on N instances sampled at the previous and current iterations.

The phenomenon inevitably makes the estimated anchors inaccurate due to the following two reasons.

First, deep Q-learning algorithms are usually run in an online manner by nature. At each iteration, the algorithm samples an (or some) instance(s) as the new training data. Therefore, it is impossible for the algorithm to estimate the full gradients as the anchors, because only a part of the whole training instances (the instances sampled at the past iterations) are accessible.

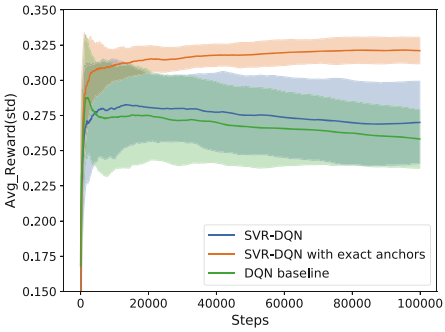

Fig. 1. Training curves of "SVR-DQN", "SVR-DQN with exact anchors" and DQN baseline. The shaded area presents one standard deviation. We focus on the influence of anchors and omit the Adam process in these algorithms.

Second, the instances sampled at different iterations are based on the Q-network with different parameters. That is, at each iteration, the agent first sample an action guided by the Q-network $Q(s, a; \theta)$ (for example, ϵ-greedy), and then use the sampled instance to update parameters. Therefore, the Q-network is continuously updated during the training, and the sampled instances at different iterations would belong to different distributions. This will make the estimated anchors cannot reflect the directions that the exact gradients should point to.

We conduct an experiment to show the impact of the inaccurate anchors. Specifically, based on the Mountain Car task, we compared the performances of deep Q-learning with SVRG (SVR-DQN) and its variation in which the anchor points *could be exactly estimated*. To do this, we first ran the existing SVR-DQN and recorded all of the sampled transitions. Then, we re-ran SVR-DQN using identical settings and training transitions at each iteration, except estimating the $\tilde{\mathbf{g}}$ in line 5 with all of the recorded transitions (denoted as "SVR-DQN with exact anchors").

In this way, the anchors $\tilde{\mathbf{g}}$ are estimated on the whole training data as that of in traditional machine learning Fig. 1 shows the training curves of the two models". We can see that compared with "SVR-DQN", "SVR-DQN with exact anchors" converged faster, better, and with lower variances. The results clearly indicate that the inaccurate estimation of the anchors can hurt the power of SVRG. We conclude that directly applying SVRG to deep Q-learning violates the basic assumptions of SVRG, and therefore limits its full potentials.

3.2 Recursive Gradient Deep Q-Learning

To address the problem and inspired by the algorithm of SARAH [10], in this paper we propose a novel algorithm called Stochastic Recursive Gradient Deep Q-Network (SRG-DQN). Different from the SVRG-based methods, SRG-DQN resorts to the recursive gradients rather than the full gradients, as the anchors. In this way, SRG-DQN gets rid of the inaccurate estimation of the anchors.

As shown in Algorithm 1, SRG-DQN contains an outer loop indexed by s. At each outer loop, N training instances $\mathcal{D}$ are sampled and the initial anchor point Δ_0^s is calculated based on all of the N sampled instances:

$$\Delta_0^s = \nabla\left(\frac{1}{N}\sum_{i=1}^{N}(y_i - Q(S_i, A_i; \theta_0^s))^2\right) = \frac{1}{N}\sum_{i=1}^{N} 2(y_i - Q(S_i, A_i; \theta_0^s))\nabla Q(S_i, A_i; \theta_0^s) \quad (8)$$

where each of the targets $y_i = R_{i+1} + \gamma \max_{a' \in \mathcal{A}(S_{i+1})} Q(S_{i+1}, a'; \theta_0^s)$ is the Q-value derived from the Bellman equation, for $i = 1, \cdots, N$. Δ_0^s is first used to update the parameters.

At each iteration of its inner loop m, the algorithm first randomly samples one training instance $(S, A, R, S') \in \mathcal{D}$. The stochastic gradient w.r.t. current up-to-date parameters, denoted as θ_m^s, is calculated as follows:

$$\mathbf{g}_m^s = \nabla(y_m - Q(S, A; \theta_m^s))^2 = 2\left(y_m - Q(S, A; \theta_m^s)\right)\nabla Q(S, A; \theta_m^s), \quad (9)$$

where the target $y_m = R + \gamma \max_{a' \in \mathcal{A}(S')} Q(S', a'; \theta_m^s)$. Similarly, the stochastic gradient w.r.t. the previous inner loop parameter θ_{m-1}^s is also calculated:

$$\mathbf{g}_{m-1}^s = 2\left(y_{m-1} - Q(S, A; \theta_{m-1}^s)\right)\nabla Q(S, A; \theta_{m-1}^s),$$

where $y_{m-1} = R + \gamma \max_{a' \in \mathcal{A}(S')} Q(S', a'; \theta_{m-1}^s)$.

Following the recursive gradient defined in [10], the final gradient at current loop, Δ_m^s, can be defined recursively. That is, using the previous loop gradient Δ_{m-1}^s as the anchor point to estimate the current loop gradient:

$$\Delta_m^s = \mathbf{g}_m^s - \mathbf{g}_{m-1}^s + \Delta_{m-1}^s. \quad (10)$$

Note that the anchor for the first loop (i.e., $\Delta_0^s, m = 0$) is the full gradient calculated in the outer loop. To further improve the performances and in light of Adam optimizer [6], we also propose to introduce the Adam process in SRG-DQN. Specifically, after the ending of each inner loop, an Adam process is executed for further updating the parameters, including calculating a biased first moment, a biased second raw moment, a bias-corrected first moment, and a bias-corrected second raw moment, and finally conducting the parameter updating. Detailed description of Adam process in SRG-DQN can be found in the supplementary material. Following the practices in [7,10], SRG-DQN takes θ_{M+1}^S as its final output. The algorithm is also suitable for a mini-batch version.

3.3 Theoretical Analysis

We analyze the convergence of SRG-DQN as follows. Proof of Theorem 1 can be found in the supplementary material. In the inner loop, the optimization in DQN can be formulated as the empirical risk minimization problem:

$$\min_{\theta \in \Theta} F(\theta) := \frac{1}{M}\sum_{i=1}^{M} f_i(\theta), \quad (11)$$

where $f_i(\theta) = [y_i - Q(S_i, A_i; \theta)]^2$. The optimization problem in DQN is nonconvex due to the composite structure of neural networks. Given the error parameter $\varepsilon > 0$, the goal is to search for an *ε-optimal point* $\theta \in \Theta$ such that

$$\mathbb{E}[\|\nabla F(\theta)\|^2] \leq \varepsilon. \tag{12}$$

First, similar to that of in [1], we give the definition of the Incremental First-order Oracle (IFO):

Definition 1. *IFO takes a point $\theta \in \Theta$ and an index $i \in \{1, 2, \ldots, M\}$ as inputs and returns the pair $\nabla f_i(\theta)$.*

Then the convergence rate of the algorithms can be measured by the oracle complexity. The *oracle complexity* is defined as the smallest number of queries to IFO leading to an ε-optimal point. Further assuming that each function $f_i(\theta)$ is β_i-smooth and bounded for $i \in [M]$ (that is, $\nabla f_i(\theta)$ is β_i-Lipschitz continuous and $|f_i(\theta)| \leq B_i,\ i \in [M], \theta \in \Theta$), we have the following Theorem 1 for SRG-DQN:

Theorem 1. *Let $\mu = \sum_{i=1}^{M} \beta_i^2/M$ and $B_{\max} = \sup_{i\in[M]}\{B_i\}$. For SRG-DQN within a single outer loop (in outer iteration $s \in [S]$), setting $\eta \leq 2/[\sqrt{\mu}(\sqrt{4M+1}+1)]$ to attain an ε-optimal point requires $\Omega\left(\sqrt{M}/\varepsilon\right)$ queries to IFO.*

Algorithm 1. Stochastic Recursive Gradient for Deep Q-Learning (SRG-DQN)

Require: Deep Q-function Q, # epochs S, epoch size M, discount factor γ, step size η
Ensure: Model parameters θ

Initialize $\theta_{M+1}^0 \leftarrow \theta_0$
for $s = 1$ **to** S **do**
  $\theta_0^s \leftarrow \theta_{M+1}^{s-1}$
  sample N transitions $\mathcal{D} = \{(S_i, A_i, R_{i+1}, S_{i+1})\}_{i=1}^N$ according to $Q(s, a; \theta_0^s)$
  $\Delta_0^s \leftarrow \frac{1}{N}\sum_{i=1}^N 2(y_i - Q(S_i, A_i; \theta_0^s))\nabla Q(S_i, A_i; \theta_0^s)$
  where $y_i = R_{i+1} + \gamma \max_{a\in\mathcal{A}(S_{k+1})} Q(S_{k+1}, a; \theta_0^s)$
  $\theta_1^s \leftarrow \theta_0^s - \eta\Delta_0^s$ {update with full gradient}
  for $m = 1$ **to** M **do**
    randomly select a transition $(S, A, R, S') \in \mathcal{D}$
    $y_m \leftarrow R + \gamma \max_{a'\in\mathcal{A}(S')} Q(S', a'; \theta_m^s)$
    $\mathbf{g}_m^s \leftarrow 2(y_m - Q(S, A; \theta_m^s))\nabla Q(S, A; \theta_m^s)$ {gradient w.r.t. up-to-date parameters}
    $y_{m-1} \leftarrow R + \gamma \max_{a'\in\mathcal{A}(S')} Q(S', a'; \theta_{m-1}^s)$
    $\mathbf{g}_{m-1}^s \leftarrow 2(y_{m-1} - Q(S, A; \theta_{m-1}^s))\nabla Q(S, A; \theta_{m-1}^s)$ {gradient w.r.t. previous-iteration parameters}
    $\Delta_m^s \leftarrow \mathbf{g}_m^s - \mathbf{g}_{m-1}^s + \Delta_{m-1}^s$ {recursive gradient which using the previous one as the anchor}
    $\theta_{m+1}^s \leftarrow \theta_m^s - \eta\Delta_m^s$ {update parameters}
  end for
  {Adam process}
end for
return θ_{M+1}^S

Remark 1. While the oracle complexity of SVRG is $\widetilde{O}\left(M + M^{2/3}/\varepsilon\right)$ for the optimization problem in DQN [7], SRG-DQN achieves a lower oracle complexity w.r.t. the number of the epoch size M, which indicates SRG-DQN has a faster convergence rate than that of SVRG for DQN.

4 Experiments

4.1 Experimental Settings

We follow [16] to conduct the experiments on benchmark RL environments, including the Cartpole, Mountain Car and Pendulum problems. Following the setups in [9], ϵ-greedy strategy is used for the exploitation and exploration, where ϵ decreases linearly from initial value 0.1 to 0.001. The transfer instances generated during the interactions between the agent and the environment are stored in the experience replay memory, which adopted a first-in-first-out mechanism to store the transition data. When performing the gradient descent, the algorithm sampled 64 transition instances from the experience replay uniformly as the training batch data. The learning frequency is set to 16, which means that the batch data is sampled once every 16 rounds. In all the experiments, the DQN algorithm in [9] is adopted as the main architecture. Our algorithms are called "SRG-DQN" and "SRG-DQN without Adam Process". The DQN optimized by SGD (called "DQN with SGD") and DQN optimized by SVRG (called "SVR-DQN") are chosen as the baselines.

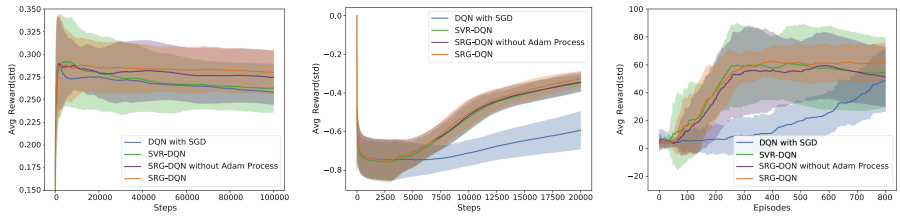

Fig. 2. Performance curves of DQN with SGD (blue), SVR-DQN (green), SRG-DQN without Adam process (purple) and SRG-DQN (orange) for three tasks, where the shaded area represents standard deviation, the 'Steps' represents the outer iteration s in Algorithm 1 and the 'Avg_Reward' represents the average rewards, the 'Episode' represents a complete trajectory and the 'Avg_Reward' represents the average return per trajectory. (Color figure online)

4.2 Experimental Results

We conduct average reward experiments in three tasks, in which the average reward is used to measure the performance of the agent. For fair comparisons, DQN structures in all algorithms are set with the same parameters.

The left part of Fig. 2 compares the performance of the four algorithms on the Mountain Car task. To encourage the car to explore, we replace the reward function from the original discrete value to a continuous function that is correlated with the car's position. Without limiting the episode length, the four algorithms all run 100,000 steps which means the faster the car reaches the goal, the higher average reward of each action will get. We omit the standard deviation of DQN with SGD in this figure, which is obviously poor. The middle part of Fig. 2 compares the performance of the four algorithms on Pendulum task. To facilitate DQN for choosing the action, we decompose it into 12 parts with equal distances. All the four algorithms do not limit the number of episodes, run 20,000 steps, and repeat 50 rounds. From the results, our algorithm achieves a fast convergence rate and has the optimal average reward with reduced variances. The right part of Fig. 2 compares the performance of the four algorithms under Cartpole task in which we need to keep the pole standing, and once it falls, the task is terminated. So we replace the average rewards for each step with the average reward for each episode. To accelerate convergence, we replace the reward function from the discrete value of 0/1 to a continuous function related to observations. The score is higher when the pole is straighter. All four algorithms run 800 episodes and repeat 10 rounds. From the results, our algorithm has an excellent average reward while significantly reducing the variance.

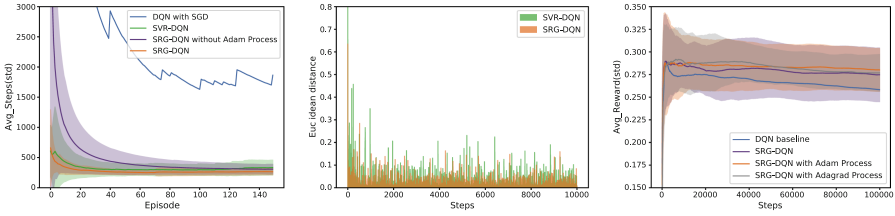

Fig. 3. Results of the experimental analysis on Mountain Car. *Left*: average steps w.r.t. episodes; *Middle*: ℓ_2 distances between the exact anchors and recursive anchors in SRG-DQN and SVR-DQN. *Right*: SRG-DQN with different optimizer processes.

4.3 Experimental Analysis

We experimentally analyze the reasons why SRG-DQN is effective. We first conduct the episode-average size experiment, in which four algorithms are run with 150 episodes and 100 rounds under the same model parameter settings. The convergence rate is measured by the average size of the episode, and the stability is evaluated by the standard deviation. The experimental results are shown in the left part of Fig. 3, where the bold line represents the average episode size of multiple experiments, and the shading represents the standard deviation. We can observe that SRG-DQN has significantly improved the convergence rate and the stability compared to the traditional DQN with SGD. Compared with SVR-DQN, SRG-DQN further shortens the average episode length, reduces the standard deviation. The orange line represents SRG-DQN. From the results, Adam

process and variance reduction algorithm can play a complementary role, and their combination can further accelerate the algorithm convergence and improve the stability of the agent. In addition, we computed the ℓ_2 distances between the exact anchors and the recursive anchors in SVR-DQN and SRG-DQN. From the results about distances in the middle part of Fig. 3, we can observe that the recursive anchors in SRG-DQN can significantly reduce the distances from the exact anchors, which is another reason why SRG-DQN can achieve better performances. We also compared the effects of the combined use of variance reduction methods and different optimizer processes. As shown in the right part of Fig. 3, adding an optimization process will further improve the performance of the algorithm, and due to the combined use of first-order gradient and second-order gradient information, the Adam process has proven to be a better choice than the Adagrad.

In order to explore whether our method really reduces the variance of gradient estimation, we compared the performances of SRG-DQN with SVR-DQN on the variance reduction for gradients. We calculated the standard deviations of the gradients with respect to the parameters on each dimension separately and then summed them up. From the results in Table 1, we can find that, compared with SVR-DQN, SRG-DQN significantly reduces the standard deviation, and it is almost completely superior to SVR-DQN at most steps. Thus, we can conclude that SRG-DQN converges to the function controlling the variance of the gradients, and achieves an improvement on SVR-DQN for variance reduction, which demonstrates the effectiveness of our stochastic recursive gradient for the variance reduction in DQN.

Table 1. The comparisons between SVR-DQN and SRG-DQN in terms of the standard deviation of the gradients on Mountain Car task, where the standard deviation is computed by summing the standard deviations of each element in the first-layer network gradient vector. We recorded the standard deviation once every 1,000 steps for the first 10,000 steps (first three rows) and last 10,000 steps (last three rows).

# Steps (first)	1,000	2,000	3,000	4,000	5,000	6,000	7,000	8,000	9,000	10,000
SVR-DQN	0.229	0.790	0.430	**0.474**	0.626	1.176	**0.500**	**0.388**	0.638	0.739
SRG-DQN	**0.122**	**0.395**	**0.389**	0.520	**0.625**	**0.632**	0.966	0.966	**0.394**	**0.562**
# Steps (last)	1,000	2,000	3,000	4,000	5,000	6,000	7,000	8,000	9,000	10,000
SVR-DQN	0.306	**0.286**	0.293	**0.212**	0.188	0.318	0.443	0.271	0.193	0.205
SRG-DQN	**0.225**	0.334	**0.254**	0.353	**0.154**	**0.247**	**0.182**	**0.229**	**0.188**	**0.203**

5 Conclusion

This paper proposes a novel deep Q-learning algorithm using stochastic recursive gradients, which reduces the variance of the gradient estimation. The proposed algorithm introduces the recursive framework for updating the stochastic

gradient and computing the anchor points. Adam process is involved for achieving a more accurate gradient direction. Theoretical analysis and empirical comparisons showed that the proposed algorithm outperformed the state-of-the-art baselines in terms of reward scores, convergence rate, and stability. The proposed stochastic recursive gradient provides an effective scheme for variance reduction in reinforcement learning.

Acknowledgements. This work was funded by the National Key R&D Program of China (2019YFE0198200), National Natural Science Foundation of China (61872338, 62102420, 61832017), Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, and Public Policy and Decision-making Research Lab of Renmin University of China.

References

1. Agarwal, A., Bottou, L.: A lower bound for the optimization of finite sums. In: Proceedings of the 32nd International Conference on Machine Learning, pp. 78–86 (2015)
2. Anschel, O., Baram, N., Shimkin, N.: Averaged-DQN: variance reduction and stabilization for deep reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning, pp. 176–185 (2017)
3. Defazio, A., Bach, F., Lacoste-Julien, S.: SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives. In: Advances in Neural Information Processing Systems, vol. 27, pp. 1646–1654 (2014)
4. Du, S.S., Chen, J., Li, L., Xiao, L., Zhou, D.: Stochastic variance reduction methods for policy evaluation. In: Proceedings of the 34th International Conference on Machine Learning, pp. 1049–1058 (2017)
5. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. In: Advances in Neural Information Processing Systems, vol. 26, pp. 315–323 (2013)
6. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Li, B., Ma, M., Giannakis, G.B.: On the convergence of SARAH and beyond. arXiv:1906.02351 (2019)
8. Mnih, V., et al.: Playing atari with deep reinforcement learning. arXiv:1312.5602 (2013)
9. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
10. Nguyen, L.M., Liu, J., Scheinberg, K., Takáč: SARAH: a novel method for machine learning problems using stochastic recursive gradient. In: Proceedings of the 34th International Conference on Machine Learning, pp. 2613–2621 (2017)
11. Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., Restelli, M.: Stochastic variance-reduced policy gradient. In: Proceedings of the 35th International Conference on Machine Learning, pp. 4023–4032 (2018)
12. Romoff, J., Henderson, P., Piché, A., Francois-Lavet, V., Pineau, J.: Reward estimation for variance reduction in deep reinforcement learning. arXiv preprint arXiv:1805.03359 (2018)

13. Roux, N.L., Schmidt, M., Bach, F.R.: A stochastic gradient method with an exponential convergence rate for finite training sets. In: Advances in Neural Information Processing Systems, vol. 25, pp. 2663–2671 (2012)
14. Sabry, M., Khalifa, A.M.A.: On the reduction of variance and overestimation of deep Q-learning. arXiv preprint arXiv:1910.05983 (2019)
15. Xu, P., Gao, F., Gu, Q.: An improved convergence analysis of stochastic variance reduced policy gradient. In: Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence, p. 191 (2019)
16. Xu, P., Gao, F., Gu, Q.: Sample efficient policy gradient methods with recursive variance reduction. In: Proceedings of the 8th International Conference on Learning Representations (2020)
17. Xu, T., Liu, Q., Peng, J.: Stochastic variance reduction for policy gradient estimation. arXiv:1710.06034 (2017)
18. Zhao, W.Y., Peng, J.: Stochastic variance reduction for deep Q-learning. In: Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems, pp. 2318–2320 (2019)