A Restart-based Rank-1 Evolution Strategy for Reinforcement Learning

Zefeng Chen[1] , **Yuren Zhou**[1,2] , **Xiaoyu He**[1] and **Siyu Jiang**[3]
[1]School of Data and Computer Science, Sun Yat-sen University
[2]Engineering Research Institute, Guangzhou College of South China University of Technology
[3]School of Software Engineering, South China University of Technology
chzfeng@mail2.sysu.edu.cn, zhouyuren@mail.sysu.edu.cn, hxyokokok@foxmail.com,
jsy2008@126.com

Abstract

Evolution strategies have been demonstrated to have the strong ability to roughly train deep neural networks and well accomplish reinforcement learning tasks. However, existing evolution strategies designed specially for deep reinforcement learning only involve the plain variants which can not realize the adaptation of mutation strength or other advanced techniques. The research of applying advanced and effective evolution strategies to reinforcement learning in an efficient way is still a gap. To this end, this paper proposes a restart-based rank-1 evolution strategy for reinforcement learning. When training the neural network, it adapts the mutation strength and updates the principal search direction in a way similar to the momentum method, which is an ameliorated version of stochastic gradient ascent. Besides, two mechanisms, i.e., the adaptation of the number of elitists and the restart procedure, are integrated to deal with the issue of local optima. Experimental results on classic control problems and Atari games show that the proposed algorithm is superior to or competitive with state-of-the-art algorithms for reinforcement learning, demonstrating the effectiveness of the proposed algorithm.

1 Introduction

In reinforcement learning (RL), faced with a specific environment, an agent attempts to learn to conduct a sequence of actions with the aim of maximizing a kind of cumulative reward [Sutton and Barto, 1998]. The popular RL algorithms (such as Q-learning and policy gradient method) and the neuroevolution approach (that is, applying black-box optimization methods to train neural networks) have been demonstrated to perform well on challenging deep RL problems. It is worth noting that among these algorithms for RL, evolution strategies (ESs) have been demonstrated to have the ability to roughly train deep neural networks and can well accomplish the deep RL tasks [Salimans *et al.*, 2017].

Existing ES variants for deep RL only involve the plain variants (such as Natural Evolution Strategies (NES) [Wierstra *et al.*, 2008] and (μ, λ)-ES [Schwefel, 1981]), but do not adopt one of the most efficient ESs, that is, the covariance matrix adaptation evolution strategy (CMA-ES) [Hansen and Ostermeier, 2014]. CMA-ES adapts the complete covariance matrix (representing the population) of the normal search distribution. It maintains and updates a Gaussian distribution during the search process. and attempts to use the contours of the Gaussian distribution to approximate that of the objective function. CMA-ES owns several important invariances, including the invariance against angle preserving transformations of the search space, the invariance against order preserving transformations of the objective function, and the scale invariance [Hansen and Kern, 2004]. Owing to its nice properties, CMA-ES has been successful in solving optimization problems in low to medium dimension. However, due to its high space and time complexity, it is unsuitable for large scale optimization [Varelas *et al.*, 2018], and also unsuited to the training of deep neural networks and the solving of deep RL problems. Particularly, when training the neural network, standard CMA-ES needs to consume much time on maintaining and updating its full covariance matrix, due to its quadratic time complexity. As a result, the research of applying CMA-ES to RL is still a gap. In reality, the issue of how to apply CMA-ES or other advanced ES variants [Li and Zhang, 2017; Varelas *et al.*, 2018] to deep RL in an efficient way is a rich research area, and has a great practical research value for deep RL problems.

To this end, this paper proposes a restart-based rank-1 evolution strategy (called R-R1-ES for short) for RL. This algorithm can be regarded as a simplified CMA-ES and can retain the excellent performance of CMA-ES. Notably, it is of linear time complexity and low space complexity, thus making it suitable for dealing with large scale black-box optimization and deep RL problems. Furthermore, compared to the original CMA-ES which searches along the coordinate axes and is more suitable for separable problem, the proposed ES variant can well deal with the dependency problem of different variables by virtue of adapting the principal search direction.

2 Preliminaries

As a class of black-box optimization algorithms, ESs belong to heuristic search algorithms inspired by natural evolution. At every generation, a population of parameter vectors are perturbed through sampling from a Gaussian distribution, and

their objective function values are evaluated. The parameter vectors with better quality are then recombined to construct new distribution for the next generation, and this procedure is iterated until the termination criterion is fulfilled. Algorithms in this class differ in how they represent the population and how they perform mutation and recombination.

Substantially, NES can be viewed as an $(1, \lambda)$-ES algorithm [Beyer, 1993]. Suppose the parameter vector and the fitness function for a specific optimization problem are θ and $F(\theta)$, respectively. And let $p_\phi(\theta)$ denote a distribution of parameter vectors θ characterized by parameters ϕ. NES represents the population as $p_\phi(\theta)$, and strives to maximize the average fitness of the population $\mathbb{E}_{\theta\sim p_\phi}[F(\theta)]$ by means of optimizing ϕ with stochastic gradient ascent method [Ruder, 2016].

Recently, scientists from OpenAI designed a version of NES applied to RL problems [Salimans *et al.*, 2017]. This variant will be referred to as OpenAI-ES hereinafter. For an RL problem, θ is the parameters of a policy π_θ, and the fitness function $F(\theta)$ denotes the stochastic reward over a full episode of agent interaction. At the t-th iteration, the population distribution p_{ϕ_t} is an isotropic multivariate Gaussian with mean θ_t and covariance $\sigma^2\mathbf{I}$ (i.e., $\mathcal{N}\left(\theta_t, \sigma^2\mathbf{I}\right)$). Different from NES which updates the mean and the covariance of the population distribution during the evolutionary process, OpenAI-ES does not attempt to evolve the covariance (reflected by σ). Thus, the mutation strength σ is fixed during the evolutionary process of OpenAI-ES.

At each iteration, λ parameter vectors $\theta_t^i \sim \mathcal{N}\left(\theta_t, \sigma^2\mathbf{I}\right)$ $(i = 1, \ldots, \lambda)$ are sampled from the Gaussian distribution $\mathcal{N}\left(\theta_t, \sigma^2\mathbf{I}\right)$, and their corresponding policies $\pi_{\theta_t^i}$ are evaluated to obtain a reward $F\left(\theta_t^i\right)$. Similar to the classical REINFORCE method [Williams, 1992] in RL, OpenAI-ES computes the approximate gradient of expected reward with respect to θ_t in the following manner:

$$\nabla_\phi \mathbb{E}_{\theta\sim p_\phi}\left[F\left(\theta\right)\right] \approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} F\left(\theta_t^i\right)\nabla_\phi \log p_\phi\left(\theta_t^i\right) \quad (1)$$

In fact, the sampling of θ_t^i in OpenAI-ES is reformulated as adding additive Gaussian noise to θ_t: $\theta_t^i = \theta_t + \sigma\varepsilon_i$ $(\varepsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}))$. Then, Eq. (1) can be simplified according to the property of Gaussian distribution. Concretely, Eq. (1) can be converted to the following formula:

$$\nabla_{\theta_t}\mathbb{E}\left[F\left(\theta\right)\right] \approx \frac{1}{\sigma\lambda}\sum_{i=1}^{\lambda} F\left(\theta_t^i\right)\frac{\theta_t^i - \theta_t}{\sigma} \quad (2)$$

Furthermore, they also made several contributions to improve the performance of ES. The technique of virtual batch normalization [Salimans *et al.*, 2016] and other reparameterizations of the neural network policy are adopted to enhance the reliability of ES. Moreover, by virtue of the high parallelization property of ES, a novel communication strategy based on common random numbers is introduced to reduce communication cost, and it can make the algorithm well scaled to a large number of parallel workers.

3 Proposed Method: R-R1-ES

This section detailedly introduces the proposed R-R1-ES.

3.1 Distribution Model

Unlike all these existing ES variants for RL, the proposed R-R1-ES utilizes the Gaussian distribution model $\mathcal{N}\left(\theta_t, \sigma_t^2\mathbf{C}_t\right)$, where $\theta_t \in \mathbb{R}^n$ (n is the dimension of the parameter vector θ) is the distribution mean, $\sigma_t > 0$ is the mutation strength, and $\mathbf{C}_t$ is the $n \times n$ covariance matrix at the t-th iteration. The covariance matrix $\mathbf{C}_t$ is set as follows [Li and Zhang, 2017]:

$$\mathbf{C}_t = (1 - c_{cov})\,\mathbf{I} + c_{cov}\mathbf{p}_t\mathbf{p}_t^T \quad (3)$$

where $c_{cov} \in (0, 1)$ is called the changing rate of covariance matrix, and $\mathbf{p}_t \in \mathbb{R}^n$ is called principal search direction.

In reality, the aforementioned model is designed for capturing the most important search direction (reflected by $\mathbf{p}_t$) and parameter dependencies, so as to endow the proposed algorithm with the ability of handling the dependency problem of different variables. In addition, the term $\mathbf{p}_t\mathbf{p}_t^T$ in Eq. (3) produces an $n \times n$ matrix whose rank is 1. This is also why we call our proposed algorithm a rank-1 ES.

To realize the aforementioned model, the proposed R-R1-ES needs to maintain the following critical variables at each generation of the evolutionary process:

- distribution mean $\theta_t \in \mathbb{R}^n$;
- mutation strength $\sigma_t > 0$;
- principal search direction $\mathbf{p}_t \in \mathbb{R}^n$.

Thus, the new model enables the R-R1-ES to achieve the adaptation of mean and mutation strength as well as the search direction.

3.2 Framework Of R-R1-ES

Algorithm 1 sketches the framework of our proposed R-R1-ES algorithm. The important procedures will be introduced in details hereinafter.

At the beginning of **Algorithm 1**, the initialization procedure initializes a random parameter vector θ_0 and several distribution parameters, including mutation strength $\sigma_0 = 0.05$, principal search direction $\mathbf{p}_0 = \mathbf{0}$ and cumulative rank rate $s_0 = 0$. For convenience, we set $F_0 = [F(\theta_0), \ldots, F(\theta_0)]$, which indicates that F_0 contains μ function values of the initialized parameter vector θ_0.

At the t-th generation, R-R1-ES samples λ random vectors $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and λ random variables $r_i \sim \mathcal{N}(\mathbf{0}, 1)$ $(i = 1, ..., \lambda)$, and constructs λ new solutions in the following manner:

$$\mathbf{x}_i = \theta_t + \sigma_t\left(\sqrt{1 - c_{cov}}\mathbf{z}_i + \sqrt{c_{cov}}r_i\mathbf{p}_t\right) \quad (4)$$

Since R-R1-ES adopts the Gaussian distribution model $\mathcal{N}\left(\theta_t, \sigma_t^2\mathbf{C}_t\right)$ where $\mathbf{C}_t$ is determined by Eq. (3), new solutions can be sampled by using Eq. (4). Then, these λ new solutions are sorted in descending order according to their reward function values, and the best μ ones are selected to form $F_{t+1} = [F(\mathbf{x}_{1:\lambda}), \ldots, F(\mathbf{x}_{\mu:\lambda})]$, , where $\mathbf{x}_{i:\lambda}$ $(i = 1, \ldots, \lambda)$ means that it is the i-th best solution among all the λ solutions. The best μ solutions will be utilized to update the distribution mean and principal search direction. The updating of the

Algorithm 1 R-R1-ES

1: **Parameters setting:** $w_i = \frac{\ln(\mu+1)-\ln i}{\mu\ln(\mu+1)-\sum_{j=1}^{\mu}\ln j}(i = 1,\ldots,\mu)$, $\mu_{eff} = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$, $c_{cov} = \frac{1}{3\sqrt{n}+5}$, $c = \frac{2}{n+7}$, $q* = 0.3$, $c_s = 0.3$, $d_\sigma = 1$;
2: **Initialization:** $\theta_0, \sigma_0 = 0.05, \mathbf{p}_0 = \mathbf{0}, s_0 = 0$;
3: Set $F_0 = [F(\theta_0), \ldots, F(\theta_0)]$;
4: **for** $t \leftarrow 0$ to T **do**
5: **for** $i \leftarrow 1$ to λ **do**
6: Sample $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $r_i \sim \mathcal{N}(\mathbf{0}, 1)$;
7: $\mathbf{x}_i \leftarrow \theta_t + \sigma_t\left(\sqrt{1-c_{cov}}\mathbf{z}_i + \sqrt{c_{cov}} r_i \mathbf{p}_t\right)$;
8: **end for**
9: Sort $\mathbf{x}_i$ as $F(\mathbf{x}_{1:\lambda}) \geq F(\mathbf{x}_{2:\lambda}) \geq \ldots \geq F(\mathbf{x}_{\lambda:\lambda})$;
10: Increase μ linearly from μ_0 to λ;
11: Set $F_{t+1} = [F(\mathbf{x}_{1:\lambda}), \ldots, F(\mathbf{x}_{\mu:\lambda})]$;
12: Update distribution mean: $\theta_{t+1} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$;
13: Update principal search direction: $\mathbf{p}_{t+1} \leftarrow (1-c)\mathbf{p}_t + \sqrt{c(2-c)\mu_{eff}}\frac{\theta_{t+1}-\theta_t}{\sigma_t}$;
14: $R_t, R_{t+1} \leftarrow$ ranks of F_t, F_{t+1} in $F_t \cup F_{t+1}$;
15: Compute rank difference: $q \leftarrow \frac{1}{\mu}\sum_{i=1}^{\mu} w_i (R_t(i) - R_{t+1}(i))$;
16: Compute cumulative rank rate: $s_{t+1} \leftarrow (1-c_s)s_t + c_s(q-q*)$;
17: Adapt mutation strength: $\sigma_{t+1} \leftarrow \sigma_t \exp\left(\frac{s_{t+1}}{d_\sigma}\right)$;
18: **if** *restart criterion is fulfilled* **then**
19: Conduct restart procedure;
20: **end if**
21: $t \leftarrow t+1$;
22: **end for**

distribution mean and principal search direction can be formulated as follows:

$$\theta_{t+1} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda} \tag{5}$$

$$\mathbf{p}_{t+1} = (1-c)\mathbf{p}_t + \sqrt{c(2-c)\mu_{eff}}\frac{\theta_{t+1}-\theta_t}{\sigma_t} \tag{6}$$

where $c \in (0,1)$ denotes the changing rate of principal search direction, $\mu_{eff} = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$, and $w_i > 0$ $(i = 1,\ldots,\mu)$ are weighting parameters satisfying $\sum_{i=1}^{\mu} w_i = 1$. Additionally, these weighting parameters are required to satisfy $w_1 \geq w_2 \geq \cdots \geq w_\mu$ for the reason that setting larger weighting parameter value for better solution can put more emphasize on better solutions. The concrete settings for these parameters can be seen in Line 1 of **Algorithm 1**.

As for the adaptation of the mutation strength, R-R1-ES adopts a slightly modified version of the rank-based success rule proposed in [Li and Zhang, 2017] (Lines 14-17 of **Algorithm 1**). The adaptation of mutation strength is based on fitness ranking, and is irrelevant to the concrete objective function value. Firstly, the elements in $F_t \cup F_{t+1}$ are sorted in descending order, and the ranks of the i-th best solutions of F_t and F_{t+1} in $F_t \cup F_{t+1}$ are expressed as $R_t(i)$ and $R_{t+1}(i)$ $(i = 1, ..., \mu)$, respectively. Then, the cumulative rank rate $s_t \in \mathbb{R}$ is computed as follow:

$$s_{t+1} = (1-c_s)\, s_t + c_s\,(q - q*) \tag{7}$$

Based on the computed cumulative rank rate, the mutation strength is adapted in the following manner:

$$\sigma_{t+1} = \sigma_t \exp\left(\frac{s_{t+1}}{d_\sigma}\right) \tag{8}$$

3.3 Mechanisms For Escaping Local Optima

In RL, optimizing the stochastic reward experienced over a full episode of agent interaction can often make an agent trapped in local optima. Thus, it is necessary to design mechanisms for escaping local optima. To this aim, our proposed R-R1-ES algorithm integrates two mechanisms: one is the adaptation of the number of elitists μ, and the other is the restart procedure.

As for the adaptation of μ, we make the value of μ linearly increase from an initial value μ_0 to λ. This adaptation can add more diversity to the updating of distribution mean at the later stage of algorithm, and can promote the balance between exploration and exploitation to a certain extent. Thus, it can make R-R1-ES less prone to getting stuck in local optima. It is noticeable that the settings of w_i $(i = 1,\ldots,\mu)$ and μ_{eff} rely on the value of μ (see Line 1 of **Algorithm 1**). Thus, when the value of μ is changed during the evolutionary process, the values of w_i and μ_{eff} are also adjusted.

As for the restart procedure, it is invoked when the mutation strength $\sigma_t < 10^{-10}$ [1] or the difference between the latest reward function value and the mean of the reward function values during the latest n generations is less than 10% [2] of the latest reward function value. When the restart procedure is invoked, both the values of population size λ and damping parameter d_σ are doubled. This is because that increasing the population size and slowing down the decaying speed of the mutation strength are conducive to the further evolution process. The novel restart procedure fully considers the stochastic characteristics of reward function in RL problems, and thus can make R-R1-ES more suitable for RL problems.

3.4 Analyses Of R-R1-ES

Complexity Of R-R1-ES

The proposed R-R1-ES is of low space and time complexity. On the one hand, R-R1-ES has a space complexity of $O(\lambda n)$ at each generation. On the other hand, since R-R1-ES utilizes a novel distribution model and the sampling procedure only involves the multiplications of scalar and vector (see Eq. (4)), the time complexity of R-R1-ES in terms of the time consumed on one evaluation of objective function is $O(n)$.

[1] The threshold value for mutation strength is set according to the suggestion in [Li and Zhang, 2017].

[2] The threshold value for the difference in reward function in the restart condition is set based on our study of parameter tuning. In fact, this value should change per problem so as to make the performance of our proposed R-R1-ES become better on different problems. However, in our experiments, for the sake of convenience and uniformity, we simply use a fixed value for different problems. To a certain extent, this setting also makes the numerical comparison fairer.

This means that R-R1-ES greatly reduces the time complexity of the original CMA-ES, and can become more efficient when dealing with the issue of training deep neural networks and solving deep RL problems.

Analysis On Updating Of Distribution Mean

Focusing on the updating rule of the distribution mean (see Eq. (5)) in our proposed R-R1-ES, since $\sum_{i=1}^{\mu} w_i = 1$, Eq. (5) is equivalent to the following formula:

$$\theta_{t+1} - \theta_t = \sum_{i=1}^{\mu} w_i \left(\mathbf{x}_{i:\lambda} - \theta_t\right) \tag{9}$$

Recall from previous section that OpenAI-ES is based on NES, and estimates the natural gradient with respect to the distribution mean in a manner shown in Eq. (2). In fact, Eq. (2) can be rewritten as follow:

$$\nabla_{\theta_t} \mathbb{E}\left[F\left(\theta\right)\right] \approx \sum_{i=1}^{\lambda} \frac{F\left(\theta_t^i\right)}{\sigma^2 \lambda} \left(\theta_t^i - \theta_t\right) \tag{10}$$

As can be seen from Eq. (9) and Eq. (10), if we regard the term $\frac{F\left(\theta_t^i\right)}{\sigma^2 \lambda}$ as a weighting parameter (in fact, this term plays a role of weighting parameter), then the updating formula of the distribution mean (i.e., Eq. (5)) adopted in our proposed R-R1-ES can serve as an estimator to the natural gradient $\nabla_{\theta_t} \mathbb{E}\left[F\left(\theta\right)\right]$.

Analysis On Updating Of Principal Search Direction

As noted above, Eq. (5) can approximate the natural gradient $\nabla_{\theta_t} \mathbb{E}\left[F\left(\theta\right)\right]$. Thus, if we replace the term $\theta_{t+1} - \theta_t$ in Eq. (6) with $\nabla_{\theta_t} \mathbb{E}\left[F\left(\theta\right)\right]$, then the updating of principal search direction can be rewritten as follow:

$$\mathbf{p}_{t+1} = \left(1 - c\right) \mathbf{p}_t + \frac{\sqrt{c\left(2 - c\right) \mu_{eff}}}{\sigma_t} \nabla_{\theta_t} \mathbb{E}\left[F\left(\theta\right)\right] \tag{11}$$

For comparison, we pay attention to the momentum method [Qian, 1999]. It is an ameliorated version of stochastic gradient descent, and is commonly used for training neural networks. When handling a maximization problem with objective function f, this method can be formulated as follow:

$$\mathbf{d}_{t+1} = \beta \mathbf{d}_t + \alpha \nabla f \tag{12}$$

where $\alpha, \beta > 0$ are two parameters.

As can be seen from Eq. (11) and Eq. (12), the principal search direction in our proposed R-R1-ES acts as a momentum term. Moreover, the principal search direction can accumulate the natural gradients with respect to the distribution mean over all the generations during the evolutionary process of R-R1-ES. In this way, solutions with good quality can be more likely to be generated.

4 Experiments

The experiments are conducted on classic control theory problems and Atari games.

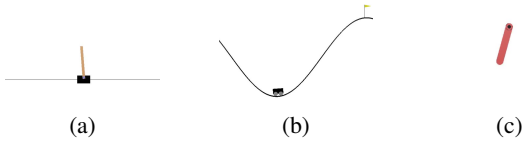


Figure 1: Classic control theory problems.

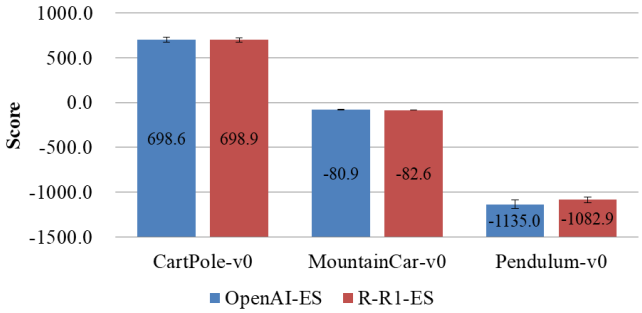


Figure 2: Comparison of reward values obtained by different algorithms on control problems.

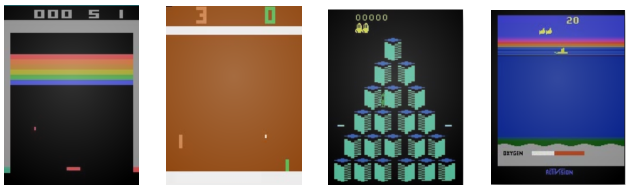
4.1 Classic Control Theory Problems

In the empirical experiments, to investigate the performance of the proposed R-R1-ES algorithm, we firstly adopt three classic control theory problems from the OpenAI Gym [Brockman *et al.*, 2016], including CartPole-v1, MountainCar-v0 and Pendulum-v0. As shown in Figure 1, these three problems attempt to achieve some kind of goal through controlling an object. The goal of CartPole-v1 is to prevent the pole on a cart from falling over, while that of MountainCar-v0 is to drive up the mountain on the right. The Pendulum-v0 aims to swing the pendulum up so it stays upright. These problems stem from the classical RL literature, and can serve as the benchmark problems for verifying the performance of RL algorithm. Thus, we adopt these problems to compare the performance of the proposed R-R1-ES algorithm and the OpenAI-ES [Salimans *et al.*, 2017].

Since these classic control theory problems are relatively simple, we directly use a simple neural network with two hidden layers (one with 30 units and one with 20 units) to act as the network to be trained by ES. Virtual batch normalization is utilized as suggested in [Salimans *et al.*, 2017].

The population size λ is set as $\lambda = 20$. Besides, the initial value μ_0 for the proposed R-R1-ES is set as $\mu_0 = 10$. The maximum number of generations T is set as follows: $T = 100$ for CartPole-v1 and Pendulum-v0, $T = 10000$ for MountainCar-v0. Further, the mechanisms for escaping local optima in the proposed R-R1-ES are temporarily removed in this part of experiments.

For each algorithm on each problem, 21 independent training runs (each run with a final reward value) are performed, and the median reward values are recorded. As shown in Figure 2, the median values obtained by R-R1-ES on CartPole-v1 and Pendulum-v0 are higher than those obtained by OpenAI-



(a) (b) (c) (d)

Figure 3: Atari 2600 games.

ES. As for the MountainCar-v0 problem, the median reward value of R-R1-ES is a bit lower than that of OpenAI-ES. In short, R-R1-ES is on par with OpenAI-ES when solving classic control theory problems.

4.2 Atari Games

In the second part, the performance of the R-R1-ES is further evaluated on four Atari games (i.e., Breakout-v0, Pong-v0, Qbert-v0 and Seaquest-v0), which are shown in Figure 3. And we compare R-R1-ES with four ES variants, including OpenAI-ES [Salimans *et al.*, 2017], NS-ES [Conti *et al.*, 2017], NSR-ES [Conti *et al.*, 2017] and Canonical ES [Chrabaszcz *et al.*, 2018].

As for the neural network architecture, we adopt the suggestion in [Chrabaszcz *et al.*, 2018], which is a bit different from the DQN [Mnih *et al.*, 2015]. And the virtual batch normalization is also utilized as suggested in [Salimans *et al.*, 2017].

As for the training of each ES variant on each game, 20 CPUs are used with a time budget of 20 hours. The population size λ and the initial value μ_0 are set to $\lambda = 798$ and $\mu_0 = 50$, respectively. As for the other parameters used in each ES variant, we follow the suggestions given by their developers, and keep their settings the same as in their original literature.

For each algorithm on each game, 21 independent training runs are performed. After each training, 30 independent evaluation runs (each evaluation run with a score evaluated from the final policy) are conducted for each algorithm, and the mean values of 30 evaluation scores are recorded. In this way, each training run is corresponding to a mean evaluation score, and the median value of all the mean evaluation scores among 21 training runs serves as the final evaluation score obtained by each algorithm on each game. The experimental results are displayed in Figures 4-7.

For the simple Pong-v0 game, as shown in Figure 4, OpenAI-ES obtain the highest scores, and the other four ES variants obtain lower scores.

Seaquest-v0 game has an easy-to-reach local optima, which does not require performing complex behavior. As can be known from [Conti *et al.*, 2017], NS-ES and NSR-ES avoid local optima by means of novelty search (NS) [Lehman and Stanley, 2011] and quality diversity (QD) [Pugh *et al.*, 2016], respectively. Our proposed R-R1-ES integrates two mechanisms (i.e., the adaptation of the number of elitists and the restart procedure) to deal with the issue of local optima. The effects of these mechanisms for escaping local optima are

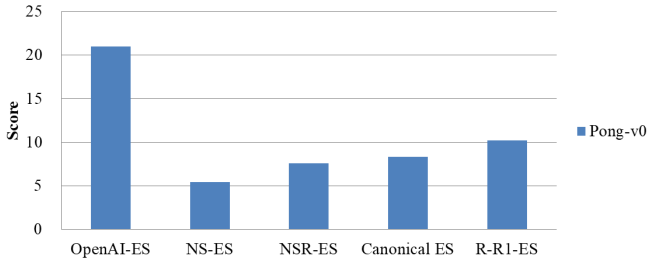


Figure 4: Comparison of evaluation scores obtained by different algorithms on Pong-v0 game.

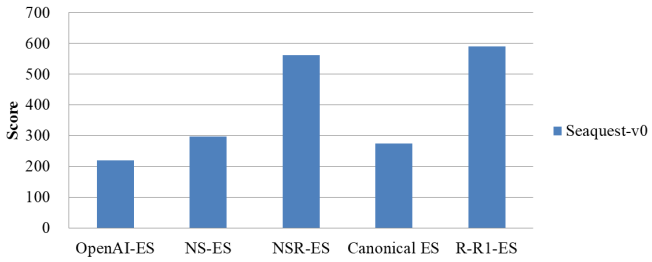


Figure 5: Comparison of evaluation scores obtained by different algorithms on Seaquest-v0 game.

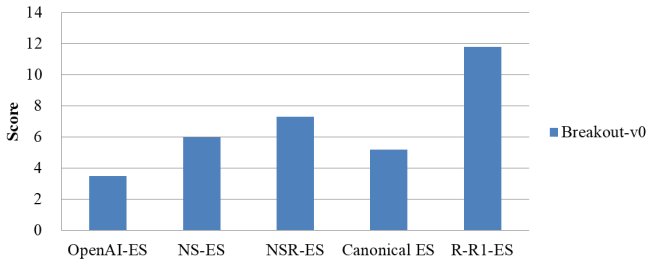


Figure 6: Comparison of evaluation scores obtained by different algorithms on Breakout-v0 game.

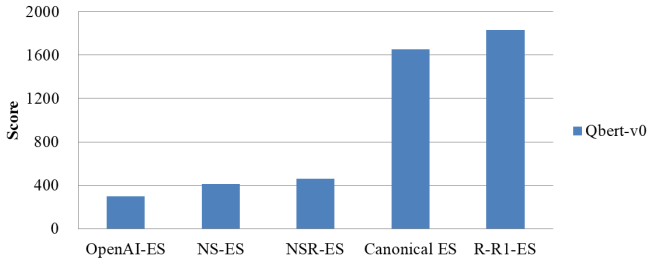


Figure 7: Comparison of evaluation scores obtained by different algorithms on Qbert-v0 game.

experimentally validated and underpinned by Figure 5 which shows that the aforementioned three ES variants (i.e., NS-ES, NSR-ES and R-R1-ES) have nice performance on Seaquest-v0 game. Among them, R-R1-ES obtains the highest score and NSR-ES obtains the second highest score.

As can be seen from Figure 6, all the five algorithms ob-

tain low scores on Breakout-v0 game. This can also provide evidence that Breakout-v0 is challenging to ES algorithms. The proposed R-R1-ES performs best, while two ES variants (NS-ES and NSR-ES) involving NS and QD also have a good performance. Compared to the aforementioned three ES variants which unite mechanism for escaping local optima, the other two ES variants (that is, OpenAI-ES and Canonical ES) perform even worse on Breakout-v0 game. This phenomenon indicates that Breakout-v0 game may also encounter the issue of local optima.

For Qbert-v0 game, both Canonical ES and R-R1-ES obtain higher scores than the other three ES variants which do not adopt the framework of (μ, λ)-ES algorithms. Since both Canonical ES and R-R1-ES belong to the family of (μ, λ)-ES algorithms and they can obtain nice performance on Qbert-v0 game, it is natural to think that their nice performance may be attributed to the superiority of the framework of (μ, λ)-ES algorithms.

To sum up, the whole performance of the proposed R-R1-ES on Atari games is superior to or competitive with existing ES variants for reinforcement learning.

5 Related Work

In the literature of neuroevolution, there exist a lot of researches which employ ES-related methods to train neural networks for RL tasks [Risi and Togelius, 2017]. In 2010, Sehnke *et al.* proposed a novel algorithm called policy gradients with parameter-based exploration (PGPE for short) based on a gradient-based search through model parameter space [Sehnke *et al.*, 2010]. Koutník *et al.* [Koutnik *et al.*, 2010; Koutník *et al.*, 2013] and Srivastava *et al.* [Srivastava *et al.*, 2012] adopted different ways to compress the policy, but they both attempted to apply an ES method to RL tasks with visual inputs in a similar manner. In 2014, Hausknecht *et al.* applied four neuroevolution algorithms with three different state representations to Atari games, and demonstrated that neuroevolution is a promising approach to general video game playing [Hausknecht *et al.*, 2014].

Recently, researchers from OpenAI [Salimans *et al.*, 2017] studied a version of ES algorithm from the class of NES [Wierstra *et al.*, 2008], and demonstrated that it can reliably train neural network policies in a set of RL benchmark environments. This version of ES can obtain competitive performance with state-of-the-art RL algorithms. Notably, it has motivated several nice works by scientists from tech companies and researchers from universities.

Following OpenAI's work, Conti *et al.* attempted to improve the performance of ES on sparse and/or deceptive control tasks by introducing novelty search (NS) [Lehman and Stanley, 2011] and quality diversity (QD) [Pugh *et al.*, 2016], and form two algorithms (called NS-ES and NSR-ES, respectively) [Conti *et al.*, 2017]. The resultant algorithms can effectively avoid local optima encountered by ES.

Zhang *et al.* investigated the relationship between the OpenAI-ES and the stochastic gradient descent method by conducting several MNIST-based experiments [Zhang *et al.*, 2017]. This work gives a nuanced insight into the behaviors of ES, and can promote more informed decisions on the application of ES within RL and other paradigms.

Lehman *et al.* demonstrated that although ES is not just a traditional finite-difference approximator [Lehman *et al.*, 2017]. Most importantly, ES is aimed to optimize the performance of the distribution of solutions rather than a single solution, and thus it can seek out solutions that are more robust to the noise in the parameter space.

Chrabaszcz *et al.* demonstrated that even a very basic canonical ES algorithm, which belongs to the family of (μ, λ)-ES algorithms [Schwefel, 1981], can obtain comparable performance with the OpenAI-ES [Chrabaszcz *et al.*, 2018]. This indicates that with the integration of the state-of-the-art advances made in ES field, we can further promote the performance of ES when solving deep RL problems.

6 Conclusion

In this paper, we have proposed an ES variant called R-R1-ES for training neural networks on RL problems. As the first attempt of applying CMA-ES to RL, our proposed R-R1-ES is based on a simplified CMA-ES for large scale black-box optimization, and involves the procedures of adapting the mutation strength, updating the distribution and the principal search direction, and escaping local optima. Strikingly, the R-R1-ES proposed for RL is of linear time complexity and low space complexity. Moreover, the updating formula of the distribution mean in our proposed R-R1-ES can serve as an estimator to the natural gradient, and the updating of the principal search direction adopted in R-R1-ES acts like the momentum method. Through the empirical experiments on some RL tasks including classic control problems and Atari games, we have demonstrated that the proposed R-R1-ES is superior to or competitive with existing ES variants for RL.

In the future research, the effect of the proposed algorithm on more tough RL tasks can be investigated with the support of more advanced hardware devices. In addition, other state-of-the-art CMA-ES variants for large scale black-box optimization and advanced techniques can be studied in depth and combined to further promote the effectiveness and efficiency of RL algorithm.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grants 61773410 and 61673403.

References

[Beyer, 1993] Hans-Georg Beyer. Toward a theory of evolution strategies: Some asymptotical results from the (1,+λ)-theory. *Evolutionary Computation*, 1(2):165–188, June 1993.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[Chrabaszcz *et al.*, 2018] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. Back to basics: Benchmarking canonical evolution strategies for playing atari. *CoRR*, abs/1802.08842, 2018.

[Conti *et al.*, 2017] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *CoRR*, abs/1712.06560, 2017.

[Hansen and Kern, 2004] Nikolaus Hansen and Stefan Kern. Evaluating the cma evolution strategy on multimodal test functions. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 282–291, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Hansen and Ostermeier, 2014] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2014.

[Hausknecht *et al.*, 2014] Matthew Hausknecht, Joel Lehman, Risto Miikkulainen, and Peter Stone. A neuroevolution approach to general atari game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):355–366, 2014.

[Koutnik *et al.*, 2010] Jan Koutnik, Faustino Gomez, and Jürgen Schmidhuber. Evolving neural networks in compressed weight space. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 619–626, New York, NY, USA, 2010. ACM.

[Koutník *et al.*, 2013] Jan Koutník, Giuseppe Cuccu, Jürgen Schmidhuber, and Faustino Gomez. Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 1061–1068, New York, NY, USA, 2013. ACM.

[Lehman and Stanley, 2011] Joel Lehman and Kenneth O. Stanley. *Novelty Search and the Problem with Objectives*. Springer New York, 2011.

[Lehman *et al.*, 2017] Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O. Stanley. ES is more than just a traditional finite-difference approximator. *CoRR*, abs/1712.06568, 2017.

[Li and Zhang, 2017] Zhenhua Li and Qingfu Zhang. A simple yet efficient evolution strategy for large scale black-box optimization. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2017.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Pugh *et al.*, 2016] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[Qian, 1999] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145 – 151, 1999.

[Risi and Togelius, 2017] Sebastian Risi and Julian Togelius. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):25–41, 2017.

[Ruder, 2016] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[Salimans *et al.*, 2016] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[Salimans *et al.*, 2017] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017.

[Schwefel, 1981] Hans Paul Schwefel. Numerical optimization of computer models. *Journal of the Operational Research Society*, 33(12):1166–1166, 1981.

[Sehnke *et al.*, 2010] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks the Official Journal of the International Neural Network Society*, 23(4):551–559, 2010.

[Srivastava *et al.*, 2012] Rupesh Kumar Srivastava, Jürgen Schmidhuber, and Faustino Gomez. Generalized compressed network search. In *Conference Companion on Genetic and Evolutionary Computation*, pages 647–648, 2012.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Varelas *et al.*, 2018] Konstantinos Varelas, Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Ouassim Ait ElHara, Yann Semet, Rami Kassab, and Frédéric Barbaresco. A comparative study of large-scale variants of cma-es. In Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and Darrell Whitley, editors, *Parallel Problem Solving from Nature – PPSN XV*, pages 3–15, Cham, 2018. Springer International Publishing.

[Wierstra *et al.*, 2008] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3381–3387, June 2008.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[Zhang *et al.*, 2017] Xingwen Zhang, Jeff Clune, and Kenneth O. Stanley. On the relationship between the openai evolution strategy and stochastic gradient descent. *CoRR*, abs/1712.06564, 2017.