Improving Evolutionary Strategies With Past Descent Directions

Florian Meier [*], **Asier Mujika**[†,*], **Marcelo Matheus Gauy, Angelika Steger**
Department of Computer Science
ETH Zürich, Switzerland
{meierflo, asierm, marcelo.matheus, steger}@inf.ethz.ch

Abstract

Evolutionary Strategies (ES) have been shown to be an effective black-box optimization technique for deep neural networks when the access to the true gradients is not possible, like in Reinforcement Learning (RL). Maheswaranathan et al. recently showed how to use surrogate gradients to improve the gradient estimation of ES. In this paper, we follow this line of research and propose a new method to incorporate surrogate gradient information. We prove that it determines the direction of the subspace of evaluated directions that is most aligned with the true gradient. As a consequence, it is guaranteed to always improve on the surrogate direction. Further, we propose that the previous gradient estimate can serve as surrogate gradient for the current search point. We theoretically prove that iteratively using past gradient estimates leads to fast convergence to the true gradient for linear functions and yields a significant improvement for general functions. Finally, we empirically test our approach and show that it considerably improves the gradient estimation of ES at no extra computational cost.

1 Introduction

Designing agents that learn to successfully interact with complex environments is one of the main open problems in artificial intelligence. In reinforcement learning (RL), an agent interacts with an environment and the goal is finding a policy that maximizes the reward. While this goal is often tackled with value function based approaches, evolutionary strategies (ES) have been shown to be a viable alternative [1, 2], and were successfully applied in a variety of different settings [3, 4, 5]. ES approaches are highly parallelizable, account for robust learning while having decent data-efficiency[1]. Moreover, black-box optimization techniques like ES do not require propagation of gradients, are tolerant to long time horizons and do not suffer from sparse reward distributions.

The idea of ES is to compute the gradient of the (smoothed) reward function with respect to the parameters by making random perturbations of the current search point. In many scenarios, surrogate gradients are available for gradient estimation [6]. Here, we define *surrogate gradients* to be directions that are correlated but usually not equal to the true gradient, e.g. they might be biased or unbiased approximations of the gradient. If surrogate gradients are available to the algorithm, it was shown in [6] that it can be beneficial to preferentially sample parameter perturbations from the subspace defined by this direction. However, the proposed algorithm requires knowing in advance the quality of the surrogate gradient, and it remained open how to obtain such surrogate gradients in an RL setting.

Parallel to that, experimental evidence has established that higher order derivatives in deep learning are usually "well behaved", in which case gradients of consecutive parameter updates correlate and

[*]Equal contribution.

[†]Author was supported by grant no. CRSII5_173721 of the Swiss National Science Foundation.

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

applying momentum speeds up convergence [7, 8, 9]. These observations suggest that past update directions are promising candidates for surrogate gradients.

Our contribution is threefold:

- First, we show theoretically how to optimally combine the surrogate gradient directions with new random search directions, even when the quality of these surrogate gradients is not known.
- Second, we propose using the last update direction as a surrogate gradient. We prove that this will optimally converge to the true gradient for linear functions and offer an improvement over ES that depends on the rate of change of the gradient.
- Third, we validate experimentally that these results transfer to practice, that is, the proposed approach computes more accurate gradients than standard ES.

2 Related Work

Evolutionary strategies [10, 11, 12] are black box optimization techniques that approximate the gradient or descent direction by taking finite differences in random directions in parameter space. The usefulness of ES for RL in combination with neural networks was demonstrated in [1], which boosted the popularity of ES methods in RL in recent years [2, 3, 4, 5]. They showed that training was efficient, despite taking a number of samples much smaller than the dimensionality of the network, which leads to very noisy gradients.

The history of descent directions was previously used to adapt the search distribution in covariance matrix adaptation ES (CMA-ES) [13]. However, this approach is closer to second-order methods as it tries to also adapt the step-size based on it and uses a full covariance matrix, which makes the algorithm quadratic and thus, impractical in high-dimensional spaces. There are different linear time approximations of CMA-ES [14] . These approximations do often not work well. For example, when optimizing a linear function, treating the covariance matrix as a diagonal will not lead to a gradient approximation that is aligned with the true gradient. Instead, it will lead to an arbitrarily large step-size in the descent direction. This is a fundamental difference to our approach. We simply try to approximate the gradient more efficiently and then feed this gradient approximation to a first-order optimization algorithm.

The closest work to ours is [6]. In this paper, biased estimates of the gradient are used to 'elongate' the search space along the direction of the biased gradient. However, the approach has two main shortcomings. First, the bias of the surrogate gradient has to be known to know how to adapt the covariance matrix. Second, once the bias of the surrogate gradient is small enough, the algorithm cannot improve on it. In contrast, in our algorithm, the expectation of the cosine is always larger than that of the biased gradient and we do not need to know how biased the surrogate gradient is. These two factors allow us to apply our algorithm iteratively, using past gradient estimates.

Applying different kinds of momentum is one of the standard tools in current deep learning and it has been shown to speed-up learning in a very wide range of tasks [15, 8, 9]. This hints, that for many problems the higher-order terms in deep learning models are "well-behaved" and thus, the gradients don't change too much after each parameter update. These approaches focus on momentum for parameter updates and in fact, our approach can be seen as a form of momentum too but on the search space of ES.

3 Gradient Estimation

We aim at minimizing a function $f : \mathbb{R}^n \to \mathbb{R}$ by steepest descent. ES computes an estimate of the gradient of a smoothed version of f and thus provides a good parameter update direction in scenarios where $\bigtriangledown f$ does not exist, or $\bigtriangledown f$ is impossible or computationally hard to compute.

3.1 Antithetic ES gradient estimator

ES considers the function f_σ that is obtained by *gaussian smoothing*

$$f_\sigma(x) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[f(x + \sigma\epsilon)] \, . \tag{1}$$

The gradient of f_σ with respect to parameters x is given by

$$\bigtriangledown f_\sigma = \frac{1}{\sigma}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[f(x+\sigma\epsilon)\epsilon], \tag{2}$$

which can be sampled by a Monte Carlo estimator that samples random vectors $\epsilon \sim \mathcal{N}(0, I)$. Often antithetic sampling is used, as it reduces variance [2]. The *antithetic ES gradient estimator* using P samples is given by

$$g = \sum_{i=1}^{P} \frac{f(x+\sigma\epsilon^i) - f(x-\sigma\epsilon^i)}{2\sigma}\epsilon^i \ , \tag{3}$$

where $\epsilon^i \sim \mathcal{N}(0, I)$ for $i \in \{1, \ldots, P\}$ and σ is a hyperparameter modulating the magnitude of the parameter perturbations.

3.2 Our one step gradient estimator

Given one surrogate gradient direction ζ, our one step gradient estimator applies the following sampling strategy. First, it estimates how much the gradient points into the direction of ζ by antithetically evaluating f in the direction of ζ. Second, it estimates the part of the gradient that is orthogonal to ζ by essentially applying ES in the subspace that is orthogonal to ζ. More precisely, it evaluates random orthogonal directions that are orthogonal to ζ. This way, our estimator detects precisely how much one should update the parameters into the surrogate direction (the step into this direction has length 0 if the gradient is orthogonal to the surrogate direction and maximal length if they are parallel). Additionally, if the surrogate direction and the gradient are not perfectly aligned, then the gradient estimate strictly improves over the surrogate direction due to the contribution from the evaluated directions orthogonal to ζ. In the following we define our estimator formally and prove that it possesses best possible alignment with the gradient that can be achieved with our sampling scheme.

We assume that k orthogonal surrogate gradient directions $\zeta^1, \ldots, \zeta^k$ are given to our estimator. Denote by $\mathbb{R}_\zeta$ the subspace of $\mathbb{R}_n$ that is spanned by the ζ^i, and by $\mathbb{R}_{\perp\zeta}$ the subspace that is orthogonal to $\mathbb{R}_{\perp\zeta}$. Further, for any vector v, we denote by $\hat{v}$ the normalized vector $\frac{v}{\|v\|}$, and we denote by $\hat{\bigtriangledown} f$ the normalized gradient $\bigtriangledown f$. Let $\hat{\epsilon}^1, \ldots, \hat{\epsilon}^P$ be random orthogonal unit vectors from $\mathbb{R}_{\perp\zeta}$. Then, our estimator is defined as

$$g_{our} = \sum_{i=1}^{k} \frac{f(x+\sigma\hat{\zeta}^i) - f(x-\sigma\hat{\zeta}^i)}{2\sigma}\hat{\zeta}^i + \sum_{i=1}^{P} \frac{f(x+\sigma\hat{\epsilon}^i) - f(x-\sigma\hat{\epsilon}^i)}{2\sigma}\hat{\epsilon}^i \ . \tag{4}$$

We write $\bigtriangledown f = \bigtriangledown f_{\|\zeta} + \bigtriangledown f_{\perp\zeta}$, where $\bigtriangledown f_{\|\zeta}$ and $\bigtriangledown f_{\perp\zeta}$ the projections of $\bigtriangledown f$ on $\mathbb{R}_\zeta$ and $\mathbb{R}_{\perp\zeta}$, respectively. In essence, the first sum in (4) computes $\bigtriangledown f_{\|\zeta}$ by assessing the quality of each surrogate gradient direction, and the second sum estimates $\bigtriangledown f_{\perp\zeta}$ similar to an orthogonalized antithetic ES gradient estimator, that samples directions from $\mathbb{R}_{\perp\zeta}$, see [2]. We remark that we require orthogonal unit directions $\hat{\epsilon}^i$ for the proofs. In practice this is nearly identical as sampling ϵ^i from a $\mathcal{N}(0, I)$ distribution, because in high-dimensional space the norm of $\epsilon^i \sim \mathcal{N}(0, I)$ is highly concentrated around 1 and the cosine of two such random vectors is almost surely very close to 0.

For the sake of analysis, we assume that f is differentiable and we assume equality for the following first order approximation

$$\frac{f(x+\sigma\hat{\epsilon}) - f(x-\sigma\hat{\epsilon})}{2\sigma} \approx \langle \bigtriangledown f(x), \hat{\epsilon}\rangle$$

In the following we will omit the x in $\bigtriangledown f(x)$. Remark that when sampling $\langle \bigtriangledown f, v^i\rangle$ for arbitrary directions v^i, then no information about search directions orthogonal to the subspace spanned by the v^is is obtained. Therefore, one can only hope for finding the best approximation of $\bigtriangledown f$ lying within the subspace spanned by the v^i's. Our first theorem states that g_{our} computes the direction in the subspace spanned by $\zeta^1, \ldots, \zeta^k, \epsilon^1, \ldots, \epsilon^P$ that is most aligned with $\bigtriangledown f$.

Theorem 1 (Optimal direction of g_{our})**.** *Let $\zeta^1, \ldots, \zeta^k, \epsilon^1, \ldots, \epsilon^P$ be orthogonal vectors in $\mathbb{R}^n$. The direction ϵ of the subspace spanned by $\zeta^1, \ldots, \zeta^k, \epsilon^1, \ldots, \epsilon^P$, that maximizes the cosine $\langle \hat{\bigtriangledown} f, \hat{\epsilon}\rangle$ between $\bigtriangledown f$ and ϵ, is given by the projection of $\bigtriangledown f$ on this subspace. Especially, ϵ is given by $g_{our} = \sum_{i=1}^{k}\langle \bigtriangledown f, \hat{\zeta^i}\rangle\hat{\zeta^i} + \sum_{i=1}^{P}\langle \bigtriangledown f, \hat{\epsilon^i}\rangle\hat{\epsilon^i}$, and $\langle \hat{\bigtriangledown} f, \hat{\epsilon}\rangle^2 = \sum_{i=1}^{k}\langle \hat{\bigtriangledown} f, \hat{\zeta^i}\rangle^2 + \sum_{i=1}^{P}\langle \hat{\bigtriangledown} f, \hat{\epsilon^i}\rangle^2$.*

The proof of this theorem follows easily from the Cauchy-Schwarz inequality and is given in the appendix.

3.3 Iterative application of our estimator

We suppose that the last gradient estimate is more aligned with the gradient than a random direction. Thus, we propose to use the previous gradient estimate as surrogate direction for the current time step in our gradient estimator. Thereby, the estimator improves over the surrogate direction in every time step, and converges to the true gradient if the gradient does not change over time. In the following, we rigorously analyze the convergence rate of this iterative gradient estimation process for linear functions, and determine the convergence value for general functions assuming some simplifying assumptions.

Denote by x_t the search point at time t and by ζ_t the parameter update step that is applied at time t, that is, $x_{t+1} = x_t + \zeta_t$. For the analysis, we assume that in order to obtain an estimate for the gradient $\bigtriangledown f(x_t)$, we compute g_{our} by evaluating the last update direction ζ_{t-1} and one random direction ϵ_t orthogonal to ζ_{t-1}:

$$\zeta_t = \langle \bigtriangledown f_t, \hat{\zeta}_{t-1} \rangle \hat{\zeta}_{t-1} + \langle \bigtriangledown f_t, \hat{\epsilon}_t \rangle \hat{\epsilon}_t \ . \tag{5}$$

We remark that restricting the analysis to one direction $\hat{\epsilon}_t$ is no restriction at all, because the right hand side of (4) can be obtained by choosing $\hat{\epsilon}_t = \sum_{i=1}^{P} \langle \bigtriangledown f_t, \hat{\epsilon}^i \rangle \hat{\epsilon}^i$ and $\hat{\zeta}_t = \sum_{i=1}^{k} \langle \bigtriangledown f_t, \hat{\zeta}^i \rangle \hat{\zeta}^i$.

The next theorem quantifies how fast the cosine between ζ_t and $\bigtriangledown f_t$ converges to 1, if f is a linear function, that is, $\bigtriangledown f_t$ does not change over time.

Theorem 2 (Convergence rate for linear functions)**.** *Let $c \approx 1$ be the constant such that the expected cosine squared between two random vectors in an $N-1$-dimensional space is $\frac{c}{N-1}$. Let $X_t = \langle \hat{\bigtriangledown} f, \hat{\zeta}_t \rangle$ be the random variable that denotes the cosine between ζ_t and $\bigtriangledown f_t$ at time t. Then, the expected drift of X_t^2 is $\mathbb{E}[X_t^2 - X_{t-1}^2 | X_{t-1} = x_{t-1}] = (1 - x_{t-1}^2)\frac{c}{N-1}$. Moreover, let $\epsilon > 0$ and define T to be the first point in time t with $X_t^2 \geq 1 - \delta$. It holds*

$$\mathbb{E}[T] \leq \frac{N-1}{c} \min\{\frac{1-\delta}{\delta}, 1 + \ln(1/\delta)\} \ . \tag{6}$$

The first bound $\mathbb{E}[T] \leq \frac{N-1}{c}\frac{1-\delta}{\delta}$ is tight for δ close to 1 and follows by an additive drift theorem, while the second bound $\mathbb{E}[T] \leq \frac{N-1}{c}(1 + \ln(1/\delta))$ is tight for δ close to 0 and follows by a variable drift theorem, see appendix.

Naturally, the linear case is not the most interesting one. However, it is hard to rigorously analyse the case of general f, because it is unpredictable how the gradient $\bigtriangledown f_t$ differs from $\bigtriangledown f_{t-1}$. Note that $\bigtriangledown f_t - \bigtriangledown f_{t-1} \approx H\zeta_{t-1}$, where H is the Hessian matrix of f at x_{t-1}. We define $\alpha_t = \langle \hat{\bigtriangledown} f_t, \hat{\bigtriangledown} f_{t-1} \rangle$ and write $\hat{\bigtriangledown} f_t = \alpha_t \hat{\bigtriangledown} f_{t-1} + \bigtriangledown f_\perp$ where $\bigtriangledown f_\perp$ is orthogonal to $\hat{\bigtriangledown} f_t$ and has norm $1 - \alpha_t^2$. Then, the first term of (5) is equal to

$$\langle \hat{\bigtriangledown} f_t, \hat{\zeta}_{t-1} \rangle^2 = \langle \alpha_t \hat{\bigtriangledown} f_{t-1} + \bigtriangledown f_\perp, \hat{\zeta}_{t-1} \rangle^2 = \left(\alpha_t \langle \hat{\bigtriangledown} f_{t-1}, \hat{\zeta}_{t-1} \rangle + \langle \bigtriangledown f_\perp, \hat{\zeta}_{t-1} \rangle \right)^2 \ . \tag{7}$$

In the following, we assume that $\bigtriangledown f_\perp$ is a direction orthogonal to $\bigtriangledown f_{t-1}$ chosen uniformly at random. Though, this assumption is not entirely true, it allows to get a grasp on the approximate cosine that our estimator is going to converge to.

Theorem 3. *Let x_{t-1} be the cosine between two vectors $\hat{\bigtriangledown} f_{t-1}$ and ζ_{t-1}. Further, let $1 \geq \alpha_t \geq 0$ and $\hat{\bigtriangledown} f_t = \alpha_t \hat{\bigtriangledown} f_{t-1} + \bigtriangledown f_\perp$, where $\bigtriangledown f_\perp$ is a random vector orthogonal to $\hat{\bigtriangledown} f_{t-1}$ with norm $\sqrt{1-\alpha_t^2}$. Choose ζ_t according to Equation* (5) *and define x_t to be the cosine between $\hat{\bigtriangledown} f_t$ and $\hat{\zeta}_t$. Then,*

$$\mathbb{E}[x_t^2 | x_{t-1}] = \left(\alpha_t^2 x_{t-1}^2 + (1-\alpha_t^2)(1-x_{t-1}^2)\frac{c}{N-1} \right) \left(1 - \frac{c}{N-1} \right) + \frac{c}{N-1} \ .$$

The last theorem implies that the evolution of the cosine depends heavily on the cosine α_t between consecutive gradients. Let $A = \frac{(1-\alpha_t^2)\frac{c}{N-1}(1-\frac{c}{N-1}) + \frac{c}{N-1}}{1-(\alpha_t^2 + (1-\alpha_t^2)\frac{c}{N-1})(1-\frac{c}{N-1}))}$. Then, the theorem implies that the drift $\mathbb{E}[x_t^2 - x_{t-1}^2 | x_{t-1}]$ is positive if $x_{t-1} \leq A$ and negative otherwise. Thus, if α_t would not change over time, we would expect x_t to converge to A.

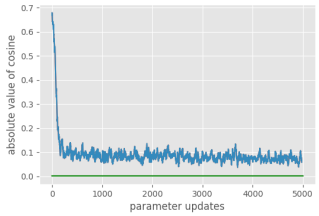

Figure 2: The blue line represents the absolute value of the cosine between the gradient before and after a parameter update. The green line represents the expected absolute value of the cosine with respect to a random vector.

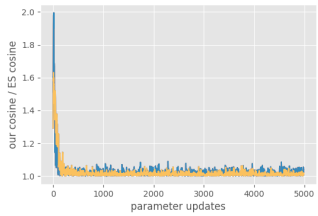

Figure 3: The lines represent the ratio between the gradient computed by our algorithm and the one computed by ES. The blue and yellow line correspond to the best models for SGD and Adam, respectively.

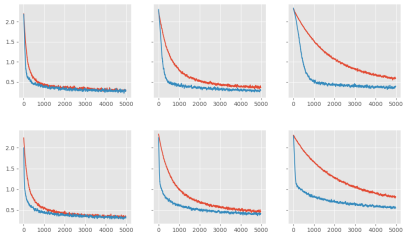

Figure 4: Training log-likelihood (y-axes) of the best three learning rates for ES (red line) with Adam (top) and SGD (bottom). Our algorithm (blue) uses the same parameters as ES.

Table 1: Results on the MNIST digit classification task of the best performing learning rate for each optimizer. While the final performance of our algorithm is still better than ES, the main advantage can be seen early on training.

Optimizer	Steps until loss < 0.6	Final loss
ES + Adam	433	0.242
Ours + Adam	182	0.216
ES + SGD	727	0.305
Ours + SGD	295	0.278

4 Experiments

In this section, we will empirically evaluate the performance of our algorithm when combined with deep neural networks. In our first set of experiments, we will train digit classifiers on MNIST. For this task, we can compute the true gradient, and compare the gradient estimates computed by standard ES and our algorithm to it. Additionally, this task allows us to study the optimization aspects of ES , while ignoring more complex phenomena that occur in Reinforcement Learning (RL), like exploration or function smoothing. Finally, we will also test our algorithm in a set of RL robotics tasks (Roboschool).

4.1 MNIST

For these experiments, we use a fully connected neural network with two hidden layers of 1000 units each with a $tanh$ non-linearity. For standard ES 128 random perturbations are evaluated at each step. For our algorithm the previous gradient estimate and 126 random perturbations are evaluated. To eliminate the noise of the function evaluations, all perturbations are evaluated on the same batch of images (the batch is resampled after every step). We use small perturbations ($\sigma = 0.001$), as the objective function is already differentiable and there is no function evaluation noise. We test both SGD and Adam optimizers with learning rates in the range $10^{0.5}, 10^{0}, \ldots, 10^{-3}$.

We train the network with our algorithm and measure three different aspects over the training process. First, we want to validate the idea that consecutive gradients are correlated in consecutive steps in deep neural networks. For this, at each step of training, we will compute the cosine of the gradient with respect to the gradient at the previous step. We do this by computing the gradient using back-propagation. Second, we will compare the gradients computed by standard ES and our algorithm to the gradient computed by back-propagation. This is done over a single network trained with our algorithm (such that ES and our algorithm are evaluated at the same search points). Third, we train the network also with standard ES, to see the effect of the improved gradient estimation in practice.

Figure 2 show consecutive gradients are highly correlated, specially taking into account the high dimensionality (1.8M parameters) of the model. In Figure 3, we can see how in the initial steps our algorithm get a gradient estimate that is considerably better than standard ES and then it converges to sligthly above what ES does. In fact, the steps in which our algorithm performs best, match the ones where the consecutive gradients are more highly correlated (see Figure 2), just as we would expect from our theoretical analysis. Finally, Figure 4 shows that our algorithm outperforms standard ES in final performance, but specially in speed of convergence, for all parameters that we tried. See also Table 1.

4.2 Robotic RL environments

For the next set of experiments, we evaluate our algorithm on three different robotics task of the Roboschool environment: RoboschoolInvertedPendulum-v1, RoboschoolHalfCheetah-v1 and RoboschoolAnt-v1. We use most of the hyper-parameters from the OpenAI implementation [3] . That is, two hidden layers of 256 units each and with a $tanh$ non-linearity. Further, we use a learning rate of 0.01, a perturbation standard deviation of $\sigma = 0.02$ and the Adam optimizer, and we also apply fitness shaping [14]. For standard ES 128 random perturbations are evaluated at each step. For our algorithm the previous gradient estimate and 126 random perturbations are evaluated.

For the Ant and Cheetah environments, we observed with this setup, that agents often get stuck in a local optima where they stay completely still, instead of running forward. As this happens for both, ES and our algorithm, we tweaked the environments in order to ensure that a true solution to the task is learned and not some some degenerate optima, we tweaked the environments in the following way. We remove the penalty for using electricity and finish the episode if the agent does not make any progress in a given amount of time. In this way, agents consistently escape the local minima. We use a $tanh$ non-linearity on the output of the network, which increased stability of training, as otherwise the output of the network would become very large without an electricity penalty.

Our approach outperforms ES in the pendulum task, but offers only a small improvement over ES in the other two tasks, see Figure 5. We observed that our approach finds parameters with larger norms. This is likely because the consecutive parameter updates are more correlated in our approach. Since we use a fixed perturbation size, we speculate, that the exploration is hindered in our approach as a consequence of the large parameter norms. Thus, that the benefits of more accurate gradients may be counteracted by the worse exploration. However, this is not an inherent issue with our approach and can be fixed, for example, by using other sources of exploration [16], or adaptive perturbation sizes.

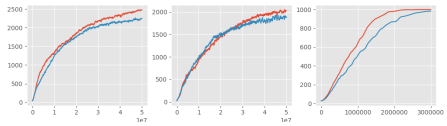

Figure 5: Performance of our algorithm (red line) and ES (blue line) on three different Roboschool tasks: Ant (left), Cheetah (center) and Pendulum (right). The plot shows the mean average reward over 9 repetitions as a function of time-steps (in thousands).

5 Conclusion

We proposed an approach that optimally incorporates surrogate gradient directions into ES, in the sense that it determines the direction with maximal cosine to the true gradient from the subspace of evaluated directions. Such a method has many applications as elucidated in [6]. Further, we showed that previous gradient estimates can iteratively be used as good candidate directions for our gradient estimator. We theoretically quantified the benefits of the proposed iterative gradient estimation. Finally, we showed that our approach in combination with deep neural networks considerably improves the gradient estimation capabilities of ES, at no extra computational cost.

While our approach offered only a small advantage in Reinforcement Learning, we showed that this is likely related to large parameter norms, that hinder exploration. However, if the exploration issue is fixed, the promising results on MNIST indicate that Evolutionary Strategies, a key algorithm in the current Reinforcement Learning toolbox, can be greatly speed-up.

[3] `https://github.com/openai/evolution-strategies-starter`

References

[1] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[2] Krzysztof Choromanski, Mark Rowland, Vikas Sindhwani, Richard E Turner, and Adrian Weller. Structured evolution with compact architectures for scalable policy optimization. *arXiv preprint arXiv:1804.02395*, 2018.

[3] Xiaodong Cui, Wei Zhang, Zoltán Tüske, and Michael Picheny. Evolutionary stochastic gradient descent for optimization of deep neural networks. In *Advances in neural information processing systems*, pages 6048–6058, 2018.

[4] Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *Advances in Neural Information Processing Systems*, pages 5400–5409, 2018.

[5] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.

[6] Niru Maheswaranathan, Luke Metz, George Tucker, Dami Choi, and Jascha Sohl-Dickstein. Guided evolutionary strategies: augmenting random search with surrogate gradients. In *International Conference on Machine Learning*, pages 4264–4273, 2019.

[7] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[8] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

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

[10] Ingo Rechenberg. Evolution strategy: Optimization of technical systems by means of biological evolution. *Fromman-Holzboog, Stuttgart*, 104:15–16, 1973.

[11] Hans-Paul Schwefel. Evolutionsstrategien für die numerische optimierung. In *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, pages 123–176. Springer, 1977.

[12] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

[13] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[14] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, pages 5027–5038, 2018.

[17] Johannes Lengler and Angelika Steger. Drift analysis and evolutionary algorithms revisited. *Combinatorics, Probability and Computing*, 27(4):643–666, 2018.

A Proof of Theorems

In this Section we proof the theorems from the main paper rigorously.

A.1 Proof of Theorem 1

Note that for this theoerem there is no distinction between the directions ζ_i and ϵ_i. For the ease of notation, we therefore denote $\zeta_1, \ldots, \zeta_k, \zeta_1, \ldots, \zeta^P$ by $\epsilon_1, \ldots, \epsilon_m$. The theorem is a simple application of the Cauchy-Schwarz inequality. We can write $\epsilon = \sum_{i=1}^{m} \alpha_i \hat{\epsilon_i}$. Denote by $\bigtriangledown f_{\|\epsilon}(x) =$

$\sum_{i=1}^{m}\langle\bigtriangledown f(x),\hat{\epsilon_i}\rangle\hat{\epsilon_i}$ the projection of $\bigtriangledown f(x)$ in the hyper plane spanned by the ϵ_is. Then, Cauchy-Schwarz implies

$$\langle\bigtriangledown f(x),\epsilon\rangle = \langle\bigtriangledown f_{\|\epsilon}(x),\epsilon\rangle \leq \|\bigtriangledown f_{\|\epsilon}(x)\|\|\epsilon\| \,, \tag{8}$$

where equality holds if and only if ϵ and $\bigtriangledown f_{\|\epsilon}(x)$ have the same direction. In particular, in this case the cosine is

$$\langle\hat{\bigtriangledown} f(x),\hat{\epsilon}\rangle^2 = \frac{\|\bigtriangledown f_{\|\epsilon}(x)\|^2}{\|\bigtriangledown f(x)\|^2} = \sum_{i=1}^{m}\langle\hat{\bigtriangledown} f(x),\hat{\epsilon_i}\rangle^2 \tag{9}$$

□

A.2 Proof of Theorem 2

When taking one sample per time step, $\hat{\epsilon}_t$ is a random unit vector orthogonal to $\hat{\zeta}_{t-1}$. We can split $\hat{\bigtriangledown} f = \bigtriangledown f_{\perp\zeta} + \bigtriangledown f_{\|\zeta}$ into an orthogonal to $\hat{\zeta}_{t-1}$part and a parallel to $\hat{\zeta}_{t-1}$ part. It holds $\langle\hat{\bigtriangledown} f_{\|\zeta},\hat{\epsilon}_t\rangle = 0$ and $\|\bigtriangledown f_{\perp\zeta}\|^2 = 1 - \langle\hat{\bigtriangledown} f,\hat{\zeta}_{t-1}\rangle^2$. Recall that $\hat{\bigtriangledown} f_{\perp\zeta} = \frac{\bigtriangledown f_{\perp\zeta}}{\|\bigtriangledown f_{\perp\zeta}\|}$, then

$$\langle\hat{\bigtriangledown} f,\hat{\epsilon}_t\rangle^2 = \langle\bigtriangledown f_{\perp\zeta},\hat{\epsilon}_t\rangle^2 = (1-\langle\hat{\bigtriangledown} f,\hat{\zeta}_{t-1}\rangle^2)\langle\hat{\bigtriangledown} f_{\perp\zeta},\hat{\epsilon}_t\rangle^2 = (1-X_{t-1}^2)\langle\hat{\bigtriangledown} f_{\perp\zeta},\hat{\epsilon}_t\rangle^2 \,, \tag{10}$$

and therefore

$$X_t^2 = \langle\hat{\bigtriangledown} f,\hat{\zeta}_{t-1}\rangle^2 + \langle\hat{\bigtriangledown} f,\hat{\epsilon}_t\rangle^2 = X_{t-1}^2 + (1-X_{t-1}^2)\langle\hat{\bigtriangledown} f_{\perp\zeta},\hat{\epsilon}_t\rangle^2 \tag{11}$$

$$\tag{12}$$

Define the random process $Y_t = 1 - X_t^2$. It holds

$$Y_t = 1 - X_{t-1}^2 + (1-X_{t-1}^2)\langle\hat{\bigtriangledown} f_{\perp\zeta},\hat{\epsilon}_t\rangle^2 = Y_{t-1}(1-\langle\hat{\bigtriangledown} f_{\perp\zeta},\hat{\epsilon}_t\rangle^2) \,, \text{ and} \tag{13}$$

$$\mathbb{E}[Y_t|Y_{t-1}=y_{t-1}] = y_{t-1}\left(1-\mathbb{E}[\langle\hat{\bigtriangledown} f_{\perp\zeta},\hat{\epsilon}_t\rangle^2]\right) = y_{t-1}\left(1-\frac{c}{N-1}\right) \,, \tag{14}$$

where we used that $\hat{\epsilon}_t$ is a random vector in the $N-1$ dimensional space that is orthogonal to $\hat{\zeta}_{t-1}$ and $\bigtriangledown f_{\perp\zeta}$ is a vector in the same subspace, which implies $\mathbb{E}[\langle\hat{\bigtriangledown} f_{\perp\zeta},\hat{\epsilon}_t\rangle^2] = \frac{c}{N-1}$

In order to derive the first bound on T, we bound the drift of Y_t for $Y_t \geq \delta$.

$$\mathbb{E}[Y_t|Y_{t-1}=y_{t-1},y_{t-1}\geq\delta] = y_{t-1} - y_{t-1}\frac{c}{N-1} \leq y_{t-1} - \delta\frac{c}{N-1} \tag{15}$$

In order to apply Theorem 4, we define the auxiliary process $Z_t = Y_t - \delta$. Then, T is the expected time that Z_t hits 0. Since $\mathbb{E}[Z_t|Z_{t-1}=z_{t-1},z_{t-1}\geq 0] \leq z_{t-1} - \delta\frac{c}{N-1}$ and $Z_0 = 1-\delta$, Theorem 4 implies that

$$\mathbb{E}[T] \leq \frac{1-\delta}{\delta}\frac{N-1}{c} \tag{16}$$

In order to apply Theorem 5, to show the second bound on T, we need to rescale Y_t such that it takes values in $\{0\}\cup[1,\infty)$. Define the auxiliary process Z_t by

$$Z_t = \begin{cases} Y_t/\delta & \text{if } Y_t \geq \delta \\ 0 & \text{if } Y_t < \delta \end{cases} \,. \tag{17}$$

Then, T is the expected time that Z_t hits 0. The process Z_t satisfies

$$\mathbb{E}\left[Z_t|Z_{t-1}=z_{t-1},z_{t-1}\geq 1\right] \leq \mathbb{E}\left[Y_t/\delta|Y_{t-1}=\delta z_{t-1},z_{t-1}\geq 1\right] \leq z_{t-1}\left(1-\frac{c}{N-1}\right) \,. \tag{18}$$

Since $Z_0 = 1/\delta$, Theorem 5 for $h(z) = z\frac{c}{N-1}$ implies that

$$\mathbb{E}[T] \leq \frac{N-1}{c} + \int_1^{1/\delta}\frac{N-1}{cu}\,\mathrm{du} = \frac{N-1}{c}(1+\ln(1/\delta)\,.$$

□

A.3 Proof of Theorem 3

Intuitively, the idea is simple. We have $\langle\hat{\bigtriangledown} f_t, \hat{\zeta}_t\rangle^2 = \langle\hat{\bigtriangledown} f_t, \hat{\zeta}_{t-1}\rangle^2 + \langle\hat{\bigtriangledown} f_t, \hat{\epsilon}_t\rangle^2$. As before we can write $\langle\hat{\bigtriangledown} f_t, \hat{\epsilon}_t\rangle^2 = (1 - \langle\hat{\bigtriangledown} f_t, \hat{\zeta}_{t-1}\rangle^2)\langle\hat{\bigtriangledown} f_{\perp\hat{\zeta}_{t-1}}, \hat{\epsilon}_t\rangle^2$ and note that $\mathbb{E}[\langle\hat{\bigtriangledown} f_{\perp\hat{\zeta}_{t-1}}, \hat{\epsilon}_t\rangle^2] = \frac{c}{N-1}$, as $\hat{\epsilon}$ is a random direction orthogonal to $\hat{\zeta}_{t-1}$. This implies that

$$\mathbb{E}[\langle\hat{\bigtriangledown} f_t, \hat{\zeta}_t\rangle^2] = \left(1 - \frac{c}{N-1}\right)\mathbb{E}[\langle\hat{\bigtriangledown} f_t, \hat{\zeta}_{t-1}\rangle^2] + \frac{c}{N-1} \tag{19}$$

To understand how the x_t evolves we need to analyze how $\langle\hat{\bigtriangledown} f_t, \hat{\zeta}_{t-1}\rangle^2$ relates to $\langle\hat{\bigtriangledown} f_{t-1}, \hat{\zeta}_{t-1}\rangle^2$. To that end, we set $\alpha_t = \langle\hat{\bigtriangledown} f_t, \hat{\bigtriangledown} f_{t-1}\rangle$ and write $\hat{\bigtriangledown} f_t = \alpha_t\hat{\bigtriangledown} f_{t-1} + \bigtriangledown f_\perp$ where $\bigtriangledown f_\perp$ is orthogonal to $\hat{\bigtriangledown} f_t$ and has norm $1 - \alpha_t^2$. Then, the first term of (5) is equal to

$$\langle\hat{\bigtriangledown} f_t, \hat{\zeta}_{t-1}\rangle^2 = \langle\alpha_t\hat{\bigtriangledown} f_{t-1} + \bigtriangledown f_\perp, \hat{\zeta}_{t-1}\rangle^2 = \left(\alpha_t\langle\hat{\bigtriangledown} f_{t-1}, \hat{\zeta}_{t-1}\rangle + \langle\bigtriangledown f_\perp, \hat{\zeta}_{t-1}\rangle\right)^2 . \tag{20}$$

It follows that

$$\mathbb{E}[\langle\hat{\bigtriangledown} f_t, \hat{\zeta}_{t-1}\rangle^2] = \alpha_t^2 x_{t-1}^2 + 2\alpha_t x_{t-1}\mathbb{E}[\langle\bigtriangledown f_\perp, \hat{\zeta}_{t-1}\rangle] + \mathbb{E}[\langle\bigtriangledown f_\perp, \hat{\zeta}_{t-1}\rangle^2] \tag{21}$$

$$= \alpha_t^2 x_{t-1}^2 + (1 - \alpha_t^2)(1 - x_t^2)\frac{c}{N-1} , \tag{22}$$

where $\mathbb{E}[\langle\bigtriangledown f_\perp, \hat{\zeta}_{t-1}\rangle] = 0$ follows from the randomness of $f_\perp$, and

$$\mathbb{E}[\langle\bigtriangledown f_\perp, \hat{\zeta}_{t-1}\rangle^2] = \mathbb{E}[\langle\bigtriangledown f_\perp, \hat{\zeta}_{t-1\perp\bigtriangledown f_{t-1}}\rangle^2] \tag{23}$$

$$= \| \bigtriangledown f_\perp\|^2\|\hat{\zeta}_{t-1\perp\bigtriangledown f_{t-1}}\|^2\mathbb{E}[\langle\frac{\bigtriangledown f_\perp}{\| \bigtriangledown f_\perp\|}, \frac{\hat{\zeta}_{t-1\perp\bigtriangledown f_{t-1}}}{\|\hat{\zeta}_{t-1\perp\bigtriangledown f_{t-1}}\|}\rangle^2] \tag{24}$$

$$= (1 - \alpha_t^2)(1 - x_t^2)\frac{c}{N-1} . \tag{25}$$

Then, plugging in Equation (21) into (19), implies the theorem. □

B Drift Theorems

For the proves of 2, we use two drif theorems from [17].

Theorem 4 (Additive Drift, Theorem 1 from [17])**.** *Let $(X_t)_{t\in\mathbb{N}_0}$ be a Markov chain with state space $S \subset [0, \infty)$ and assume $X_0 = n$. Let T be the earliest point in time $t \geq 0$ such that $X_t = 0$. If there exists $c > 0$ such that for all $x \in S$, $x > 0$ and for all $t \geq 0$ we have*

$$\mathbb{E}[X_{t+1}|X_t = x] \leq x - c .$$

Then,

$$\mathbb{E}[T] \leq \frac{n}{c} .$$

Theorem 5. *Variable Drift, Theorem 4 from [17]] Let $(X_t)_{t\in\mathbb{N}}$ be a Markov chain with state space $S \subset \{0\} \cup [1, \infty)$ and with $X_0 = n$. Let T be the earliest point in time $t \geq 0$ such that $X_t = 0$. Suppose furthermore that there is a positive, increasing function $h : [1, \infty) \to \mathbb{R}_{>0}$ such that for all $x \in S$, $x > 0$ we have for all $t \geq 0$*

$$\mathbb{E}[X_{t+1}|X_t = x] \leq x - h(x) .$$

Then,

$$\mathbb{E}[T] \leq \frac{1}{h(1)} + \int_1^n \frac{1}{h(u)}\,\mathrm{du} .$$