2nd International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI' 2020), 1-3 April 2020, Berlin, Germany

Oral

Topic: <*Machine Learning*>

Sample-Efficient Covariance Matrix Adaptation Evolutional Strategy via Simulated Rollouts in Neural Networks

H. Xue [1], S. Böttger [1], N. Rottmann [1], H. Pandya [2], R. Bruder [1], G. Neumann [3], A. Schweikard [1] and E. Rueckert [1]

[1] University of Luebeck, Institute of Robotics and Cognitive Systems, Luebeck, Germany
[2] University of Lincoln, School of Computer Science, College of Science, Lincoln, UK
[3] Karlsruhe Institute of Technology, Bosch Center for Artificial Intelligence, University of Tuebingen, Germany

Abstract: Gradient-free reinforcement learning algorithms often fail to scale to high dimensions and require a large number of rollouts. In this paper, we propose learning a predictor model that allows simulated rollouts in a rank-based black-box optimizer Covariance Matrix Adaptation Evolutional Strategy (CMA-ES) to achieve higher sample-efficiency. We validated the performance of our new approach on different benchmark functions where our algorithm shows a faster convergence compared to the standard CMA-ES. As a next step, we will evaluate our new algorithm in a robot cup flipping task.

Keywords: CMA-ES, Reinforcement Learning, Dynamic Movement Primitives, Cup Flipping

1. Introduction

Reinforcement Learning (RL) has become a popular approach in robotics [2], where an agent learns a policy from scratch based on the cost. In this paper, we investigate an episodic reinforcement learning problem. Several approaches have been proposed. One categorization of these learning approaches is whether it is a gradient-based approaches or a gradient-free approach. Gradient-based approaches are efficient but sensitive to the design of the cost function, whereas gradient-free approaches remained less affected by the cost function design but less efficient. In this work, we focus on one state-of-the-art gradient-free algorithm, Covariance Matrix Adaptation Evolutional Strategy (CMA-ES) [5].

However, one problem of the CMA-ES is its limited performance in high feature dimensions, leading to larger number of rollouts or convergence in local optima. In real robot control tasks, fast convergence to optimal policies is essential [6][7]. The goal of this paper is to enhance the sample-efficiency of the original CMA-ES algorithm to achieve faster convergence.

In order to enhance of the performance of CMA-ES, several variants have been proposed on top of that. CMA-ES with Active Update [16] adapts the covariance matrix by considering all the offsprings. Some other approaches, e.g., Mirrored Sampling [17], Orthogonal Sampling [18] and Quasi-Gaussian Sampling [21] introduce new ways of proposing offsprings. In Mirrored Sampling, two offsprings are generated symmetrically with one random vector so that the samples spread evenly in the sampling space. Orthogonal Sampling bases itself on Mirrored Sampling, where offspring vectors are orthnormalized by Gram-Schimdt process. In Quasi-Gaussian Sampling, a uniform sampling in unit ball instead of Gaussian distribution is performed so that trust-region effect is enabled. CMA-ES with Increasing Population Size [24] schemes an increasing population size after restart to achieve a more global search. [15] introduces a computationally efficient CMA-ES for large scale optimization by applying Cholesky decomposition into covariance matrix to reduce time and memory. Another work is close relation is [22], where they replaced the original ranking of the candidate solutions in CMA-ES by an approximate ranking using local weighted regression. Some other appraoches suggest online selection strategy to search for the best variant fit into the current optimization function [19][20]. In these approaches, the best variant is chosen via automatic machine learning.

Our approach is categorized as a variant of changing the sampling scheme of the offsprings. However, distinct from the above variants, where some adaptations are only valid under the inherent uni-modal Gaussian distribution, our approach can theoretically be applied to any other black-box optimizers with arbitrary sampling distribution.

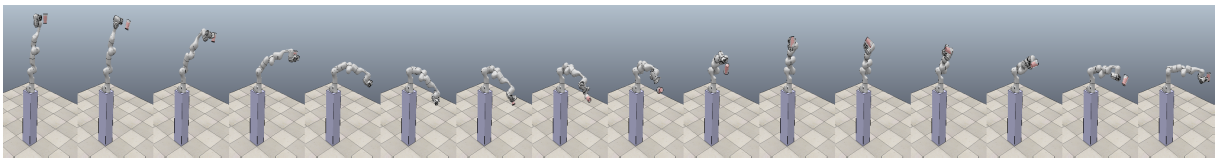

Fig. 4. Illustration of cup flipping task performed on Franka Panda robot in V-REP simulator

Moreover, our approach can be easily combined with previous variants. In this work, we validate this idea in one black-box optimizer CMA-ES. The idea arises from the observation that the samples in previous iterations only contribute indirectly to the update of Gaussian covariance and mean, causing low data-efficiency. One way to improve it is to learn a global predictor model based on all of the tested samples. The idea of learning a predictor model and proposing candidate solutions is also closely related to [12], where they address the problem of automatic machine learning by learning a predictor model mapping the current configuration of the algorithm and the dataset feature to the performance score.

The main contributions of this paper are as follows:
(i) Integration of a predictor model into standard CMA-ES for further performance enhancement with no extra efforts on tunning predictor model hyper-parameters.
(ii) Formulation of the cup flipping task as an RL problem, introducing proper objective function considering the arm constraints.
(iii) Evaluation of CMA-ES with active update and Mirrored Sampling on the flipping task.

2. Methods

In this section, we give a brief overview on our new algorithm CMA-ES with Simulated Rollouts (CMA-ES-SR) and the trajectory formulation using dynamic movement primitives.

2.1. Covariance Matrix Adaptation Evolutional Strategies with Simulated Rollouts (CMA-ES-SR)

CMA-ES is an optimizer that searches for the optimal parameters θ_{opt} that minimizes the cost C. In standard CMA-ES, a multi-variate Gaussian distribution is used to characterize the distribution of candidate solutions (samples). In each generation, N_{pop} candidate solutions are drawn such that $\theta_{1:N_{pop}} \sim \mu + \sigma\mathcal{N}(0, \sum)$, where the mean vector $\mu \in \mathbb{R}^D$ and D represents the dimension of θ, the step size $\sigma \in \mathbb{R}^1$ determines the degree of exploration and $\sum \in \mathbb{R}^{D\times D}$ is the covariance matrix. After each sample is tested, μ, σ and $\sum$ get updated to increase the sampling probability of better candidate solutions. A detailed explanation on the update rule is listed in [25].

However, the standard CMA-ES only uses the previous samples for updating μ, σ and $\sum$, which is data-inefficient. We suggest integrating a predictor model $\mathcal{M}$, such that

$$\mathcal{M}: \boldsymbol{\theta} \rightarrow C \text{ with } \boldsymbol{\theta} \in \mathbb{R}^D, C \in \mathbb{R}^1. \quad (1)$$

The predictor $\mathcal{M}$ is fit to all the tested candidate solutions in each iteration of CMA-ES. With an available model, more promising candidate solutions can be proposed than random samples, leading to faster convergence [12]. In this paper, we use a multi-layer perceptron as it is a universal function approximator

Algorithm 1 CMA-ES-SR
Let N be the number of samples drawn from the current distribution and passed to the model, D be the input dimension. N_{best} is the number of the best samples according to model.

Initialize $N = min(4096, N^d_{pop})$, $N_{best} = 2$, Model M ▷ Can increase 4096
repeat(for each training episode)
  Draw N samples $\Theta = \{\theta_1, ..., \theta_n\}$ under current Gaussian distribution
  Pass Θ to M to get the predicted cost $(\Theta, \hat{C}) = \{(\theta_1, \hat{c}_1)..., (\theta_n, \hat{c}_n)\}$
  Take the N_{best} highest ranking samples Θ_{model} from Θ based on C
  $\Theta_{model} = \{\theta_1, ..., \theta_{N_{best}}\}$
  Take the $N_{pop} - N_{best}$ samples Θ_{random} randomly from $\Theta \setminus \Theta_{model}$
  $\Theta_{pop} = \Theta_{model} \cup \Theta_{random}$
  Evaluate Θ_{model} and Θ_{random}, get $C_{pop} = C_{model} \cup C_{random}$
  Save (Θ_{pop}, C_{pop}) to replay buffer B
  Update M for all samples in B
  Compute $\mu_{model}, \sigma_{model}, \mu_{random}, \sigma_{random}$ from C_{model}, C_{random}
  if $\mu_{model} - \sigma_{model} < \mu_{random} - \sigma_{random}$ **then** ▷ Trust the predictor more
    $N_{best} += 1$
  else ▷ Trust the random samples more
    $N_{best} -= 1$
  end if
  $N_{best} = max(2, min(N_{best}, \frac{N_{pop}}{2}))$
  Update all parameters in standard CMA-ES algorithm
until max iteration or target value is reached
return $(\theta_{best}, c_{best})$

Fig. 1. CMA-ES-SR algorithm

[13]. However, any arbitrary predictor model can be used in general.

The algorithm is shown in **Fig. 1**. With a learned predictor model, N samples are drawn $\theta_{1:N} \sim \mu + \sigma\mathcal{N}(0, \sum)$, with $N \gg N_{pop}$. The best N_{best} solutions are picked according to the model prediction. The final N_{pop} candidate solutions Θ_{pop} consist of the N_{best} predictor-proposed solutions Θ_{model} and $N_{pop} - N_{best}$ samples Θ_{random} randomly drawn from the Gaussian distribution. The value of N_{best} adjusts itself based on the quality of Θ_{model} and Θ_{random}. Meanwhile, we also design a heuristic determining to which extent we trust the model. It is measured by the quality of Θ_{model} and Θ_{random}, where the mean and variance of cost values from both are calculated. We use the optimistic bound similar to the acquisition function in Gaussian process. Since CMA-ES minimizes the objective function, the optimistic bound is calculated by subtracting the variance. When the quality of Θ_{model} is better than that of Θ_{random}, we trust the model more by incrementing N_{best} by one.

Additionally, we set an upper bound for N_{best} to avoid the dominance of the predictor-proposed solutions over the random solutions. Without this upper bound, one potential consequence is that the final candidate solution contains mainly predictor-proposed solutions, i.e., over-trust on the predictor. In the case where the predictor fails to fit the cost landscape but happens to render better solutions than random samples, the algorithm will converge to local optima. For small input dimensions, we also restrict the upper bound of N_{best} and N so that the final set of candidate solutions still follow the Gaussian

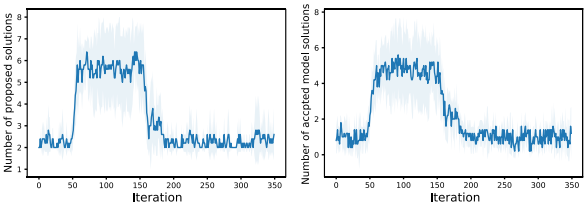

Fig. 3. MLP Model contribution on Ackley with input dimension of 32. N_{pop} is 14.

distribution. Otherwise, the final offsprings can cluster due to the model-fitted landscape especially in low dimension, and no longer follow the original Gaussian distribution. This can cause less exploration and a non-desired update in μ, σ and $\sum$. In high dimensions, N samples remain sparse in space and the final offspring distribution will not be affected by the model-fitted landscape. In short, one advantage of our algorithm is the preservation of the Gaussian distribution on both θ_{model} and θ_{random}. Therefore, it does not affect the Gaussian parameter update.

The hyperparameters of this algorithm are the boundaries of N and N_{best}. Typically, a batch forward is computationally cheap. One can increase the upper bound of N to exceed 4096 if sufficient computation power is available. The lower bound of N (N_{pop}^{D}) scales exponentially with the number of dimensions D, the base can be chosen as values other than N_{pop} as long as the number of model proposed samples N_{best} remains sparse in low dimensions. The default setting of $N_{pop}/2$ takes into account that half of the offsprings affect the update.

The details of model learning are presented in Supplementary Information, Section 5.

2.2. Dynamic Movement Primitives (DMP)

DMPs is an approach to characterize smooth trajectory profiles of a robot [9][10][11]. The trajectory expressiveness is achieved by combining a second-order spring-damper system with a learnable external forcing function $f(t)$.

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f(t), \quad (2)$$

$$f(t) = \frac{\sum_{i=1}^{N} \Psi_i(t) w_i}{\sum_{i=1}^{N} \Psi_i(t)}. \quad (3)$$

The system describes the trajectory in terms of the position y, velocity $\dot{y}$ and acceleration $\ddot{y}$ given the goal position g and the damping coefficients α and β. Forcing function $f(t)$ adds to the trajectory complexity by incorporating a set of weighted sum of N basis functions $\Psi_i(t)$, which can either be stroke-based or rhythmic-based. Variable t denotes the discrete time. For a cup flipping task, we applied stroke-based basis functions

$$\Psi_i(t) = \exp\left[(t - b_i)^2 / 2h_i\right]. \quad (4)$$

It is characterized as a set of Gaussian Basis Functions (GBFs) with pre-defined mean b_i and width h_i.

3. Results and Discussion

We evaluated the performance of CMA-ES-SR on different benchmark optimization problems using the same neural network with no additional tunning. In addition, we started to investigate an episodic RL problem [7] where the goal for a 7-DoF robot arm is to flip a cup filled with liquid around 360 degrees while achieving minimal spillage.

3.1 Benchmarks

For the same benchmark function, we are also interested in the performance enhancement of CMA-ES-SR in different input dimensions. And the detailed settings of benchmarks are shown in Supplementary Information, **Table 3.**

3.2 Performance of CMA-ES-SR on benchmarks

In order to evaluate the performance of CMA-ES-SR compared to the standard CMA-ES, we quantified the following metrics:
(i) The convergence acceleration rate P,
(ii) The best cost value found within a fixed number of iterations,
(iii) The number of predictor-proposed samples N_{best} w.r.t. the number of generations (iterations) and the number of predictor-proposed samples used for mean and covariance update,
The convergence acceleration P is defined as $(I_1 - I_2)/\min(I_1, I_2)$, where I_1 and I_2 refer to the minimal number of generations required to achieve a certain threshold in cost value respectively from CMA-ES and CMA-ES-SR.

The learning curve of CMA-ES-SR on some exemplary benchmarks are shown in **Fig. 2**. It can be observed that our algorithm achieves faster convergence than the original CMA-ES in cases where the learned model is capable of generalizing the cost landscape. Under the circumstance where the model fails to learn the cost landscape, it does not affect the overall optimization process and behaves similarly as the standard CMA-ES. This corresponds to the case of Rosenbrock function, where the cost value is of large magnitude. This poses challenges on regression using MLP and the predictor fails to fit or generalize with our current configuration. Nonetheless, a similar learning curve as the standard CMA-ES can still be observed. Detailed statistics on the convergence acceleration rate P on all tested benchmarks are illustrated in **Table 1**. If one compare the same benchmark of different input dimensions, a consistent performance boost with increasing input dimension can be observed on average.

We also show metric(iii) for one benchmark in **Fig. 3** as an example. Most of the benchmarks also register similar patterns. It can be observed that the number of predictor-proposed solutions N_{best} nearly reaches its upper bound, and the number of accepted solutions proposed by the model takes similar value as N_{best}. This shows predictor-proposed samples are of higher quality than random samples. It can be concluded that the model indeed contributes to higher-quality solutions than random samples when N_{best} reaches its upper bound. At this stage, a faster convergence. The quality of proposed samples is highly dependent on the current step size, mean vector, data distribution and trained model. In the later phase, random samples are at least as good as model-proposed samples, CMA-ES-SR behaves similarly as standard CMA-ES and the algorithm starts to converge.

3.3. Performance of CMA-ES on flipping task

For the flipping task, we used the package Pycma [4]. The trajectories are trained from scratch on two robot arms Franka Panda and KUKA-iiwa R820 in V-REP simulator [14]. All the settings are the same except for the joint constraints, where KUKA-iiwa has stricter joint constraints. Hence, the performance on KUKA-iiwa is restricted. We demonstrated one learned trajectory of Panda in **Fig. 4**. It can be illustrated that the robot arm learns to flip the cup vertically down and finally stopped at an upright pose, with no spillage. We validated the performance five times with a Gaussian noise of zero-mean and variance of 0.2 applied on each joint, shown in **Table 2**.

Table 2. Learned trajectory performance per robot type

Robot Type	Spillage (%)	$\varphi_{\max}$ (°)	φ_{end} (°)
Franka Panda	0±0	179.89±0.04	0.18±0.06
KUKA iiwa R820	23±6	136.01±0.07	1.39±0.08

4. Conclusions

Reducing the number of required rollouts in robot tasks requires sample-efficient RL algorithms. The popular RL algorithm CMA-ES fails to scale to high dimensions. To improve the sample efficiency, we extended the standard CMA-ES by learning a predictor to propose high-quality samples. We tested our new algorithm CMA-ES-SR on different benchmark optimization functions and showed that CMA-ES-SR outperforms the standard CMA-ES by at least 50% in terms of convergence speed. With the increasing dimension, the performance gain is higher. The limitation is the additional overhead in fitting the model. In addition, we demonstrated how to learn a cup flipping task in 7-DoF robot arms which features fast robot motion and fulfills the joint angle and angular velocity constraints. The future work is to evaluate the performance of CMA-ES-SR on the flipping task and extend to different predictor models.

References

[1]. M. Jamil, X. S. Yang, A literature survey of benchmark functions for global optimization problems. arXiv preprint arXiv:1308.4008, 2013

[2]. J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11), 2013, pp.1238-1274.

[3]. A. R. Conn, K. Scheinberg, L. N. Vicente, *Introduction to derivative-free optimization*, Siam., 2009.

[4]. N. Hansen, Y. Akimoto, and P. Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019.

[5]. N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In Proceedings of *IEEE international conference on evolutionary computation*, 1996, pp. 312-317.

[6]. M. P. Deisenroth, G. Neumann, J. Peters, A survey on policy search for robotics, *Foundations and Trends in Robotics*, 2(1–2), 2013, pp. 1-142.

[7]. A. Kupcsik, M. P. Deisenroth, J. Peters, A. P. Loh, P. Vadakkepat, G. Neumann, Model-based contextual policy search for data-efficient generalization of robot skills. Artificial Intelligence, 247, 2017, pp. 415-439.

[8]. S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015

[9]. S. Schaal, Dynamic movement primitives-a framework for motor control in humans and humanoid robotics, In Adaptive motion of animals and machines, Springer, 2006, pp. 261-280.

[10]. E. Rückert, A. d'Avella, Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems. *Frontiers in computational neuroscience*, 7, 2013, 138.

[11]. A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2), 2013, pp. 328-373.

[12]. F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent* optimization, 2011, pp. 507-523

[13]. T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Transactions on Neural Networks, 6(4), 1995, pp. 911-917

[14]. E. Rohmer, S. P. Singh, M. Freese, CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework, In Proceedings of *the International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[15]. I. Loshchilov, A computationally efficient limited memory CMA-ES for large scale optimization. In Proceedings of *the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 397-404.

[16]. D. V. Arnold, & N. Hansen, Active covariance matrix adaptation for the (1+ 1)-CMA-ES. In Proceedings of the 12th annual conference on Genetic and evolutionary computation, 2010, pp. 385-392.

[17]. D. Brockhoff, A. Auger, N. Hansen, D. V. Arnold, T.Hohm, Mirrored sampling and sequential selection for evolution strategies. In *International Conference on Parallel Problem Solving from Nature*, 2010, pp. 11-21

[18]. H. Wang, M. Emmerich, T. Bäck, Mirrored orthogonal sampling with pairwise selection in evolution strategies. In Proceedings of *the 29th Annual ACM Symposium on Applied Computing*, 2014, pp. 154-156.

[19]. D. Vermetten, S. van Rijn, T. Bäck, C. Doerr, Online selection of CMA-ES variants. In Proceedings of *the Genetic and Evolutionary Computation Conference*, 2019, pp. 951-959.

[20]. S. van Rijn, H. Wang, B. van Stein, T. Bäck, Algorithm configuration data mining for cma evolution strategies. In Proceedings of *the Genetic and Evolutionary Computation Conference*, 2017, pp. 737-744

[21]. B. Bischl, O. Mersmann, H. Trautmann, M. Preuß, Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In Proceedings of the *14th annual conference on Genetic and evolutionary computation*, 2012, pp. 313-320.

[22]. Z. Bouzarkouna, A. Auger, D. Y. Ding, Local-meta-model CMA-ES for partially separable functions. In Proceedings of *the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 869-876.

[23]. I. Loshchilov, F. Hutter. Fixing weight decay regularization in adam , 2018

[24]. A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size. In *2005 IEEE congress on evolutionary computation*, pp. 1769-1776, 2005.

[25]. A. Auger, N. Hansen, Tutorial CMA-ES: evolution strategies and covariance matrix adaptation. In Proceedings of *the 14th annual conference companion on Genetic and evolutionary computation*, 2012, pp. 827-848.

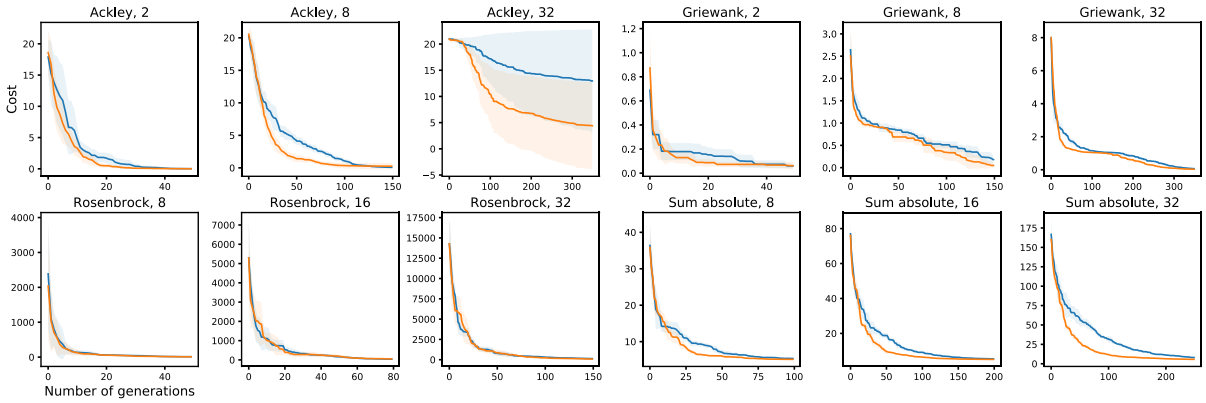

Fig. 2. Learning curves on exemplary benchmarks: The tasks are specified by a function and the feature dimension. and the incumbent settings (the best solutions found so far) are shown. The population size is $\lfloor 4 + 3\ln(D) \rfloor$. For each problem, we ran our algorithm CMA-ES-SR and standard CMA-ES algorithm five times, the mean and variance of the learning curves are shown.

Table 1. Performance of CMA-ES-SR on different benchmarks, where the mean and variance of final converged value in the given iteration are shown. The colums P_{50}, P_{75}, P_{90} refers to the convergence acceleration rate with the threshold set as 50%, 75%, 90% percentile of the final convergence value of standard CMA-ES.

Benchmark Function, Dimension	Iterations	CMA-ES		CMA-ES-SR (NN)				
		mean	var	mean	var	P_{50} (%)	P_{75} (%)	P_{90} (%)
Ackley, 2	50	0.018	0.01	**0.012**	0.01	75	22	43
Ackley, 8	150	0.080	0.04	0.254	0.48	14	71	108
Ackley, 32	350	12.963	9.83	4.408	8.32	91	161	261
DejongsF5, 2	100	**0.399**	0.49	**0.399**	0.49	29	7	13
DejongsF5, 8	100	**1.402**	1.29	2.747	2.41	-300	0	62
DejongsF5, 20	200	1.408	0.74	**0.896**	0.61	0	-33	160
Griewank, 2	50	**0.061**	0.02	0.062	0.03	0	-25	114
Griewank, 8	150	0.181	0.10	**0.049**	0.05	100	32	33
Griewank, 32	350	0.064	0.01	**0.031**	0.00	-29	95	14
Michalewicz, 2	50	-1.841	0.06	**-1.866**	0.01	-50	-67	-75
Michalewicz, 10	100	-4.415	0.87	**-4.491**	1.64	107	16	30
Michalewicz, 20	200	-6.875	1.24	**-7.417**	0.94	467	1343	178
Rosenbrock, 2	50	**0.031**	0.04	0.547	0.85	100	-50	-25
Rosenbrock, 8	100	**5.977**	0.91	6.346	0.84	0	0	17
Rosenbrock, 16	150	**14.999**	0.56	16.238	1.81	0	-14	5
Rosenbrock, 32	250	53.157	42.57	**33.594**	1.63	40	-21	3
Sphere, 8	100	0.022	0.01	**0.010**	0.00	50	50	70
Sphere, 16	150	0.167	0.06	**0.031**	0.01	25	40	94
Sphere, 32	200	1.693	0.51	**0.277**	0.08	25	47	131
Sum absolute, 8	100	5.376	0.11	**5.139**	0.02	-25	46	72
Sum absolute, 16	200	5.398	0.11	**5.104**	0.01	-11	33	67
Sum absolute, 32	250	8.023	0.52	**5.672**	0.11	33	146	125
Averaged Performance	/	4.467	/	**3.093**	/	**33**	**86**	**68**

Supplementary Information

5. Algorithm Details

The network architecture we used is a three-layer MLP with 1024 nodes on each layer. We added two batch normalization layers [8] after non-linear activation layer PReLU. We used AdamW [23] as the optimizer with the default learning rate of 10^{-3}. Loss Metric is mean square error with mean reduction. In order to prevent overfitting, we conducted early stopping with a tolerance of nine iterations and performed train-test split on the experience in the replay buffer. The validation dataset consists of the samples collected from the last optimization iteration. Since CMA-ES updates the Gaussian mean and variance in an accumulative manner, we assume the query points in current iteration are not far away from the query points in the last iteration. A small error in validation set infers similar error for the query points in current iteration given the trained network. In the first iteration, we did not perform any training as no dataset from previous iterations is available. In the second iteration, only samples from the first iteration are available, hence, we defined the validation set to be one-fifth of the samples retrieved in the first iteration. The remaining samples serve as training samples. From the third iteration on, the training and validation dataset were chosen as explained above. In each iteration, we retrained MLP from scratch. All of the samples were pre-processed to have zero-mean and unit-variance in each dimension.

6. Experiment Details

We validated the performance of CMA-ES and CME-ES-SR on different benchmark optimization problems suggested in [1]. The problem settings differ from each other in the domain for each input feature. The details of the problem setting are shown in **Table 3**. The initial mean vector μ_0 passed to CMA-ES(-SR) was chosen uniformly from the input domain, while the initial variance σ_0 passed to CMA-ES optimizer to be $0.3(\mu_{max} - \mu_{max})$. For each benchmark and each input dimension, we ran the experiment five times respectively for CMA-ES, CMA-ES-SR with two models.

Table 3. The settings of benchmark optimization functions

Optimization Problems	Domain μ_0
Ackley	[-32,32]
DeJong F5	[-65,65]
Griewank	[-50,50]
Michalewicz	[0, π]
Rosenbrock	[-2,2]
Sphere	[-10,10]
Sum of absolute value	[-10,10]

In the cup flipping task, the learnable parameters θ describe the robot trajectories τ in joint space. The parameters θ consist of 10 weight parameters w_i for GBFs, two meta-parameters α and g. We characterized each of the seven joint trajectory profiles with one DMP so that the total number of learnable parameters is 84. β is chosen as $\alpha/4$, so that the system is critically-damped [9]. The trajectory time is fixed as five seconds. The cost function is defined as

$$\begin{aligned} C_1 &= a_1 C_{spi} + a_2(180 - \varphi_{max}) + \\ &a_3\varphi_{end} + a_4\delta_{col}C_{col}, \\ C_2 &= C_{worst} + a_5\delta_{con}C_{con}, \end{aligned} \tag{6}$$

where C_1 is the case when joint angle and angular velocity constraints are satisfied and C_2 corresponds to the case of violation. The term φ_{max} is the largest difference of the cup normal vector to the vector (0,0,1) in 3D space throughout the whole trajectory, while the term φ_{end} refers to the same angle difference but at the end of the trajectory. Initially, cup normal vector is (0,0,1). C_{worst} denotes the worst possible of C_1. δ_{col} and δ_{con} are indicator functions telling whether collision happens and robot constraints are met. The cost maximizes the rotation angle of the cup φ_{max} while achieving minimal spillage C_{spi} and the upright final pose φ_{end}. When the constraints are not satisfied, the cost is C_{worst} plus an extra cost for exceeding the joint angle and joint velocity constraints. With such design, the robot arm constraints must be first satisfied so that it can learn to perform flipping.