Efficient Natural Evolution Strategies

Evolution Strategies and Evolutionary Programming Track

Yi Sun
IDSIA
Manno 6928, Switzerland
yi@idsia.ch

Daan Wierstra
IDSIA
Manno 6928, Switzerland
daan@idsia.ch

Tom Schaul
IDSIA
Manno 6928, Switzerland
tom@idsia.ch

Jürgen Schmidhuber
IDSIA
Manno 6928, Switzerland
juergen@idsia.ch

ABSTRACT

Efficient Natural Evolution Strategies (eNES) is a novel alternative to conventional evolutionary algorithms, using the natural gradient to adapt the mutation distribution. Unlike previous methods based on natural gradients, eNES uses a *fast* algorithm to calculate the inverse of the *exact* Fisher information matrix, thus increasing both robustness and performance of its evolution gradient estimation, even in higher dimensions. Additional novel aspects of eNES include optimal fitness baselines and importance mixing (a procedure for updating the population with very few fitness evaluations). The algorithm yields competitive results on both unimodal and multimodal benchmarks.

Keywords

evolution strategies, natural gradient, optimization

1. INTRODUCTION

Evolutionary algorithms aim to optimize a 'fitness' function that is either unknown or too complex to model directly. They allow domain experts to search for good or near-optimal solutions to numerous difficult real-world problems in areas ranging from medicine and finance to control and robotics.

Typically, three objectives have to be kept in mind when developing evolutionary algorithms—we want (1) robust performance; (2) few (potentially costly) fitness evaluations; (3) scalability with problem dimensionality.

We recently introduced Natural Evolution Strategies (NES; [8]), a new class of evolutionary algorithms less ad-hoc than traditional evolutionary methods. Here we propose a novel algorithm within this framework. It retains the theoretically well-founded nature of the original NES while addressing its shortcomings w.r.t. the above objectives.

NES algorithms maintain and iteratively update a multinormal mutation distribution. Parameters are updated by estimating a *natural evolution gradient*, i.e. the natural gradient on the parameters of the mutation distribution, and following it towards better expected fitness. Well-known advantages of natural gradient methods include isotropic convergence on ill-shaped fitness landscapes [2]. This avoids drawbacks of 'vanilla' (regular) gradients which are prone to slow or premature convergence [4].

Our algorithm calculates the natural evolution gradient using the *exact* Fisher information matrix (FIM) and the Monte Carlo-estimated gradient. In conjunction with the techniques of *optimal fitness baselines* and *fitness shaping* this yields robust performance (objective 1).

To reduce the number of potentially costly evaluations (objective 2), we introduce *importance mixing*, a kind of steady-state enforcer which keeps the distribution of the new population conformed to the current mutation distribution.

To keep the computational cost manageable in higher problem dimensions (objective 3), we derive a novel, efficient algorithm for computing the inverse of the exact Fisher information matrix (previous methods were either inefficient or approximate).

The resulting algorithm, *Efficient Natural Evolution Strategies* (eNES), is elegant, requires no additional heuristics and has few parameters that need tuning. It performs consistently well on both unimodal and multimodal benchmarks.

2. EVOLUTION GRADIENTS

First let us introduce the algorithm framework and the concept of evolution gradients. The objective is to maximize a d-dimensional unknown fitness function $f : \mathbb{R}^d \to \mathbb{R}$, while keeping the number of function evaluations – which are considered costly – as low as possible. The algorithm iteratively evaluates a population of size n individuals $\mathbf{z}_1 \dots \mathbf{z}_n$ generated from the mutation distribution $p(\mathbf{z}|\theta)$. It then uses the fitness evaluations $f(\mathbf{z}_1) \dots f(\mathbf{z}_n)$ to adjust parameters θ of the mutation distribution.

Let $J(\theta) = \mathbb{E}[f(\mathbf{z})|\theta]$ be the expected fitness under mutation distribution $p(\mathbf{z}|\theta)$, namely,

$$J(\theta) = \int f(\mathbf{z})\, p(\mathbf{z}|\theta)\, d\mathbf{z}.$$

The core idea of our approach is to find, at each iteration, a small adjustment $\delta\theta$, such that the expected fitness

$J(\theta + \delta\theta)$ is increased. The most straightforward approach is to set $\delta\theta \propto \nabla_\theta J(\theta)$, where $\nabla_\theta J(\theta)$ is the gradient on $J(\theta)$. Using the 'log likelihood trick', the gradient can be written as

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta \int f(\mathbf{z})\, p(\mathbf{z}|\theta)\, d\mathbf{z} \\ &= \int f(\mathbf{z})\, \nabla_\theta p(\mathbf{z}|\theta)\, d\mathbf{z} \\ &= \int f(\mathbf{z})\, \frac{p(\mathbf{z}|\theta)}{p(\mathbf{z}|\theta)} \nabla_\theta p(\mathbf{z}|\theta)\, d\mathbf{z} \\ &= \int p(\mathbf{z}|\theta) \cdot \left(f(\mathbf{z})\, \nabla_\theta \ln p(\mathbf{z}|\theta)\right) d\mathbf{z},.\end{aligned}$$

The last term can be approximated using Monte Carlo:

$$\nabla_\theta^s J(\theta) = \frac{1}{n} \sum\nolimits_{i=1}^{n} f(\mathbf{z}_i)\, \nabla_\theta \ln p(\mathbf{z}_i|\theta),$$

where $\nabla_\theta^s J(\theta)$ denotes the estimated evolution gradient.

In our algorithm, we assume that $p(\mathbf{z}|\theta)$ is a Gaussian distribution with parameters $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$, where $\mathbf{x}$ represents the mean, and $\mathbf{A}$ represents the Cholesky decomposition of the covariance matrix $\mathbf{C}$, such that $\mathbf{A}$ is upper triangular matrix and[1] $\mathbf{C} = \mathbf{A}^\top \mathbf{A}$. The reason why we choose $\mathbf{A}$ instead of $\mathbf{C}$ as primary parameter is twofold. First, $\mathbf{A}$ makes explicit the $d(d+1)/2$ independent parameters determining the covariance matrix $\mathbf{C}$. Second, the diagonal elements of $\mathbf{A}$ are the square roots of the eigenvalues of $\mathbf{C}$, so $\mathbf{A}^\top \mathbf{A}$ is always positive semidefinite. In the rest of the text, we assume θ is column vector of dimension $d_s = d + d(d+1)/2$ with elements in $\langle \mathbf{x}, \mathbf{A} \rangle$ arranged as

$$\left[\left(\theta^0\right)^\top, \left(\theta^1\right)^\top \ldots \left(\theta^d\right)^\top\right]^\top.$$

Here $\theta^0 = \mathbf{x}$ and $\theta^k = [a_{k,k} \ldots a_{k,d}]^\top$ for $1 \leq k \leq d$, where $a_{i,j}$ $(i \leq j)$ denotes the (i,j)-th element of $\mathbf{A}$.

Now we compute

$$\begin{aligned}\mathbf{g}(\mathbf{z}|\theta) &= \nabla_\theta \ln p(\mathbf{z}|\theta) \\ &= \nabla_\theta \{ \frac{d}{2} \ln 2\pi - \frac{1}{2} \ln |\mathbf{A}|^2 \\ &\quad - \frac{1}{2} \left(\mathbf{A}^{-\top}(\mathbf{z} - \mathbf{x})\right)^\top \left(\mathbf{A}^{-\top}(\mathbf{z} - \mathbf{x})\right) \},\end{aligned}$$

where $\mathbf{g}(\mathbf{z}|\theta)$ is assumed to be a d_s-dimensional column vector. The gradient w.r.t. $\mathbf{x}$ is simply

$$\nabla_\mathbf{x} \ln p(\mathbf{z}|\theta) = \mathbf{C}^{-}(\mathbf{z} - \mathbf{x}).$$

The gradient w.r.t. $a_{i,j}$ $(i \leq j)$ is given by

$$\frac{\partial}{\partial a_{i,j}} \ln p(\mathbf{z}|\theta) = r_{i,j} - \delta(i,j)\, a_{i,i}^{-1},$$

where $r_{i,j}$ is the (i,j)-th element of

$$\mathbf{R} = \mathbf{A}^{-\top} (\mathbf{z} - \mathbf{x}) (\mathbf{z} - \mathbf{x})^\top \mathbf{C}^{-}$$

and $\delta(i,j)$ is the Kronecker Delta function.

From $\mathbf{g}(\mathbf{z}|\theta)$, the mutation gradient $\nabla_\theta^s J(\theta)$ can be computed as $\nabla_\theta^s J(\theta) = \frac{1}{n} \mathbf{G}\mathbf{f}$, where $\mathbf{G} = [\mathbf{g}(\mathbf{z}_1|\theta) \ldots \mathbf{g}(\mathbf{z}_n|\theta)]$, and $\mathbf{f} = [f(\mathbf{z}_1) \ldots f(\mathbf{z}_n)]^\top$. We update θ by $\delta\theta = \eta \nabla_\theta^s J(\theta)$, where η is an empirically tuned step size.

[1]For any matrix $\mathbf{Q}$, $\mathbf{Q}^{-}$ denotes its inverse and $\mathbf{Q}^\top$ denotes its transpose.

3. NATURAL GRADIENT

Vanilla gradient methods have been shown to converge slowly in fitness landscapes with ridges and plateaus. Natural gradients [1] constitute a principled approach for dealing with such problems. The natural gradient, unlike the vanilla gradient, has the advantage of always pointing in the direction of the steepest ascent. Furthermore, since the natural gradient is invariant w.r.t. the particular parameterization of the mutation distribution, it can cope with ill-shaped fitness landscapes and provides isotropic convergence properties, which prevents premature convergence on plateaus and avoids overaggressive steps on ridges [1].

In this paper, we consider a special case of the natural gradient $\tilde{\nabla}_\theta J$, defined as

$$\begin{gathered}\delta\theta^\top \tilde{\nabla}_\theta J = \max_{\delta\theta} J(\theta + \delta\theta), \\ \text{s.t. } KL(\theta + \delta\theta || \theta) = \varepsilon,\end{gathered}$$

where ε is an arbitrarily small constant and $KL(\theta' || \theta)$ denotes the Kullback-Leibler divergence between distributions $p(\mathbf{z}|\theta')$ and $p(\mathbf{z}|\theta)$. The constraints impose a geometry on θ which differs from the plain Euclidean one. With $\varepsilon \to 0$, the natural gradient $\tilde{\nabla}_\theta J$ satisfies the necessary condition $\mathbf{F}\tilde{\nabla}_\theta J = \nabla_\theta J$, with $\mathbf{F}$ being the Fisher information matrix:

$$\mathbf{F} = \mathbb{E}\left[\nabla_\theta \ln p(\mathbf{z}|\theta)\, \nabla_\theta \ln p(\mathbf{z}|\theta)^\top\right].$$

If $\mathbf{F}$ is invertible, which may not always be the case, the natural gradient can be uniquely identified by $\tilde{\nabla}_\theta J = \mathbf{F}^{-} \nabla_\theta J$, or estimated from data using $\mathbf{F}^{-} \nabla_\theta^s J$. The adjustment $\delta\theta$ can then be computed by

$$\delta\theta = \eta \mathbf{F}^{-} \nabla_\theta^s J.$$

In the following sub-sections, we show that the FIM can in fact be computed exactly, that it is invertible, and that there exists an efficient[2] algorithm to compute the inverse of the FIM.

3.1 Derivation of the Exact FIM

In the original NES [8], we compute the natural evolution gradient using the empirical Fisher information matrix, which is estimated from the current population. This approach has three important disadvantages. First, the empirical FIM is not guaranteed to be invertible, which could result in unstable estimations. Second, a large population size would be required to approximate the exact FIM up to a reasonable precision. Third, it is highly inefficient to invert the empirical FIM, a matrix with $O(d^4)$ elements.

We circumvent these problems by computing the exact FIM directly from mutation parameters θ, avoiding the potentially unstable and computationally costly method of estimating the empirical FIM from a population which in turn was generated from θ.

In eNES, the mutation distribution is the Gaussian defined by $\theta = \langle \mathbf{x}, \mathbf{A} \rangle$, the precise FIM $\mathbf{F}$ can be computed analytically. Namely, the (m,n)-th element in $\mathbf{F}$ is given by

$$(\mathbf{F})_{m,n} = \frac{\partial \mathbf{x}^\top}{\partial \theta_m} \mathbf{C}^{-} \frac{\partial \mathbf{x}}{\partial \theta_n} + \frac{1}{2} \operatorname{tr}\left(\mathbf{C}^{-} \frac{\partial \mathbf{C}}{\partial \theta_m} \mathbf{C}^{-} \frac{\partial \mathbf{C}}{\partial \theta_n}\right),$$

where θ_m, θ_n denotes the m-th and n-th element in θ. Let i_m, j_m be the a_{i_m, j_m} such that it appears at the $(d+m)$-th

[2]Normally the FIM would involve $d_s^2 = O(d^4)$ parameters, which is intractable for most practical problems.

position in θ. First, notice that

$$\frac{\partial \mathbf{x}^\top}{\partial x_i}\mathbf{C}^-\frac{\partial \mathbf{x}}{\partial x_j} = \left(\mathbf{C}^-\right)_{i,j},$$

and

$$\frac{\partial \mathbf{x}^\top}{\partial a_{i_1,j_1}}\mathbf{C}^-\frac{\partial \mathbf{x}}{\partial a_{i_2,j_2}} = \frac{\partial \mathbf{x}^\top}{\partial x_i}\mathbf{C}^-\frac{\partial \mathbf{x}}{\partial a_{j,k}} = 0.$$

So the upper left corner of the FIM is $\mathbf{C}^-$, and $\mathbf{F}$ has the following shape

$$\mathbf{F} = \left[\begin{array}{cc} \mathbf{C}^- & \mathbf{0} \\ \mathbf{0} & \mathbf{F_A} \end{array}\right].$$

The next step is to compute $\mathbf{F_A}$. Note that

$$\left(\mathbf{F_A}\right)_{m,n} = \frac{1}{2}\,\mathrm{tr}\left[\mathbf{C}^-\frac{\partial \mathbf{C}}{\partial a_{i_m,j_m}}\mathbf{C}^-\frac{\partial \mathbf{C}}{\partial a_{i_n,j_n}}\right].$$

Using the relation

$$\frac{\partial \mathbf{C}}{\partial a_{i,j}} = \frac{\partial}{\partial a_{i,j}}\mathbf{A}^\top\mathbf{A} = \frac{\partial \mathbf{A}^\top}{\partial a_{i,j}}\mathbf{A} + \mathbf{A}^\top\frac{\partial \mathbf{A}}{\partial a_{i,j}},$$

and the properties of the trace, we get

$$\begin{aligned}\left(\mathbf{F_A}\right)_{m,n} &= \mathrm{tr}\left[\mathbf{A}^-\frac{\partial \mathbf{A}}{\partial a_{i_m,j_m}}\mathbf{A}^-\frac{\partial \mathbf{A}}{\partial a_{i_n,j_n}}\right] \\ &\quad + \mathrm{tr}\left[\frac{\partial \mathbf{A}}{\partial a_{i_m,j_m}}\mathbf{C}^-\frac{\partial \mathbf{A}^\top}{\partial a_{i_n,j_n}}\right].\end{aligned}$$

Computing the first term gives us

$$\mathrm{tr}\left[\mathbf{A}^-\frac{\partial \mathbf{A}}{\partial a_{i_m,j_m}}\mathbf{A}^-\frac{\partial \mathbf{A}}{\partial a_{i_n,j_n}}\right] = \left(\mathbf{A}^-\right)_{j_n,i_m}\left(\mathbf{A}^-\right)_{j_m,i_n}.$$

Note that since $\mathbf{A}$ is upper triangular, $\mathbf{A}^-$ is also upper triangular, so the first summand is non-zero iff

$$i_n = i_m = j_n = j_m.$$

In this case, $\left(\mathbf{A}^-\right)_{j_n,i_m} = \left(\mathbf{A}^-\right)_{j_m,i_n} = a^{-1}_{j_n,i_m}$, so

$$\mathrm{tr}\left[\mathbf{A}^-\frac{\partial \mathbf{A}}{\partial a_{i_m,j_m}}\mathbf{A}^-\frac{\partial \mathbf{A}}{\partial a_{i_n,j_n}}\right] = a^{-2}_{i_m,i_n}\delta\left(i_m,i_n,j_m,j_n\right).$$

Here $\delta\left(\cdot\right)$ is the generalized Kronecker Delta function, i.e. $\delta\left(i_m,i_n,j_m,j_n\right) = 1$ iff all four indices are the same. The second term is computed as

$$\mathrm{tr}\left[\frac{\partial \mathbf{A}}{\partial a_{i_m,j_m}}\mathbf{C}^-\frac{\partial \mathbf{A}^\top}{\partial a_{i_n,j_n}}\right] = \left(\mathbf{C}^-\right)_{j_n,j_m}\delta\left(i_n,i_m\right).$$

Therefore, we have

$$\left(\mathbf{F_A}\right)_{m,n} = \left(\mathbf{C}^-\right)_{j_n,j_m}\delta\left(i_n,i_m\right) + a^{-2}_{i_m,i_n}\delta\left(i_m,i_n,j_m,j_n\right).$$

It can easily be proven that $\mathbf{F_A}$ itself is a block diagonal matrix with d blocks along the diagonal, with sizes ranging from d to 1. Therefore, the precise FIM is given by

$$\mathbf{F} = \left[\begin{array}{cccc} \mathbf{F}_0 & & & \\ & \mathbf{F}_1 & & \\ & & \ddots & \\ & & & \mathbf{F}_d \end{array}\right],$$

with $\mathbf{F}_0 = \mathbf{C}^-$ and block $\mathbf{F}_k$ $(d \geq k \geq 1)$ given by

$$\mathbf{F}_k = \left[\begin{array}{cc} a^{-2}_{k,k} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array}\right] + \mathbf{D}_k.$$

Here $\mathbf{D}_k$ is the lower-right square submatrix of $\mathbf{C}^-$ with dimension $d+1-k$, e.g. $\mathbf{D}_1 = \mathbf{C}^-$, and $\mathbf{D}_d = \left(\mathbf{C}^-\right)_{d,d}$.

We prove that the FIM given above is invertible if $\mathbf{C}$ is invertible. $\mathbf{F}_k$ $(1 \leq k \leq d)$ being invertible follows from the fact that the submatrix $\mathbf{D}_k$ on the main diagonal of a positive definite matrix $\mathbf{C}^-$ must also be positive definite, and adding $a^{-2}_{k,k} > 0$ to the diagonal would not decrease any of its eigenvalues. Also note that $\mathbf{F}_0 = \mathbf{C}^-$ is invertible, so $\mathbf{F}$ is invertible.

It is worth pointing out that the block diagonal structure of $\mathbf{F}$ partitions parameters θ into $d+1$ orthogonal groups $\theta^0 \ldots \theta^k$, which suggests that we could modify each group of parameters without affecting other groups. We will need this intuition in the next section.

3.2 Iterative Computation of FIM Inverse

The exact FIM is a block diagonal matrix with $d{+}1$ blocks. Normally, inverting the FIM requires d matrix inversions. However, we can explore the structure of each sub-block in order to make the inverse of $\mathbf{F}$ more efficient, both in terms of time and space complexity.

First, we realize that $\mathbf{F}_d$ is simply a number, so its inversion is given by $\mathbf{F}_d^- = \left(\left(\mathbf{C}^-\right)_{d,d} + a^{-2}_{d,d}\right)^{-1}$, and similarly $\mathbf{D}_d^- = \left(\left(\mathbf{C}^-\right)_{d,d}\right)^{-1}$. Now, letting k vary from $d-1$ to 1, we can compute $\mathbf{F}_k^-$ and $\mathbf{D}_k^-$ directly from $\mathbf{D}_{k+1}^-$. By block matrix inversion

$$\left[\begin{array}{cc} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{array}\right]^- = \left[\begin{array}{cc} \mathbf{Q}_1^- & -\mathbf{P}_{11}^-\mathbf{P}_{12}\mathbf{Q}_2^- \\ -\mathbf{Q}_2^-\mathbf{P}_{21}\mathbf{P}_{11}^- & \mathbf{Q}_2^- \end{array}\right],$$

using

$$\mathbf{Q}_1 = \mathbf{P}_{11} - \mathbf{P}_{12}\mathbf{P}_{22}^-\mathbf{P}_{21},\ \mathbf{Q}_2 = \mathbf{P}_{22} - \mathbf{P}_{21}\mathbf{P}_{11}^-\mathbf{P}_{12},$$

and the Woodbury identity

$$\begin{aligned}\mathbf{Q}_2^- &= \left[\mathbf{P}_{22} + \mathbf{P}_{21}\left(-\mathbf{P}_{11}^-\right)\mathbf{P}_{21}^\top\right]^- \\ &= \mathbf{P}_{22}^- - \mathbf{P}_{22}^-\mathbf{P}_{21}\left[-\mathbf{P}_{11} + \mathbf{P}_{21}^\top\mathbf{P}_{22}^-\mathbf{P}_{21}\right]^-\mathbf{P}_{21}^\top\mathbf{P}_{22}^-,\end{aligned}$$

(also noting that in our case, $\mathbf{P}_{11}$ is a number $\left(\mathbf{C}^-\right)_{k,k}$), we can state

$$\mathbf{Q}_2^- = \mathbf{P}_{22}^- - \frac{\left(\mathbf{P}_{22}^-\mathbf{P}_{21}\right)\left(\mathbf{P}_{22}^-\mathbf{P}_{21}\right)^\top}{\mathbf{P}_{21}^\top\mathbf{P}_{22}^-\mathbf{P}_{21} - \mathbf{P}_{11}}.$$

This can be computed directly from $\mathbf{P}_{22}^-$, i.e. $\mathbf{D}_{k+1}^-$. Skipping the intermediate steps, we propose the following algorithm for computing $\mathbf{F}_k^-$ and $\mathbf{D}_k^-$ from $\mathbf{D}_{k+1}^-$:

$$\begin{aligned} &\mathbf{v} = \left(\mathbf{C}^-\right)_{k+1:d,k},\ w = \left(\mathbf{C}^-\right)_{k,k},\ \mathbf{u} = \mathbf{D}_{k+1}^-\mathbf{v}, \\ &s = \mathbf{v}^\top\mathbf{u},\ q = \left(w-s\right)^{-1},\ q_F = \left(w_F - s\right)^{-1}, \\ &c = -w^{-1}\left(1+qs\right),\ c_F = -w_F^{-1}\left(1+q_F s\right), \\ &\mathbf{F}_k^- = \left[\begin{array}{cc} q_F & c_F\mathbf{u}^\top \\ c_F\mathbf{u}^\top & \mathbf{D}_{k+1}^- + q_F\mathbf{u}\mathbf{u}^\top \end{array}\right], \\ &\mathbf{D}_k^- = \left[\begin{array}{cc} q & c\mathbf{u}^\top \\ c\mathbf{u}^\top & \mathbf{D}_{k+1}^- + q\mathbf{u}\mathbf{u}^\top \end{array}\right]. \end{aligned}$$

Here $\left(\mathbf{C}^-\right)_{k+1:d,k}$ is the sub-vector in $\mathbf{C}^-$ at column k, and row $k+1$ to d. A single update from $\mathbf{D}_{k+1}^-$ to $\mathbf{F}_k^-$ and $\mathbf{D}_k^-$ requires $O\left(\left(d-k\right)^2\right)$ floating point multiplications. The overall complexity of computing all sub-blocks $\mathbf{F}_k^-$, $1 \le k \le d$, is thus $O\left(d^3\right)$.

The algorithm is efficient both in time and storage in the sense that, on one hand, there is no explicit matrix inversion, while on the other hand, the space for storing $\mathbf{D}_k$ (including $\mathbf{F}_k$, if not needed later) can be reclaimed immediately after each iteration, which means that at most $O\left(d^2\right)$ storage is required. Note also that $\mathbf{F}_k^-$ can be used directly to compute $\delta\theta^k$, using $\delta\theta^k = \mathbf{F}_k^- \mathbf{G}^k \mathbf{f}$, where

$$\begin{aligned}\mathbf{G}^k &= \left[\mathbf{g}^k\left(\mathbf{z}_1\right), \ldots, \mathbf{g}^k\left(\mathbf{z}_n\right)\right] \\ &= \left[\nabla_{\theta^k} \ln p\left(\mathbf{z}|\theta\right), \ldots, \nabla_{\theta^k} \ln p\left(\mathbf{z}|\theta\right)\right]\end{aligned}$$

is the submatrix of $\mathbf{G}$ w.r.t. the mutation gradient of θ^k.

To conclude, the algorithm given above efficiently computes the inverse of the exact FIM, required for computing the natural mutation gradient.

4. OPTIMAL FITNESS BASELINES

The concept of *fitness baselines*, first introduced in [8], constitutes an efficient variance reduction method for estimating $\delta\theta$. However, baselines as found in [5] are suboptimal w.r.t. the variance of $\delta\theta$, since this FIM may not be invertible. It is difficult to formulate the variance of $\delta\theta$ directly. However, since the exact FIM is invertible and can be computed efficiently, we can in fact compute an optimal baseline for minimizing the variance of $\delta\theta$, given by

$$\begin{aligned}\mathrm{Var}\left(\delta\theta\right) &= \eta^2 \mathbb{E}\big[\left(\mathbf{F}^- \nabla_\theta^s J - \mathbb{E}\left[\mathbf{F}^- \nabla_\theta^s J\right]\right)^\top \\ &\quad \cdot \left(\mathbf{F}^- \nabla_\theta^s J - \mathbb{E}\left[\mathbf{F}^- \nabla_\theta^s J\right]\right)\big],\end{aligned}$$

where $\nabla_\theta^s J$ is the estimated evolution gradient, which is given by

$$\nabla_\theta^s J = \frac{1}{n} \sum_{i=1}^n \left[f\left(z_i\right) - b\right] \nabla_\theta \ln p\left(\mathbf{z}_i|\theta\right).$$

The scalar b is called the fitness baseline. Adding b does not affect the expectation of $\nabla_\theta^s J$, since

$$\begin{aligned}\mathbb{E}\left[\nabla_\theta^s J\right] &= \nabla_\theta \int \left(f\left(\mathbf{z}\right) - b\right) p\left(\mathbf{z}|\theta\right) d\mathbf{z} \\ &= \nabla_\theta \int f\left(\mathbf{z}\right) p\left(\mathbf{z}|\theta\right) d\mathbf{z}.\end{aligned}$$

However, the variance depends on the value of b, i.e.

$$\begin{aligned}\mathrm{Var}\left(\delta\theta\right) &\propto b^2 \mathbb{E}\left[\left(\mathbf{F}^- \mathbf{G}\mathbf{1}\right)^\top \left(\mathbf{F}^- \mathbf{G}\mathbf{1}\right)\right] \\ &\quad -2b\mathbb{E}\left[\left(\mathbf{F}^- \mathbf{G}\mathbf{f}\right)^\top \left(\mathbf{F}^- \mathbf{G}\mathbf{1}\right)\right] + \text{const.}\end{aligned}$$

Here $\mathbf{1}$ denotes a n-by-1 vector filled with 1s. The optimal value of the baseline is

$$b = \frac{\mathbb{E}\left[\left(\mathbf{F}^- \mathbf{G}\mathbf{f}\right)^\top \left(\mathbf{F}^- \mathbf{G}\mathbf{1}\right)\right]}{\mathbb{E}\left[\left(\mathbf{F}^- \mathbf{G}\mathbf{1}\right)^\top \left(\mathbf{F}^- \mathbf{G}\mathbf{1}\right)\right]}.$$

Assuming individuals are i.i.d., b can be approximated from data by

$$b \simeq \frac{\sum_{i=1}^n f\left(\mathbf{z}_i\right) \left(\mathbf{F}^- \mathbf{g}\left(\mathbf{z}_i\right)\right)^\top \left(\mathbf{F}^- \mathbf{g}\left(\mathbf{z}_i\right)\right)}{\sum_{i=1}^n \left(\mathbf{F}^- \mathbf{g}\left(\mathbf{z}_i\right)\right)^\top \left(\mathbf{F}^- \mathbf{g}\left(\mathbf{z}_i\right)\right)}.$$

In order to further reduce the estimation covariance, we can utilize a parameter-specific baseline for each parameter θ_j individually, which is given by

$$b_j = \frac{\mathbb{E}\left[\left(\mathbf{h}_j \mathbf{G}\mathbf{f}\right)\left(\mathbf{h}_j \mathbf{G}\mathbf{1}\right)\right]}{\mathbb{E}\left[\left(\mathbf{h}_j \mathbf{G}\mathbf{1}\right)\left(\mathbf{h}_j \mathbf{G}\mathbf{1}\right)\right]} \simeq \frac{\sum_{i=1}^n f\left(\mathbf{z}_i\right) \left(\mathbf{h}_j \mathbf{g}\left(\mathbf{z}_i\right)\right)^2}{\sum_{i=1}^n \left(\mathbf{h}_j \mathbf{g}\left(\mathbf{z}_i\right)\right)^2}.$$

Here $\mathbf{h}_j$ is the j-th row vector of $\mathbf{F}^-$.

However, parameter-specific baseline values θ_j might reduce variance too much, which harms the performance of the algorithm. Additionally, adopting different baseline values for correlated parameters may affect the underlying structure of the parameter space, rendering estimations unreliable. To address both of these problems, we follow the intuition that if the (m,n)-th element in the FIM is 0, then parameters θ_m and θ_n are orthogonal and only weakly correlated. Therefore, we propose using the *block fitness baseline*, i.e. a single baseline b^k for each group of parameters θ^k, $0 \le k \le d$. Its formulation is given by

$$\begin{aligned}b^k &= \frac{\mathbb{E}\left[\left(\mathbf{F}_k^- \mathbf{G}^k \mathbf{f}\right)\left(\mathbf{F}_k^- \mathbf{G}^k \mathbf{1}\right)\right]}{\mathbb{E}\left[\left(\mathbf{F}_k^- \mathbf{G}^k \mathbf{1}\right)\left(\mathbf{F}_k^- \mathbf{G}^k \mathbf{1}\right)\right]} \\ &\simeq \frac{\sum_{i=1}^n f\left(\mathbf{z}_i\right) \left(\mathbf{F}_k^- \mathbf{g}^k\left(\mathbf{z}_i\right)\right)^\top \left(\mathbf{F}_k^- \mathbf{g}^k\left(\mathbf{z}_i\right)\right)}{\sum_{i=1}^n \left(\mathbf{F}_k^- \mathbf{g}^k\left(\mathbf{z}_i\right)\right)^\top \left(\mathbf{F}_k^- \mathbf{g}^k\left(\mathbf{z}_i\right)\right)}.\end{aligned}$$

Here $\mathbf{F}_k^-$ denotes the inverse of the k-th diagonal block of $\mathbf{F}^-$, while $\mathbf{G}^k$ and $\mathbf{g}^k$ denote the submatrices corresponding to differentiation w.r.t. θ^k.

In a companion paper [7], we empirically investigated the convergence properties when using the various types of baseline. We found block fitness baselines to be very robust, whereas uniform and parameter-specific baselines sometimes led to premature convergence.

The main complexity for computing the optimal fitness baseline pertains to the necessity of storing a potentially large gradient matrix $\mathbf{G}$, with dimension $d_s \times n \sim O\left(nd^2\right)$. The time complexity, in this case, is $O\left(nd^3\right)$ since we have to multiply each $\mathbf{F}_k^-$ with $\mathbf{G}^k$. For large problem dimensions, the storage requirement may not be acceptable since n also scales with d. We solve this problem by introducing a time-space tradeoff which reduces the storage requirement to $O\left(d^2\right)$ while keeping the time complexity unchanged. In particular, we first compute for each k, a scalar $u_k = \mathbf{a}_k^- \left(\mathbf{z} - \mathbf{x}\right)$, where $\mathbf{a}_k^-$ is the k-th row vector of $\mathbf{A}^-$. Then, for $1 \le i \le n$, we compute the vector $\mathbf{v} = \left(\mathbf{C}^-\right)_{k:d} \left(\mathbf{z} - \mathbf{x}\right)$, where $\left(\mathbf{C}^-\right)_{k:d}$ is the submatrix of $\mathbf{C}^-$ by taking rows k to d. The gradient $\mathbf{g}^k\left(\mathbf{z}_i\right)$ can be computed from u_k and $\mathbf{v}$, and used to compute $\mathbf{F}_k^- \mathbf{g}^k\left(\mathbf{z}_i\right)$ directly. The storage for $\mathbf{g}^k\left(\mathbf{z}_i\right)$ can be immediately reclaimed. Finally, the complexity of computing $\mathbf{g}^k\left(\mathbf{z}_i\right)$ for all i is $O\left(nd\left(d-k\right)\right)$, so the total complexity of computing every element in $\mathbf{G}$ would still be $O\left(nd^3\right)$.

5. IMPORTANCE MIXING

At each generation, we evaluate n new individuals generated from mutation distribution $p\left(\mathbf{z}|\theta\right)$. However, since small updates ensure that the KL divergence between consecutive mutation distributions is generally small, most new individuals will fall in the high density area of the previous mutation distribution $p\left(\mathbf{z}|\theta'\right)$. This leads to redundant fitness evaluations in that same area.

Our solution to this problem is a new procedure called importance mixing, which aims to *reuse* fitness evaluations from the previous generation, while ensuring the updated population conforms to the new mutation distribution.

Importance mixing works in two steps: In the first step, rejection sampling is performed on the previous population, such that individual $\mathbf{z}$ is accepted with probability

$$\min\left\{1,(1-\alpha)\frac{p(\mathbf{z}|\theta)}{p(\mathbf{z}|\theta')}\right\}.$$

Here $\alpha \in [0,1]$ is the *minimal refresh rate*. Let n_a be the number of individuals accepted in the first step. In the second step, reverse rejection sampling is performed as follows: Generate individuals from $p(\mathbf{z}|\theta)$ and accept $\mathbf{z}$ with probability

$$\max\left\{\alpha,1-\frac{p(\mathbf{z}|\theta')}{p(\mathbf{z}|\theta)}\right\}$$

until $n-n_a$ new individuals are accepted. The n_a individuals from the old generation and $n-n_a$ newly accepted individuals together constitute the new population. Note that only the fitnesses of the newly accepted individuals need to be evaluated. The advantage of using importance mixing is twofold: On the one hand, we reduce the number of fitness evaluations required in each generation, on the other hand, if we fix the number of newly evaluated fitnesses, then many more fitness evaluations can potentially be used to yield more reliable and accurate gradients.

The minimal refresh rate α lower bounds the expected proportion of newly evaluated individuals $\rho = \mathbb{E}\left[\frac{n-n_a}{n}\right]$, namely $\rho \geq \alpha$, with the equality holding iff $\theta = \theta'$. In particular, if $\alpha = 1$, all individuals from the previous generation will be discarded, and if $\alpha = 0$, ρ depends only on the distance between $p(\mathbf{z}|\theta)$ and $p(\mathbf{z}|\theta')$. Normally we set α to be a small positive number, e.g. 0.01, to avoid too low an acceptance probability at the second step when $p(\mathbf{z}|\theta')/p(\mathbf{z}|\theta) \simeq 1$.

It can be proven that the updated population conforms to the mutation distribution $p(\mathbf{z}|\theta)$. In the region where $(1-\alpha)p(\mathbf{z}|\theta)/p(\mathbf{z}|\theta') \leq 1$, the probability that an individual from previous generations appears in the new population is

$$p(\mathbf{z}|\theta')\cdot(1-\alpha)p(\mathbf{z}|\theta)/p(\mathbf{z}|\theta') = (1-\alpha)p(\mathbf{z}|\theta).$$

The probability that an individual generated from the second step entering the population is $\alpha p(\mathbf{z}|\theta)$, since

$$\max\left\{\alpha,1-p(\mathbf{z}|\theta')/p(\mathbf{z}|\theta)\right\} = \alpha.$$

So the probability of an individual entering the population is just $p(\mathbf{z}|\theta)$ in that region. The same result holds also for the region where $(1-\alpha)p(\mathbf{z}|\theta)/p(\mathbf{z}|\theta') > 1$.

In a companion paper [7], we measured the usefulness of importance mixing, and found that it reduces the number of required fitness evaluations by a factor 5. Additionally, it reduced the algorithm's sensitivity to the population size.

The computational complexity of importance mixing can be analyzed as follows. For each generated individual $\mathbf{z}$, the probability $p(\mathbf{z}|\theta)$ and $p(\mathbf{z}|\theta')$ need to be evaluated, requiring $O(d^2)$ computations. The total number of individuals generated is bounded by n/α in the worst case, and is close to n on average.

6. FITNESS SHAPING

For problems with wildly fluctuating fitnesses, the gradient is disproportionately distorted by extreme fitness values, which can lead to premature convergence or numerical instability. To overcome this problem, we use *fitness shaping*, an order-preserving nonlinear fitness transformation function [8]. The choice of (monotonically increasing) fitness shaping function is arbitrary, and should therefore be considered to be one of the tuning parameters of the algorithm. We have empirically found that ranking-based shaping functions work best for various problems. The shaping function used for all experiments in this paper was fixed to $f'(\mathbf{z}) = 2i-1$ for $i > 0.5$ and $f'(\mathbf{z}) = 0$ for $i < 0.5$, where i denotes the relative rank of $f(\mathbf{z})$ in the population, scaled between $0\dots1$.

7. EFFICIENT NES

Integrating all the algorithm elements introduced above, the Efficient Natural Evolution Strategy (with block fitness baselines) can be summarized as

```
1   initialize $\mathbf{A} \leftarrow \mathbf{I}$
2   **repeat**
3     compute $\mathbf{A}^{-}$, and $\mathbf{C}^{-} = \mathbf{A}^{-}\mathbf{A}^{-\top}$
4     update population using importance mixing
5     evaluate $f(\mathbf{z}_i)$ for new $\mathbf{z}_i$
6     compute rank-based fitness shaping $\hat{f}$
7     **for** $k = d$ **to** $0$
8       compute the exact FIM inverse $\mathbf{F}_k^{-}$
9       $\mathbf{u} \leftarrow \mathbf{0}$, $\mathbf{v} \leftarrow \mathbf{0}$, $s_1 \leftarrow 0$, $s_2 \leftarrow 0$
10      **for** $i = 1$ **to** $n$
11        $\mathbf{q} \leftarrow \mathbf{F}_k^{-}\mathbf{g}^k(\mathbf{z}_i)$
12        $\mathbf{u} \leftarrow \mathbf{u} + \hat{f}(\mathbf{z}_i)\mathbf{q}$
13        $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{q}$
14        $s_1 \leftarrow s_1 + \hat{f}(\mathbf{z}_i)\mathbf{q}^\top\mathbf{q}$
15        $s_2 \leftarrow s_2 + \mathbf{q}^\top\mathbf{q}$
16      **end**
17      $b^k \leftarrow s_1/s_2$
18      $\delta\theta^k \leftarrow \mathbf{u} - b^k\mathbf{v}$
19    **end**
20    $\theta \leftarrow \theta + \eta\delta\theta$
21  **until** stopping criteria is met
```

Note that vectors $\mathbf{u}$ and $\mathbf{v}$ in line 18 correspond to $\mathbf{F}_k^{-}\mathbf{G}^k\mathbf{f}$ and $\mathbf{F}_k^{-}\mathbf{G}^k\mathbf{1}$, respectively. Summing up the analysis from previous sections, the time complexity of processing a single generation is $O(nd^3)$, while the space complexity is just $O(d^2+nd)$, where $O(nd)$ comes from the need of storing the population. Assuming that n scales linearly with d, our algorithms scales linearly in space and quadratically in time w.r.t. the number of parameters, which is $O(d^2)$. This is a significant improvement over the original NES, whose complexity is $O(d^4)$ in space and $O(d^6)$ in time.

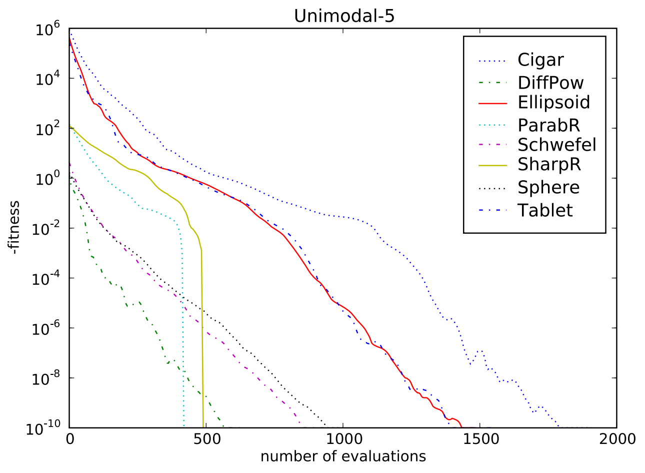
Implementations of eNES are available in both Python and Matlab[3].

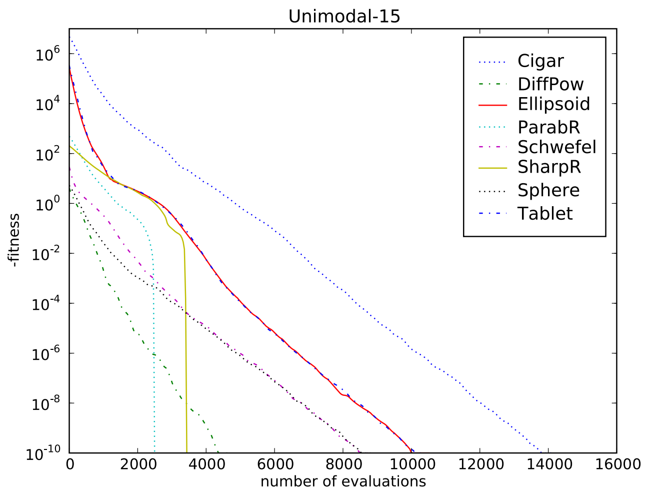
8. EXPERIMENTS

The tunable parameters of Efficient Natural Evolution Strategies are comprised of the population size n, the learning rate η, the refresh rate α and the fitness shaping function. In addition, three kinds of fitness baselines can be used.

We empirically find a good and robust choice for the learning rate η to be 1.0. On some (but not all) benchmarks the

[3]The Python code is part of the PyBrain machine learning library (`www.pybrain.org`) and the Matlab code is available at `www.idsia.ch/~sun/enes.html`





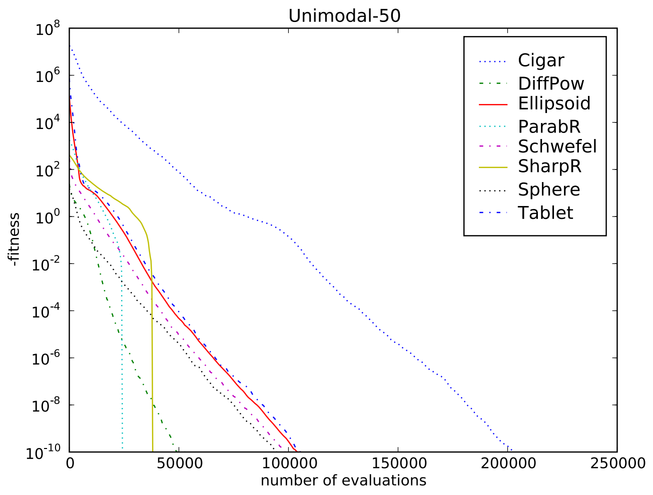


Figure 1: Results on the unimodal benchmark functions for dimension 5, 15 and 50 (from top to bottom).

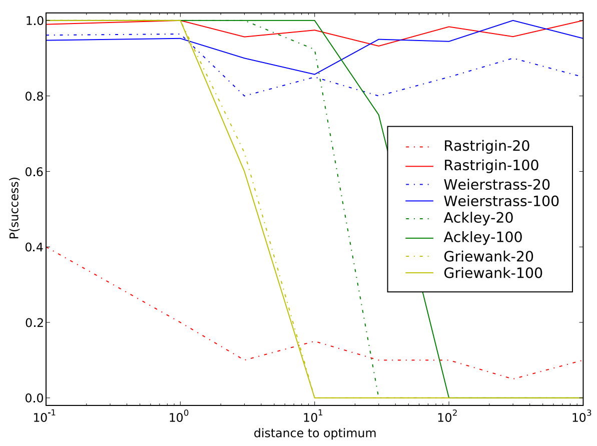


Figure 2: Success percentages varying with initial distances for the multimodal test functions using population sizes 20 and 100.

performance can be further improved by more aggressive updates. Therefore, the only parameter that needs tuning in practice is the population size, which is dependent on both the expected ruggedness of the fitness landscape and the problem dimensionality.

8.1 Benchmark Functions

We empirically validate our algorithm on 9 unimodal and 4 multimodal functions out of the set of standard benchmark functions from [6] and [3], that are typically used in the literature, for comparison purposes and for competitions. We randomly choose the inital guess at average distance 1 from the optimum. In order to prevent potentially biased results, we follow [6] and consistently transform (by a combined rotation and translation) the functions' inputs, making the variables non-separable and avoiding trivial optima (e.g. at the origin). This immediately renders many other methods virtually useless, since they cannot cope with correlated mutation directions. eNES, however, is invariant under translation and rotation. In addition, the rank-based fitness shaping makes it invariant under order-preserving transformations of the fitness function.

8.2 Performance on Benchmark Functions

We ran eNES on the set of unimodal benchmark functions with dimensions 5, 15 and 50 with population sizes 50, 250 and 1000, respectively, using $\eta = 1.0$ and a target precision of 10^{-10}. Figure 1 shows the average performance over 20 runs (5 runs for dimension 50) for each benchmark function. We left out the Rosenbrock function on which eNES is one order of magnitude slower than on the other functions (e.g. 150,000 evaluations on dimension 15). Presumably this is due to the fact that the principal mutation direction is updated too slowly on complex curvatures. Note that SharpR and ParabR are unbounded functions, which explains the abrupt drop-off.

For the experiments on the multimodal benchmark functions we varied the distance of the initial guess to the optimum between 0.1 and 1000. Those runs were performed on dimension 2 with a target precision of 0.01, since here the

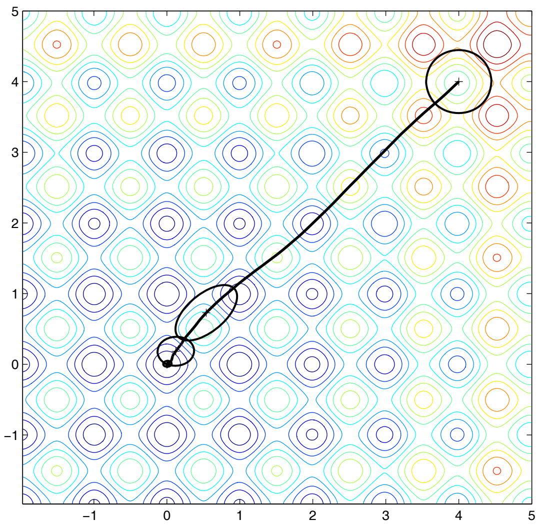


Figure 3: Evolution path and mutation distributions for a typical run on Rastrigin. Ellipsoids correspond to 0.5 standard deviations of the mutation distributions in generations 1, 20, 40.

focus was on avoiding local maxima. We compare the results for population size 20 and 100 (with $\eta = 1.0$). Figure 2 shows, for all tested multimodal functions, the percentage of 100 runs where eNES found the global optimum (as opposed to it getting stuck in a local extremum) conditioned on the distance from the initial guess to the optimum.

Note that for Ackley and Griewank the success probability drops off sharply at a certain distance. For Ackley this is due to the fitness landscapes providing very little global structure to exploit, whereas for Giewank the reason is that the local optima are extremely large, which makes them virtually impossible to escape from[4]. Figure 3 shows the evolution path of a typical run on Rastrigin, and the ellipses corresponding to the mutation distribution at different generations, illustrating how eNES jumps over local optima to reach the global optimum.

For three functions we find that eNES finds the global optimum reliably, even with a population size as small as 20. For the other one, Rastrigin, the global optimum is only reliably found when using a population size of 100.

9. DISCUSSION

Unlike most evolutionary algorithms, eNES boasts a relatively clean derivation from first principles. Using a full multinormal mutation distribution and fitness shaping, the eNES algorithm is invariant under translation and rotation and under order-preserving transformations of the fitness function.

Comparing our empirical results to CMA-ES [3], considered by many to be the 'industry standard' of evolutionary computation, we find that eNES is competitive but slower, especially on higher dimensions. However, eNES is faster on DiffPow for all dimensions. On multimodal benchmarks eNES is competitive with CMA-ES as well, as compared to the results in [8]. Our results, together with the ones on the non-Markovian double pole balancing task reported in [7], collectively show that eNES can compete with state of the art evolutionary algorithms on standard benchmarks.

Future work will also address the problems of automatically determining good population sizes and dynamically adapting the learning rate. Moreover, we plan to investigate the possibility of combining our algorithm with other methods (e.g. Estimation of Distribution Algorithms) to accelerate the adaptation of covariance matrices, improving performance on fitness landscapes where directions of ridges and valleys change abruptly (e.g. the Rosenbrock benchmark).

[4]A solution to this would be to start with a significantly larger initial $\mathbf{C}$, instead of $\mathbf{I}$

10. CONCLUSION

Efficient NES is a novel alternative to conventional evolutionary algorithms, using a natural evolution gradient to adapt the mutation distribution. Unlike previous natural gradient methods, eNES *quickly* calculates the inverse of the *exact* Fisher information matrix. This increases robustness and accuracy of the evolution gradient estimation, even in higher-dimensional search spaces. Importance mixing prevents unnecessary redundancy embodied by individuals from earlier generations. eNES constitutes a competitive, theoretically well-founded and relatively simple method for artificial evolution. Good results on standard benchmarks affirm the promise of this research direction.

11. ACKNOWLEDGMENTS

This research was funded by SNF grants 200020-116674/1, 200021-111968/1 and 200021-113364/1.

12. REFERENCES

[1] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[2] S. Amari and S. C. Douglas. Why natural gradient? volume 2, pages 1213–1216 vol.2, 1998.

[3] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[4] J. Peters and S. Schaal. Natural actor-critic. *Neurocomput.*, 71(7-9):1180–1190, 2008.

[5] J. R. Peters. *Machine learning of motor skills for robotics*. PhD thesis, Los Angeles, CA, USA, 2007. Adviser-Stefan Schaal.

[6] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore, 2005.

[7] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Stochastic search using the natural gradient. *To appear in: Proceedings of the International Conference on Machine Learning (ICML-2009)*, 2009.

[8] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural evolution strategies. In *Proceedings of the Congress on Evolutionary Computation (CEC08), Hongkong*. IEEE Press, 2008.