Journal of Machine Learning Research 15 (2014) 949-980 Submitted 4/11; Revised 5/13; Published 3/14

Natural Evolution Strategies

Daan Wierstra DAAN@DEEPMIND.COM
Tom Schaul TOM@DEEPMIND.COM
DeepMind Technologies Ltd.
Fountain House, 130 Fenchurch Street
London, United Kingdom

Tobias Glasmachers TOBIAS.GLASMACHERS@INI.RUB.DE
Institute for Neural Computation
Universitätsstrasse 150
Ruhr-University Bochum, Germany

Yi Sun YI@IDSIA.CH
Google Inc.
1600 Amphitheatre Pkwy
Mountain View, United States

Jan Peters MAIL@JAN-PETERS.NET
Intelligent Autonomous Systems Institute
Hochschulstrasse 10
Technische Universität Darmstadt, Germany

Jürgen Schmidhuber JUERGEN@IDSIA.CH
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
University of Lugano (USI)/SUPSI
Galleria 2
Manno-Lugano, Switzerland

Editor: Una-May O'Reilly

Abstract

This paper presents Natural Evolution Strategies (NES), a recent family of black-box optimization algorithms that use the natural gradient to update a parameterized search distribution in the direction of higher expected fitness. We introduce a collection of techniques that address issues of convergence, robustness, sample complexity, computational complexity and sensitivity to hyperparameters. This paper explores a number of implementations of the NES family, such as general-purpose multi-variate normal distributions and separable distributions tailored towards search in high dimensional spaces. Experimental results show best published performance on various standard benchmarks, as well as competitive performance on others.

Keywords: natural gradient, stochastic search, evolution strategies, black-box optimization, sampling

1. Introduction

Many real world optimization problems are too difficult or complex to model directly. Therefore, they might best be solved in a 'black-box' manner, requiring no additional information on the objective function (i.e., the 'fitness' or 'cost') to be optimized besides fitness evaluations at certain points in parameter space. Problems that fall within this category are numerous, ranging from applications in health and science (Winter et al., 2005; Shir and Bäck, 2007; Jebalia et al., 2007) to aeronautic design (Hasenjäger et al., 2005; Klockgether and Schwefel, 1970) and control (Hansen et al., 2009).

Numerous algorithms in this vein have been developed and applied in the past fifty years, in many cases providing good and even near-optimal solutions to hard tasks, which otherwise would have required domain experts to hand-craft solutions at substantial cost and often with worse results. The near-infeasibility of finding globally optimal solutions resulted in a fair amount of heuristics in black-box optimization algorithms, leading to a proliferation of complicated yet frequently highly performant methods.

In this paper, we introduce Natural Evolution Strategies (NES), a novel black-box optimization framework which boasts a relatively clean derivation, yet achieves state-of-the-art performance (with the help of well-chosen heuristic methods). The core idea, similar to the framework of estimation of distribution algorithms (EDAs) (Mühlenbein and Paass, 1996; Larrañaga, 2002; Pelikan et al., 2000) and many evolution strategies approaches (e.g., Ostermeier et al. 1994), is to maintain and iteratively update a search distribution from which search points are drawn and subsequently evaluated. However, NES updates the search distribution in the direction of higher expected fitness using the natural gradient (whereas EDAs, for example, typically use maximum likelihood methods to *fit* the distribution of search points).

1.1 Continuous Black-Box Optimization

The problem of black-box optimization has spawned a wide variety of approaches. A first class of methods was inspired by classic optimization methods, including simplex methods such as Nelder-Mead (Nelder and Mead, 1965), as well as members of the quasi-Newton family of algorithms. Simulated annealing (Kirkpatrick et al., 1983), a popular method introduced in 1983, was inspired by thermodynamics, and is in fact an adaptation of the Metropolis-Hastings algorithm. Other methods, such as those inspired by evolution, have been developed from the early 1950s on. These include the broad class of genetic algorithms (Holland, 1975; Goldberg, 1989), differential evolution (Storn and Price, 1997), estimation of distribution algorithms (Larrañaga, 2002; Pelikan et al., 2000; Bosman and Thierens, 2000; Bosman et al., 2007; Pelikan et al., 2006), particle swarm optimization (Kennedy and Eberhart, 2001), and the cross-entropy method (Rubinstein and Kroese, 2004).

Evolution strategies (ES), introduced by Ingo Rechenberg and Hans-Paul Schwefel in the 1960s and 1970s (Rechenberg and Eigen, 1973; Schwefel, 1977), were designed to cope with high-dimensional continuous-valued domains and have remained an active field of research for more than four decades (Beyer and Schwefel, 2002). ESs involve evaluating the fitness of real-valued genotypes in batch ('generation'), after which the best genotypes are kept, while the others are discarded. Survivors then procreate (by slightly mutating all of their genes) in order to produce the next batch of offspring. This process, after several generations, was shown to lead to reasonable to excellent results for many difficult optimization problems. The algorithm framework has been developed extensively over the years to include the representation of correlated mutations by the use of a full covariance matrix. This allowed the framework to capture interrelated dependencies by exploiting the covariances while 'mutating' individuals for the next generation. The culminating algorithm, the covariance matrix adaptation evolution strategy (CMA-ES; Hansen and Ostermeier, 2001), has proven successful in numerous studies (e.g., Friedrichs and Igel, 2005; Muller et al., 2002; Shepherd et al., 2006). While evolution strategies have shown to be effective at black-box optimization, analyzing the actual dynamics of the procedure turns out to be difficult, the considerable efforts of various researchers notwithstanding (Beyer, 2001; Jägersküpper, 2007; Jebalia et al., 2010; Auger, 2005; Schaul, 2012f).

1.2 The NES Family

Natural Evolution Strategies (NES) are a family of evolution strategies which iteratively update a search distribution by using an estimated gradient on its distribution parameters.

The general procedure is as follows: the parameterized search distribution is used to produce a batch of search points, and the fitness function is evaluated at each such point. The distribution's parameters (which include strategy parameters) allow the algorithm to adaptively capture the (local) structure of the fitness function. For example, in the case of a Gaussian distribution, this comprises the mean and the covariance matrix. From the samples, NES estimates a search gradient on the parameters towards higher expected fitness. NES then performs a gradient ascent step along the *natural gradient*, a second-order method which, unlike the plain gradient, renormalizes the update w.r.t. uncertainty. This step is crucial, since it prevents oscillations, premature convergence, and undesired effects stemming from a given parameterization (see Section 2.3 and Figure 2 for an overview on how the natural gradient addresses those issues). The entire process reiterates until a stopping criterion is met.

All members of the 'NES family' operate based on the same principles. They differ in the type of distribution and the gradient approximation method used. Different search spaces require different search distributions; for example, in low dimensionality it can be highly beneficial to model the full covariance matrix. In high dimensions, on the other hand, a more scalable alternative is to limit the covariance to the diagonal only. In addition, highly multi-modal search spaces may benefit from more heavy-tailed distributions (such as Cauchy, as opposed to the Gaussian). A last distinction arises between distributions where we can analytically compute the natural gradient, and more general distributions where we need to estimate it from samples.

1.3 Paper Outline

This paper builds upon and extends our previous work on Natural Evolution Strategies (Wierstra et al., 2008; Sun et al., 2009a,b; Glasmachers et al., 2010a,b; Schaul et al., 2011), and is structured as follows: Section 2 presents the general idea of search gradients as described in Wierstra et al. (2008), explaining stochastic search using parameterized distributions while doing gradient ascent towards higher expected fitness. The limitations of the plain gradient are exposed in Section 2.2, and subsequently addressed by the introduction of the natural gradient (Section 2.3), resulting in the canonical NES algorithm.

Section 3 then regroups a collection of techniques that enhance NES's performance and robustness. This includes fitness shaping (designed to render the algorithm invariant w.r.t.

order-preserving fitness transformations (Wierstra et al., 2008), Section 3.1), and adaptation sampling which is a novel technique for adjusting learning rates online (Section 3.2). We provide a novel formulation of NES for the whole class of multi-variate versions of distributions with rotation symmetries (Section 3.3). As special cases we summarize techniques for multivariate Gaussian search distributions, constituting the most common case (Section 3.4). Finally, in Section 3.5, we develop the breadth of the framework, motivating its usefulness and deriving a number of NES variants with different search distributions.

The ensuing experimental investigations show the competitiveness of the approach on a broad range of benchmarks (Section 5). The paper ends with a discussion on the effectiveness of the different techniques and types of distributions and an outlook towards future developments (Section 6).

2. Search Gradients

The core idea of Natural Evolution Strategies is to use *search gradients* (first introduced in Berny, 2000, 2001) to update the parameters of the search distribution. We define the search gradient as the sampled gradient of expected fitness. The search distribution can be taken to be a multinormal distribution, but could in principle be any distribution for which we can find derivatives of its log-density w.r.t. its parameters. For example, useful distributions include Gaussian mixture models and the Cauchy distribution with its heavy tail.

If we use θ to denote the parameters of density $\pi(\mathbf{z}\,|\,\theta)$ and $f(\mathbf{z})$ to denote the fitness function for samples $\mathbf{z}$, we can write the expected fitness under the search distribution as

$$J(\theta) = \mathbb{E}_\theta[f(\mathbf{z})] = \int f(\mathbf{z})\; \pi(\mathbf{z}\,|\,\theta)\; d\mathbf{z}. \tag{1}$$

The so-called ‘log-likelihood trick’ enables us to write

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int f(\mathbf{z})\; \pi(\mathbf{z}\,|\,\theta)\; d\mathbf{z} \\
&= \int f(\mathbf{z})\; \nabla_\theta \pi(\mathbf{z}\,|\,\theta)\; d\mathbf{z} \\
&= \int f(\mathbf{z})\; \nabla_\theta \pi(\mathbf{z}\,|\,\theta)\; \frac{\pi(\mathbf{z}\,|\,\theta)}{\pi(\mathbf{z}\,|\,\theta)}\; d\mathbf{z} \\
&= \int \Big[f(\mathbf{z})\; \nabla_\theta \log \pi(\mathbf{z}\,|\,\theta) \Big]\; \pi(\mathbf{z}\,|\,\theta)\; d\mathbf{z} \\
&= \mathbb{E}_\theta \left[f(\mathbf{z})\; \nabla_\theta \log \pi(\mathbf{z}\,|\,\theta) \right].
\end{aligned}$$

From this form we obtain the estimate of the search gradient from samples $\mathbf{z}_1 \ldots \mathbf{z}_\lambda$ as

$$\nabla_\theta J(\theta) \approx \frac{1}{\lambda} \sum_{k=1}^{\lambda} f(\mathbf{z}_k)\; \nabla_\theta \log \pi(\mathbf{z}_k\,|\,\theta), \tag{2}$$

where λ is the population size. This gradient on expected fitness provides a search direction in the space of search distributions. A straightforward gradient ascent scheme can thus iteratively update the search distribution

$$\theta \leftarrow \theta + \eta \nabla_\theta J(\theta),$$

where η is a learning rate parameter. Algorithm 1 provides the pseudocode for this very general approach to black-box optimization by using a search gradient on search distributions.

Algorithm 1: Canonical Search Gradient algorithm

```
input: $f$, $\theta_{init}$
repeat
    for $k = 1 \ldots \lambda$ do
        draw sample $\mathbf{z}_k \sim \pi(\cdot|\theta)$
        evaluate the fitness $f(\mathbf{z}_k)$
        calculate log-derivatives $\nabla_\theta \log \pi(\mathbf{z}_k|\theta)$
    end
    $\nabla_\theta J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_\theta \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
    $\theta \leftarrow \theta + \eta \cdot \nabla_\theta J$
until stopping criterion is met
```

Using the search gradient in this framework is similar to evolution strategies in that it iteratively generates the fitnesses of batches of vector-valued samples—the ES's so-called candidate solutions. It is different however, in that it represents this 'population' as a parameterized distribution, and in the fact that it uses a search gradient to update the parameters of this distribution, which is computed using the fitnesses.

2.1 Search Gradient for Gaussian Distributions

In the case of the 'default' d-dimensional multi-variate normal distribution, the parameters of the Gaussian are the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ (candidate solution center) and the covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ (mutation matrix). Let θ denote these parameters: $\theta = \langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle$. To sample efficiently from this distribution we need a square root of the covariance matrix, that is, a matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$ fulfilling $\mathbf{A}^\top \mathbf{A} = \boldsymbol{\Sigma}$. Then $\mathbf{z} = \boldsymbol{\mu} + \mathbf{A}^\top \mathbf{s}$ transforms a standard normal vector $\mathbf{s} \sim \mathcal{N}(0, \mathbb{I})$ into a sample $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Here, $\mathbb{I} = \text{diag}(1, \ldots, 1) \in \mathbb{R}^{d\times d}$ denotes the identity matrix. Let

$$\begin{aligned}
\pi(\mathbf{z} \,|\, \theta) &= \frac{1}{(\sqrt{2\pi})^d |\det(\mathbf{A})|} \cdot \exp\left(-\frac{1}{2} \left\| \mathbf{A}^{-1} \cdot (\mathbf{z} - \boldsymbol{\mu}) \right\|^2 \right) \\
&= \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \cdot \exp\left(-\frac{1}{2} (\mathbf{z} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{z} - \boldsymbol{\mu}) \right)
\end{aligned}$$

denote the density of the multinormal search distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

In order to calculate the derivatives of the log-likelihood with respect to individual elements of θ for this multinormal distribution, first note that

$$\log \pi(\mathbf{z}|\theta) = -\frac{d}{2} \log(2\pi) - \frac{1}{2} \log \det \boldsymbol{\Sigma} - \frac{1}{2} (\mathbf{z} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{z} - \boldsymbol{\mu}).$$

We will need its derivatives, that is, $\nabla_{\boldsymbol{\mu}} \log \pi (\mathbf{z}|\theta)$ and $\nabla_{\boldsymbol{\Sigma}} \log \pi (\mathbf{z}|\theta)$. The first is trivially

$$\nabla_{\boldsymbol{\mu}} \log \pi (\mathbf{z}|\theta) = \boldsymbol{\Sigma}^{-1} (\mathbf{z} - \boldsymbol{\mu}) , \tag{3}$$

while the latter is

$$\nabla_{\boldsymbol{\Sigma}} \log \pi (\mathbf{z}|\theta) = \frac{1}{2}\boldsymbol{\Sigma}^{-1} (\mathbf{z} - \boldsymbol{\mu}) (\mathbf{z} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} - \frac{1}{2}\boldsymbol{\Sigma}^{-1}. \tag{4}$$

Using these derivatives to calculate $\nabla_\theta J$, we can then update parameters $\theta = \langle \boldsymbol{\mu}, \boldsymbol{\Sigma} \rangle$ as $\theta \leftarrow \theta + \eta \nabla_\theta J$ using learning rate η. This produces a new center $\boldsymbol{\mu}$ for the search distribution, and simultaneously adapts its associated covariance matrix $\boldsymbol{\Sigma}$. To summarize, we provide the pseudocode for following the search gradient in the case of a multinormal search distribution in Algorithm 2.

Algorithm 2: Search Gradient algorithm: Multinormal distribution

```
input: $f$, $\boldsymbol{\mu}_{init}$,$\boldsymbol{\Sigma}_{init}$
repeat
    for $k = 1 \ldots \lambda$ do
        draw sample $\mathbf{z}_k \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
        evaluate the fitness $f(\mathbf{z}_k)$
        calculate log-derivatives:
          $\nabla_{\boldsymbol{\mu}} \log \pi (\mathbf{z}_k|\theta) = \boldsymbol{\Sigma}^{-1} (\mathbf{z}_k - \boldsymbol{\mu})$
          $\nabla_{\boldsymbol{\Sigma}} \log \pi (\mathbf{z}_k|\theta) = -\frac{1}{2}\boldsymbol{\Sigma}^{-1} + \frac{1}{2}\boldsymbol{\Sigma}^{-1} (\mathbf{z}_k - \boldsymbol{\mu}) (\mathbf{z}_k - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}$
    end
    $\nabla_{\boldsymbol{\mu}} J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_{\boldsymbol{\mu}} \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
    $\nabla_{\boldsymbol{\Sigma}} J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_{\boldsymbol{\Sigma}} \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
    $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta \cdot \nabla_{\boldsymbol{\mu}} J$
    $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma} + \eta \cdot \nabla_{\boldsymbol{\Sigma}} J$
until stopping criterion is met
```

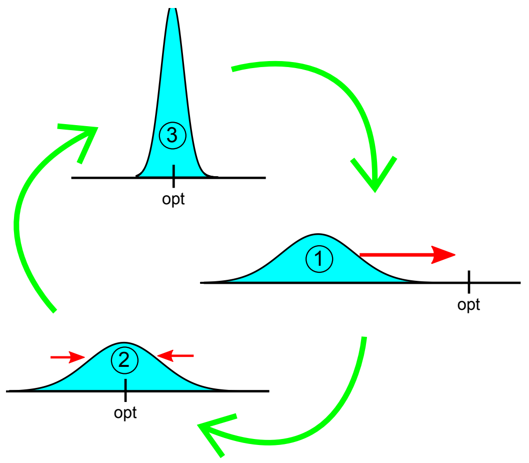
2.2 Limitations of Plain Search Gradients

As the attentive reader will have realized, there exists at least one major issue with applying the search gradient as-is in practice: It is impossible to *precisely locate* a (quadratic) optimum, even in the one-dimensional case. Let $d = 1$, $\theta = \langle \mu, \sigma \rangle$, and samples $z \sim \mathcal{N}(\mu, \sigma)$. Equations (3) and (4), the gradients on μ and σ, become

$$\begin{aligned} \nabla_\mu J &= \frac{z - \mu}{\sigma^2}, \\ \nabla_\sigma J &= \frac{(z - \mu)^2 - \sigma^2}{\sigma^3}, \end{aligned}$$

and the updates, assuming simple hill-climbing (i.e., a population size $\lambda = 1$) read:

$$\begin{aligned} \mu &\leftarrow \mu + \eta \frac{z - \mu}{\sigma^2}, \\ \sigma &\leftarrow \sigma + \eta \frac{(z - \mu)^2 - \sigma^2}{\sigma^3}. \end{aligned}$$



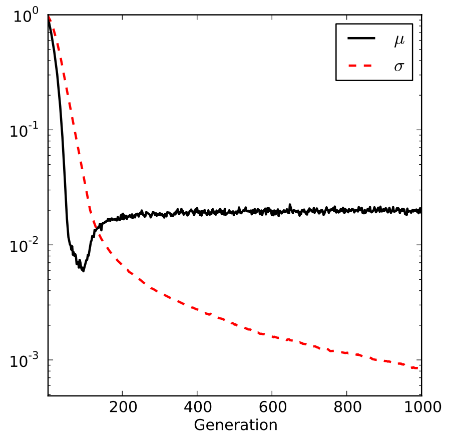


Figure 1: **Left:** Schematic illustration of how the search distribution adapts in the one-dimensional case: from (1) to (2), μ is adjusted to make the distribution cover the optimum. From (2) to (3), σ is reduced to allow for a precise localization of the optimum. The step from (3) to (1) then is the problematic case, where a small σ induces a largely overshooting update, making the search start over again. **Right:** Progression of μ (black) and σ (red, dashed) when following the search gradient towards minimizing $f(\mathbf{z}) = \mathbf{z}^2$, executing Algorithm 2. Plotted are median values over 1000 runs, with a small learning rate $\eta = 0.01$ and $\lambda = 10$, both of which mitigate the instability somewhat, but still show the failure to precisely locate the optimum (for which both μ and σ need to approach 0).

For any objective function f that requires locating an (approximately) quadratic optimum with some degree of precision (e.g., $f(\mathbf{z}) = \mathbf{z}^2$), σ must decrease, which in turn increases the variance of the updates, as $\Delta\mu \propto \frac{1}{\sigma}$ and $\Delta\sigma \propto \frac{1}{\sigma}$ for a typical sample z. In fact, the updates become increasingly unstable, the smaller σ becomes, an effect which a reduced learning rate or an increased population size can only delay but not avoid. Figure 1 illustrates this effect. Conversely, whenever $\sigma \gg 1$ is large, the magnitude of a typical update is severely reduced.

Clearly, this update is not at all *scale-invariant*: Starting with $\sigma \gg 1$ makes all updates minuscule, whereas starting with $\sigma \ll 1$ makes the first update huge and therefore unstable.

This effect need not occur in gradient-based search in general. Here it is rather a consequence of the special situation that the gradient controls both position and variance of a distribution over the same search space dimension. Note that this situation is generic for all translation and scale-invariant families of search distributions. We conjecture that this limitation constitutes one of the main reasons why search gradients have not been developed before: typically, with a naive parameterization, the plain search gradient's performance can be both unstable and unsatisfying; however, the natural gradient extension (introduced in Section 2.3) tackles these issues, and renders search gradients into a viable optimization method by making updates invariant with respect to the particular parameterization used.

2.3 Using the Natural Gradient

Instead of using the plain stochastic gradient for updates, NES follows the *natural gradient*. The natural gradient was first introduced into the field of machine learning by Amari in 1998, and has been shown to possess numerous advantages over the plain gradient (Amari, 1998; Amari and Douglas, 1998). Natural gradients help mitigate the slow convergence of plain gradient ascent in optimization landscapes with ridges and plateaus.

The plain gradient ∇J simply follows the steepest ascent in the space of the actual parameters θ of the distribution. This means that for a given small step-size ε, following it will yield a new distribution with parameters chosen from the hypersphere of radius ϵ and center θ that maximizes J. In other words, the Euclidean distance in parameter space is used to measure the distance between subsequent distributions. Clearly, this makes the update dependent on the particular parameterization of the distribution, therefore a change of parameterization leads to different gradients and different updates. See also Figure 2 for an illustration of how this effectively renormalizes updates w.r.t. uncertainty.

The key idea of the natural gradient is to remove this dependence on the parameterization by relying on a more 'natural' measure of distance $D(\theta'||\theta)$ between probability distributions $\pi\left(\mathbf{z}|\theta\right)$ and $\pi\left(\mathbf{z}|\theta'\right)$. One such natural distance measure between two probability distributions is the Kullback-Leibler divergence (Kullback and Leibler, 1951). The natural gradient can then be formalized as the solution to the constrained optimization problem

$$\begin{aligned} \max_{\delta\theta} J\left(\theta + \delta\theta\right) &\approx J\left(\theta\right) + \delta\theta^\top \nabla_\theta J, \\ s.t.\ D\left(\theta + \delta\theta || \theta\right) &= \varepsilon, \end{aligned} \tag{5}$$

where $J\left(\theta\right)$ is the expected fitness of Equation (1), and ε is a small increment size. Now, we have for $\lim \delta\theta \rightarrow 0$,

$$D\left(\theta + \delta\theta || \theta\right) = \frac{1}{2}\delta\theta^\top \mathbf{F}\left(\theta\right)\delta\theta,$$

where

$$\begin{aligned} \mathbf{F} &= \int \pi\left(\mathbf{z}|\theta\right) \nabla_\theta \log \pi\left(\mathbf{z}|\theta\right) \nabla_\theta \log \pi\left(\mathbf{z}|\theta\right)^\top d\mathbf{z} \\ &= \mathbb{E}\left[\nabla_\theta \log \pi\left(\mathbf{z}|\theta\right) \nabla_\theta \log \pi\left(\mathbf{z}|\theta\right)^\top\right] \end{aligned}$$

is the *Fisher information matrix* of the given parametric family of search distributions. The solution to the constrained optimization problem in Equation (5) can be found using a Lagrangian multiplier (Peters, 2007), yielding the necessary condition

$$\mathbf{F}\delta\theta = \beta\nabla_\theta J,$$

for some constant $\beta > 0$. The direction of the natural gradient $\widetilde{\nabla}_\theta J$ is given by $\delta\theta$ thus defined. If $\mathbf{F}$ is invertible,[1] the natural gradient amounts to

$$\widetilde{\nabla}_\theta J = \mathbf{F}^{-1}\nabla_\theta J(\theta).$$

1. Care has to be taken because the Fisher matrix estimate may not be (numerically) invertible even if the exact Fisher matrix is.

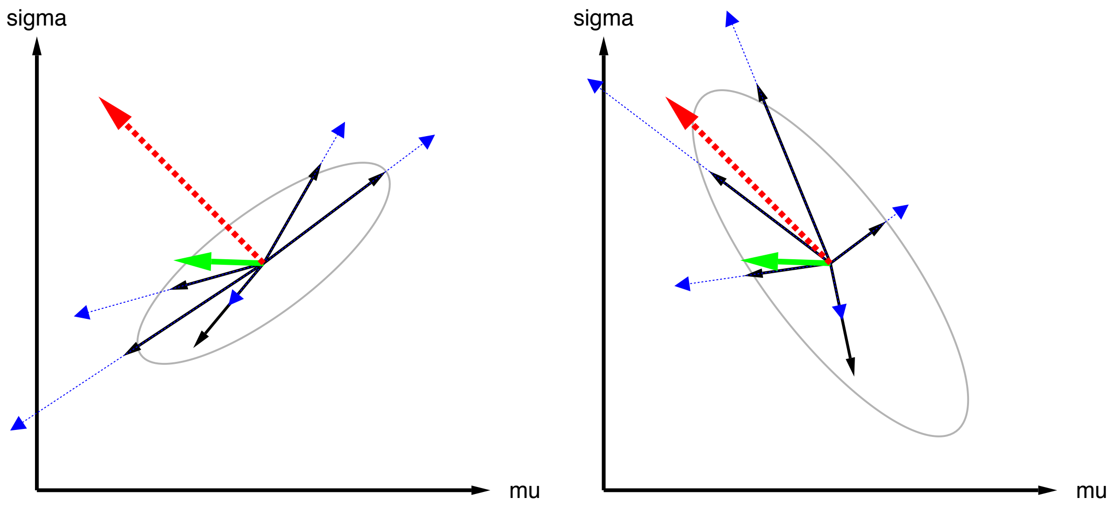


Figure 2: Illustration of plain versus natural gradient in parameter space. Consider two parameters, for example, $\theta = (\mu, \sigma)$, of the search distribution. In the plot on the left, the solid (black) arrows indicate the gradient samples $\nabla_\theta \log \pi(\mathbf{z} \mid \theta)$, while the dotted (blue) arrows correspond to $f(\mathbf{z}) \cdot \nabla_\theta \log \pi(\mathbf{z} \mid \theta)$, that is, the same gradient estimates, but scaled with fitness. Combining these, the bold (green) arrow indicates the (sampled) fitness gradient $\nabla_\theta J$, while the bold dashed (red) arrow indicates the corresponding natural gradient $\tilde{\nabla}_\theta J$.

Being random variables with expectation zero, the distribution of the black arrows is governed by their covariance, indicated by the gray ellipse. Notice that this covariance is a quantity in *parameter space* (where the θ reside), which is not to be confused with the covariance of the distribution in the *search space* (where the samples $\mathbf{z}$ reside).

In contrast, solid (black) arrows on the right represent $\tilde{\nabla}_\theta \log \pi(\mathbf{z} \mid \theta)$, and dotted (blue) arrows indicate the *natural* gradient samples $f(\mathbf{z}) \cdot \tilde{\nabla}_\theta \log \pi(\mathbf{z} \mid \theta)$, resulting in the natural gradient (dashed red).

The covariance of the solid arrows on the right hand side turns out to be the inverse of the covariance of the solid arrows on the left. This has the effect that when computing the natural gradient, directions with high variance (uncertainty) are penalized and thus shrunken, while components with low variance (high certainty) are boosted, since these components of the gradient samples deserve more trust. This makes the (dashed red) natural gradient a much more trustworthy update direction than the (green) plain gradient.

The Fisher matrix can be estimated from samples, reusing the log-derivatives $\nabla_\theta \log \pi(\mathbf{z}|\theta)$ that we already computed for the gradient $\nabla_\theta J$. Then, updating the parameters following the natural gradient instead of the steepest gradient leads us to the general formulation of NES, as shown in Algorithm 3.

Algorithm 3: Canonical Natural Evolution Strategies

```
input: $f$, $\theta_{init}$
repeat
    for $k = 1 \ldots \lambda$ do
        draw sample $\mathbf{z}_k \sim \pi(\cdot|\theta)$
        evaluate the fitness $f(\mathbf{z}_k)$
        calculate log-derivatives $\nabla_\theta \log \pi(\mathbf{z}_k|\theta)$
    end
    $\nabla_\theta J \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_\theta \log \pi(\mathbf{z}_k|\theta) \cdot f(\mathbf{z}_k)$
    $\mathbf{F} \leftarrow \frac{1}{\lambda} \sum_{k=1}^{\lambda} \nabla_\theta \log \pi\left(\mathbf{z}_k|\theta\right) \nabla_\theta \log \pi\left(\mathbf{z}_k|\theta\right)^\top$
    $\theta \leftarrow \theta + \eta \cdot \mathbf{F}^{-1} \nabla_\theta J$
until stopping criterion is met
```

3. Performance and Robustness Techniques

In the following we will present and introduce crucial heuristics to improves NES's performance and robustness. Fitness shaping (Wierstra et al., 2008) is designed to make the algorithm invariant w.r.t. arbitrary yet order-preserving fitness transformations (Section 3.1). Adaptation sampling, a novel technique for adjusting learning rates online, is introduced in Section 3.2.

In sections 3.3 and 3.4 we describe two crucial techniques to enhance performance of the NES algorithm as applied to multinormal distributions: Exponential parameterization guarantees that the covariance matrix stays positive-definite, and second, a novel method for changing the coordinate system into a "natural" one is laid out, which makes the algorithm computationally efficient.

3.1 Fitness Shaping

NES uses rank-based fitness shaping in order to render the algorithm *invariant* under monotonically increasing (i.e., rank preserving) transformations of the fitness function. For this purpose, the fitness of the population is transformed into a set of utility values $u_1 \geq \cdots \geq u_\lambda$. Let $\mathbf{z}_i$ denote the i^{th} best individual (the i^{th} individual in the population, sorted by fitness, such that $\mathbf{z}_1$ is the best and $\mathbf{z}_\lambda$ the worst individual). Replacing fitness with utility, the gradient estimate of Equation (2) becomes, with slight abuse of notation,

$$\nabla_\theta J(\theta) = \sum_{k=1}^{\lambda} u_k \; \nabla_\theta \log \pi(\mathbf{z}_k \,|\, \theta).$$

The choice of utility function can in fact be seen as a free parameter of the algorithm. Throughout this paper we will use the following

$$u_k = \frac{\max\left(0, \log(\frac{\lambda}{2}+1) - \log(k)\right)}{\sum_{j=1}^{\lambda} \max\left(0, \log(\frac{\lambda}{2}+1) - \log(j)\right)} - \frac{1}{\lambda},$$

which is directly related to the one employed by CMA-ES (Hansen and Ostermeier, 2001), for ease of comparison. In our experience, however, this choice has not been crucial to performance, as long as it is monotonous and based on ranks instead of raw fitness (e.g., a function which simply increases linearly with rank).

3.2 Adaptation Sampling

To reduce the burden of determining appropriate hyper-parameters such as the learning rate, we develop a new online adaptation or meta-learning technique (Schaul and Schmidhuber, 2010), called *adaptation sampling*, that can automatically adapt the settings.

We model this situation as follows: Let π_θ be a distribution with hyper-parameter θ and $\psi(\mathbf{z})$ a quality measure for each sample $\mathbf{z} \sim \pi_\theta$. Our goal is to adapt θ such as to maximize the quality ψ. A straightforward method to achieve this, henceforth dubbed *adaptation sampling*, is to evaluate the quality of the samples $\mathbf{z}'$ drawn from $\pi_{\theta'}$, where $\theta' \neq \theta$ is a slight variation of θ, and then perform hill-climbing: Continue with the new θ' if the quality of its samples is significantly better (according, for example, to a Mann-Whitney U-test), and revert to θ otherwise. Note that this proceeding is similar to the NES algorithm itself, but applied at a meta-level to algorithm parameters instead of the search distribution. The goal of this adaptation is to maximize the *pace* of progress over time, which is slightly different from maximizing the fitness function itself.

Virtual adaptation sampling is a lightweight alternative to adaptation sampling that is particularly useful whenever evaluating ψ is expensive :

- do importance sampling on the existing samples $\mathbf{z}_i$, according to $\pi_{\theta'}$:

  $$w'_i = \frac{\pi(\mathbf{z}|\theta')}{\pi(\mathbf{z}|\theta)}$$

  (this is always well-defined, because $\mathbf{z} \sim \pi_\theta \Rightarrow \pi(\mathbf{z}|\theta) > 0$).

- compare $\{\psi(\mathbf{z}_i)\}$ with weights $\{w_i = 1, \forall i\}$ and $\{\psi' = \psi(\mathbf{z}_i), \forall i\}$ with weights $\{w'_i\}$, using a weighted generalization of the Mann-Whitney test.

Beyond determining whether θ or θ' is better, choosing a non-trivial confidence level ρ allows us to avoid parameter drift, as θ is only updated if the improvement is significant enough. There is one caveat, however: the rate of parameter change needs to be adjusted such that the two resulting distributions are not too similar (otherwise the difference won't be statistically significant), but also not too different, (otherwise the weights w' will be too small and again the test will be inconclusive). If, however, we explicitly desire large adaptation steps on θ, we have the possibility of interpolating between adaptation sampling and virtual adaptation sampling by drawing a few new samples from the distribution $\pi_{\theta'}$ (each assigned weight 1), where it is overlapping least with π_θ.

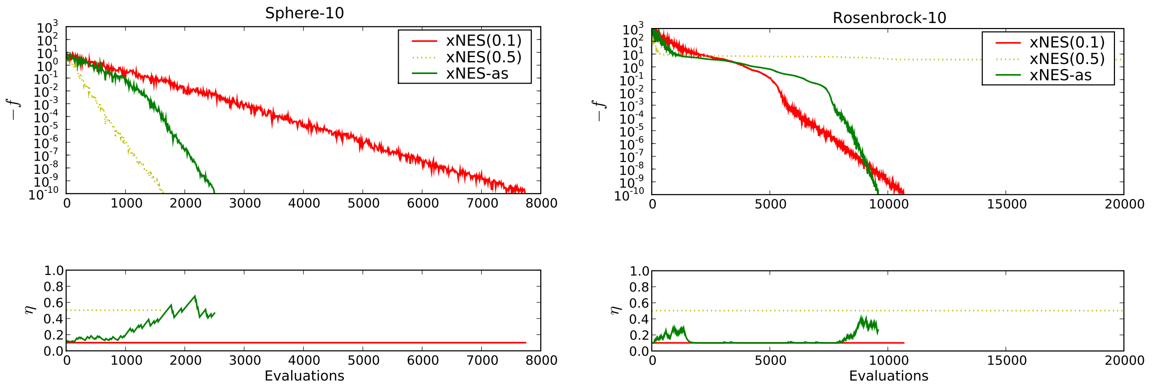


Figure 3: Illustration of the effect of adaptation sampling. We show the increase in fitness during a NES run (above) and the corresponding learning rates (below) on two setups: 10-dimensional sphere function (left), and 10-dimensional Rosenbrock function (right). Plotted are three variants of xNES (Algorithm 5): fixed default learning rate of $\eta = 0.1$ (dashed, red) fixed large learning rate of $\eta = 0.5$ (dotted, yellow), and an adaptive learning rate starting at $\eta = 0.1$ (green). We see that for the (simple) Sphere function, it is advantageous to use a large learning rate, and adaptation sampling automatically finds that one. However, using the overly greedy updates of a large learning rate fails on harder problems (right). Here adaptation sampling really shines: it boosts the learning rate in the initial phase (entering the Rosenbrock valley), then quickly reduces it while the search needs to carefully navigate the bottom of the valley, and boosts it again at the end when it has located the optimum and merely needs to zoom in precisely.

For NES algorithms, the most important parameter to be adapted by adaptation sampling is the learning rate η, starting with a conservative guess. This is because half-way into the search, after a local attractor has been singled out, it may well pay off to increase the learning rate in order to more quickly converge to it.

In order to produce variations η' which can be judged using the above-mentioned U-test, we propose a procedure similar in spirit to Rprop-updates (Riedmiller and Braun, 1993; Igel and Hüsken, 2003), where the learning rates are either increased or decreased by a multiplicative constant whenever there is evidence that such a change will lead to better samples.

More concretely, when using adaptation sampling for NES we test for an improvement with the hypothetical distribution θ' generated with $\eta' = 1.5\eta$. Each time the statistical test is successful with a confidence of at least $\rho = \frac{1}{2} - \frac{1}{3(d+1)}$ (this value was determined empirically) we increase the learning rate by a factor of $1+c'$, up to at most $\eta = 1$. Otherwise we bring it closer to its initial value: $\eta \leftarrow (1-c')\eta + c'\eta_{init}$. We use $c' = \frac{1}{10}$ (again, an empirically robust choice). The final procedure is summarized in algorithm 4. We append the ending "-as" to denote algorithm variants using adaptation sampling.

Figure 3 illustrates the effect of the virtual adaptation sampling strategy on two different 10-dimensional unimodal benchmark functions, the Sphere function f_1 and the Rosenbrock function f_8 (see Section 5.2 for details). We find that, indeed, adaptation sampling boosts the learning rates to the appropriate high values when quick progress can be made (in the presence of an approximately quadratic optimum), but keeps them at carefully low values otherwise.

Algorithm 4: Adaptation sampling

```
input  : $\eta_{\sigma,t}$,$\eta_{\sigma,\mathrm{init}}$, $\theta_t$, $\theta_{t-1}$, $\{(\mathbf{z}_k, f(\mathbf{z}_k))\}$, $c'$, $\rho$
output: $\eta_{\sigma,t+1}$
compute hypothetical $\theta'$, given $\theta_{t-1}$ and using $3/2\eta_{\sigma,t}$
for $k = 1 \dots \lambda$ do
    $w'_k = \dfrac{\pi(\mathbf{z}_k|\theta')}{\pi(\mathbf{z}_k|\theta)}$
end
$S \leftarrow \{\mathrm{rank}(\mathbf{z}_k)\}$
$S' \leftarrow \{w'_k \cdot \mathrm{rank}(\mathbf{z}_k)\}$
if weighted-Mann-Whitney$(S, S') < \rho$ then
    return $(1 - c') \cdot \eta_\sigma + c' \cdot \eta_{\sigma,\mathrm{init}}$
else
    return $\min((1 + c') \cdot \eta_\sigma, 1)$
end
```

3.3 Rotationally Symmetric Distributions

In this section we derive the computation of the natural gradient for a rather general class of distributions, namely multi-variate distributions arising from linear transformations of rotationally symmetric distributions. For many important cases, such as multi-variate Gaussians, this allows us to obtain the Fisher matrix in closed form. The resulting strategy updates are computationally efficient.

For a sample $\mathbf{z} \in \mathbb{R}^d$ let $r = \|\mathbf{z}\|$ denote its radial component. Let $Q_{\boldsymbol{\tau}}(\mathbf{z})$ be a family of rotationally symmetric distributions with parameter vector $\boldsymbol{\tau}$. From this invariance property we deduce that the density can be written as $Q_{\boldsymbol{\tau}}(\mathbf{z}) = q_{\boldsymbol{\tau}}(r^2)$ for some family of functions $q_{\boldsymbol{\tau}} : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$. In the following we consider classes of search distributions with densities

$$\begin{aligned} \pi\big(\mathbf{z} \,\big|\, \boldsymbol{\mu}, \mathbf{A}, \boldsymbol{\tau}\big) &= \frac{1}{|\det(\mathbf{A})|} \cdot q_{\boldsymbol{\tau}}\big(\| \left(\mathbf{A}^{-1}\right)^\top (\mathbf{z} - \boldsymbol{\mu})\|^2\big) \\ &= \frac{1}{\sqrt{\det(\mathbf{A}^\top \mathbf{A})}} \cdot q_{\boldsymbol{\tau}}\big((\mathbf{z} - \boldsymbol{\mu})^\top (\mathbf{A}^\top \mathbf{A})^{-1} (\mathbf{z} - \boldsymbol{\mu})\big) \end{aligned} \tag{6}$$

with additional transformation parameters $\boldsymbol{\mu} \in \mathbb{R}^d$ and invertible $\mathbf{A} \in \mathbb{R}^{d \times d}$. If needed, $\mathbf{A}$ can be restricted to any continuous sub-group of the invertible matrices like, for example, diagonal matrices. The function $q_{\boldsymbol{\tau}}$ is the accordingly transformed density of the random variable $\mathbf{s} = \left(\mathbf{A}^{-1}\right)^\top (\mathbf{z} - \boldsymbol{\mu})$. This setting is rather general. It covers many important families of distributions and their multi-variate forms, most prominently multi-variate Gaussians. In addition, properties of the radial distribution such as its tail (controlling whether large mutations are common or rare) can be controlled with the parameter $\boldsymbol{\tau}$.

3.3.1 Local "Natural" Coordinates

In principle the computation of the natural gradient, involving the computation of gradient and Fisher matrix, is straight-forward. However, the parameters $\boldsymbol{\mu}$ and $\mathbf{A}$ have d and $d(d+1)/2$ dimensions, which makes a total of $d(d+3)/2 \in \mathcal{O}(d^2)$. Let d' denote the dimensionality of the radial parameters $\boldsymbol{\tau}$, and for simplicity we assume that d' is fixed and does not grow with the dimensionality of the search space. Then the Fisher matrix has $\mathcal{O}(d^4)$ entries, and its inversion costs $\mathcal{O}(d^6)$ operations. It turns out that we can do much better. The following derivation is a generalization of the proceeding found in Glasmachers et al. (2010b).

The above encoding by means of transformations of a rotation invariant normal form of the distribution hints at the introduction of a canonical local coordinate system in which the normal form becomes the current search distribution. It turns out from Equation (6) that the dependency of the distribution on $\mathbf{A}$ is only in terms of the symmetric positive definite matrix $\mathbf{A}^\top\mathbf{A}$. In the Gaussian case this matrix coincides with the covariance matrix. Instead of performing natural gradient steps on the manifolds of invertible or positive definite symmetric matrices we introduce a one-to-one encoding with a vector space representation. The matrix exponential, restricted to the vector space of symmetric matrices, is a global map for the manifold of symmetric positive definite matrices. Thus, we introduce "exponential" local coordinates $(\boldsymbol{\delta}, \mathbf{M}) \mapsto (\boldsymbol{\mu}_{\text{new}}, \mathbf{A}_{\text{new}}) = \left(\boldsymbol{\mu} + \mathbf{A}^\top\boldsymbol{\delta}, \mathbf{A}\exp\left(\frac{1}{2}\mathbf{M}\right)\right)$. These coordinates are local in the sense that the current search distribution is encoded by $(\boldsymbol{\delta}, \mathbf{M}) = (0, 0)$. It turns out that in these coordinates the Fisher matrix takes the rather simple form

$$\mathbf{F} = \begin{pmatrix} \mathbb{I} & v \\ v^\top & c \end{pmatrix} \quad \text{with} \quad v = \frac{\partial^2 \log \pi(\mathbf{z})}{\partial(\boldsymbol{\delta}, \mathbf{M})\partial\boldsymbol{\tau}} \in \mathbb{R}^{(m-d')\times d'} \quad \text{and} \quad c = \frac{\partial^2 \log \pi(\mathbf{z})}{\partial\boldsymbol{\tau}^2} \in \mathbb{R}^{d'\times d'}. \tag{7}$$

Note that for distributions without radial parameters $\boldsymbol{\tau}$, such as Gaussians, we obtain $\mathbf{F} = \mathbb{I}$. Thus, in local coordinates the otherwise computationally intensive operations of computing and inverting the Fisher matrix are trivial, and the vanilla gradient coincides with the natural gradient. For this reason we call the above local coordinates also *natural* exponential coordinates. For non-trivial parameters $\boldsymbol{\tau}$ we use the Woodbury identity to compute the inverse of the Fisher matrix as

$$\mathbf{F}^{-1} = \begin{pmatrix} \mathbb{I} & v \\ v^\top & c \end{pmatrix}^{-1} = \begin{pmatrix} \mathbb{I} + Hvv^\top & -Hv \\ -Hv^\top & H \end{pmatrix}$$

with $H = (c - v^\top v)^{-1}$, and exploiting $H^\top = H$. It remains to compute the gradient. We obtain the three components of derivatives of log-probabilities

$$\nabla_{\boldsymbol{\delta},\mathbf{M},\boldsymbol{\tau}}\big|_{\boldsymbol{\delta}=0,\mathbf{M}=0} \log \pi\left(\mathbf{z} \,|\, \boldsymbol{\mu}, \mathbf{A}, \boldsymbol{\tau}, \boldsymbol{\delta}, \mathbf{M}\right) = g = \left(g_{\boldsymbol{\delta}}, g_{\mathbf{M}}, g_{\boldsymbol{\tau}}\right),$$

$$\begin{aligned}
g_{\boldsymbol{\delta}} =& - 2 \cdot \frac{q'_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)}{q_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)} \cdot \mathbf{s}, \\
g_{\mathbf{M}} =& - \frac{1}{2}\mathbb{I} - \frac{q'_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)}{q_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)} \cdot \mathbf{s}\mathbf{s}^\top, \\
g_{\boldsymbol{\tau}} =& \frac{1}{q_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2)} \cdot \nabla_{\boldsymbol{\tau}}\, q_{\boldsymbol{\tau}}(\|\mathbf{s}\|^2),
\end{aligned}$$

where $q'_{\boldsymbol{\tau}} = \frac{\partial}{\partial(r^2)} q_{\boldsymbol{\tau}}$ denotes the derivative of $q_{\boldsymbol{\tau}}$ with respect to r^2, and $\nabla_{\boldsymbol{\tau}}\, q_{\boldsymbol{\tau}}$ denotes the gradient w.r.t. $\boldsymbol{\tau}$. The sample-wise natural gradient becomes

$$\mathbf{F}^{-1} \cdot g = \begin{pmatrix} (g_{\boldsymbol{\delta}}, g_{\mathbf{M}}) - Hv(v^\top (g_{\boldsymbol{\delta}}, g_{\mathbf{M}}) - g_{\boldsymbol{\tau}}) \\ H(v^\top (g_{\boldsymbol{\delta}}, g_{\mathbf{M}}) - g_{\boldsymbol{\tau}}) \end{pmatrix},$$

which can be computed efficiently in only $\mathcal{O}(d^2)$ operations (assuming fixed d'). This is in contrast to $\mathcal{O}(d^6)$ operations required for a naïve inversion of the full Fisher matrix.

3.3.2 Sampling from Radial Distributions

In order to use this class of distributions for search we need to be able to draw samples from it. The central idea is to first draw a sample $\mathbf{s}$ from the 'standard' density $\pi(\mathbf{s} \,|\, \boldsymbol{\mu} = 0, \mathbf{A} = \mathbb{I}, \boldsymbol{\tau})$, which is then transformed into the sample $\mathbf{z} = \mathbf{A}^\top \mathbf{s} + \boldsymbol{\mu}$, corresponding to the density $\pi(\mathbf{z} \,|\, \mathbf{A}, \boldsymbol{\mu}, \boldsymbol{\tau})$. In general, sampling $\mathbf{s}$ can be decomposed into sampling the (squared) radius component $r^2 = \|\mathbf{z}\|^2$ and a unit vector $\mathbf{v} \in \mathbb{R}^d$, $\|\mathbf{v}\| = 1$. The squared radius has the density

$$\tilde{q}_{\boldsymbol{\tau}}(r^2) = \int\limits_{\|\mathbf{z}\|^2 = r^2} Q_{\boldsymbol{\tau}}(\mathbf{z})\, d\mathbf{z} = \frac{2\pi^{d/2}}{\Gamma(d/2)} \cdot (r^2)^{(d-1)/2} \cdot q_{\boldsymbol{\tau}}(r^2),$$

where $\Gamma(\cdot)$ denotes the gamma function. In the following we assume that we have an efficient method of drawing samples from this one-dimensional density. Besides the radius we draw a unit vector $\mathbf{u} \in \mathbb{R}^d$ uniformly at random, for example by normalizing a standard normally distributed vector. Then $\mathbf{s} = r \cdot \mathbf{u}$ is effectively sampled from $\pi(\mathbf{s} \,|\, \boldsymbol{\mu} = 0, \mathbf{A} = \mathbb{I}, \boldsymbol{\tau})$, and the composition $\mathbf{z} = r\mathbf{A}^\top \mathbf{u} + \boldsymbol{\mu}$ follows the density $\pi(\mathbf{z} \,|\, \boldsymbol{\mu}, \mathbf{A}, \boldsymbol{\tau})$. In many special cases, however, there are more efficient ways of sampling $\mathbf{s} = r \cdot \mathbf{u}$ directly.

3.4 Techniques for Multinormal Distributions

Multi-variate Gaussians are the most prominent class of search distributions for evolution strategies. An advantageous property of Gaussians is that the Fisher information matrix is known analytically. A large share of the previous work on NES has dealt with the development of efficient techniques for this important special case.

Here we deal with this prominent case in the above introduced framework. The natural gradient is

$$\nabla_{\boldsymbol{\delta}} J = \sum_{k=1}^{\lambda} f(\mathbf{z}_k) \cdot \mathbf{s}_k,$$
$$\nabla_{\mathbf{M}} J = \sum_{k=1}^{\lambda} f(\mathbf{z}_k) \cdot (\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}),$$

where $\mathbf{s}_k$ is the k-th best sample in the batch in local coordinates, and $\mathbf{z}_k$ is the same sample in task coordinates. The resulting algorithm, outlined in Algorithm 5 is known as exponential NES (xNES). It is demonstrated in the algorithm how the covariance factor $\mathbf{A}$ can be decomposed into a scalar step size $\sigma > 0$ and a normalized covariance factor $\mathbf{B}$ fulfilling $\det(\mathbf{B}) = 1$. This decoupling of shape ($\mathbf{B}$) from scale (σ) information allows for adaptation of the two orthogonal components with independent learning rates. All NES variants for multi-variate Gaussians have a complexity of $\mathcal{O}(d^3)$ computations per covariance matrix update. This complexity can be reduced to $\mathcal{O}(d^2)$ by computing the updates in local non-exponential coordinates.

It was shown that the NES principle is also compatible with elitist selection (Glasmachers et al., 2010a), resulting in the natural gradient hillclimber (1+1)-xNES. We refer to the papers by Sun et al. (2009a,b) and Glasmachers et al. (2010b,a) for further technical details on these algorithms.

Algorithm 5: Exponential Natural Evolution Strategies (xNES) (multinormal case)

```
input: $f$, $\boldsymbol{\mu}_{init}$, $\boldsymbol{\Sigma}_{init} = \mathbf{A}^\top \mathbf{A}$

initialize  $\sigma \leftarrow \sqrt[d]{|\det(\mathbf{A})|}$
            $\mathbf{B} \leftarrow \mathbf{A}/\sigma$
repeat
    for $k = 1 \ldots \lambda$ do
        draw sample $\mathbf{s}_k \sim \mathcal{N}(0, \mathbb{I})$
        $\mathbf{z}_k \leftarrow \boldsymbol{\mu} + \sigma \mathbf{B}^\top \mathbf{s}_k$
        evaluate the fitness $f(\mathbf{z}_k)$
    end
    sort $\{(\mathbf{s}_k, \mathbf{z}_k)\}$ with respect to $f(\mathbf{z}_k)$ and compute utilities $u_k$

    compute gradients  $\nabla_{\boldsymbol{\delta}} J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot \mathbf{s}_k$    $\nabla_{\mathbf{M}} J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot (\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I})$
                       $\nabla_{\sigma} J \leftarrow \mathrm{tr}(\nabla_{\mathbf{M}} J)/d$    $\nabla_{\mathbf{B}} J \leftarrow \nabla_{\mathbf{M}} J - \nabla_{\sigma} J \cdot \mathbb{I}$

                       $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta_{\boldsymbol{\delta}} \cdot \sigma \mathbf{B} \cdot \nabla_{\boldsymbol{\delta}} J$
    update parameters  $\sigma \leftarrow \sigma \cdot \exp(\eta_{\sigma}/2 \cdot \nabla_{\sigma} J)$
                       $\mathbf{B} \leftarrow \mathbf{B} \cdot \exp(\eta_{\mathbf{B}}/2 \cdot \nabla_{\mathbf{B}} J)$
until *stopping criterion is met*
```

3.5 Beyond Multinormal Distributions

The large share of literature on the multinormal case does not properly reflect the generality of the NES algorithm. It was shown by Schaul et al. (2011) that certain classes of problems can profit from tailored search distributions.

One simple yet important variant, inspired by Ros and Hansen (2008), is to use separable search distributions to improve the update complexity from cubic or quadratic to linear. This is compatible with the exponential parameterization of xNES. Cheap updates are a prerequisite for search in high-dimensional spaces, for example, for training recurrent neural networks. The resulting algorithm is called separable NES (SNES). Within our framework of linearly transformed rotationally symmetric distributions we obtain this case by restricting $\mathbf{A}$ to the group of (invertible) diagonal transformation matrices.

Another direction is the extension of NES to multivariate Cauchy distributions. These heavy-tailed distributions have undefined expectation and infinite variance. In this case the natural gradient is not defined. Still, the NES principle can be generalized by means of invariance properties. This is because three seemingly unrelated properties coincide for local natural coordinates. Let us assume the absence of radial parameters $\boldsymbol{\tau}$, then (i) due to Equation (7) the Fisher matrix is the identity, (ii) plain and natural gradient coincide, and (iii) by construction the current search distribution is invariant under orthogonal transformations. We obtain the following alternative characterization of NES: The NES algorithm performs its gradient updates based on a local coordinate system in which the orthogonal group (induced by the standard inner product) leaves the search distribution invariant.

This characterization for multi-variate distributions turns out to be robust. The argument stays valid while we iteratively grow the distribution's tail. In the limit of infinite variance the natural gradient interpretation breaks down, while the characterization by invariance is unaffected.

NES with heavy-tailed distributions are most useful as (1+1) hillclimbers: Assume a lucky exploratory sample from the distribution's heavy tail has managed to escape a bad local optimum. In a population-based algorithm this effect would be "corrupted" by weighted averaging and the better attractor may be missed, while a hillclimber can jump to the new position. It has been demonstrated in Schaul et al. (2011) that heavy-tailed distributions can improve the performance of NES on highly multi-modal problems.

4. Connection to CMA-ES

It turns out that xNES is closely related to the seminal Covariance Matrix Adaptation Evolution Strategy (CMA-ES; Hansen and Ostermeier, 2001) algorithm. It has been noticed by Glasmachers et al. (2010b) that in first order Taylor approximation the exponential xNES update coincides with the so-called rank-μ update of CMA-ES. Akimoto et al. (2010) have show rigorously that CMA-ES is in fact following an approximate natural gradient, and thus, arguably, can be seen as a member of the NES family. In the remainder of this section, we will point out similarities and differences between it and xNES.

The relation between CMA-ES and xNES is clearest when considering the CMA-ES variant with rank-μ update (in the terminology of this study, rank-λ-update), since this one does not feature evolution paths. Both xNES and CMA-ES parameterize the search distribution with three functionally different parameters for mean, scale, and shape of the

Algorithm 6: Separable NES (SNES)

```
input: $f$, $\boldsymbol{\mu}_{init}$, $\boldsymbol{\sigma}_{init}$
repeat
    for $k = 1 \ldots \lambda$ do
        draw sample $\mathbf{s}_k \sim \mathcal{N}(0, \mathbb{I})$
        $\mathbf{z}_k \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma}\mathbf{s}_k$
        evaluate the fitness $f(\mathbf{z}_k)$
    end
    sort $\{(\mathbf{s}_k, \mathbf{z}_k)\}$ with respect to $f(\mathbf{z}_k)$ and compute utilities $u_k$
    compute gradients   $\nabla_{\boldsymbol{\mu}} J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot \mathbf{s}_k$
                        $\nabla_{\boldsymbol{\sigma}} J \leftarrow \sum_{k=1}^{\lambda} u_k \cdot (\mathbf{s}_k^2 - 1)$
    update parameters   $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta_{\boldsymbol{\mu}} \cdot \boldsymbol{\sigma} \cdot \nabla_{\boldsymbol{\mu}} J$
                        $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} \cdot \exp(\eta_{\boldsymbol{\sigma}}/2 \cdot \nabla_{\boldsymbol{\sigma}} J)$
until stopping criterion is met;
```

distribution. xNES uses the parameters $\boldsymbol{\mu}$, σ, and $\mathbf{B}$, while the covariance matrix is represented as $\sigma^2 \cdot \mathbf{C}$ in CMA-ES, where $\mathbf{C}$ can be any positive definite symmetric matrix. Thus, the representation of the scale of the search distribution is shared among σ and $\mathbf{C}$ in CMA-ES, and the role of the additional parameter σ is to allow for an adaptation of the step size on a faster time scale than the full covariance update. In contrast, the NES updates of scale and shape parameters σ and $\mathbf{B}$ are decoupled.

The update of the center parameter $\boldsymbol{\mu}$ is very similar to the update of the center of the search distribution in CMA-ES, see Hansen and Ostermeier (2001). The utility function exactly takes the role of the weights in CMA-ES, which assumes a fixed learning rate of one.

For the covariance matrix, the situation is more complicated. We deduce the update rule

$$\begin{aligned}\boldsymbol{\Sigma}_{\text{new}} =& (\mathbf{A}_{\text{new}})^\top \cdot \mathbf{A}_{\text{new}} \\ =& \mathbf{A}^\top \cdot \exp\left(\eta_{\boldsymbol{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}\right)\right) \cdot \mathbf{A}\end{aligned}$$

for the covariance matrix, with learning rate $\eta_{\boldsymbol{\Sigma}} = \eta_{\mathbf{A}}$. This term is closely connected to the exponential parameterization of the natural coordinates in xNES, while CMA-ES is formulated in global linear coordinates. The connection of these updates can be shown either by applying the xNES update directly to the natural coordinates without the exponential parameterization (Akimoto et al., 2010), or by approximating the exponential map by its first order Taylor expansion. Akimoto et al. (2010) established the same connection directly in coordinates based on the Cholesky decomposition of $\boldsymbol{\Sigma}$, see Sun et al. (2009a,b). The arguably simplest derivation of the equivalence relies on the invariance of the natural gradient under coordinate transformations, which allows us to perform the computation, w.l.o.g., in natural coordinates. We use the first order Taylor approximation of the matrix exponential to obtain

$$\exp\left(\eta_{\mathbf{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}\right)\right) \approx \mathbb{I} + \eta_{\mathbf{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}\right),$$

so the first order approximate update yields

$$\begin{aligned}
\mathbf{\Sigma}'_{new} =& \mathbf{A}^\top \cdot \left(\mathbb{I} + \eta_{\mathbf{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{s}_k \mathbf{s}_k^\top - \mathbb{I}\right)\right) \cdot \mathbf{A} \\
=& \left(1 - U \cdot \eta_{\mathbf{\Sigma}}\right) \cdot \mathbf{A}^\top \mathbf{A} + \eta_{\mathbf{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{A}^\top \mathbf{s}_k\right) \left(\mathbf{A}^\top \mathbf{s}_k\right)^\top \\
=& \left(1 - U \cdot \eta_{\mathbf{\Sigma}}\right) \cdot \mathbf{\Sigma} + \eta_{\mathbf{\Sigma}} \cdot \sum_{k=1}^{\lambda} u_k \left(\mathbf{z}_k - \boldsymbol{\mu}\right) \left(\mathbf{z}_k - \boldsymbol{\mu}\right)^\top
\end{aligned}$$

with $U = \sum_{k=1}^{\lambda} u_k$, from which the connection to the CMA-ES rank-μ-update is obvious (see Hansen and Ostermeier, 2001, and note that $U = 1$ for CMA-ES.).

It is interesting to note that both xNES and CMA-ES use different learning rates for the mean and covariance components of the search distribution. Thus, in a strict sense they do not follow the natural gradient. Instead they follow a systematically transformed direction with altered (typically reduced) covariance component. This makes intuitive sense since the d components of the mean can be estimated more robustly from a fixed size sample than the $\mathcal{O}(d^2)$ covariance parameters. The theoretical implications of using differently scaled updates for the two components are yet to be explored.

CMA-ES uses the well-established technique of evolution paths to smooth out random effects over multiple generations. This technique is particularly valuable when working with minimal population sizes, which is the default for both algorithms. Thus, evolution paths are expected to improve stability; further interpretations have been provided by Hansen and Ostermeier (2001). However, the presence of evolution paths complicates matters since the state of the CMA-ES algorithms is not completely described by its search distribution. Another difference between xNES and CMA-ES is the exponential parameterization of the updates in xNES, which results in a multiplicative update equation for the covariance matrix, in contrast to the additive update of CMA-ES. The multiplicative covariance update is coherent with the multiplicative (and also exponential) update of the step size σ.

A valuable perspective offered by the natural gradient updates in xNES is the derivation of the updates of the center $\boldsymbol{\mu}$, the step size σ, and the normalized transformation matrix $\mathbf{B}$, all from the *same* principle of natural gradient ascent. In contrast, the updates applied in CMA-ES result from different heuristics for each parameter. This connection might provide an interesting perspective on some of the methods employed by CMA-ES.

5. Experiments

In this section, we empirically validate the new algorithms, to determine how NES algorithms perform compared to state-of-the-art evolution strategies, identifying specific strengths and limitations of the different variants.

We conduct a broad series of experiments on standard benchmarks, as well as more specific experiments testing special capabilities. In total, four different algorithm variants are tested and their behaviors compared qualitatively as well as quantitatively, w.r.t. different modalities.

We start by detailing and justifying the choices of hyperparameters, then we proceed to evaluate the performance of a number of different variants of NES (with and without adaptation sampling) on a broad collection of benchmarks. We also conduct experiments using the separable variant on high-dimensional problems.

5.1 Experimental Setup and Hyperparameters

Across all NES variants, we distinguish three hyperparameters: the population size λ, the learning rates η and the utility function u (because we always use fitness shaping, see Section 3.1). In particular, for the multivariate Gaussian case (xNES) we have the three learning rates $\eta_{\boldsymbol{\mu}}$, η_σ, and $\eta_{\mathbf{B}}$.

It is highly desirable to have good default settings that scale with the problem dimension and lead to robust performance on a broad class of benchmark functions. Table 1 provides such default values as functions of the problem dimension d for xNES. We borrowed several of the settings from CMA-ES (Hansen and Ostermeier, 2001), which seems natural due to the apparent similarity. Both the population size λ and the learning rate $\eta_{\boldsymbol{\mu}}$ are the same as for CMA-ES, even if this learning rate never explicitly appears in CMA-ES. For the utility function we copied the weighting scheme of CMA-ES, but we shifted the values such that they sum to zero, which is the simplest form of implementing a fitness baseline; Jastrebski and Arnold (2006) proposed a similar approach for CMA-ES. The remaining parameters were determined via an empirical investigation, aiming for robust performance. In addition, in the separable case (SNES) the number of parameters in the covariance matrix is reduced from $d(d+1)/2 \in \mathcal{O}(d^2)$ to $d \in \mathcal{O}(d)$, which allows us to increase the learning rate $\eta_{\boldsymbol{\sigma}}$ by a factor of $d/3 \in \mathcal{O}(d)$, a choice which has proven robust in practice (Ros and Hansen, 2008).

The algorithm variants that we will be evaluating below are xNES (Algorithm 5), "xNES-as", that is xNES using adaptation sampling (Section 3.2), and the separable SNES (Algorithm 6). A Python implementation of all these is available within the open-source machine learning library PyBrain (Schaul et al., 2010), and implementations in different languages can be found at `http://www.idsia.ch/~tom/nes.html`.

5.2 Black-box Optimization Benchmarks

For a practitioner it is important to understand how NES algorithms compare to other methods on a wide range black-box optimization scenarios. Thus, we evaluate our algorithm on all the benchmark functions of the 'Black-Box Optimization Benchmarking' collection (BBOB) from the GECCO Workshop for Real-Parameter Optimization. The collection consists of 24 noise-free functions (12 unimodal, 12 multimodal; Hansen et al., 2010a) and 30 noisy functions (Hansen et al., 2010b). In order to make our results fully comparable, we also use the identical setup (Hansen and Auger, 2010), which transforms the pure benchmark functions to make the parameters non-separable (for some) and avoid trivial optima at the origin. The framework permits restarts until the budget of function evaluations ($10^5 d$) is

parameter	default value
λ	$4 + \lfloor 3\log(d) \rfloor$
$\eta_{\boldsymbol{\mu}}$	1
$\eta_\sigma = \eta_{\mathbf{B}}$	$\frac{(9 + 3\log(d))}{5d\sqrt{d}}$
$\eta_{\boldsymbol{\sigma}}$	$\frac{(3 + \log(d))}{5\sqrt{d}}$
u_k	$\frac{\max\left(0, \log(\frac{\lambda}{2}+1) - \log(i)\right)}{\sum_{j=1}^{\lambda} \max\left(0, \log(\frac{\lambda}{2}+1) - \log(j)\right)} - \frac{1}{\lambda}$
c'	$\frac{1}{10}$

Table 1: Default parameter values for xNES, xNES-as and SNES (including the utility function) as a function of problem dimension d.

used up, which we trigger whenever the variance of the distribution becomes too small, that is, when $\sqrt[d]{\det \boldsymbol{\Sigma}} < 10^{-20}$.

The results in this subsection are very similar[2] to those published in the 2012 edition of the BBOB Workshop at GECCO, we refer the interested reader to Schaul (2012b,c,e,d) for additional results and analysis. Figure 4 provides a compounded overview of the results on dimensions 5 and 20, on noisy or noiseless functions, and a direct comparison to all algorithms benchmarked in the 2009 edition of BBOB, which shows that xNES is among the best algorithms, assuming that the budget of function evaluations surpasses 100 times the dimension. We find (Schaul, 2012c) that xNES-as significantly outperforms all algorithms from the BBOB 2009 competition on function f_{115} and for limited budgets on f_{18}, f_{118} and f_{119}, while underperforming compared to the winners on a set of other functions.

Figures 5 (noise-free functions) and Figure 6 (noisy functions) show how performance scales with dimension, on a detailed function-by-function level, for both xNES and xNES-as. Using adaptation sampling improves performance most significantly on simple benchmarks like the sphere function or its noisy siblings (f_1, f_{101}, f_{102}, f_{103}, f_{107}) and only hurts significantly on f_{13}, in high dimensions (see Schaul, 2012e for the full comparison, and result tables with statistical significance tests). While the presented default settings are weak for highly multi-modal functions like f_3, f_4 or f_{15}, larger population sizes and sophisticated restart strategies may alleviate this issue.

Figure 7 compares the loss ratios (in terms of expected running time) given fixed budgets of evaluations, for xNES, xNES-as, and BIPOP-CMA-ES (Hansen, 2009a,b), the winner of the 2009 BBOB edition as reference point. BIPOP-CMA-ES is significantly better on most noisy functions, but the differences are much more subtle on the noiseless ones. In fact, a

2. The difference lies with the stopping criterion used for restarts, which was not compliant in the 2012 results.

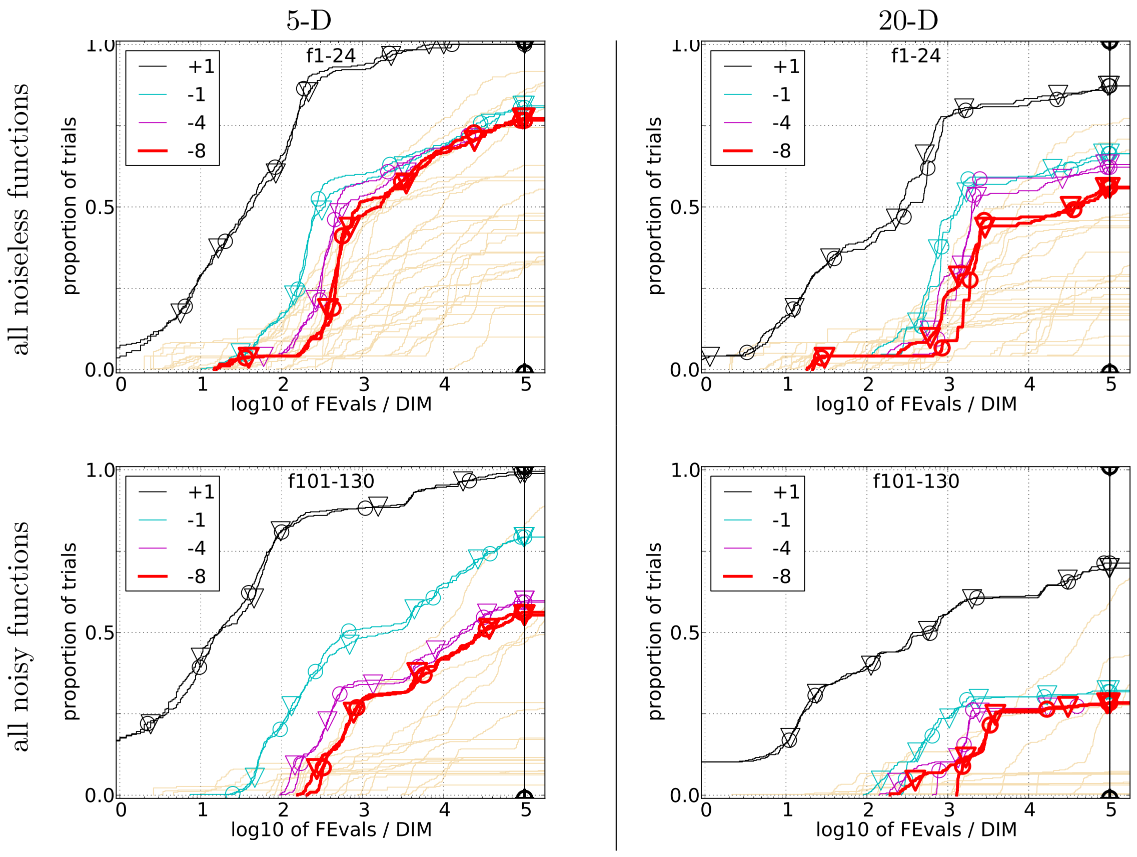


Figure 4: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right), over all noiseless functions (top row) and all noisy functions (bottom row). The ECDF is taken of the number of function evaluations divided by dimension d to reach a target value $f_{\text{opt}} + 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for xNES ($\circ$) and xNES_as ($\triangledown$). Light beige lines show the ECDF for target precision 10^{-8} of all algorithms benchmarked during BBOB-2009. From this high-level perspective, both xNES variants appear among the best tested algorithms, with a small advantage for using adaptation sampling.

direct comparison (Schaul, 2012d) found that xNES-as is close in performance to BIPOP-CMA-ES across a large fraction of the benchmark functions; but there is some diversity as well, with xNES-as being significantly better on 6 of the functions and significantly worse on 18 of them.

5.3 Separable NES for Neuroevolution

The SNES algorithm is expected to perform at least as well as xNES on separable problems, while it should show considerably worse performance in the presence of highly dependent variables. These are indeed the the findings in Schaul (2012a), where SNES was bench-

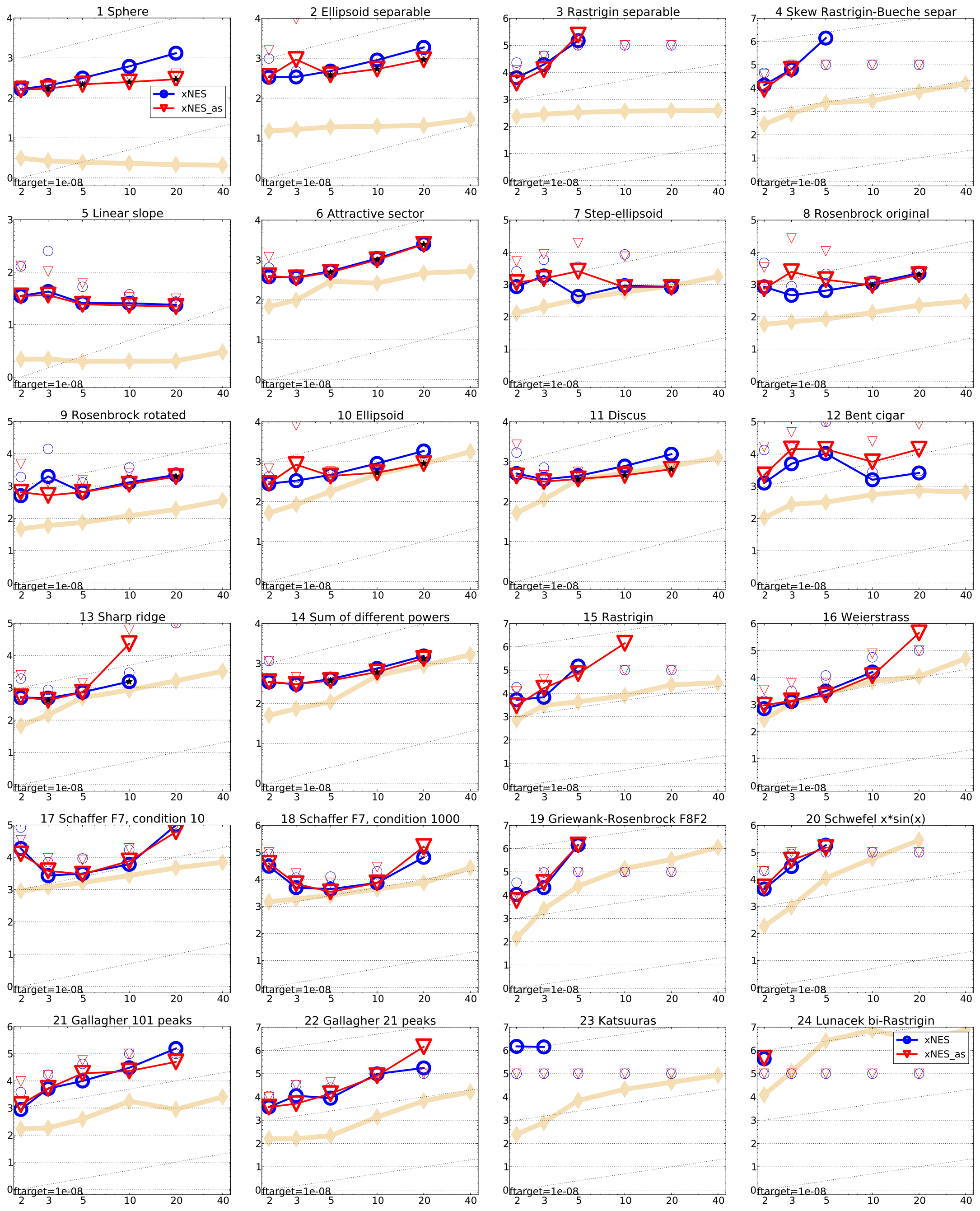


Figure 5: Expected running time (ERT, as $\log_{10}$ value of the number of f-evaluations divided by dimension) on all noise-free functions (f_1 and f_{24}) for target precision 10^{-8}, versus dimension on the horizontal axis. Blue circles $\circ$ refer to xNES, red triangles $\triangledown$ refer to xNES_as, and the light beige lines show the performance of the best-performing algorithm from the BBOB-2009 entrants (the best, individually for each dimension-function pair). Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate statistically better result compared to all other algorithms with $p < 0.01$.

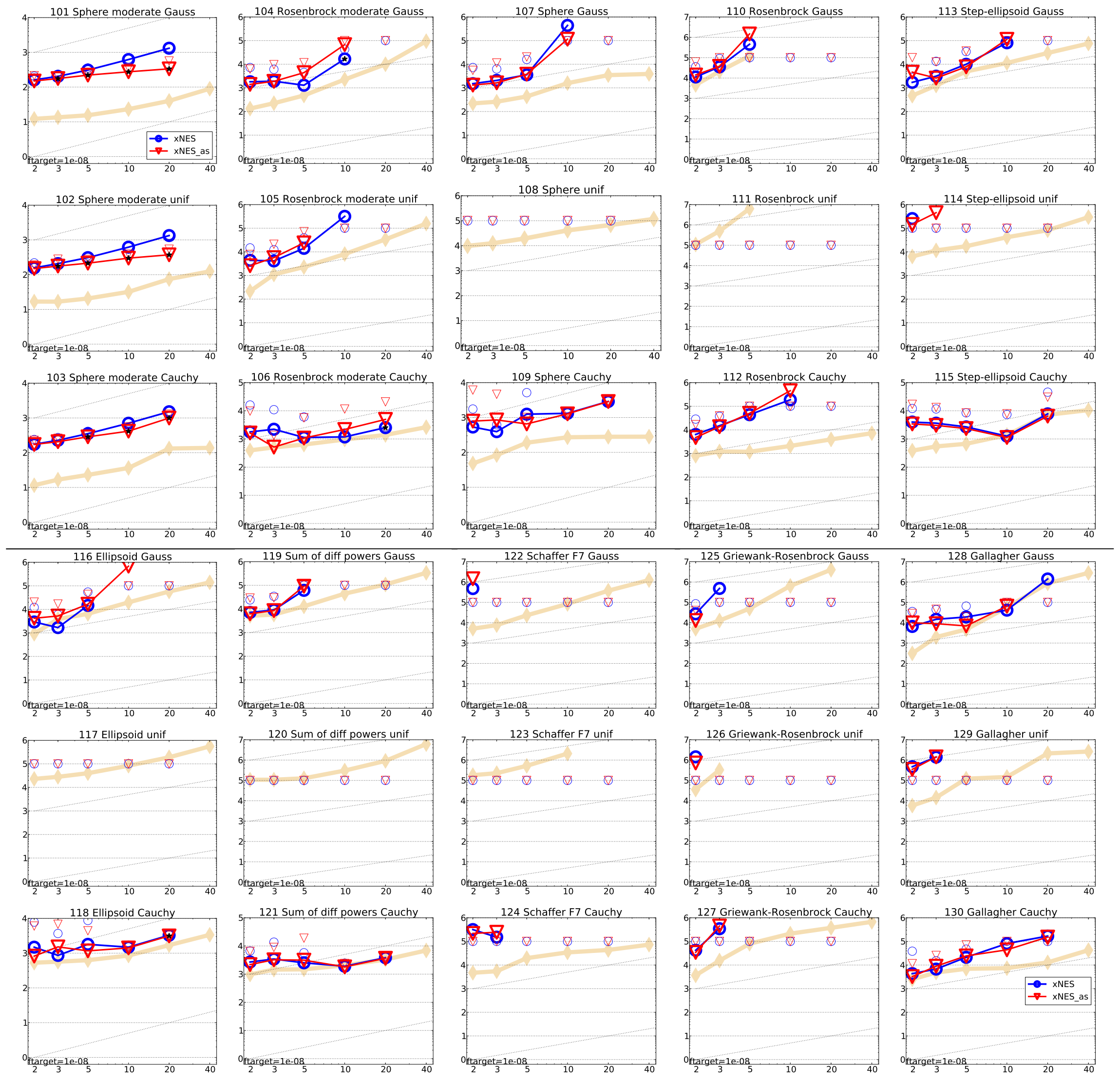


Figure 6: Expected running time (ERT in number of f-evaluations, divided by dimension) on all noisy functions (f_{101} and f_{130}) for target precision 10^{-8}, versus dimension (see Figure 5 for details). We observe that, despite using small population sizes and the same parameter settings than on the noise-free benchmarks, xNES achieves state-of-the art performance on a subset of the functions.

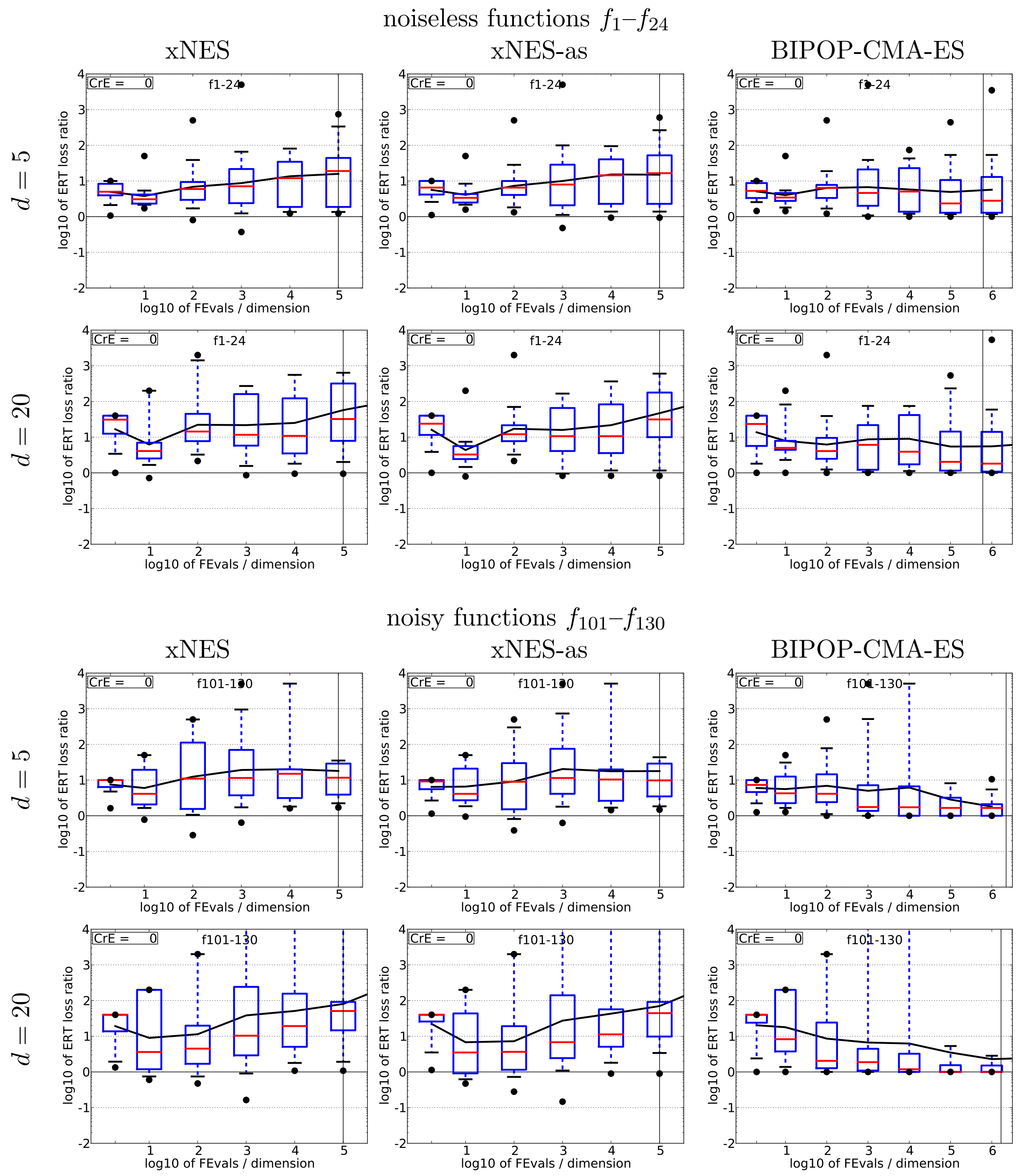


Figure 7: Loss ratio of expected running time (ERT), given a budget of function evaluations (here, smaller is better), for xNES (left column), xNES-as (middle column) and BIPOP-CMA-ES (right column). The target value f_t used for a given budget is the smallest (best) recorded function value such that ERT(f_t) $\leq$ FEvals for the presented algorithm. Shown is FEvals divided by the respective best ERT(f_t) from BBOB-2009 for all functions (noiseless f_1–f_{24}, top rows, and noisy f_{101}–f_{130}, bottom rows) in 5-D and 20-D. Black line: geometric mean. Box-Whisker error bar: 25-75%-ile (box) with median (red), 10-90%-ile (caps), and minimum and maximum ERT loss ratio (black points).

marked on the entire BBOB suite. In contrast to the BBOB setups, which are based on tricky fitness functions in *small* problem dimensions, this section illustrates how SNES scales with dimension, on a classical neuro-evolutionary controller design problem.

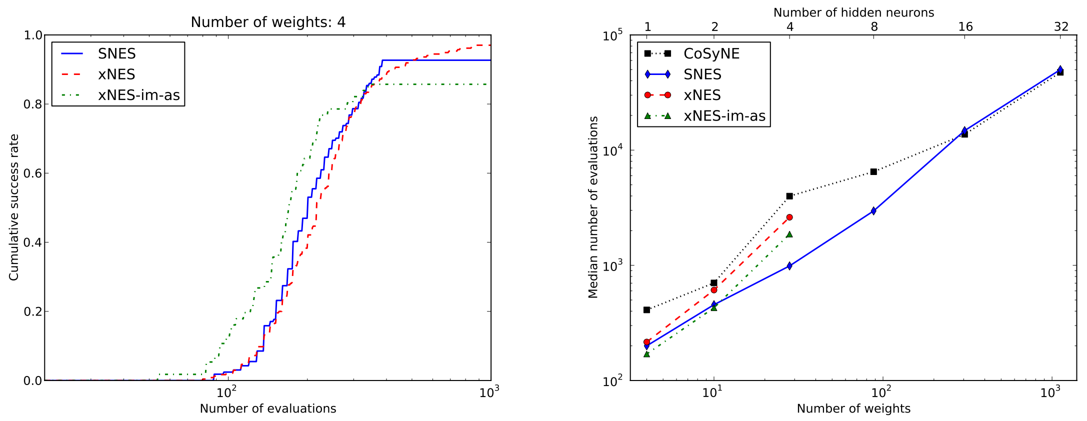


Figure 8: **Left:** Plotted are the cumulative success rates on the non-Markovian double-pole balancing task after a certain number of evaluations, empirically determined over 100 runs for each algorithm, using a single tanh-unit ($n = 1$) (i.e., optimizing 4 weights). We find that all three algorithm variants give state-of-the-art results, with a slightly faster but less robust performance for xNES-as. **Right:** Median number of evaluations required to solve the same task, but with increasing number of neurons (and corresponding number of weights). We limited the runtime to one hour per run, which explains why no results are available for xNES on higher dimensions (cubic time complexity). The fact that SNES quickly outperforms xNES, also in number of function evaluations, indicates that the benchmark is (sufficiently close to) separable, and it is unnecessary to use the full covariance matrix. For reference we also plot the corresponding results of the previously best performing algorithm CoSyNE (Gomez et al., 2008).

SNES is well-suited for neuroevolution problems because they tend to be high-dimensional, multi-modal, but with highly redundant global optima (there is not a unique set of weights that defines the optimal behavior). We use it to find a controller for non-Markovian double pole balancing, a task which involves balancing two differently sized poles hinged on a cart that moves on a finite track. The single control consists of the force F applied to the cart, and observations include the cart's position and the poles' angles, but no velocity information, which makes this task partially observable. It provides a perfect testbed for algorithms focusing on learning fine control with memory in continuous state and action spaces (Wieland, 1991). The controller is represented by a simple recurrent neural network, with three inputs, (position x and the two poles' angles β_1 and β_2), and a variable number n of tanh units in the output layer, which are fully connected (recurrently), resulting in a total of $n(n + 3)$ weights to be optimized. The activation of the first of these recurrent neurons directly determines the force to be applied. We use the implementation found in PyBrain (Schaul et al., 2010).

An evaluation is considered a success if the poles do not fall over for $100,000$ time steps. We experimented with recurrent layers of sizes $n = 1$ to $n = 32$ (corresponding to between 4 and 1120 weights). It turns out that a single recurrent neuron is sufficient to solve the task (Figure 8, left). In fact, both the xNES and SNES results are state-of-the-art, outperforming the previously best algorithm (CoSyNE; Gomez et al., 2008, with a median of 410 evaluations) by a factor two.

In practical scenarios however, we cannot know the best network size a priori, and thus the prudent choice consists in overestimating the required size. An algorithm that graciously scales with problem dimension is therefore highly desirable, and we find (Figure 8, right) that SNES is exhibiting precisely that behavior. The fact that SNES outperforms xNES with increasing dimension, also in number of function evaluations, indicates that the benchmark is not ill-conditioned, and it is unnecessary to use the full covariance matrix. We conjecture that this is a property shared with the majority of neuroevolution problems that have enough weights to exhibit redundant global optima (some of which can be found without considering all parameter covariances).

Additional SNES results, for example, on the Lennard-Jones benchmark for atom clusters (with dimension $d > 200$) can be found in Schaul et al. (2011).

6. Discussion and Conclusion

Our results on the BBOB benchmarks show that NES algorithms perform well across a wide variety of black-box optimization problems. We have demonstrated advantages and limitations of specific variants, and as such established the generality and flexibility of the NES framework. Experiments with heavy-tailed and separable distributions demonstrate the viability of the approach on high-dimensional domains. We obtained best reported results on the difficult task of training a neural controller for double pole-balancing.

Technique	Issue addressed	Applicability limited to	Relevant section
Natural gradient	Scale-invariance, many more	-	2.3
Fitness shaping	Robustness	-	3.1
Adaptation sampling	Performance, sensitivity	-	3.2
Exponential parameterization	Covariance constraints	Multivariate	3.4
Natural coordinate system	Computational efficiency	Multivariate	3.4

Table 2: Summary of enhancing techniques

Table 2 summarizes the various techniques we introduced. The plain search gradient suffers from premature convergence and lack of scale invariance (see Section 2.2). Therefore, we use the natural gradient instead, which turns NES into a viable optimization method. To improve performance and robustness, we introduced several novel techniques. Fitness shaping makes the NES algorithm invariant to order-preserving transformations of the fitness function, thus increasing robustness. Adaptation sampling adjusts the learning rates online, which yields highly performant results on standard benchmarks. Finally, the exponential parameterization is crucial for maintaining positive-definite covariance matrices, and the use of the natural coordinate system guarantees computational feasibility.

NES applies to general parameterizable distributions. In this paper, we have experimentally investigated two variants, adjusted to the particular properties of different problem classes. We demonstrated the power of the xNES variant using a full multinormal distribution, which is invariant under arbitrary translations and rotations, on the canonical suite of standard benchmarks. Additionally, we showed that the restriction of the covariance matrix to a diagonal parameterization (SNES) allows for scaling to very high dimensions, on the difficult non-Markovian double pole balancing task.

Acknowledgments

This research was funded through the 7th framework program of the European Union, under grant number 231576 (STIFF project) and ICT-270327 CompLACS, SNF grants 200020-116674/1, 200021-111968/1 and 200020-122124/1, and SSN grant Sinergia CRSIK0-122697. We thank Faustino Gomez for helpful suggestions and his CoSyNE data, as well as Andreas Krause for insightful discussions. We particularly thank the anonymous reviewers for their many constructive remarks.

References

Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. Bidirectional relation between CMA evolution strategies and natural evolution strategies. In *Parallel Problem Solving from Nature (PPSN)*, 2010.

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

S. Amari and S. C. Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98)*, volume 2, pages 1213–1216, 1998.

A. Auger. Convergence results for the (1,λ)-SA-ES using the theory of ϕ-irreducible Markov chains. *Theoretical Computer Science*, 334(1-3):35 – 69, 2005.

A. Berny. Selection and reinforcement learning for combinatorial optimization. In *Parallel Problem Solving from Nature PPSN VI*, volume 1917, pages 601–610. Springer Berlin / Heidelberg, 2000.

A. Berny. *Statistical machine learning and combinatorial optimization*, pages 287–306. Springer-Verlag, London, UK, 2001.

H.-G. Beyer. *The Theory of Evolution Strategies.* Springer-Verlag, New York, USA, 2001.

H.-G. Beyer and H.-P. Schwefel. Evolution strategies: a comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

P. A. N. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: the IDEA. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 767–776, London, UK, 2000. Springer-Verlag.

P. A. N. Bosman, J. Grahl, and D. Thierens. Adapted maximum-likelihood Gaussian models for numerical optimization with continuous EDAs. Technical report, 2007.

F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117, 2005.

T. Glasmachers, T. Schaul, and J. Schmidhuber. A natural evolution strategy for multi-objective optimization. In *Parallel Problem Solving from Nature (PPSN)*, 2010a.

T. Glasmachers, T. Schaul, Y. Sun, D. Wierstra, and J. Schmidhuber. Exponential natural evolution strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, Portland, USA, 2010b.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675.

F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 2008.

N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *GECCO (Companion)*, pages 2389–2396, 2009a.

N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 noisy testbed. In *GECCO (Companion)*, pages 2397–2402, 2009b.

N. Hansen and A. Auger. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical report, INRIA, 2010.

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

N. Hansen, A. S. P. Niederberger, L. Guzzella, and P. Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *Transactions on Evolutionary Computation*, 13:180–197, 2009.

N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Noiseless function definitions. Technical report, INRIA, 2010a.

N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Noisy functions definitions. Technical report, INRIA, 2010b.

M. Hasenjäger, B. Sendhoff, T. Sonoda, and T. Arima. Three dimensional evolutionary aerodynamic design optimization with CMA-ES. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 2173–2180, New York, NY, USA, 2005. ACM.

J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, 1975.

C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50:2003, 2003.

J. Jägersküpper. Analysis of a simple evolutionary algorithm for minimization in Euclidean spaces. *Theoretical Computer Science*, 379(3):329–347, 2007.

G. A. Jastrebski and D. V. Arnold. Improving evolution strategies through active covariance matrix adaptation. In *IEEE Congress on Evolutionary Computation*, 2006.

M. Jebalia, A. Auger, M. Schoenauer, F. James, and M. Postel. Identification of the isotherm function in chromatography using CMA-ES. In *IEEE Congress on Evolutionary Computation*, pages 4289–4296, 2007.

M. Jebalia, A. Auger, and N. Hansen. Log-linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. *Algorithmica*, pages 1–36, 2010.

J. Kennedy and R. Eberhart. *Swarm Intelligence.* Morgan Kaufmann, San Francisco, CA, 2001.

S. Kirkpatrick, C. D. Gelatt, Jr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

J. Klockgether and H. P. Schwefel. Two-phase nozzle and hollow core jet experiments. In *Proc. 11th Symp. Engineering Aspects of Magnetohydrodynamics*, pages 141–148, 1970.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. ISSN 00034851.

P. Larrañaga. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, 2002.

H. Mühlenbein and G. Paass. From recombination of genes to the estimation of distributions I. binary parameters. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, PPSN IV, pages 178–187, London, UK, 1996. Springer-Verlag.

S. D. Muller, J. Marchetto, S. Airaghi, and P. Koumoutsakos. Optimization based on bacterial chemotaxis. *IEEE Transactions on Evolutionary Computation*, 6:6–16, 2002.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

A. Ostermeier, A. Gawelczyk, and N. Hansen. Step-size adaption based on non-local use of selection information. In *The Third Conference on Parallel Problem Solving from Nature*, pages 189–198, London, UK, 1994. Springer-Verlag.

M. Pelikan, D. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. In *American Control Conference*, volume 5, pages 3289 –3293 vol.5, 2000.

M. Pelikan, K. Sastry, and E. C. Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., 2006.

J. Peters. *Machine Learning of Motor Skills for Robotics.* PhD thesis, Department of Computer Science, University of Southern California, 2007.

I. Rechenberg and M. Eigen. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution.* Frommann-Holzboog Stuttgart, 1973.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591. IEEE Press, 1993.

R. Ros and N. Hansen. A simple modification in CMA-ES achieving linear time and space complexity. In R. et al., editor, *Parallel Problem Solving from Nature, PPSN X*, pages 296–305. Springer, 2008.

R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning (Information Science and Statistics).* Springer, 2004.

T. Schaul. Benchmarking separable natural evolution strategies on the noiseless and noisy black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012a.

T. Schaul. Benchmarking exponential natural evolution strategies on the noiseless and noisy black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012b.

T. Schaul. Benchmarking natural evolution strategies with adaptation sampling on the noiseless and noisy black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012c.

T. Schaul. Comparing natural evolution strategies to BIPOP-CMA-ES on noiseless and noisy black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012d.

T. Schaul. Investigating the impact of adaptation sampling in natural evolution strategies on black-box optimization testbeds. In *Black-box Optimization Benchmarking Workshop, Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012e.

T. Schaul. Natural evolution strategies converge on sphere functions. In *Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012f.

T. Schaul and J. Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010.

T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010.

T. Schaul, T. Glasmachers, and J. Schmidhuber. High dimensions and heavy tails for natural evolution strategies. In *Genetic and Evolutionary Computation Conference*, 2011.

H.-P. Schwefel. Numerische optimierung von computer-modellen mittels der evolutionsstrategie, 1977.

J. Shepherd, D. McDowell, and K. Jacob. Modeling morphology evolution and mechanical behavior during thermo-mechanical processing of semi-crystalline polymers. *Journal of the Mechanics and Physics of Solids*, 54(3):467 – 489, 2006.

O. M. Shir and T. Bäck. The second harmonic generation case-study as a gateway for ES to quantum control problems. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 713–721, New York, NY, USA, 2007. ACM.

R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11:341–359, December 1997. ISSN 0925-5001.

Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Stochastic search using the natural gradient. In *International Conference on Machine Learning (ICML)*, 2009a.

Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber. Efficient natural evolution strategies. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2009b.

A. Wieland. Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks (Seattle, WA)*, pages 667–673, 1991.

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.

S. Winter, B. Brendel, and C. Igel. Registration of bone structures in 3D ultrasound and CT data: Comparison of different optimization strategies. *International Congress Series*, 1281:242 – 247, 2005.