Stochastic Search using the Natural Gradient

Keywords: stochastic search, natural gradient, evolution strategies

Sun Yi YI@IDSIA.CH
Daan Wierstra DAAN@IDSIA.CH
Tom Schaul TOM@IDSIA.CH
Jürgen Schmidhuber JUERGEN@IDSIA.CH
IDSIA, Galleria 2, Manno 6928, Switzerland

Abstract

To optimize unknown 'fitness' functions, we present Natural Evolution Strategies, a novel algorithm that constitutes a principled alternative to standard stochastic search methods. It maintains a multinormal distribution on the set of solution candidates. The Natural Gradient is used to update the distribution's parameters in the direction of higher expected fitness, by efficiently calculating the inverse of the exact Fisher information matrix whereas previous methods had to use approximations. Other novel aspects of our method include optimal fitness baselines and importance mixing, a procedure adjusting batches with minimal numbers of fitness evaluations. The algorithm yields competitive results on a number of benchmarks.

1. Introduction

Stochastic search methods aim to optimize a 'fitness' function that is either unknown or too complex to model directly. This general framework is known as 'black box' optimization (Hansen & Ostermeier, 2001). It allows 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 stochastic search algorithms: (1) robust performance in terms of fitness; (2) limiting the number of (potentially costly) fitness evaluations; (3) scalability with problem dimensionality. We address these issues through a new, principled method for stochastic search in continuous search spaces, which is less ad-hoc than traditional stochastic search methods.

Our algorithm maintains and iteratively updates a multinormal distribution on the search space of solution candidates. Its parameters are updated by estimating and following a *natural search gradient*, (i.e., the natural gradient on the parameters of the search distribution), towards better expected fitness. Numerous advantages of the natural gradient have been demonstrated, including its ability of providing isotropic convergence on ill-shaped function landscapes, avoiding slow or premature convergence to which 'vanilla' (regular) gradients are especially prone (Amari & Douglas, 1998; Peters & Schaal, 2008).

In our algorithm, the natural search gradient is calculated using the *exact* Fisher information matrix (FIM) and the Monte Carlo-estimated gradient, yielding robust performance (objective 1). To reduce the number of potentially costly evaluations (objective 2), we introduce *importance mixing*, which entails the reuse of samples from previous batches while keeping the sample distribution conformed to the current search 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, *Natural Evolution Strategies* (NES), is elegant, requires no additional heuristics and has few parameters that need tuning.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

2. Search Gradients

First let us introduce the algorithm framework and the concept of search 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 batch n samples $\mathbf{z}_1 \dots \mathbf{z}_n$ generated from the search 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 search distribution.

Let $J(\theta) = \mathbb{E}[f(\mathbf{z})|\theta]$ be the expected fitness under search 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

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

The last term can be approximated using Monte Carlo:

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

where $\nabla^s_\theta J(\theta)$ denotes the estimated search 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

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

Here $\theta^0 = \mathbf{x}$ and $\theta^k = [a_{k,k} \dots a_{k,d}]^\top$ for $1 \le k \le d$, where $a_{i,j}$ $(i \le j)$ 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. $a_{i,j}$ $(i \le j)$ can be derived as

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

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

3. Using the Natural Gradient

Vanilla gradient methods have been shown to converge slowly in fitness landscapes with ridges and plateaus. Natural gradients (Amari et al., 1996; Kakade, 2002) 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 search 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 (Amari, 1998).

In this paper, we consider a special case of the natural gradient $\tilde{\nabla}_\theta J$, with the metric in parameter space defined by the KL divergence (Peters, 2007). The natural gradient is thus defined by the necessary condition

$$\mathbf{F} \tilde{\nabla}_\theta J = \nabla_\theta J,$$

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

with $\mathbf{F}$ being the Fisher information matrix:

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

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 $\mathbf{F}^- \nabla^s_\theta J$. The adjustment $\delta\theta$ can then be computed by

$$\delta\theta = \eta \mathbf{F}^- \nabla^s_\theta J.$$

As we show below, the FIM can indeed be computed exactly and is invertible.

The original NES (Wierstra et al., 2008b) algorithm computes the natural search gradient using the empirical Fisher information matrix, which is estimated from the current batch. 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 batch 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 $O\left(d^4\right)$ elements.

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

In NES, the search 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

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

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 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

$$\mathbf{F} = \begin{bmatrix} \mathbf{C}^- & \mathbf{0} \\ \mathbf{0} & \mathbf{F_A} \end{bmatrix}.$$

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

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

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

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

Computing the first term gives us

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

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

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

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

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

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

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_0 & & & \\ & \mathbf{F}_1 & & \\ & & \ddots & \\ & & & \mathbf{F}_d \end{bmatrix},$$

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

$$\mathbf{F}_k = \begin{bmatrix} a^{-2}_{k,k} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \mathbf{D}_k.$$

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 $\mathbf{D}_d = (\mathbf{C}^-)_{d,d}$.

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 $a_{k,k}^{-2} > 0$ 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 $\theta^0 \dots \theta^k$, which suggests that we could modify each group of parameters without affecting other groups.

Normally, computing the inverse of $\mathbf{F}$, of size d^2 by d^2, would be of complexity $O\left(d^6\right)$, which is intractable for most practical problems. Realizing that we can invert each block $\mathbf{F}_k$ separately, the complexity can be reduced to $O\left(d^4\right)$. In a companion paper (Sun et al., 2009), we present an iterative method for computing $\mathbf{F}^-$ which further reduces the time complexity from $O\left(d^4\right)$ to $O\left(d^3\right)$. Additionally it shows that the space complexity can be reduced to $O\left(d^2\right)$ which is linear in the number of parameters.

4. Optimal Fitness Baselines

The concept of *fitness baselines*, first introduced in (Wierstra et al., 2008b), constitutes an efficient variance reduction method for estimating $\delta\theta$. However, baselines as found in (Peters, 2007) 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

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

where $\nabla_\theta^s J$ is the estimated search 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.

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

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 samples 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, the main disadvantage of parameter-specific baselines is that different baselines values are used for closely correlated parameters, which renders gradient estimations less reliable. In order to address such 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, $0 \leq k \leq d$. 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.

5. Importance Mixing

In each batch, we evaluate n new samples generated from search distribution $p\left(\mathbf{z}|\theta\right)$. However, since small updates ensure that the KL divergence between consecutive search distributions is generally small, most

new samples will fall in the high density area of the previous search distribution $p(\mathbf{z}|\theta')$. 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 batch, while ensuring the updated batch conforms to the new search distribution.

Importance mixing works in two steps: In the first step, rejection sampling is performed on the previous batch, such that sample $\mathbf{z}$ 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 samples accepted in the first step. In the second step, reverse rejection sampling is performed as follows: Generate samples 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 samples are accepted. The n_a samples from the old batch and $n-n_a$ newly accepted samples together constitute the new batch. Note that only the fitnesses of the newly accepted samples 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 batch, 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 samples $\rho = \mathbb{E}\left[\frac{n-n_a}{n}\right]$, namely $\rho \geq \alpha$, with the equality holding iff $\theta = \theta'$. In particular, if $\alpha = 1$, all samples from the previous batch 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 batch conforms to the search distribution $p(\mathbf{z}|\theta)$. In the region where $(1-\alpha)\,p(\mathbf{z}|\theta)/p(\mathbf{z}|\theta') \leq 1$, the probability that a sample from previous batches appears in the new batch 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 a sample generated from the second step entering the batch is $\alpha p(\mathbf{z}|\theta)$, since

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

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

6. The Algorithm

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

```
1   initialize A ← I
2   repeat
3     compute A⁻, and C⁻ = A⁻A⁻ᵀ
4     update batch using importance mixing
5     evaluate f(z_i) for new z_i
6     compute the gradient G
7     for k = d to 0
8       compute the exact FIM inverse F_k⁻
9       compute the baseline b^k
10      δθ^k ← F_k⁻ G^k (f − b^k)
11    end
12    θ ← θ + ηδθ
13  until stopping criteria is met
```

Assuming that n scales linearly with d, the complexity of our algorithm is $O(d^3)$ (Sun et al., 2009). This is a significant improvement over the original NES, whose complexity is $O(d^6)$.

Implementations of NES are available in both Python and Matlab[2].

7. Experiments

The tunable parameters of Natural Evolution Strategies are comprised of the batch 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 performance can be further improved by more aggressive updates. Therefore, the only parameter that needs tuning in practice is the batch size, which is dependent on both the expected ruggedness of the fitness landscape and the problem dimensionality.

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

[2]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`

use *fitness shaping*, an order-preserving nonlinear fitness transformation function (Wierstra et al., 2008b). 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 $\hat{f}(\mathbf{z}) = 2i - 1$ for $i > 0.5$ and $\hat{f}(\mathbf{z}) = 0$ for $i < 0.5$, where i denotes the relative rank of $f(\mathbf{z})$ in the batch, scaled between $0 \dots 1$.

7.1. Benchmark Functions

We empirically validate our algorithm on 9 unimodal functions out of the set of standard benchmark functions from (Suganthan et al., 2005) and (Hansen & Ostermeier, 2001), 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 (Suganthan et al., 2005) 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 search directions. NES, 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.

7.2. Fitness Baselines and Importance Mixing

We introduced optimal fitness baselines to increase the algorithm's robustness, we can thus determine their effectiveness by comparing the probability of premature convergence, for each type of baseline, on a diverse set of benchmarks. Importance mixing, on the other hand, was designed to reduce the required number of fitness evaluations.

In order to measure the benefits of both enhancements, as well as their interplay, we conducted a set of experiments for 16 different settings. Each set consisted in 10 independent runs on each of the 8 unimodal benchmark functions (on dimension 5), using $n = 50$ and $\eta = 1.0$. We varied the value of α between 0.0 and 1.0, the latter corresponding to not using importance mixing at all. We compare the three types of optimal fitness baselines introduced in section 4 to using no baseline. Table 1 summarizes the results.

Not using any baseline shows equivalent behavior to using the uniform fitness baseline. In both cases, the algorithm tends to prematurely converge on ParabR and SharpR, to a lesser degree also on Cigar. In contrast, when using the parameter-specific baseline, we find that the algorithm consistently fails on Ellipsoid and Tablet, while working well on ParabR, SharpR and Cigar. Finally, block fitness baselines are very robust and have not been found to prematurely converge in any of our experiments.

Baseline	α	evaluations	premature convergence
None	0.0	1285 ± 739	52%
None	0.1	1456 ± 533	42%
None	0.2	2011 ± 650	40%
None	1.0	8306 ± 2756	35%
Uniform	0.0	1251 ± 646	50%
Uniform	0.1	1488 ± 650	37%
Uniform	0.2	2060 ± 775	42%
Uniform	1.0	8510 ± 3158	33%
Specific	0.0	1405 ± 866	33%
Specific	0.1	2181 ± 2801	29%
Specific	0.2	2430 ± 1769	27%
Specific	1.0	7973 ± 2407	25%
Block	0.0	1329 ± 662	0%
Block	0.1	1813 ± 725	0%
Block	0.2	2481 ± 919	0%
Block	1.0	8199 ± 2321	0%

Table 1. Average number of evaluations and percentage of runs that prematurely converged, while varying α and the type of fitness baseline used.

Importance mixing is clearly beneficial to performance in all cases, but slightly decreases the robustness. The latter is not an issue when using block fitness baselines, which frees us from the requirement of tuning α, as a value of 0.0 consistently gives the best performance. However, taking into consideration computation time, it can be prudent to use a slightly larger α, for the reasons given in section 5.

For the following experiments, we consistently use block fitness baselines and set $\alpha = 0.01$.

7.3. Performance on Benchmark Functions

We ran NES on the set of unimodal benchmark functions on dimension 50 with batch size 1000, using $\eta = 1.0$ and a target precision of 10^{-10}. Figure 1 shows the average performance over 5 runs for each benchmark function. We left out the Rosenbrock function on which NES is one order of magnitude slower than on the other functions (approximately 2,000,000 evaluations). Presumably this is due to the fact that

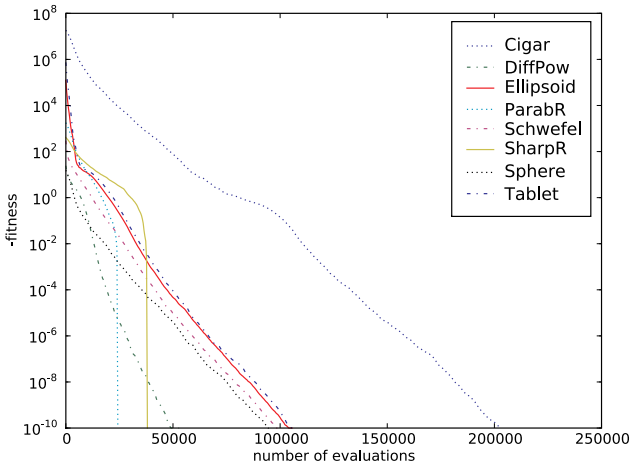

Figure 1. Results for 8 unimodal benchmark functions on dimension 50, averaged over 5 runs.

the principal search direction is updated too slowly on complex curvatures. Note that SharpR and ParabR are unbounded functions, which explains the abrupt drop-off.

7.4. Non-Markovian Double Pole Balancing

Non-Markovian double pole balancing is a challenging 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). We used the implementation as found in (Gomez & Miikkulainen, 1997).

We employ NES to optimize the parameters of the controller of the cart, which is implemented as a simple recurrent neural network, with three inputs (position x and the two poles' angles β_1 and β_2), three hidden sigmoid units, and one output, resulting in a total of 21 weights to be optimized.

An evaluation is considered a *success* iff the poles do not fall over for $100,000$ time steps. Using a batch size of 100, the average number of evaluations until success, over 50 runs, was 1753. Only 6% of the runs did not reach success within the limit of 10000 evaluations. Table 2 shows results of other premier algorithms applied to this task, as reported in the literature. NES clearly outperforms all other methods except CoSyNE.

8. Discussion

Unlike most stochastic search algorithms, NES boasts a relatively clean derivation from first principles. The relationship of NES to methods from other fields, notably evolution strategies (Hansen & Ostermeier, 2001) and policy gradients (Peters & Schaal, 2008; Kakade, 2002), should be evident to readers familiar with both of these domains, as it marries the concept of fitness-based black box optimization from evolutionary methods with the concept of Monte Carlo-based gradient estimation from the policy gradient framework.

Using both a full multinormal search distribution and fitness shaping, the NES algorithm is invariant to translation and rotation and to order-preserving transformations of the fitness function. We empirically showed that fitness baselines significantly improve the algorithm's robustness. We also measured the usefulness of importance mixing, which reduces the number of required fitness evaluations by a factor 5, and renders the algorithm's performance less sensitive to the batch size hyperparameter, because the number of effectively evaluated fitness values in each batch is adjusted dynamically.

Comparing our empirical results to CMA-ES (Hansen & Ostermeier, 2001), considered by many to be the 'industry standard' of stochastic search, we find that NES is competitive but slower on most but not all standard benchmark functions, especially on higher dimensions. On the difficult double-pole balancing benchmark, however, NES yields faster and more robust results. Furthermore, the results in a companion paper (Sun et al., 2009) show that NES is also competitive with CMA-ES on multimodal benchmarks. Our results collectively show that NES can compete with state-of-the-art stochastic search algorithms on standard benchmarks. It holds a lot of promise for ongoing real-world experiments.

Future work will address the problems of automatically determining good batch sizes and dynamically adapting the learning rate. 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). Moreover, realizing that stochastic search based on the natural gradient is not limited to any particular distribution, we can derive the FIM inverse for other (e.g. heavy-tailed) distributions using the same methodology.

Method	SANE	ESP	NEAT	CMA-ES	CoSyNE	FEM	NES
Evaluations	262,700	7,374	6,929	3,521	1,249	2,099	1,753

Table 2. Non-Markovian double pole balancing. Shown are the average numbers of evaluations for SANE (Moriarty & Miikkulainen, 1996), ESP (Gomez & Miikkulainen, 1997), NEAT (Stanley & Miikkulainen, 2002), CMA-ES (Hansen & Ostermeier, 2001), CoSyNE (Gomez et al., 2006), FEM (Wierstra et al., 2008a), and NES.

9. Conclusion

NES constitutes a competitive, theoretically well-founded and relatively simple method for stochastic search. Unlike previous natural gradient methods, NES *quickly* calculates the inverse of the *exact* Fisher information matrix. This increases robustness and accuracy of the search gradient estimation, even in higher-dimensional search spaces. Furthermore, importance mixing prevents unnecessary redundancy embodied by samples from earlier batches. Good results on standard benchmarks and the difficult non-Markovian double pole balancing task affirm the promise of this research direction.

Acknowledgments

We thank Fred Ducatelle for his valuable and timely input. This research was funded by SNF grants 200020-116674/1, 200021-111968/1 and 200021-113364/1.

References

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

Amari, S., Cichocki, A., & Yang, H. (1996). A new learning algorithm for blind signal separation. *Advances in Neural Information Processing Systems*. The MIT Press.

Amari, S., & Douglas, S. C. (1998). Why natural gradient? *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1998.* (pp. 1213–1216 vol.2).

Gomez, F., & Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, *5*, 317–342.

Gomez, F., Schmidhuber, J., & Miikkulainen, R. (2006). Efficient non-linear control through neuroevolution. *Proceedings of the 16th European Conference on Machine Learning (ECML 2006)*.

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

Kakade, S. (2002). A natural policy gradient. In *Advances in neural information processing systems 12*.

Moriarty, D. E., & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, *22*, 11–32.

Peters, J., & Schaal, S. (2008). Natural actor-critic. *Neurocomput.*, *71*, 1180–1190.

Peters, J. R. (2007). *Machine learning of motor skills for robotics*. Doctoral dissertation, Los Angeles, CA, USA. Adviser-Stefan Schaal.

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*, 99–127.

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

Sun, Y., Wierstra, D., Schaul, T., & Schmidhuber, J. (2009). Efficient natural evolution strategies. *To appear in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2009)*.

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

Wierstra, D., Schaul, T., Peters, J., & Schmidhuber, J. (2008a). Fitness expectation maximization. In *Parallel problem solving from nature (ppsn)*.

Wierstra, D., Schaul, T., Peters, J., & Schmidhuber, J. (2008b). Natural evolution strategies. *Proceedings of the Congress on Evolutionary Computation (CEC08), Hongkong*. IEEE Press.