An Adaptive Scheme for Real Function Optimization Acting as a Selection Operator

First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, 2000

Arnaud Berny
IRIN, Université de Nantes
2, rue de la Houssinière
BP 92208
44322 Nantes cedex 3 France
berny@irin.univ-nantes.fr

Abstract- We propose an adaptive scheme for real function optimization whose dynamics is driven by selection. The method is parametric and relies explicitly on the Gaussian density seen as an infinite search population. We define two gradient flows acting on the density parameters, in the spirit of neural network learning rules, which maximize either the function expectation relatively to the density or its logarithm. The first one leads to reinforcement learning and the second one leads to selection learning. Both can be understood as the effect of three operators acting on the density: translation, scaling, and rotation. Then we propose to approximate those systems with discrete time dynamical systems by means of three different methods: Monte Carlo integration, selection among a finite population, and reinforcement learning. This work synthesizes previously independent approaches and intends at showing that evolutionary strategies and reinforcement learning are strongly related.

1 Introduction

In this paper we present a general method for real function optimization. The main idea is to replace the function optimization by the optimization of its expectation relatively to a Gaussian probability measure. This is a parametric approach since we search for an optimal mean and covariance matrix of a Gaussian density. The search is achieved by gradient flows acting on the density parameters. One of the interests of these flows is that their effect can be made equivalent to a selection operator, for example the fitness proportional operator.

This work is also an attempt at bringing together such fields as reinforcement learning and evolutionary strategies in showing that some of their algorithms can be seen as numerical approximations of the above flows. Reinforcement learning is a machine learning technique where the only signal available to a learning agent is a scalar measuring the degree of success or failure of his previous action. The aim of the agent is to maximize this reinforcement signal and one common approach to this problem is to make more likely actions that led to higher reinforcement signals. Reinforcement learning therefore lies between supervised learning techniques such as backpropagation and unsupervised techniques. Reinforcement learning has been applied to associative learning with neural networks [Gul90], and also to function optimization, both in discrete spaces [WP91] and Euclidean spaces [Wil92]. However, although one-dimensional real function optimization by such means has been studied, it has not been generalized in a satisfactory way to Euclidean spaces of arbitrary dimension. We introduce algorithms living in arbitrary dimension that can still be understood as reinforcement learning heuristics.

The second field to which we refer is that of evolutionary strategies [BHS97], especially algorithms implementing correlated mutations [HO97]. In this work, we try to give a theoretical background to such techniques, for which we do not use the term mutation, since we consider that so-called correlated mutations stem from the application of selection to an implicit Gaussian density. We can however reintroduce mutation in a more classical way in the form of noise added to parameters, in the spirit of stochastic algorithms.

In a previous paper [Ber00], we have introduced a Gaussian density in the context of optimization in discrete spaces. We have used a decomposition of the covariance matrix which was not satisfactory and led to numerical instabilities. In the present paper, we use a different decomposition which separates orientation and scaling of the covariance matrix. Such a decomposition has been applied successfully to evolutionary strategies [HO97]. The decomposition relies on orthogonal matrices, which led us to turn to Riemannian gradient flows in the special orthogonal group, that is the group of rotation matrices. Such flows have many different applications in the field of neural networks [DV93], signal processing [CAL98, BA99] or analog computation [Bro88, Won95]. To the best of our knowledge, they have not been applied to continuous function optimization.

The paper is organized as follows. Section 2 introduces a general framework for statistical optimization of real functions. We give the definition of statistical criteria to be optimized and we consider multivariate Gaussian densities as infinite search populations. Section 3 is a derivation of the dynamics acting on each parameter of the Gaussian density leading to three geometric operators: translation, scaling, and rotation. Section 4 presents three iterative methods for approximating those systems, namely, Monte Carlo integration, reinforcement learning, and selection learning. We also address some of the difficulties arising from the approximation of flows on the special orthogonal group. In section 5, we present some early experiments with two simple problems to demonstrate the separate and combined effect of geometric operators.

2 A Statistical Framework for Optimization

2.1 Introduction

The problem is to maximize a (Lebesgue) measurable function $f : \mathbf{R}^n \to \mathbf{R}$, also called a fitness function. The optimal set of f is $U = \{x \in \mathbf{R}^n | f(x) = \sup_{\mathbf{R}^n} f\}$ and we make the assumption that U is non empty. We recast the optimization of the function f as the problem of finding a probability measure over $\mathbf{R}^n$ maximizing the function expectation relatively to the probability measure. Let $M(\mathbf{R}^n)$ be the set of probability measures over $\mathbf{R}^n$. For each probability measure $\mu \in M(\mathbf{R}^n)$, we define the criterion

$$J_r(\mu) = \int f d\mu$$

which is the expectation of f relatively to μ, also written $\boldsymbol{E}_\mu(f)$. The optimal set of J is $V = \{\mu \in M(\mathbf{R}^n) | J(\mu) = \sup_{M(\mathbf{R}^n)} J\}$. An important observation is that Dirac measures charging points in U are in V, and we will look for such solutions. A Dirac measure δ_x is defined by

$$\delta_x(A) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{otherwise,} \end{cases}$$

for each (Lebesgue) measurable set $A \subseteq \mathbf{R}^n$.

The notation J_r indicates that the maximization of J_r by means of gradient dynamical systems leads to reinforcement learning algorithms. We also define two other criteria, J_p and J_b, which lead respectively to proportional and Boltzmann selection learning. Proportional selection learning arises from the fact that J_p is the logarithm of J_r and Boltzmann selection learning is obtained by replacing f by $e^{\beta f}$, where $\beta > 0$ is a control parameter. The definition of J_p requires that f be positive, on the contrary to the definition of J_b, since $e^{\beta f} > 0$. Tab. 1 summarizes all three criteria.

Criterion	Sign of f	Dynamics
$J_r(\mu) = \boldsymbol{E}_\mu(f)$	Any	Reinforcement
$J_p(\mu) = \log \boldsymbol{E}_\mu(f)$	Positive	Proportional Sel.
$J_b(\mu) = \log \boldsymbol{E}_\mu(e^{\beta f})$	Any	Boltzmann Sel.

Table 1: Statistical Criteria.

2.2 The Gaussian Density as an infinite search population

We restrict ourselves to finite dimensional approximations of the problem of finding an element in the set V, or put another way, we only consider probability measures defined by a density function described by a finite number of parameters. The Gaussian density seems to be a natural choice. It is completely described by its mean and covariance matrix and leads to the Dirac measure in the limit where its variances vanish.

For any real-valued n-dimensional vector x, we consider the Gaussian density

$$p(x) = \frac{(\det C)^{1/2}}{(2\pi)^{n/2}} \times \exp\left(-\frac{1}{2}(x-m)^t C(x-m)\right),$$

where C is the inverse of the covariance matrix, m is the mean and $\det C$ is the determinant of C. We will denote by $\mathcal{SO}(n)$ the special orthogonal group , that is the group of $n \times n$ matrices A such that $A^{-1} = A^t$ and $\det A = 1$. Since C is symmetric definite positive, it can be diagonalized using an orthogonal matrix M. We also assume $\det M = 1$ and finally $M \in \mathcal{SO}(n)$ since in case $\det M = -1$, we can bring M back into $\mathcal{SO}(n)$ with an appropriate permutation matrix. We decompose C as $M\Delta^2 M^t$, where Δ is the diagonal matrix of general diagonal element $\delta_i > 0$. Such a decomposition has been previously applied in the context of evolutionary strategies [HO97]. Substituting C in the density, we obtain

$$\begin{aligned} p(x) &= \frac{\det \Delta}{(2\pi)^{n/2}} \\ &\quad \times \exp\left(-\frac{1}{2}(x-m)^t M\Delta^2 M^t (x-m)\right). \quad (1) \end{aligned}$$

We will also use the more general form $p = A \times e^B$, where

$$A = \frac{\det \Delta}{(2\pi)^{n/2}}, \quad B = -\frac{1}{2}(x-m)^t M\Delta^2 M^t (x-m).$$

Any of the three criteria is then a differentiable function of the parameters m, M, and Δ. We have to find some optimal parameters (m^*, M^*, Δ^*) such that $J_r(m^*, M^*, \Delta^*) = \max J_r$. The function $m \mapsto J_r(m, M, \Delta)$ can be regarded as a smoothed version of the fitness function f at scale Δ and orientation M.

At the core of the approximation methods presented in this paper lies the ability to sample from the Gaussian density p. We first assume that we can efficiently sample from a normal Gaussian density (zero mean and identity covariance matrix). Let Z be such a random vector. In order to produce samples from the density p as defined in (1), we introduce the random vector X defined as the image of Z under the affine transformation $T : z \mapsto x = M\Delta^{-1}z + m$. The inverse transformation is $T^{-1} : x \mapsto z = \Delta M^t(x-m)$. By the corresponding change of variable in the density of Z, we obtain the desired result. The transformation T is the composition of three transformations, scaling, rotation, and translation, which can be summarized by the following diagram:

$$z \mapsto \Delta^{-1}z \mapsto M\Delta^{-1}z \mapsto M\Delta^{-1}z + m\,.$$

The complexity of T is thus of order n^2 (multiplication by M). Most samples from the normal Gaussian density lie inside the unit ball and their images under T lie inside the image of the unit ball under T which is an ellipsoid (see Fig. 1).

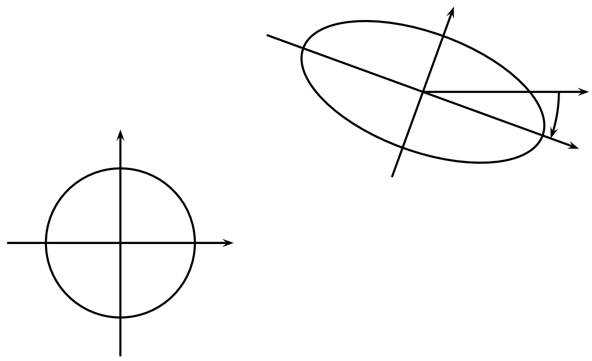


Figure 1: Effect of T on the unit circle in dimension 2.

3 Dynamics

3.1 Introduction

We first consider the criterion J_r. Let θ be one of the parameters of the Gaussian density, that is m, M, or Δ. The main idea is to evolve θ in the direction of the gradient of J_r relatively to θ in order to reach a local optimum of J_r. To compute $\operatorname{grad}_\theta J_r$, we first exchange the gradient operator and the integral, which gives

$$\operatorname{grad}_\theta J_r = \int f(x) \operatorname{grad}_\theta p(x) dx\,. \quad (2)$$

We take advantage of the fact that p is Gaussian and write

$$\begin{aligned} \operatorname{grad}_\theta p(x) &= \operatorname{grad}_\theta \log p(x) \times p(x) \\ &= \left(\frac{\operatorname{grad}_\theta A}{A} + \operatorname{grad}_\theta B\right) \times p(x)\,. \end{aligned}$$

The $p(x)$ term allows us to consider the integral in Eq. (2) as an expectation, which gives

$$\operatorname{grad}_\theta J_r = \boldsymbol{E}_p\left(f(x) \times \operatorname{grad}_\theta \log p(x)\right) .$$

We now consider the criterion $J_p = \log J_r$ whose gradient is simply

$$\frac{\operatorname{grad}_\theta J_r}{J_r}$$

from the derivative of the logarithm. We have thus

$$\operatorname{grad}_\theta J_p = \boldsymbol{E}_p\left(\frac{f(x)}{\boldsymbol{E}_p(f)} \times \operatorname{grad}_\theta \log p(x)\right) .$$

We interpret the operator $\boldsymbol{E}_p\left(\frac{f(x)}{\boldsymbol{E}_p(f)} \times \cdot\right)$ as an expectation operator. Let us introduce the probability density q which is the image of p under the transformation

$$p(dx) \longrightarrow q(dx) = \frac{f(x)\, p(dx)}{\boldsymbol{E}_p(f)} .$$

Since we require f to be positive to define J_p, q is effectively a probability density with total mass 1. We relate the transformation $p \longrightarrow q$ to the fitness proportional selection operator. If we think of p as an infinite population model, then q is the new population resulting from the application of the fitness proportional selection operator. We finally have

$$\operatorname{grad}_\theta J_p = \boldsymbol{E}_q\left(\operatorname{grad}_\theta \log p(x)\right) .$$

We have the following dynamical systems

$$\frac{d\theta}{dt} = \boldsymbol{E}_p\left(f(x) \times \operatorname{grad}_\theta \log p(x)\right) \tag{3}$$

$$\frac{d\theta}{dt} = \boldsymbol{E}_p\left(\frac{f(x)}{\boldsymbol{E}_p(f)} \times \operatorname{grad}_\theta \log p(x)\right) \tag{4}$$

$$\frac{d\theta}{dt} = \boldsymbol{E}_q\left(\operatorname{grad}_\theta \log p(x)\right) , \tag{5}$$

where (4) and (5) are identical but their expressions suggest different approximation techniques, as we will see in Sec. 4. It should be clear that dynamical systems (3) and (4) are equivalent in the sense that they have the same field of directions, hence the equivalence between reinforcement learning and selection learning. In particular, they have the same singular points.

In the remainder of the section, we will derive equations of the form (5) (selection learning) and leave the case of systems (3) and (4) to the reader.

3.2 Translation Operator

We study the case $\theta = m$. The Euclidean gradient of the density relatively to the mean is

$$\begin{aligned}\operatorname{grad}_m \log p(x) &= \operatorname{grad}_m B \\ &= C(x - m) .\end{aligned}$$

System (5) becomes

$$\frac{dm}{dt} = C\left(\boldsymbol{E}_q(x) - m\right) . \tag{6}$$

The effect of system (6) is to make m closer to the mean of the selected population.

If we introduce the operator $\mathcal{L}$ defined by

$$\mathcal{L}(f)(x) = \int \left(f(z) - f(x)\right)(z - x)\, p(dz),$$

we can write system (6) as

$$\frac{dm}{dt} = C\left(\mathcal{L}f\right)(m),$$

which is at the core of the evolutionary algorithm presented in [Sal98], up to the matrix C since the author assumes a normal Gaussian density with identity covariance matrix.

Using the variable $z = T^{-1}(x)$ in the normal space, we further simplify system (6) and obtain

$$\frac{dm}{dt} = M\Delta\boldsymbol{E}_q(z). \tag{7}$$

Apart from the expectation operator, dm/dt can be calculated in a time proportional to n^2.

Singular points of system (7) are such that $\Delta = 0$ or $\boldsymbol{E}_q(z) = 0$. The last condition can also be written $\boldsymbol{E}_q(x) = \boldsymbol{E}_p(x)$, which means that the mean of the density p is invariant under selection.

3.3 Scaling Operator

We study the case $\theta = \Delta$ which we consider as a vector rather than as a matrix. We compute the gradient of $\log p(x)$ componentwise, which gives

$$\begin{aligned}\frac{\partial \log p(x)}{\partial \delta_i} &= \frac{1}{A}\frac{\partial A}{\partial \delta_i} + \frac{\partial B}{\partial \delta_i} \\ &= \frac{1}{\delta_i}\left(1 - z_i^2\right) .\end{aligned}$$

Dynamical system (5) may now be written

$$\frac{d\delta_i^2}{dt} = 1 - \boldsymbol{E}_q\left(z_i^2\right) . \tag{8}$$

These equations generalize the search strategy described in [Wil92]. Their effect is that the search scope is broaden when good solutions are found far from the mean and, inversely, narrowed when good solutions are found close to the mean. Scaling occurs independently along each of the directions defined by the rotation matrix M.

We have not taken into account the constraints $\delta_i > 0$. A simple approach is to introduce n unbounded parameters λ_i, $i = 1, \ldots, n$, such that $\delta_i = e^{\lambda_i}$. With this change of variables, system (8) becomes

$$\frac{d\lambda_i}{dt} = 1 - \boldsymbol{E}_q\left(z_i^2\right) .$$

3.4 Rotation Operator

We study the case $\theta = M$. The first idea is to use the matrix of general element $\frac{\partial J_r}{\partial m_{ij}}$ as a gradient, where m_{ij} is the general element of M. However, the resulting dynamical system will not ensure that M stays orthogonal. This is the reason why we directly consider a differential equation on $\mathcal{SO}(n)$.

We first observe that only B is useful to compute the gradient of J_r relatively to M and we express B as a trace

$$\begin{aligned} B &= -\frac{1}{2}\|z\|^2 \\ &= -\frac{1}{2}\operatorname{Tr}\left(zz^t\right) \\ &= -\frac{1}{2}\operatorname{Tr}\left(\Delta M^t(x-m)(x-m)^t M\Delta\right) \\ &= -\frac{1}{2}\operatorname{Tr}\left((x-m)(x-m)^t M\Delta^2 M^t\right) , \end{aligned}$$

where we have used $\operatorname{Tr}(RS) = \operatorname{Tr}(SR)$. We identify the last expression with the energy function from which Brockett derives isospectral matrix flows [Bro88]. The gradient of B at M can be computed from the difference $B(N) - B(M)$ for $N \in \mathcal{SO}(n)$ in the neighborhood of M. Let $\mathcal{A}(n)$ be the space of $n \times n$ antisymmetric matrices, that is matrices A such that $A^t = -A$. The matrix Me^A, $A \in \mathcal{A}(n)$, is in the neighborhood of M in $\mathcal{SO}(n)$ for A in the neighborhood of O. With $N = Me^A$, one finds

$$\operatorname{grad}_M B = \frac{1}{2}\left[M\Delta^2 M^t,\, (x-m)(x-m)^t\right] M .$$

where $[R, S] = RS - SR$ is the Lie bracket. The above expression is similar to the natural gradient found in [CAL98, BA99] in the context of signal processing. System (5) becomes

$$\frac{dM}{dt} = \left[M\Delta^2 M^t,\, \boldsymbol{E}_q\left((x-m)(x-m)^t\right)\right] M , \tag{9}$$

where we have dropped the $1/2$ factor. Since R and S are symmetric matrices, the Lie Bracket $[R, S]$ is twice the antisymmetric part of the product RS, which ensures that $M_t \in \mathcal{SO}(n)$, for all $t \in \mathbf{R}$, provided that $M_0 \in \mathcal{SO}(n)$.

If we are interested in the singular points of system (9), we may rewrite it as

$$\frac{dM}{dt} = \left[C, \boldsymbol{E}_q\left((x-m)(x-m)^t\right)\right] M.$$

Since $M \in \mathcal{SO}(n)$, the only singular points are points at which the Lie bracket vanishes, or put another way, points at which C and C_q commute, where $C_q = \boldsymbol{E}_q\left((x-m)(x-m)^t\right)$. As we have seen in Sec. 3.2, if we exclude the case $\Delta = 0$, $m = \boldsymbol{E}_q(x)$ at singular points, hence C_q is exactly the covariance matrix of q. Let $C_p = C^{-1}$ be the covariance matrix of p. Then, at singular points, C_p^{-1} and C_q commutes and the same holds for C_p and C_q. Provided that C_p and C_q have distinct eigenvalues, it can be shown that C_p and C_q have the same eigenvectors and consequently define the same orientation. The conclusion is that the covariance matrix of the density p is invariant under selection at singular points of system (6).

4 Numerical Approximation

All approximation schemes presented in this section iteratively update m, M, and Δ with discrete time recurrent equations after a sampling step and an estimation step. We will successively review Monte Carlo estimation, selection learning, and reinforcement learning. We first adopt a general point of view which is suitable for m or Δ, but not for M.

4.1 Monte Carlo Estimation

This method consists in sampling N individuals from the density p and averaging $f(x) \times \mathrm{grad}_\theta \log p(x)$ and $f(x)$ to yield a direct estimation of the right side of Eq. (4). This is a classical Monte Carlo estimation method.

Let (x^i), $i = 1, \ldots, N$, be N independently and identically distributed (iid) individuals with common density p. Using this population, we simply update θ with

$$\theta_{t+1} = \theta_t + \alpha \sum_{i=1}^{N} \frac{f(x^i)}{\sum_{j=1}^{N} f(x^j)} \mathrm{grad}_\theta \log p(x^i) ,$$

where $0 < \alpha < 1$ is the learning rate. With such an approximation scheme, updating complexity is $Nn + n^2$ for m and Nn for Δ.

4.2 Reinforcement Learning

This is the numerical approximation which has been used in [Ber00] in the case of optimization over the space of fixed-length binary strings. It is close to classical reinforcement learning rules such as those appearing in [Wil92] and also to stochastic gradient techniques [Duf96].

We first recall that $\mathrm{grad}_\theta\, p(x) = \mathrm{grad}_\theta \log p(x) \times p(x)$, which we have already used in the derivation of the systems. In the context of reinforcement learning, $\mathrm{grad}_\theta \log p(x)$ is called the eligibility of x and, roughly speaking, is a measure of the influence of x on the evolution of the density parameters. We choose a stochastic approximation of (3) which only involves a single sample and compensates this weakness by means of a comparison technique. The updating rule is

$$\theta_{t+1} = \theta_t + \alpha \left(f(x_t) - b_t \right) \times \mathrm{grad}_\theta \log p(x_t) ,$$

where x_t is the current sample and b_t is a first order linear estimation of the expectation $\boldsymbol{E}_p(f)$, that is,

$$b_{t+1} = \gamma b_t + (1 - \gamma) f(x_t) ,$$

where $0 < \gamma < 1$. In the context of reinforcement learning, b_t is called the baseline. The complexity requirements of the reinforcement learning approach are lower than those of population-based approximations: n^2 for m, and n for Δ.

4.3 Selection Learning

Expression of system (5) suggests an estimation by sampling from the density q which results from proportional selection of the density p. From a practical point of view, at each iteration, this method starts with the sampling of N individuals from the density p. Each individual is given a weight proportional to its fitness. Then P new individuals are sampled from the initial population with the empirical reweighted law. This new population is used to estimate expectations relatively to q.

Let (x^i), $i = 1, \ldots, N$, be N iid vectors with common density p. Let

$$\hat{p} = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^i}$$

be their empirical measure. We then reweight $\hat{p}$ according to the fitness of each sample and define the empirical measure

$$\hat{q} = \sum_{i=1}^{N} \frac{f(x^i)}{\sum_{j=1}^{N} f(x^j)} \delta_{x^i} .$$

We sample P new individuals (y^i), $i = 1, \ldots, P$, independently with common law $\hat{q}$, which is equivalent to applying a fitness proportional selection operator to the population (x^i), $i = 1, \ldots, N$.

Using the selected population, we simply update θ with

$$\theta_{t+1} = \theta_t + \alpha \times \left(\frac{1}{P} \sum_{i=1}^{P} \mathrm{grad}_\theta \log p(y^i) \right) .$$

From the definition of system (5) and its approximation, we finally come to the conclusion that PBIL-like algorithms [BC95, GFD99], in discrete or continuous domains, are actually stochastic approximations of gradient dynamical systems. This can be made more explicit by considering Boltzmann selection instead of proportional selection. The reweighted empirical measure is then

$$\tilde{q} = \sum_{i=1}^{N} \frac{e^{\beta f(x^i)}}{\sum_{j=1}^{N} e^{\beta f(x^j)}} \delta_{x^i} .$$

In the limit $\beta \to \infty$, $\tilde{q}$ is uniform over the set of individuals with the highest fitness. If there is only one best individual x^*, then the parameter is updated with the rule

$$\theta_{t+1} = \theta_t + \alpha \ \mathrm{grad}_\theta \log p(x^*) \ ,$$

which is very close to the original PBIL algorithm.

We have simplified equations with the help of the affine transformation T and expressed systems with normal variable z rather than with variable x. There is thus a need for denoting the selected population in the normal space. Let (z^i), $i = 1, \ldots, N$, be the normal population and (ζ^i), $i = 1, \ldots, P$, be the selected population in the normal space. We have the relations $x^i = T(z^i)$, $i = 1, \ldots, N$, and $y^i = T(\zeta^i)$, $i = 1, \ldots, P$, which can be summed up by the following diagram:

$$\begin{tikzcd}
(z^i) \arrow[r, "T"] \arrow[d, "f \circ T\text{-selection}"'] & (x^i) \arrow[d, "f\text{-selection}"] \\
(\zeta^i) \arrow[r, "T"] & (y^i)
\end{tikzcd}$$

Wherever the variable z appears in a quantity $\mathrm{grad}_\theta \log p(x)$, we use individuals (ζ^i) rather than (y^i). The complexity requirements are basically those of the Monte Carlo method (replacing N by P), but one should also take into account the complexity arising from selection.

4.4 Adding Mutation

In the context of stochastic algorithms, one usually adds a noise term to updating rules. The objective is twofold: on one hand, it makes convergence to hyperbolic singular points impossible, and on the other hand, it allows to escape local maxima to some extent. The second part of the objective leads to annealing strategies controlling the variance of the noise, for example starting with a high variance and gradually decreasing it [Duf96]. Such techniques can be applied to our updating rules and can be seen as the effect of a mutation operator. The simplest choice as a mutation noise for both m and Δ is an additive centered Gaussian random process with a covariance matrix of the form $\sigma^2 I$, where I is the $n \times n$ identity matrix. In that case, the mutation is applied to the whole population p, or put another way, directly to m and Δ after selection and updating. In the case of M, one should be aware of keeping M in $\mathcal{SO}(n)$. We propose to perturb, before updating, each individual in the selected normal population (ζ^i) with an additive centered Gaussian random noise of the same form as in the case of m and Δ. With this mutation scheme, the resulting matrix M will still lie in $\mathcal{SO}(n)$.

4.5 Approximating the Rotation Operator

First order discrete time recurrent equations are not adapted to approximate dynamical systems of the form (9) evolving rotation matrices. The problem is to ensure $M \in \mathcal{SO}(n)$ at each time step. We first simplify system (9) using the affine transformation T, which gives

$$\frac{dM}{dt} = \boldsymbol{E}_q \left((M\Delta z) \left(\Delta^{-1} z\right)^t - \left(M \Delta^{-1} z\right) (\Delta z)^t \right) \ ,$$

where algebraic complexity is of order n^2. However, the approximation scheme increases the complexity. The above system can be written

$$\frac{dM}{dt} = MA \, ,$$

where A is the antisymmetric matrix

$$\boldsymbol{E}_q \left((\Delta z) \left(\Delta^{-1} z\right)^t - \left(\Delta^{-1} z\right) (\Delta z)^t \right) \ .$$

We first consider that A is constant over a time interval of length α which corresponds to the learning rate. Under this assumption, the updating rule of M is

$$M_{t+1} = M_t e^{\alpha A} \, .$$

We need an approximation of the matrix exponential $e^{\alpha A}$. We have chosen the Crank-Nicholson method [PTVF92] which approximates the exponential by the orthogonal matrix

$$\left(1 - \frac{\alpha}{2} A\right)^{-1} \left(1 + \frac{\alpha}{2} A\right) \ .$$

Finally, M is updated with the rule

$$M_{t+1} = M_t \left(1 - \frac{\alpha}{2} A\right)^{-1} \left(1 + \frac{\alpha}{2} A\right) \ .$$

The drawback is that a matrix inversion is required.

5 Experiments

We have carried out some experiments to demonstrate the effect of each operator (translation, scaling, rotation) acting on a Gaussian density in dimension 2. We are not trying to compete with state-of-the-art evolutionary algorithms or other numerical techniques.

The test function landscape is a cylinder, that is

$$f(x) = \begin{cases} 1 & \text{if } \|x\| < 1, \\ 0 & \text{otherwise,} \end{cases}$$

where $x \in \mathbf{R}^2$. In each of the three following experiments, only one operator is acting on the density, or put another way, only one parameter of the Gaussian density, m, Δ, or M, is evolved, the other ones are kept constant. We have chosen a reinforcement learning approximation algorithm with $\gamma = 0.9$. Table 2 gives initial states of the system and the value of simulation parameters. Every 400 iteration, we have drawn the ellipse corresponding to the current parameters of the Gaussian density, which is the image of the unit circle. The results are shown on Fig. 2.

Operator	m_0	Δ_0	Gain	Iterations
Translation	$\begin{pmatrix} 2 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	10^{-3}	10000
Scaling	$\begin{pmatrix} 2 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	10^{-3}	30000
Rotation	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} e^2 & 0 \\ 0 & 1 \end{pmatrix}$	5×10^{-4}	30000

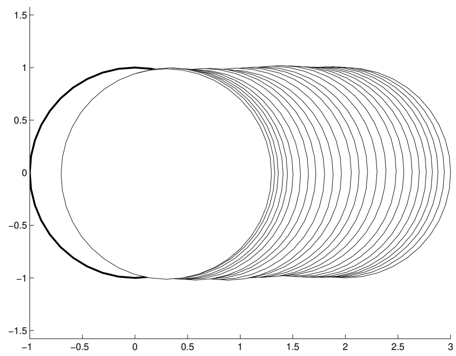
Table 2: Initial states of the system and simulation parameters. The initial orientation matrix M_0 is the identity matrix.

In a second experiment, we have chosen the quadratic test function $f(x) = -x_1^2 - 2x_2^2$ to demonstrate the effect of the three operators acting together with the following initial conditions:

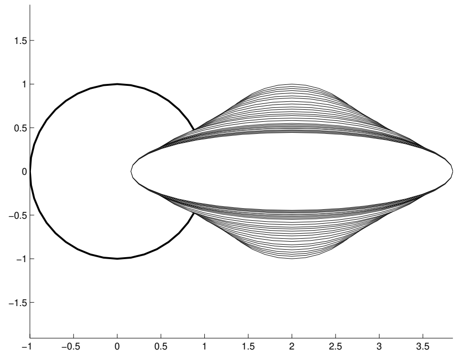
$$M_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \quad m_0 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad \Delta_0 = \begin{pmatrix} 1 & 0 \\ 0 & e \end{pmatrix}.$$

Fig. 3 shows successive ellipses corresponding to successive density parameters and level curves of the fitness function.

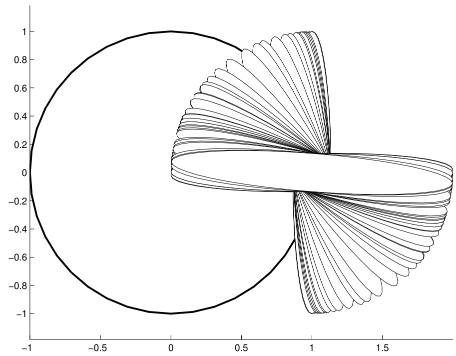
It should be clear that the effect of rotation is useful only if the change in the expected fitness does not vanish: the shape of the Gaussian density is symmetric and it will not evolve in order to be oriented along the



(a) Translation



(b) Scaling



(c) Rotation

Figure 2: Operators acting separately.

gradient of the fitness function since any gain would be compensated by a symmetric loss. Rotation may then take a limited part in the success of the search, which also holds true for the scaling operator.

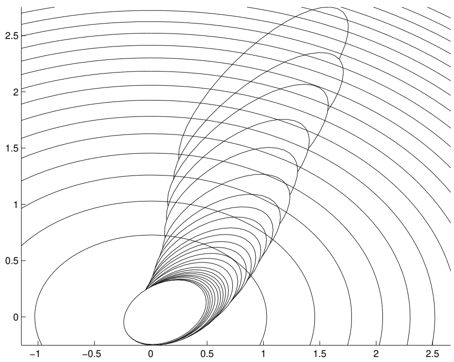


Figure 3: Operators acting together.

6 Conclusion

We have presented a general adaptive method for continuous real function optimization in arbitrary dimension. It relies on the maximization of statistical criteria relatively to a multivariate Gaussian density seen as an infinite search population. The main criteria are the expectation of the fitness function and its logarithm. Maximizing the former leads to reinforcement learning and maximizing the latter to proportional selection learning. Boltzmann selection has also been considered. To search for optimal density parameters, we have derived gradient flows acting on those parameters which locally maximize one of the criteria. Since selection and reinforcement learning dynamical systems are defined by the same field of directions, they are equivalent.

We have first worked with deterministic dynamical systems and only turned to stochastic ones for approximation purposes. We have reviewed three approximation methods, namely, Monte Carlo integration, reinforcement learning, and selection learning.

One of the objectives of this work was to keep updating complexity as low as possible. However, approximating gradient flows on the special orthogonal group requires a matrix inversion, whose complexity is of order n^3. The aim is now to look for more efficient approximation schemes. Complexity requirements can nevertheless be kept at a reasonable level if the rotation operator is not applied at each time step.

Future work consists in the study of the asymptotic behavior of the dynamical systems, both deterministic and stochastic ones. One desirable property would be the convergence of the Gaussian density to a Dirac measure located at locally maximal points. Also of interest is the experimental study of the algorithms with benchmark functions, and their comparison with other optimization methods.

Bibliography

[BA99] J. Basak and S. Amari. Blind separation of uniformly distributed signals: a general approach. *IEEE transactions on neural networks*, 10(5):1173–1185, September 1999.

[BC95] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In *Proceedings of the 12th Annual Conference on Machine Learning*, pages 38–46, 1995.

[Ber00] A. Berny. Statistical machine learning and combinatorial optimization. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, Lecture Notes in Natural Computing. Springer-Verlag, 2000.

[BHS97] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, April 1997.

[Bro88] R.W. Brockett. Dynamical systems that sort lists, diagonalize matrices and solve linear programming problems. In *Proceedings of the 27th conference on decision and control*, pages 799–803, Austin, Texas, December 1988.

[CAL98] T. Chen, S.I. Amari, and Q. Lin. A unified algorithm for principal and minor components extraction. *Neural Networks*, 11(3):385–390, 1998.

[Duf96] M. Duflo. *Algorithmes stochastiques*. Springer-Verlag, 1996.

[DV93] J. Dehaene and J. Vanderwalle. Locally implementable learning with isospectral matrix flows. In M. Verleysen, editor, *Proceedings of the European symposium on artificial neural networks*, pages 189–194, April 1993.

[GFD99] M. Gallagher, M. Frean, and T. Downs. Real-valued evolutionary optimization using a flexible probability density estimator. In *GECCO' 99*, pages 840–847, 1999.

[Gul90] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3:671–692, 1990.

[HO97] N. Hansen and A. Ostermeier. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: the $(\mu/\mu_I, \lambda)$-CMA-ES. In *Proceedings of the 5th European congress on intelligent techniques and soft computing*, pages 650–654, 1997.

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C*. Cambridge University Press, second edition, 1992.

[Sal98] R. Salomon. Evolutionary algorithms and gradient search: similarities and differences. *IEEE Transactions on Evolutionary Computation*, 2(2):45–55, July 1998.

[Wil92] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[Won95] W.S. Wong. Matrix representation and gradient flows for NP-hard problems. *Journal of Optimization Theory and Applications*, 87(1):197–220, October 1995.

[WP91] R.J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.