Foundations and Recent Advances on Natural Evolution Strategies

Tutorial @ CEC'23

Masahiro Nomura (Tokyo Institute of Technology)

Tutorial Goals

We will explain Natural Evolution Strategies (NES)

- NES = BBO framework that uses the ***natural*** gradient to update distribution params.
- Natural gradient = gradient accounting for parameter space geometry

The goals of this tutorial are to enable the participants:

- to understand core ideas and promising variants of NES
- to know recent advances for practical challenges
- to use NES algorithms in research and/or industrial applications
- to recognize future challenges in this area

Tutorial Outline

Part 1: Introduction to NES

Part 2: Theoretical Foundations

Part 3: Recent Advances

Part 4: Practical Guide

Part 5: Conclusion

Part 1:
Introduction to NES

Black-Box Optimization on Continuous Domain

This tutorial: focus on continuous black-box optimization

Goal of black-box optimization (BBO):

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmax}}\ f(x)$$

- X is continuous space

- Objective function is "**black-box**"
  - Not assume availability of gradient
  - We only use function evaluation value

Stochastic Relaxation for BBO [WSG+14]

- We want to optimize f(x), but the gradient wrt. x is not available
- Instead of directly finding x*, NES searches the distribution parameter θ*:

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmax}}\ f(x)$$

Stochastic Relaxation

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\ \mathbb{E}_{x \sim p(x;\theta)}[f(x)] =: J(\theta)$$

Gradient for Distribution Parameter [WSG+14]

- Suppose we can find $\nabla_\theta \log p(x;\theta)$
- By using the so-called 'log-likelihood trick'*:

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta \int f(x)p(x;\theta)dx \\ &= \int f(x)\nabla p(x;\theta)dx \\ &= \int f(x)\nabla \log p(x;\theta)p(x;\theta)dx \\ &= \mathbb{E}_{x\sim p(x;\theta)}[f(x)\nabla_\theta \log p(x;\theta)]\end{aligned}$$

* Interchangeability of integration and differentiation is assumed

Gradient for Distribution Parameter [WSG+14]

- Suppose we can find $\nabla_\theta \log p(x;\theta)$
- By using the so-called 'log-likelihood trick'*:

$$\nabla_\theta J(\theta) = \nabla_\theta \int f(x)p(x;\theta)dx$$

Exchange of integration and differentiation

$$= \int f(x)\nabla p(x;\theta)dx$$

$$= \int f(x)\nabla \log p(x;\theta)p(x;\theta)dx$$

$$= \mathbb{E}_{x\sim p(x;\theta)}[f(x)\nabla_\theta \log p(x;\theta)]$$

* Interchangeability of integration and differentiation is assumed

Gradient for Distribution Parameter [WSG+14]

- Suppose we can find $\nabla_\theta \log p(x;\theta)$
- By using the so-called 'log-likelihood trick'*:

$$\nabla_\theta J(\theta) = \nabla_\theta \int f(x)p(x;\theta)dx$$

$$= \int f(x)\nabla p(x;\theta)dx$$

$$= \int f(x)\nabla \log p(x;\theta)p(x;\theta)dx$$

$$\nabla \log p(x;\theta) = \frac{\nabla p(x;\theta)}{p(x;\theta)}$$

$$= \mathbb{E}_{x\sim p(x;\theta)}[f(x)\nabla_\theta \log p(x;\theta)]$$

* Interchangeability of integration and differentiation is assumed

Gradient for Distribution Parameter [WSG+14]

- Suppose we can find $\nabla_\theta \log p(x;\theta)$
- By using the so-called 'log-likelihood trick'*:

$$\nabla_\theta J(\theta) = \nabla_\theta \int f(x)p(x;\theta)dx$$

$$= \int f(x)\nabla p(x;\theta)dx$$

$$= \int f(x)\nabla \log p(x;\theta)p(x;\theta)dx$$

$$= \mathbb{E}_{x\sim p(x;\theta)}[f(x)\nabla_\theta \log p(x;\theta)]$$

cannot calculate analytically
⇒ appriximation

* Interchangeability of integration and differentiation is assumed

Stochastic Relaxation for BBO: Basic Algorithm

For i = 1, ...,λ: (λ is population size)

- generate solution x_i from the distribution p(x_i; θ)
- evaluate x_i on objective function f
- calculate derivative of log-likelihood on x_i

Approximate the gradient: $$\nabla_\theta J(\theta) \approx \frac{1}{\lambda} \sum_{i=1}^{\lambda} f(x_i) \nabla_\theta \log p(x_i; \theta)$$

Update the distribution parameter: $$\theta \leftarrow \theta + \eta \cdot \nabla_\theta J(\theta)$$

Multivariate Gaussian Distribution

- One can use any distribution whose derivative of log-likelihood is obtained
- Most widely used one: multivariate Gaussian distribution (MGD)
- Distribution parameters:
  - Mean vector : $m \in \mathbb{R}^d$
  - Covariance matrix: $\Sigma \in \mathbb{R}^{d \times d}$
- Density of the distribution:

$$p(x;\theta) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \cdot \exp\left(-\frac{1}{2}(x-m)^\top \Sigma^{-1}(x-m)\right)$$

Vanilla gradient for Multivariate Gaussian Distribution

- We first calculate the log-likelihood for estimating gradient:

$$\log p(x;\theta) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det(\Sigma) - \frac{1}{2}(x-m)^\top \Sigma^{-1}(x-m)$$

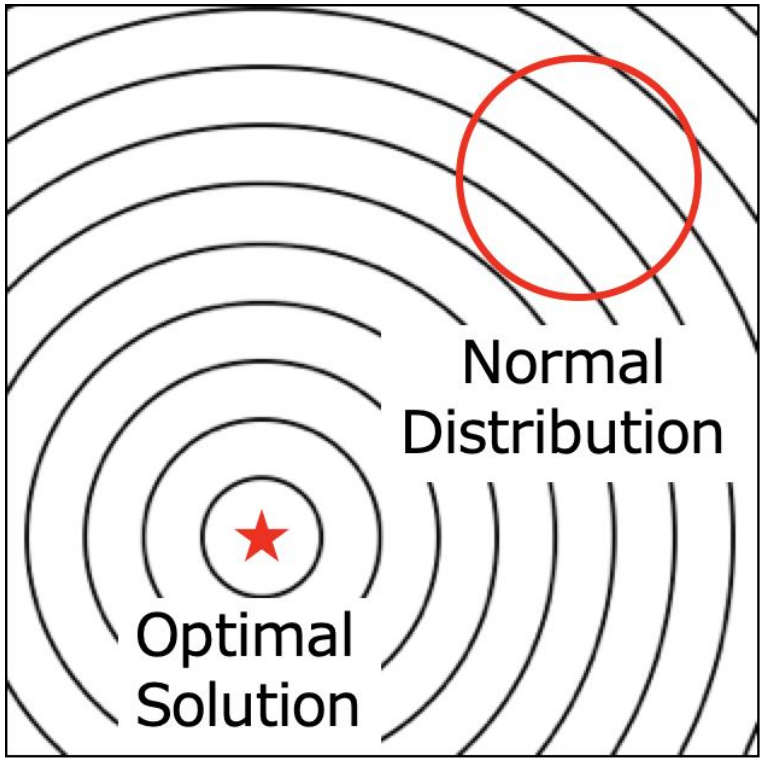
- (Vanilla) gradient for the log-likelihood:

$$\nabla_m \log p(x;\theta) = \Sigma^{-1}(x-m)$$

$$\nabla_\Sigma \log p(x;\theta) = \frac{1}{2}\Sigma^{-1}(x-m)(x-m)^\top - \frac{1}{2}\Sigma^{-1}$$

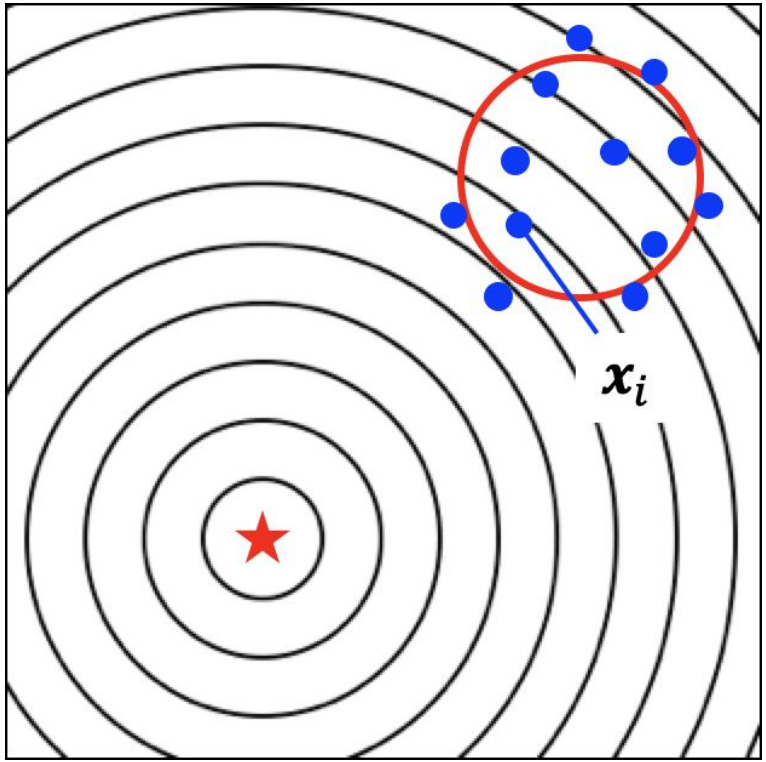
Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. Updates the parameters
4. Repeats until the criterion is met



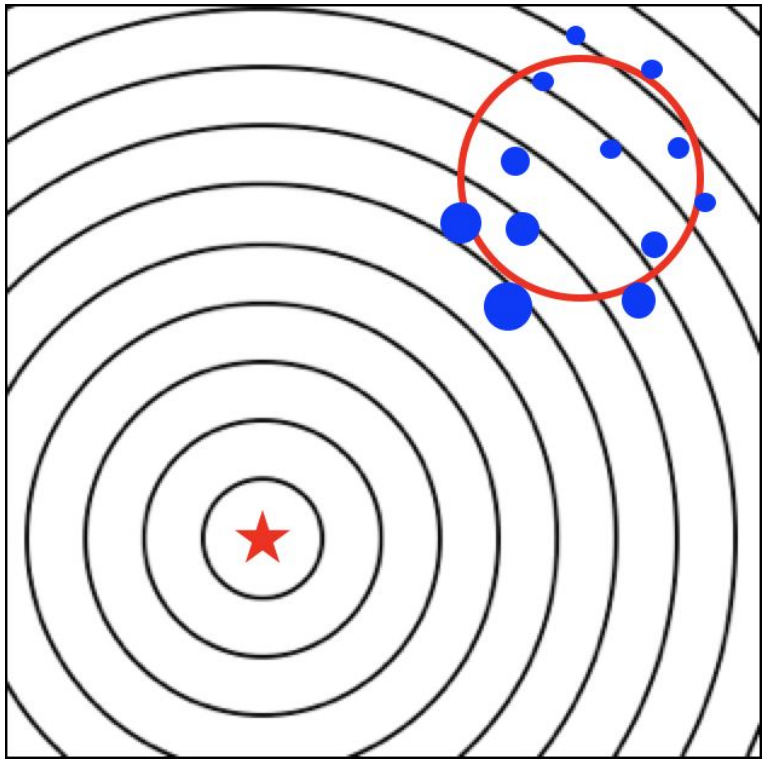
Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. Updates the parameters
4. Repeats until the criterion is met



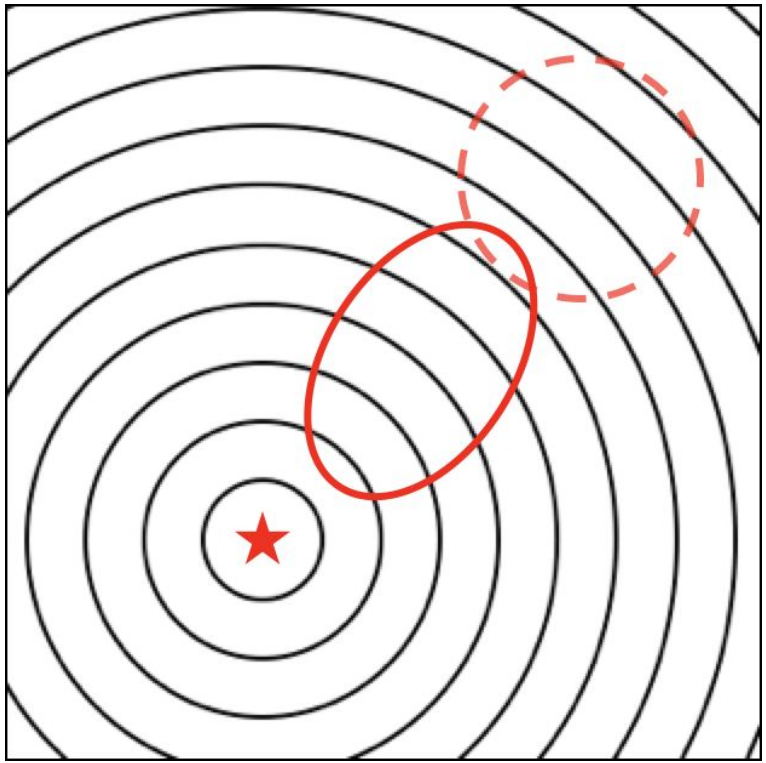
Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. Updates the parameters
4. Repeats until the criterion is met



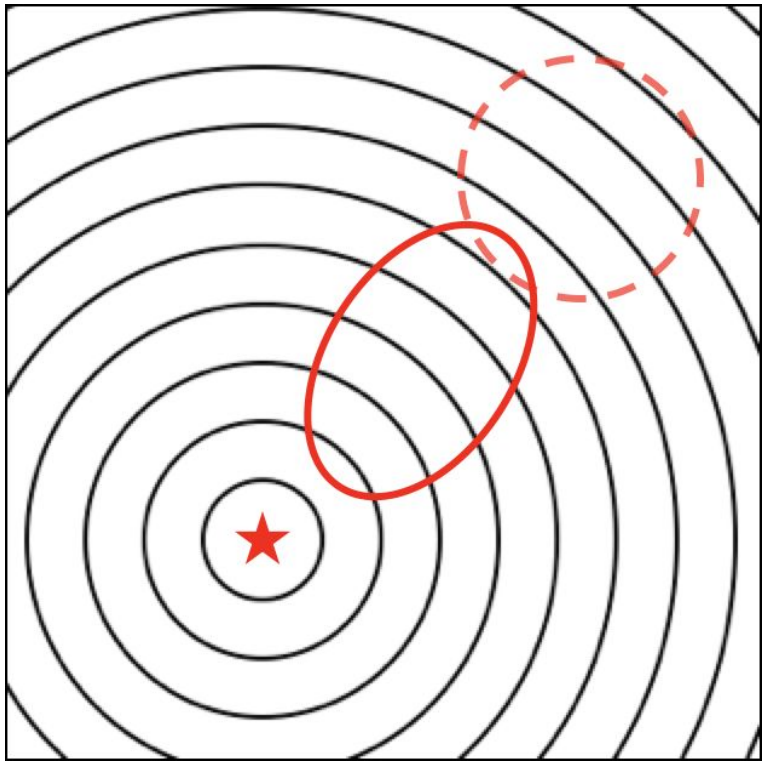
Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. Updates the parameters
4. Repeats until the criterion is met



Illustrative Behavior with Multivariate Gaussian (MGD)

1. Generates solutions from the MGD
2. Evaluates solutions and estimates gradient
3. Updates the parameters
4. Repeats until the criterion is met



Critical Issue of Vanilla Gradient

- Rethinking vanilla gradient:
  - The vanilla gradient can be characterized as the steepest ascent within the small **Euclidean distance** in the parameter space

$$\Delta_\theta^* = \lim_{\epsilon \to 0} \underset{\|\Delta_\theta\| = \epsilon}{\operatorname{argmax}} J(\theta + \Delta_\theta)$$

  - The update depends on the choice of parameterization
- Actually, the vanilla gradient can show unstable and unsatisfying performance*

More natural distance (divergence) in probability distribution space is needed

* this depends on the choice of parameterization.
In some cases, vanilla gradient and natural gradient coincides (*local coordinate*)

Natural Gradient: Motivation

- Natural gradient [AD98,Ama98]:
  - considering parameter geometry in probability space
- Natural gradient can be formalized as the solution of constrained optimization:

$$\max_{\Delta_\theta} J(\theta + \Delta_\theta) \approx J(\theta) + \nabla_\theta J(\theta)^\top \Delta_\theta,$$
$$\text{s.t.} \quad D_{\mathrm{KL}}(p_{\theta+\Delta_\theta} || p_\theta) = \epsilon$$

  - point: Kullback-Leibler (KL) divergence rather than Euclidean distance
    - KL div. is commonly used to measure the diff. of distributions

Relation between KL divergence and FIM

- Fisher information matrix (FIM) is the curvature of the KL divergence:

$$D_{\mathrm{KL}}(p_{\theta+\Delta_\theta}||p_\theta) = \frac{1}{2}\Delta_\theta^\top F \Delta_\theta + \mathcal{O}(\|\Delta_\theta\|^3) \ (\Delta_\theta \to 0)$$

where *F* is the FIM given θ:

$$F = \mathbb{E}_{x\sim p(x;\theta)}[\nabla_\theta \log p(x;\theta) \nabla_\theta \log p(x;\theta)^\top]$$

- Now, we rewrite the constrained optimization problem using FIM

Form of Natural Gradient [WSG+14]

- We want to solve the following constrained optimization problem:

$$\max_{\Delta_\theta} J(\theta) + \nabla_\theta J(\theta)^\top \Delta_\theta \quad \text{s.t.} \quad \frac{1}{2}\Delta_\theta^\top F \Delta_\theta = \epsilon$$

- Using Lagrangian multiplier,

$$\mathcal{L}(\theta, \Delta_\theta, c) = J(\theta) + \nabla_\theta J(\theta)^\top \Delta_\theta - c\left(\frac{1}{2}\Delta_\theta^\top F \Delta_\theta - \epsilon\right)$$

$$\nabla_{\Delta_\theta}\mathcal{L}(\theta, \Delta_\theta, c) = \nabla_\theta J(\theta) - cF\Delta_\theta$$

$$\therefore F\Delta_\theta^* = c^{-1}\nabla_\theta J(\theta)$$

- If F is invertible, the natural gradient $\tilde{\nabla}_\theta J(\theta)$ is given by Δ_θ^*; thus, (ingnoring const.,)

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

Fitness Shaping [GSY+10]

- Fitness shaping: replace function values with *weights* $(w_1 \geq \cdots \geq w_\lambda)$

$$F^{-1}\nabla_\theta J(\theta) \approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} f(x_i) F^{-1}\nabla_\theta \log p(x_i;\theta)$$

Fitness Shaping

$$\sum_{i=1}^{\lambda} w_i F^{-1}\nabla_\theta \log p(x_{i:\lambda};\theta)$$

$x_{i:\lambda}$: *i*-th best solution

NES uses only ranking rather than function value itself

- It makes NES invariant under monotonically increasing transformation of *f*
- Moreover, effect of speeding up is theoretically suggested [Bey14]

FIM for Multivariate Gaussian Distribution

In natural gradient method, calculating the FIM inverse is needed

For MGD, FIM (inverse) can be calculated as follows [ANOK10]:

$$F = \begin{bmatrix} \Sigma^{-1} & O \\ O & \frac{1}{2}\Sigma^{-1} \otimes \Sigma^{-1} \end{bmatrix} \text{ and } F^{-1} = \begin{bmatrix} \Sigma & O \\ O & 2\Sigma \otimes \Sigma \end{bmatrix}$$

$\otimes$: Kronecker product

By combining FIM inverse and the vanilla gradient (and fitness shaping),

$$m^{(t+1)} = m^{(t)} + \eta_m \sum_{i=1}^{\lambda} w_i (x_{i:\lambda} - m^{(t)})$$

$$\Sigma^{(t+1)} = \Sigma^{(t)} + \eta_\Sigma \sum_{i=1}^{\lambda} w_i ((x_{i:\lambda} - m^{(t)})(x_{i:\lambda} - m^{(t)})^\top - \Sigma^{(t)})$$

Another: choose a coordinate system such that the FIM becomes Identity matrix *I*

'local' coordinate system

Exponential Parameterization [GSY+10]

- Caution: need to ensure that cov. remains positive definite
  - However, the updated cov. may not be positive definite matrix

- Elegant fix: using exponential map under (natural) *local coordinates*

Let $\Sigma = AA^\top$

$$A^{(t+1)} = A^{(t)} \cdot \exp\left(\frac{\eta_\Sigma}{2} \sum_{i=1}^{\lambda} w_i (z_{i:\lambda} z_{i:\lambda}^\top - I)\right)$$

where $z_{i:\lambda} = A^{(t)^{-1}}(x_{i:\lambda} - m^{(t)})$

We can always obtain a feasible covariance matrix

Decomposing Covariance into Step-Size and Shape

- We decompose the A as $A = \sigma B$
  - step-size: $\sigma \in \mathbb{R}_+$
  - normalization transformation matrix: $B \in \mathbb{R}^{d \times d}$
- We can update each parameter separately:

$$\sigma^{(t+1)} = \sigma^{(t)} \cdot \exp\left(\frac{\eta_\sigma}{2} \cdot \mathrm{Tr}(G_M)/d\right)$$

$$B^{(t+1)} = B^{(t)} \cdot \exp\left(\frac{\eta_B}{2} \cdot \left(G_M - \frac{\mathrm{Tr}(G_M)}{d} I\right)\right)$$

where $G_M = \sum_{i=1}^{\lambda} w_i (z_{i:\lambda} z_{i:\lambda} - I)$

- $\det(B) = 1$ ⇒ We can have a clear interpretation:
  - σ controls volume (step-size), B controls shape

xNES (Exponential NES) [GSY+10]

Step 1: Sampling and Evaluation (i=1,...,λ)

$$z_i \sim \mathcal{N}(0, I), \quad x_i = m^{(t)} + \sigma^{(t)} B^{(t)} z_i$$

Step 2: Estimating Natural Gradient with Fitness Shaping

$$G_\delta = \sum_{i=1}^{\lambda} w_i z_{i:\lambda}, \quad G_M = \sum_{i=1}^{\lambda} w_i (z_{i:\lambda} z_{i:\lambda} - I)$$

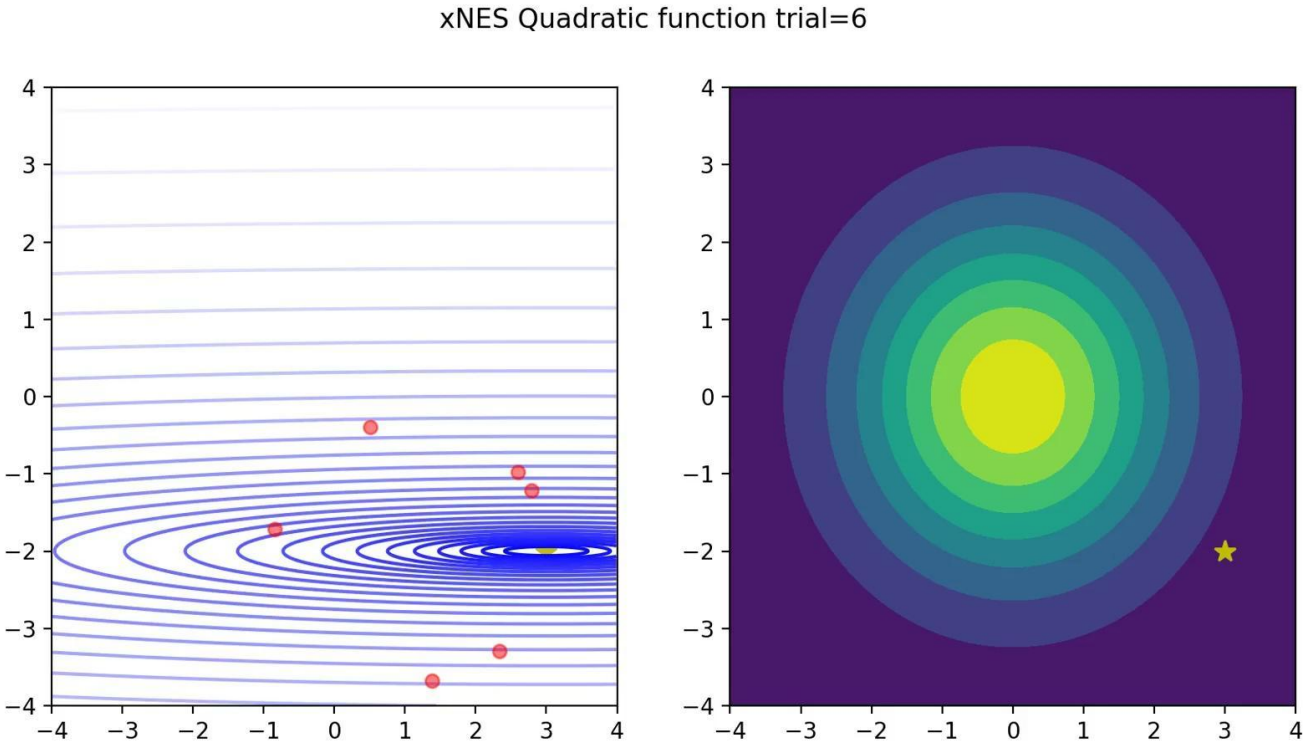
$$G_\sigma = \mathrm{Tr}(G_M)/d, \quad G_B = G_M - G_\sigma \cdot I$$

Step 3: Updating of Distribution Parameters

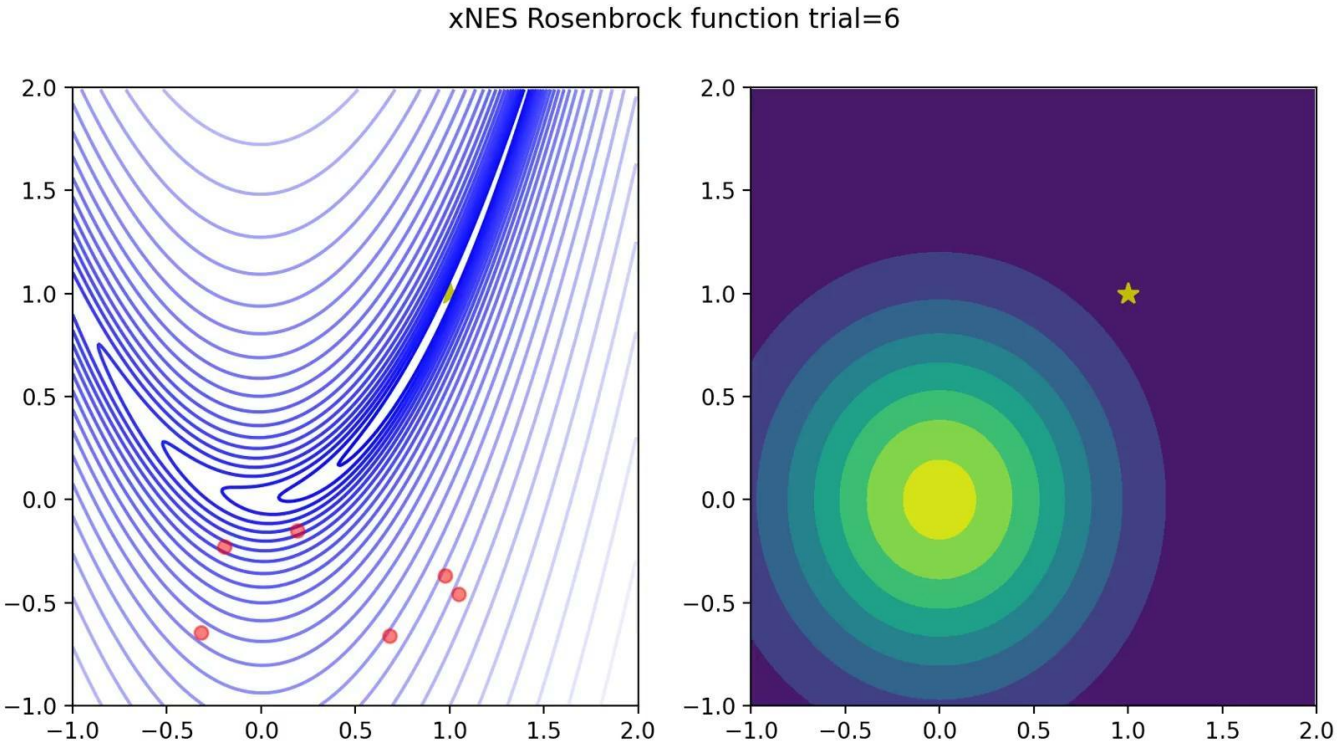
$$m^{(t+1)} = m^{(t)} + \eta_m \sigma^{(t)} B^{(t)} G_\delta, \quad \sigma^{(t+1)} = \sigma^{(t)} \cdot \exp(\eta_\delta / 2 \cdot G_\delta),$$

$$B^{(t+1)} = B^{(t)} \cdot \exp(\eta_B / 2 \cdot G_B)$$

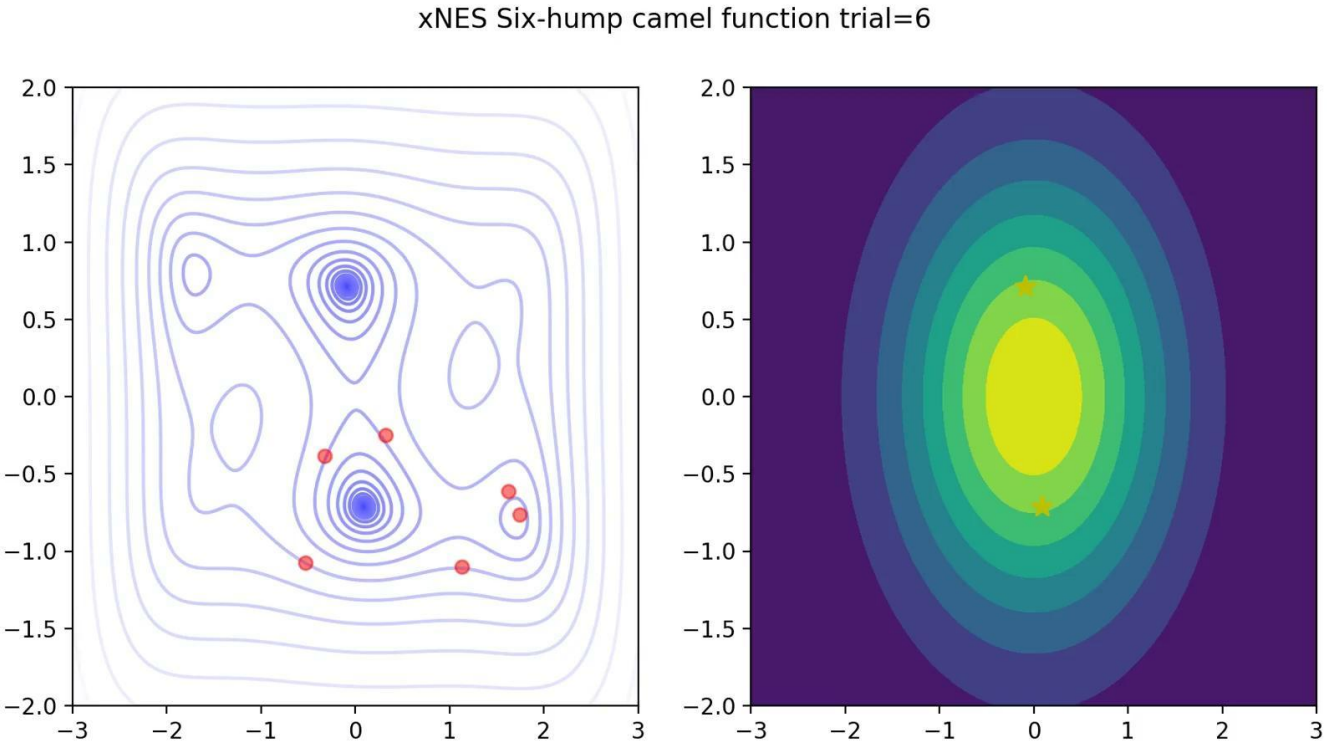
xNES Behavior: Quadratic function



xNES Behavior: Rosenbrock function



xNES Behavior: Six-hamp camel function



Comparison to Estimation of Distribution Algorithms

Both: iteratively

- sample points from the distribution and evaluate them
- update the distribution by using the evaluated points

Differences

- EDA: maximum likelihood estimation to fit distribution of sampled points
- NES: natural gradient method for expected function value

- While design principles are different, update equations sometimes coincide

Part 2:
Theoretical Foundations

Rethinking NES in view of Natural Gradient Method

- Derivation of NES started with the natural gradient for expected objective:

$$J(\theta) = \mathbb{E}_{x \sim p(x;\theta)}[f(x)]$$

- With fitness shaping, it no longer makes sense as the original natural gradient

$$f(x_{1:\lambda})/\lambda, \cdots, f(x_{\lambda:\lambda})/\lambda \Longrightarrow w_1, \cdots, w_\lambda$$

  - **Can NES no longer be considered a natural gradient method?**

- NES estimates the natural gradient for another objective
  - This is formalized by **Information Geometric Optimization**

Information Geometric Optimization [OAAH17]

- Information Geometric Optimization (IGO):
  - stochastic optimization framework that uses probability distribution
  - IGO formalizes several popular BBO methods
    - e.g., NES, rank-μ CMA-ES, PBIL, cGA
- From IGO perspective, we can derive NES as a natural gradient method

IGO: Formulation

- IGO objective: $J_{\theta^{(t)}}(\theta) = \mathbb{E}_{x \sim p(x;\theta)}[W^f_{\theta^{(t)}}(x)]$

Objective function depends on the current parameter $\theta^{(t)}$

- Quantile function q(x) for weight*:

$$q^{\geq}_{\theta^{(t)}}(x) = \Pr_{y \sim p(y;\theta^{(t)})}[f(y) \geq f(x)]$$

0 for the best
1 for the worst

- With a non-increasing function $w : [0,1] \to \mathbb{R}$, weight function is defined:

$$W^f_{\theta^{(t)}}(x) = w(q^{\geq}_{\theta^{(t)}}(x))$$

* Assuming Lebesgue measure of level-set of objective is 0, which is common in continuous optimization

IGO: Natural Gradient

- Natural gradient of IGO: Let us denote $\tilde{\nabla}_\theta \log p(x;\theta) = F^{-1}\nabla_\theta \log p(x;\theta)$

$$\tilde{\nabla}_\theta J_{\theta^{(t)}}(\theta) = \mathbb{E}_{x\sim p(x;\theta)}[W^f_{\theta^{(t)}}(x)\tilde{\nabla}_\theta \log p(x;\theta)]$$

$$\approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} W^f_{\theta^{(t)}}(x)\tilde{\nabla}_\theta \log p(x;\theta)$$

- We also need to estimate the weights $W^f_{\theta^{(t)}}(x) = w(q^{\geq}_{\theta^{(t)}}(x))$

  q(x) is the prob. of sampling a solution that is better than x =>

$$W^f_{\theta^{(t)}}(x) \approx w((\mathrm{rk}(x) - 1/2)/\lambda)$$ rk() is the ranking of x wrt. f

Other Distributions

- IGO principle can be applied to other distributions
- For many distributions belonging to the exponential family, (inverse of) FIM can be computed in a closed form (i.e. without estimating FIM)
  - Bernoulli distribution (PBIL, cGA)
  - MGD (xNES, rank-μ update CMA-ES)
  - Categorical distribution
- There are cases where estimating FIM is necessary (e.g. restricted Boltzmann machines; RBM)

xNES is an Instance of IGO

- Let $\Sigma = AA^\top$
- When parameterizing the Gaussian distribution by

$$\theta = (m, R), \quad R = \ln(A^{-1}\Sigma(A^\top)^{-1})$$

R is a kind of 'local' coordinate

- IGO Update:

$$m^{(t+1)} = m^{(t)} + \eta_m \sum_{i=1}^{\lambda} w_i (x_{i:\lambda} - m^{(t)})$$

$$R^{(t+1)} = \eta_\Sigma \sum_{i=1}^{\lambda} w_i (z_{i:\lambda} z_{i:\lambda}^\top - I)$$

- By transforming it to cov. in 'global' coordinate, i.e., $\Sigma^{(t+1)} = A^{(t)} \exp(R^{(t+1)}) A^{(t)\top}$ we obtain the same update as xNES

CMA-ES [HMK03,Han16]: Quick Overview

- one of the most powerful methods for BBO
- also uses MGD parameterized by N(m, σ^2 C)
  - m: d-dim mean vector, σ: (scalar) step-size, C: pos. def. matrix
- Update for each parameter:
  - m: **weighted recombination (natural gradient update)**
  - σ: specific method such as cumulative step-size adaptation
  - C: **rank-μ update (natural gradient update)** & rank-one update
- CMA-ES = **natural gradient update** + step-size adaptation + rank-one update

the relationship with the natural gradient was discovered after the development of CMA-ES [ANOK10]

Rank-μ update CMA-ES = natural gradient update
(combination of weighted recombination & rank-μ update)

Rank-μ Update CMA-ES is an Instance of IGO

- Natural gradient Update with MGD:

$$m^{(t+1)} = m^{(t)} + \eta_m \sum_{i=1}^{\lambda} w_i (x_{i:\lambda} - m^{(t)})$$

$$\Sigma^{(t+1)} = \Sigma^{(t)} + \eta_\Sigma \sum_{i=1}^{\lambda} w_i ((x_{i:\lambda} - m^{(t)})(x_{i:\lambda} - m^{(t)})^\top - \Sigma^{(t)})$$

- This precisely corresponds to the rank-μ update CMA-ES
  - The natural gradient plays a crucial role in CMA-ES

Cautions regarding Natural Gradient Update

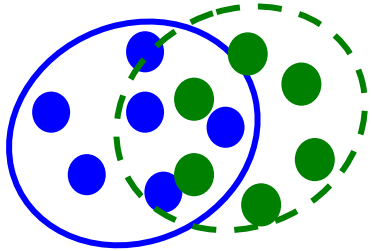
- The natural gradient does **not** explain everything in the efficiency
- The step-size adaptation & rank-one update in the CMA-ES are **not** explained from natural gradient perspective, but they drastically improve performance
  - evolution path (i.e. a sum of consecutive) is quite effective, especially for small population size setting
- Recognizing the limitations of the natural gradient method contributes to the development of practically useful algorithms

Part 3:
Recent Advances

Improving Natural Gradient Estimation by Sample Reuse

- Usually, IGO algorithms *discard* solutions after evaluation once
  - This is a waste, so can't we reuse such old samples in some way?
- IGO includes two components that should be estimated
  - Quantile (prob. of sampling a better point than x)
  - Natural gradient
- For this purpose, **importance sampling** has been utilized [SAOO15]

current samples
& distribution



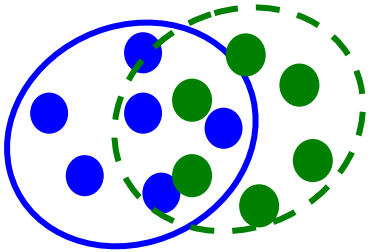
old samples
& distributions

Estimation Target and Monte-Carlo Estimation

- Estimation target: $\mathbb{E}_{x\sim p(x;\theta^{(t)})}[g(x)]$
  - for quantile $q^{\geq}_{\theta^{(t)}}(x_i^{(t-k)})$: $g(x) = \mathbb{1}\{f(x) > f(x_i^{(t-k)})\} \quad (\because \mathbb{E}[\mathbb{1}\{\cdot\}] = \Pr[\cdot])$
  - for natural gradient : $g(x) = W^f_{\theta^{(t)}}(x)\tilde{\nabla}_\theta \log p(x;\theta^{(t)})$

current samples & distribution

old samples & distributions



- (Naive) Monte-Carlo estimation:

$$\frac{1}{\lambda}\sum_{i=1}^{\lambda} g(x_i^{(t)})$$

  - This uses only current samples, and **discards old samples**

Sample Reuse via Importance Sampling [SAOO15]

- Importance sampling: utilize obs. in previous *K* iterations

current distribution

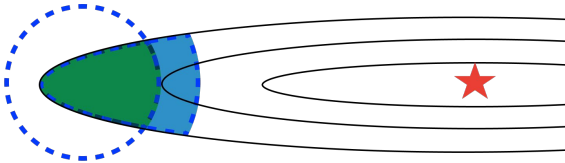
$$\frac{1}{\lambda(K+1)} \sum_{k=0}^{K} \sum_{i=1}^{\lambda} g(x_i^{(t-k)}) \frac{p(x_i^k; \theta^{(t)})}{p(x_i^k; \theta^{(t-k)})}$$

old distributions

- Assign large values to samples that seem to be drawn from the p(θ^t)
  - old samples that are ***similar*** to current => ***utilized***
  - old samples that are ***far*** to current => ***ignored***
- More sophisticated estimator was also used in [SAOO15]

Performance Issues of xNES

- Performance of xNES degrades on ill-conditioned problems
- Why?
  - More promising regions are located inside the distribution
  - As a result, step-size tends to be shrinked
- This is reasonable considering the improvement of the function value in **one step**



- But our ultimate goal is to improve the quality of the solution obtained by repeating this process, not necessarily the result in a single step

Practical Improvements for xNES

- To address the inefficiency, several improvements have been proposed:

- **Distance weight** [FNKO11]:
  - Modifying weight function to speed-up moving and enlarging distribution in the direction of promising regions
- **Emphasis on distribution expansion** [NSFO21]:
  - Faster adaptation of the distribution by emphasizing the enlargement in the direction in which it occurs

How to Design Weight Function?

- Weights are often used in NES and CMA-ES:
  - For the upper solution, $w_i \propto \log\left(\frac{\lambda}{2}+1\right) - \log(i)$
  - For the lower solution, values such as 0 or -1/λ are used
- Quality gain analysis [Arn05,AAH20]:
  - analyze expected improvement of function value at one step
  - optimal weight is similar to the above weight* $\Sigma = \sigma^2 I$
    - **Note:** the optimal weight is derived for **isotropic** Gaussian distribution
    - Not clear whether the weight is good for **'general'** Gaussian

* the result is obtained on (infinite dimensional) sphere [Arn05] and convex quadratic [AAH20] functions

DX-NES: Distance Weight [FNKO11]

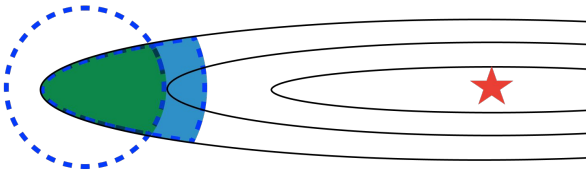
- New weighting strategy for 'general' Gaussian distribution
- **Core idea**: Moving mean vector and/or enlarging distribution in the direction of outer promising distribution

$$w_i^{\mathrm{dist}} \propto \hat{w}_i \exp(\alpha \| A^{-1}(x - m) \|)$$

① Mahalanobis distance from mean

$$\hat{w}_i = \max\left(0, \log\left(\frac{\lambda}{2} + 1\right) - \log(i)\right)$$

② assign 0 for bad solutions by 'max(0,・)' operation



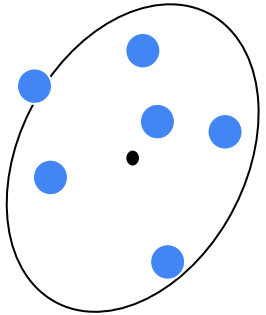
Give larger weight to solutions generated in the outer promising region (**blue** region)

Mirrored sampling

Naive Sampling

- All λ samples are generated i.i.d (i=1,...,λ)

$$z_i \sim \mathcal{N}(0, I), \quad x_i = m + \sigma B z_i$$

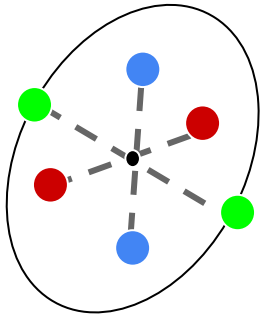


Mirrored Sampling

- Generates pairs of solutions symmetrically with respect to the mean vector (i=1,...,λ/2)

$$z_{2i-1} \sim \mathcal{N}(0, I), \; z_{2i} = -z_{2i-1}$$

$$x_{2i-1} = m + \sigma B z_{2i-1}, \; x_{2i} = m + \sigma B z_{2i}$$



Additional Discussions on Mirrored Sampling

- Empirical mean of samples matches the mean vector
  - may enable accurately estimating natural gradient wrt mean
    - DX-NES(-IC) shows superior performance in multimodal problems
  - more detailed investigation is interesting work

- Note: direct application for CMA-ES is not straightforward
  - causes bias for step-size adaptation [ABH11]
  - debiasing (e.g. by modifying strategy parameters [GK20]) is needed

Emphasizing Expansion [NSFO21]

- Key observation: Shrink is typical, but expansion is relatively rare
  - To fully utilize the information of expansion, we emphasize it!
- We quantify the amount of change in each direction of distribution

  Second moment change (e_i: eigenvector, i=1,..,d):

$$\tau_i = \frac{e_i^\top (B^{(t+1)} B^{(t+1)\top} - B^{(t)} B^{(t)\top}) e_i}{e_i^\top B^{(t)} B^{(t)\top} e_i}$$

$\tau_i > 0$ => distribution has been **expanded** in the direction of e_i

Emphasizing Expansion [NSFO21] (2)

- Expansion matrix Q: (γ is hyperparameter)

$$Q := \gamma \sum_{i=1}^{d} \mathbb{1}\{\tau_i > 0\} e_i e_i^\top + I$$

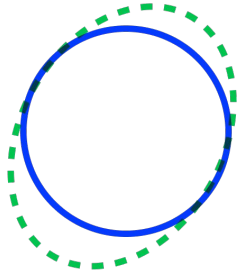
- Emphasizing expansion is performed:

$$B^{(t+1)} \leftarrow Q B^{(t+1)} / \sqrt[d]{\det(Q)}$$

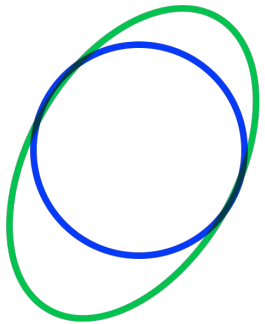
$$\sigma^{(t+1)} \leftarrow \sigma^{(t+1)} \cdot \sqrt[d]{\det(Q)}$$

To keep det(B)=1, normalization is performed

(blue) before cov.
(green) after cov.



Emphasizing Expansion



for High-Dimensional Problems

- The complexity:
  - time complexity*: $\mathcal{O}(d^3 + \lambda d^2)$ (decomposition, update cov.)
  - space complexity: $\mathcal{O}(d^2 + \lambda d)$ (matrix, population)
  - difficult to apply it for high-dimensional (e.g. 1000) problems
- Main reason of this complexity:
  - # of params. (i.e. degree of freedom) of covariance is $\mathcal{O}(d^2)$
- Basic policy: reducing # of params. of covariance (i.e. restricting the representation of covariance)

* This complexity is reduced to O(λd^2) if we decompose covariance at every O(d/λ) iteration

Pros. and Cons. of Restricted Representation

- Advantage:
  - reducing (time and space) complexity
  - making learning efficient
    - according to # params. O(d^2), learning rate is set to O(1/d^2)
    - reduction of # params. enables to set higher learning rate
- Disadvantage:
  - variable dependencies cannot be captured if restriction is not sufficient
    - optimization can fail or be inefficient on problems with strong variable dependencies

Use of Diagonal Covariance Matrix [RH08]

- Simple and effective form of covariance: Diagonal representation

$$\Sigma = D$$ (D is diagonal matrix)

- Natural gradient corresponds to extracting the diagonal components of the natural gradient in the case of full covariance matrix
  - This update corresponds to rank-μ update of *Sep-CMA** scalable variant
- Efficient for ill-conditioned problems
- **Limitation**: variable dependencies are completely ignored in cov.

* In Sep-CMA, Σ = σ^2 D is used because step-size adaptation is performed

VD-CMA [AAH14]

- VD-CMA represents the covariance matrix using 2d parameters:

$$\Sigma = \sigma^2 D(I + vv^\top)D$$

d-dim vector diagonal matrix

- Thanks to vector v, *VD-CMA partially addresses variable dependencies*
- v and D are updated by natural grad.* (+ rank-one update and step-size adapt.):

$$v^{(t+1)} = v^{(t)} + \eta_v \tilde{\nabla}_v J_{\theta^{(t)}}(\theta)$$

$$D^{(t+1)} = D^{(t)} + \eta_D \tilde{\nabla}_D J_{\theta^{(t)}}(\theta)$$

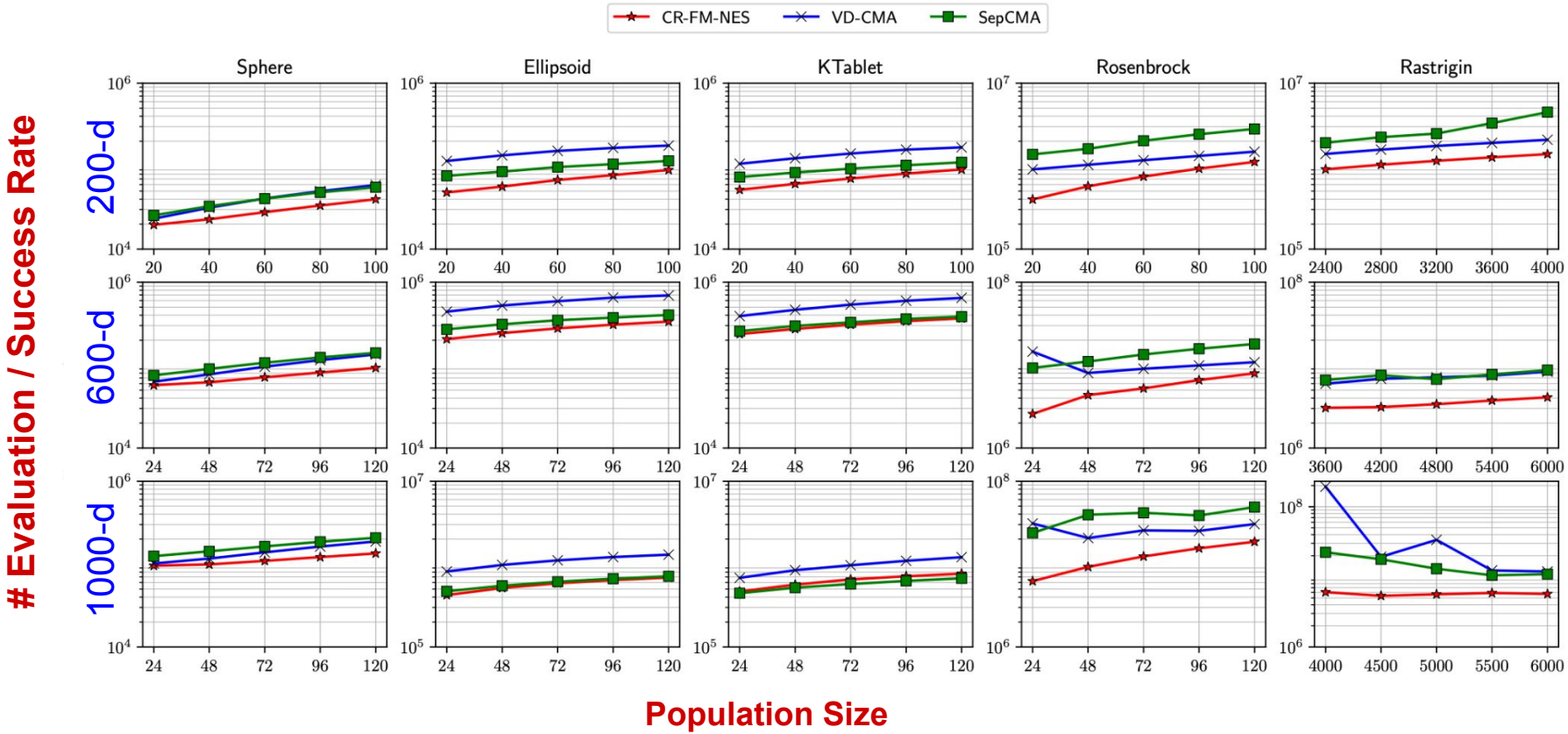
All computations are performed in **linear** time O(d)

* To avoid singularity of the FIM, a modified FIM is used in VD-CMA

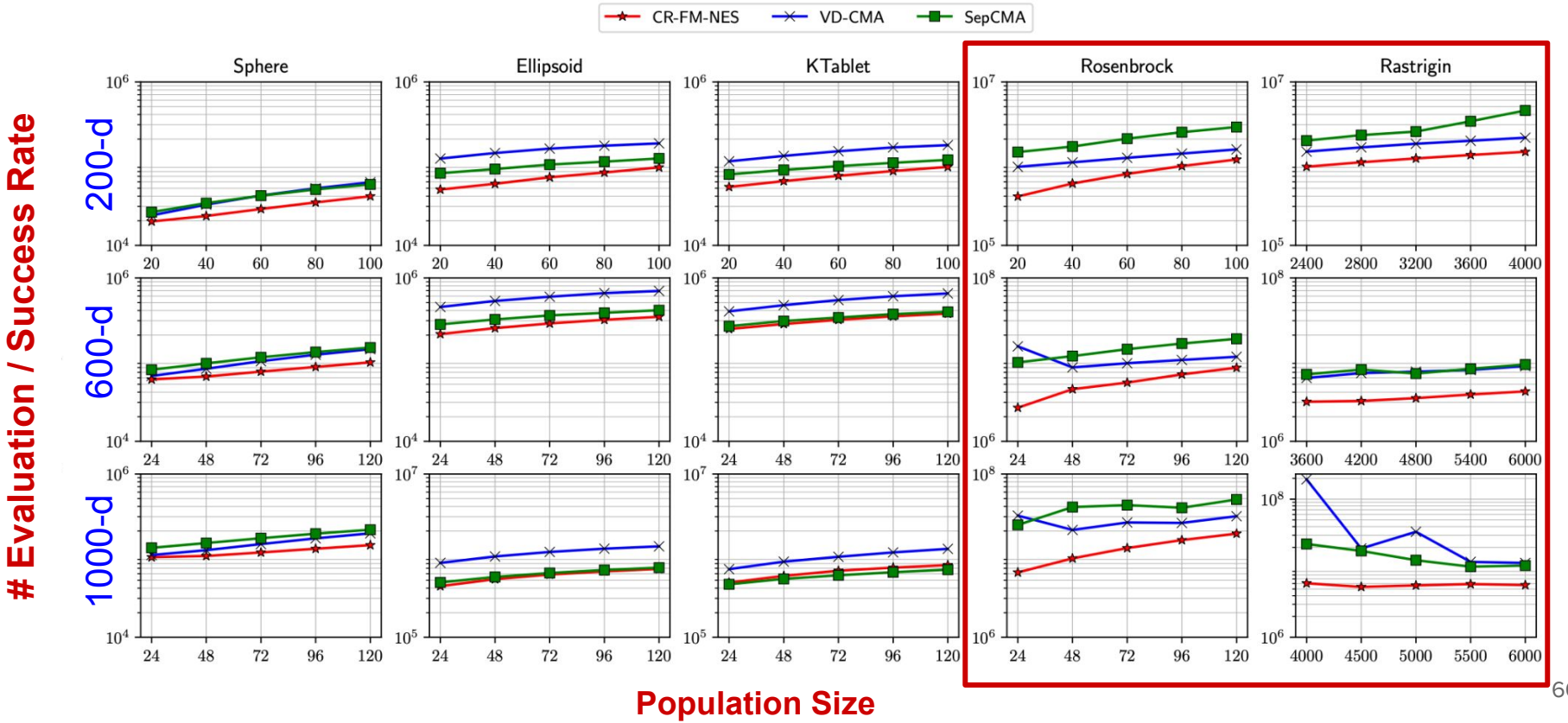
CR-FM-NES [NO22]

- CR-FM-NES uses the same representation as VD-CMA
  - all computations are also performed in linear time
- Instead of step-size adaptation for σ, natural gradient update is used
- Several recent improvements are included
  - distance weight
  - mirrored sampling
  - rank-one update

Comparison: CR-FM-NES vs. VD-CMA vs. Sep-CMA



Comparison: CR-FM-NES vs. VD-CMA vs. Sep-CMA



Towards Completely Hyperparameter-Free Methods

- Internal parameters often have recommended values
  - They are set to ensure they perform well across a wide range of problems
  - However, in ***difficult*** problems (e.g. multimodal and noisy), these settings are often insufficient
- Increasing population size often helps to solve such difficult problems
  - It is difficult to know the appropriate values beforehand in BBO scenarios
  - Tuning can be expensive, so it would be preferable to avoid

Can we adapt the hyperparameter online?

Learning Rate Adaptation vs. Population Size Adaptation

- Important observation: Increasing population size has effect similar to decreasing mean-vector learning rate [MA17]
  - CMA-ES with a small population size solves multimodal problems, through (fixed) appropriate setting of learning rate
- **Learning rate adaptation** vs. **population size adaptation**
  - Learning rate adaptation is more practically useful
  - E.g. parallel implementation (population size may be set to # of workers)
- Introduction of latest LRA [NAO23] (GECCO'23, best paper nominated)
  - While this focus LRA for CMA, but same principle is applicable to NES

Learning Rate Adaptation: Setup

- Notations:
  - distribution parameters: $\theta_m = m, \theta_\Sigma = \mathrm{vec}(\Sigma)$
    - "vec" is the vectorization operator
  - original updates: $\Delta_m^{(t)} = m^{(t+1)} - m^{(t)}, \Delta_\Sigma^{(t)} = \mathrm{vec}(\Sigma^{(t+1)} - \Sigma^{(t)})$
  - learning rate factors: $\eta_m^{(t)}, \eta_\Delta^{(t)}$
- Modified updates for mean and covariance:
  - $\theta_m^{(t+1)} = \theta_m^{(t)} + \eta_m^{(t)} \Delta_m^{(t)}$
  - $\theta_\Sigma^{(t+1)} = \theta_\Sigma^{(t)} + \eta_\Sigma^{(t)} \Delta_\Sigma^{(t)}$

Learning Rate Adaptation: Main Concept

- We adapt the learning rate based on the signal-to-noise ratio (SNR):

$$\mathrm{SNR} := \frac{\|\mathbb{E}[\Delta]\|_F^2}{\mathrm{Tr}(F\,\mathrm{Cov}[\Delta])} = \frac{\|\mathbb{E}[\Delta]\|_F^2}{\mathbb{E}[\|\Delta\|_F^2] - \|\mathbb{E}[\Delta]\|_F^2}$$

F: Fisher information matrix

- **Insight**: in noisy problems, when noise becomes dominant, SNR -> 0
  - To improve function value, maintaining a positive SNR is crucial
  - If we consider (well-structured) multimodal problems as a kind of noisy problems, similar arguments can be applied
- We keep SNR = αη (α: hyperparameter)

Learning Rate Adaptation: Main Concept (2)

- Assume LR is sufficiently small over *n* iterations
  - param. don't change much ⇒ updates are considered to be i.i.d
- *n* steps update:

$$\theta^{(t+n)} = \theta^{(t)} + \eta \sum_{k=0}^{n-1} \Delta^{(t+k)}$$

$$\approx \theta^{(t)} + \mathcal{D}\left(n\eta\mathbb{E}[\Delta], n\eta^2 \mathrm{Cov}[\Delta]\right)$$

- *n* = *1/η* & small *η* ⇒ more concentrated update than *η=1*
- The SNR over *n* iterations: $\frac{\|\mathbb{E}[\Delta]\|_F^2}{\eta \,\mathrm{Tr}(F\,\mathrm{Cov}[\Delta])} = \frac{1}{\eta}\mathrm{SNR}$
- Keeping SNR = *αη* ...
  - Keeping the SNR as *α* over *n* = *1/η* iterations **independently** of *η*

Learning Rate Adaptation: SNR Estimation

- We introduce moving averages for each m and Σ

$$\mathcal{E}^{(t+1)} = (1-\beta)\mathcal{E}^{(t)} + \beta\tilde{\Delta}^{(t)},$$
$$\mathcal{V}^{(t+1)} = (1-\beta)\mathcal{V}^{(t)} + \beta\|\tilde{\Delta}^{(t)}\|_2^2$$

local coordinate

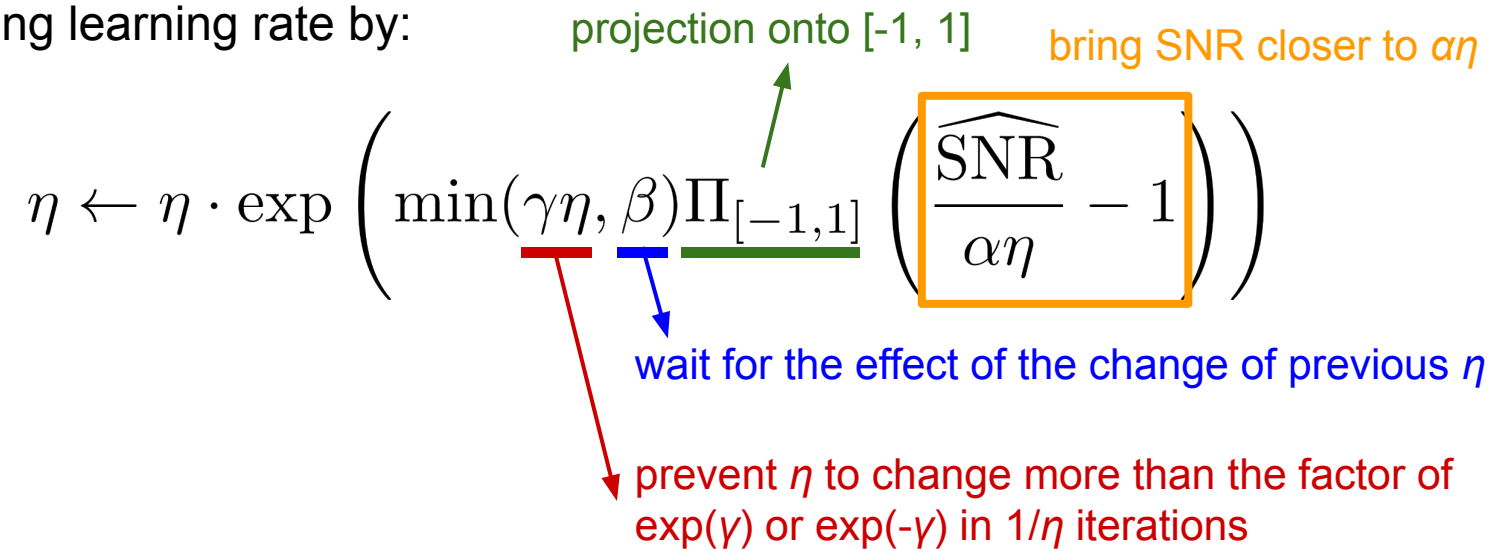
$$\tilde{\Delta}_m = \sqrt{\Sigma}^{-1}\Delta_m$$
$$\tilde{\Delta}_\Sigma = 2^{-\frac{1}{2}}\mathrm{vec}(\sqrt{\Sigma}^{-1}\mathrm{vec}^{-1}(\Delta_\Sigma)\sqrt{\Sigma}^{-1})$$

- The SNR is estimated as:

$$\mathrm{SNR} := \frac{\mathbb{E}[\tilde{\Delta}]^2}{\mathrm{Tr}(\mathrm{Cov}[\tilde{\Delta}])} = \frac{\mathbb{E}[\tilde{\Delta}]^2}{\mathbb{E}[\|\tilde{\Delta}\|^2] - \|\mathbb{E}[\tilde{\Delta}]\|^2},$$
$$\approx \frac{\|\mathcal{E}\|_2^2 - \frac{\beta}{2-\beta}\mathcal{V}}{\mathcal{V} - \|\mathcal{E}\|_2^2} =: \widehat{\mathrm{SNR}}$$

Update Equation of Learning Rate Adaptation

- Adapting learning rate by:

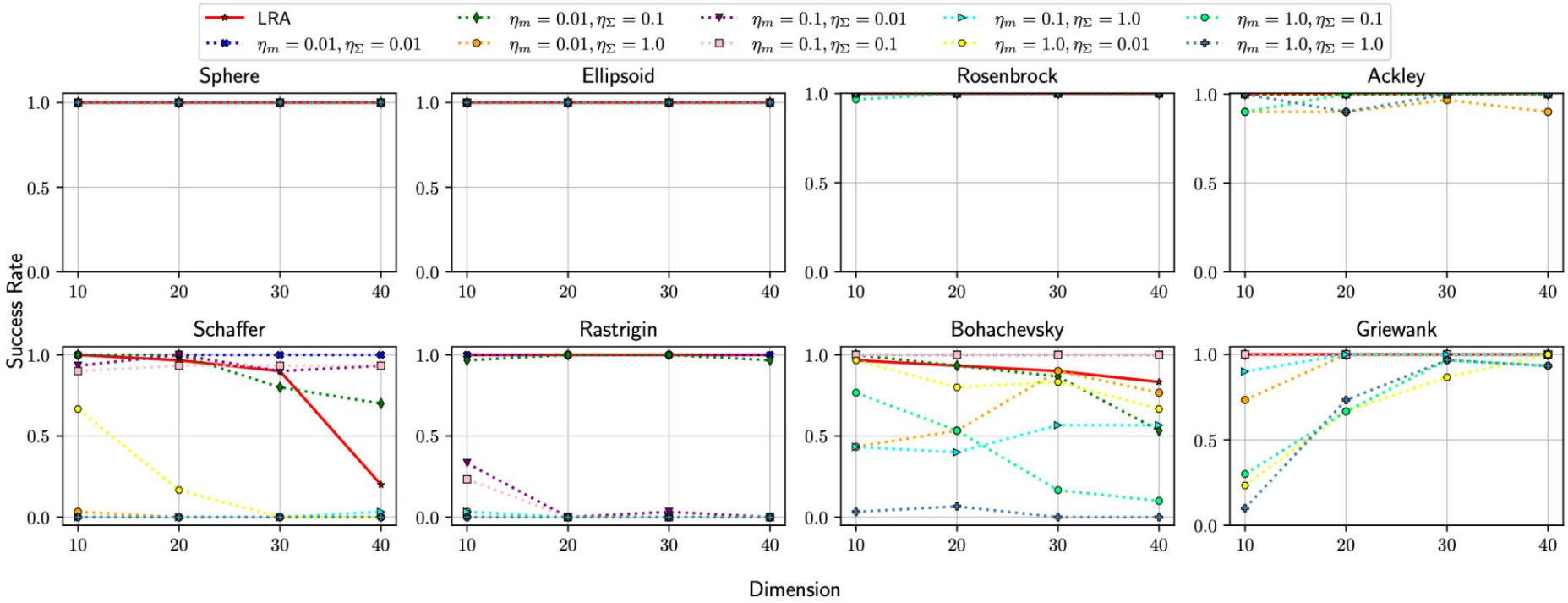


- We set the upper bound to 1 to prevent unstable behavior

Experiments: Research Questions and Setups

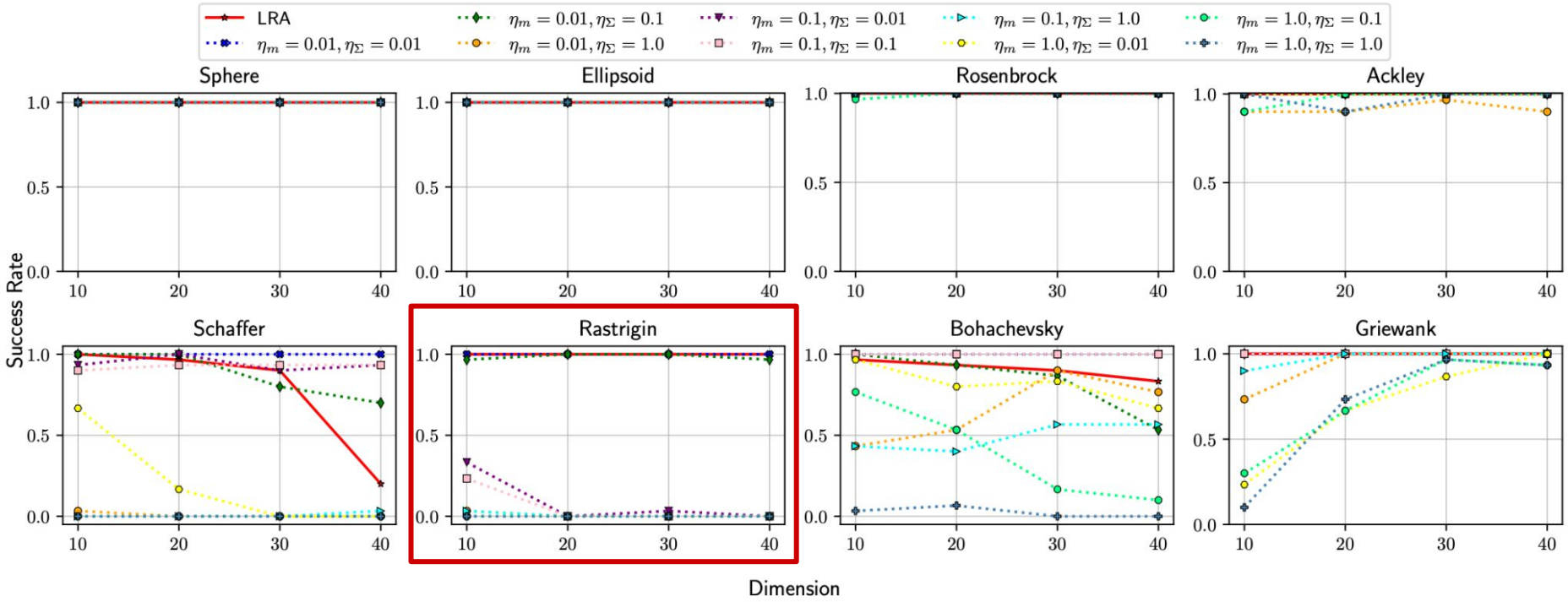
- We compare *LRA-CMA* with CMA with fixed learning rates
- Benchmark problems
  - unimodal (Sphere, Ellipsoid, Rosenbrock) and multimodal (Ackley, Schaffer, Rastrigin, Bohachevsky, Griewank)
  - In noisy problems, we considered additive Gaussian noise
- Population size: recommended value $\lambda = 4 + \lfloor 3 \ln d \rfloor$

Success Rate versus (10-40)Dim. (Noiseless Problems)



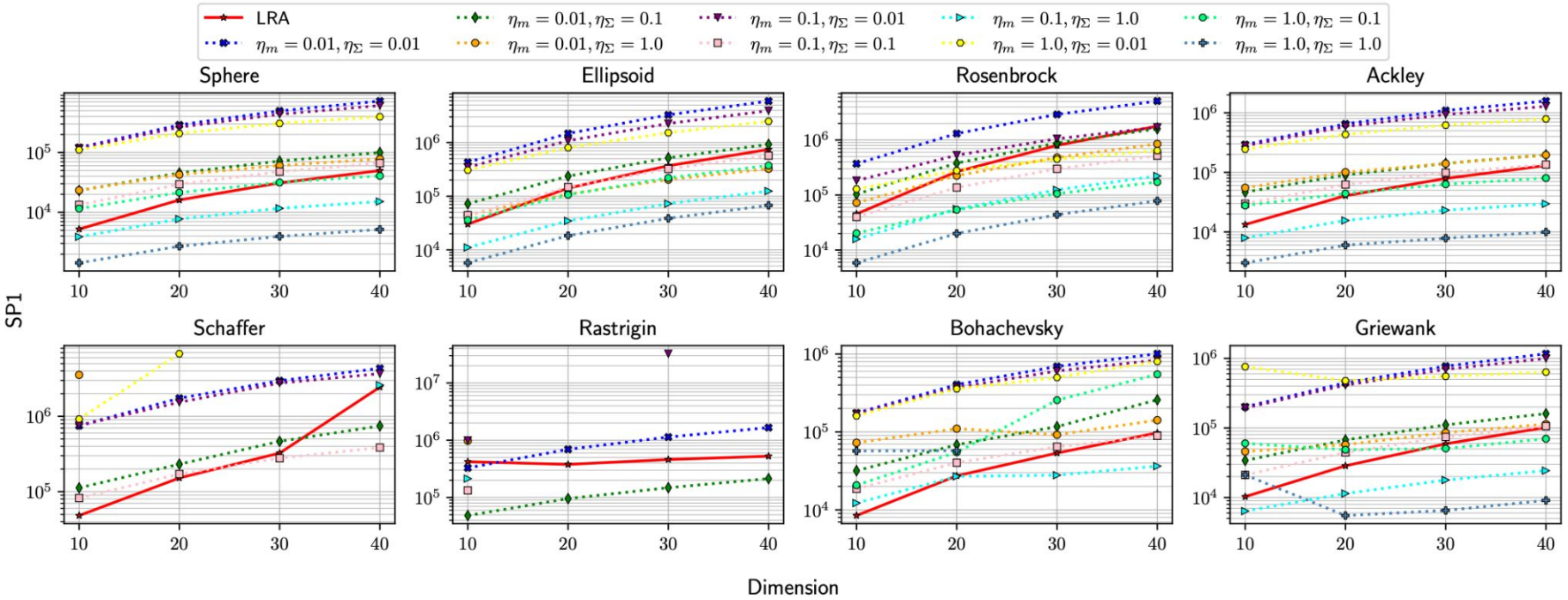
For multimodal, CMA with high η often *failed*, but CMA with small η had a high SR
LRA-CMA had a relatively *good SR without tuning*

Success Rate versus (10-40)Dim. (Noiseless Problems)



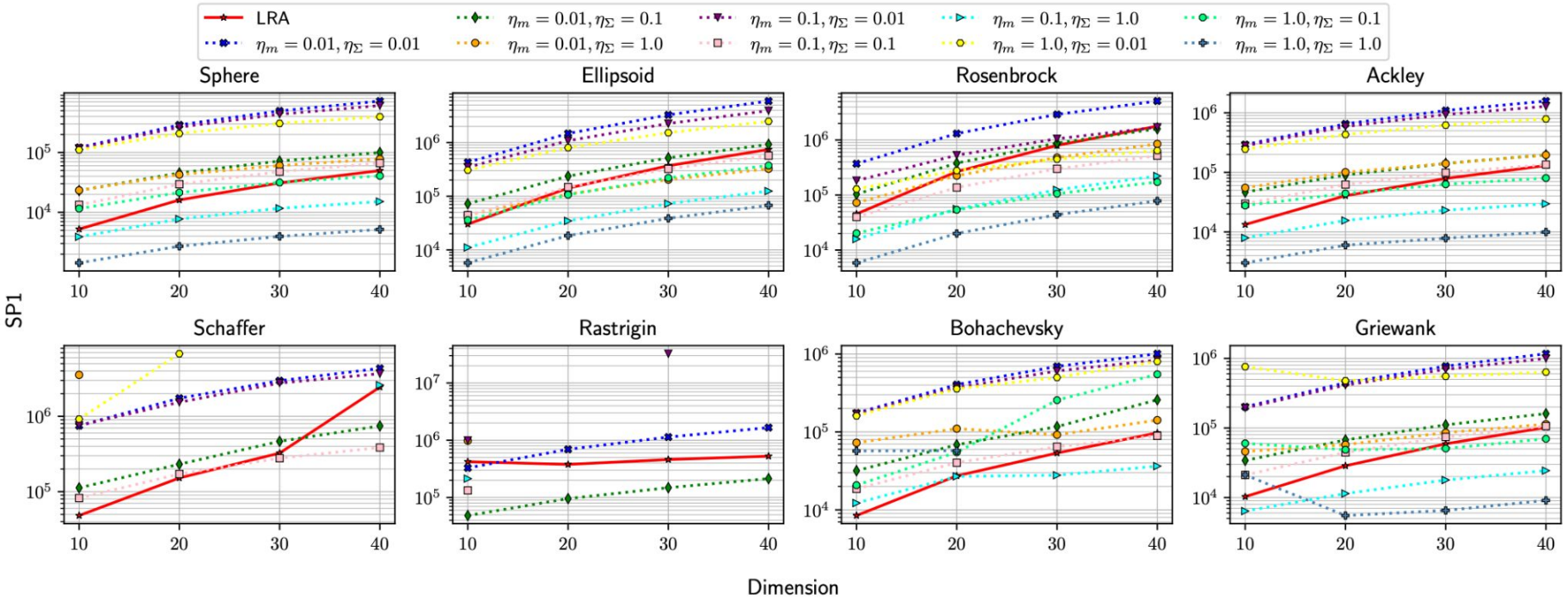
LRA with default pop. (e.g. λ=15 for d=40) succeeded in all trials on the Rastrigin

SP1 versus (10-40)Dim. (Noiseless Problems)



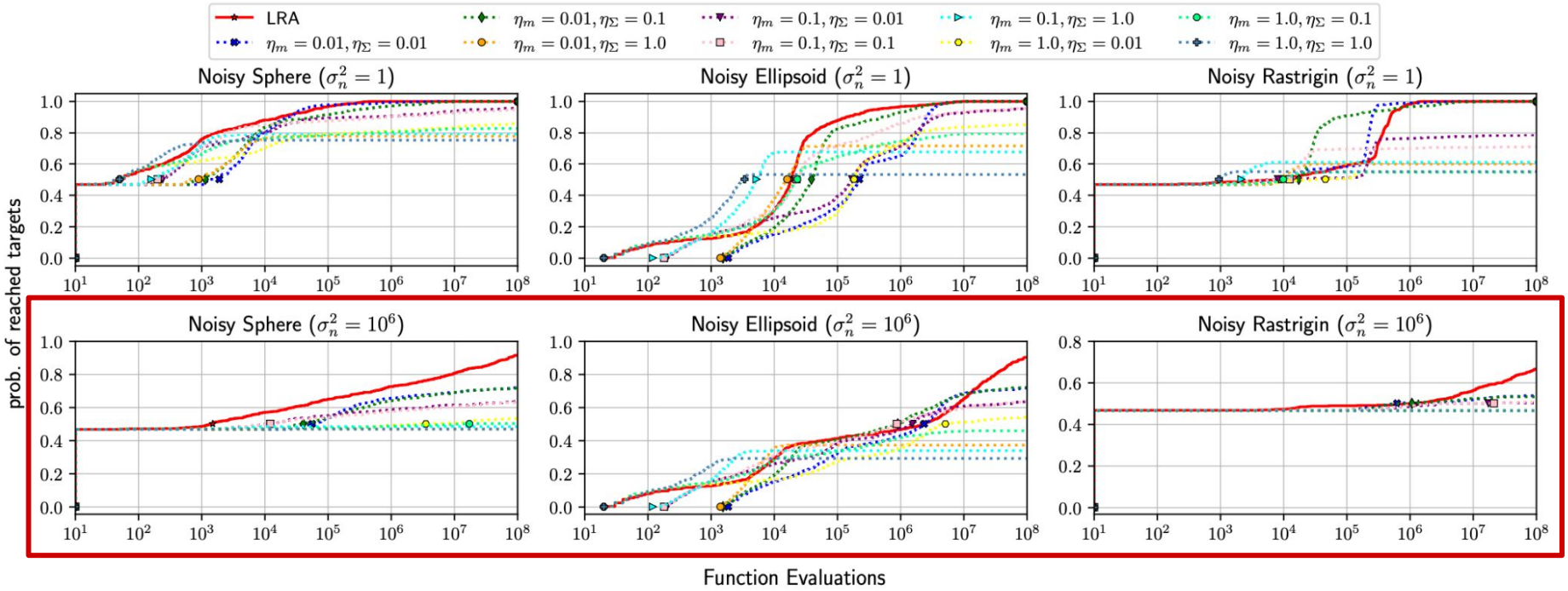
CMA with high η ⇒ worse on multimodal, CMA-ES with small η ⇒ slow on unimodal
Clear trade-off in efficiency exists depending on η

SP1 versus (10-40)Dim. (Noiseless Problems)



LRA shows stable and relatively good performance without expensive tuning

Empirical cumulative density function on noisy problems



CMA with fixed η had stopped improving the function value
In contrast, *LRA continued to improve* it even in strong noise

Part 4:
Practical Guide

Topics of this part

- How to understand behavior of NES/CMA-ES?
  - typical metrics for logging
  - typical behavior on benchmark problems
- How to run NES/CMA-ES in practical situations?
  - popular Python libraries
  - how to set initial configurations
    - injecting domain knowledge for efficient optimization
    - automatic determination by using results on similar tasks
- The content of this part is inspired by [akimotolab/CMAES_Tutorial]

Logging Metrics

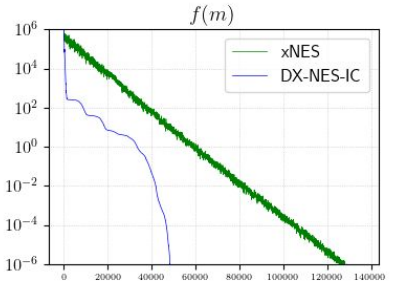
- Function value f(x)
  - may not be sufficient to judge whether optimization is finished
- Mean vector
  - useful for checking validity of range of box constraint
- Step-size σ
  - useful for checking convergence
- ratios of (sqrt) eigenvalues
  - if ratios significantly differ, normalizing scaling variables may be needed

Our Focus: DX-NES-IC

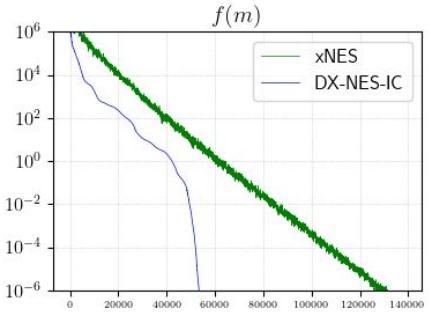
- DX-NES-IC shows better performance than xNES
  - Distance weight
  - Mirrored sampling
  - Switching learning rate
  - Emphasizing expansion

- Note: The behavior to be examined now do not differ significantly across NES/CMA-ES methods

40-Dim. k-Tablet



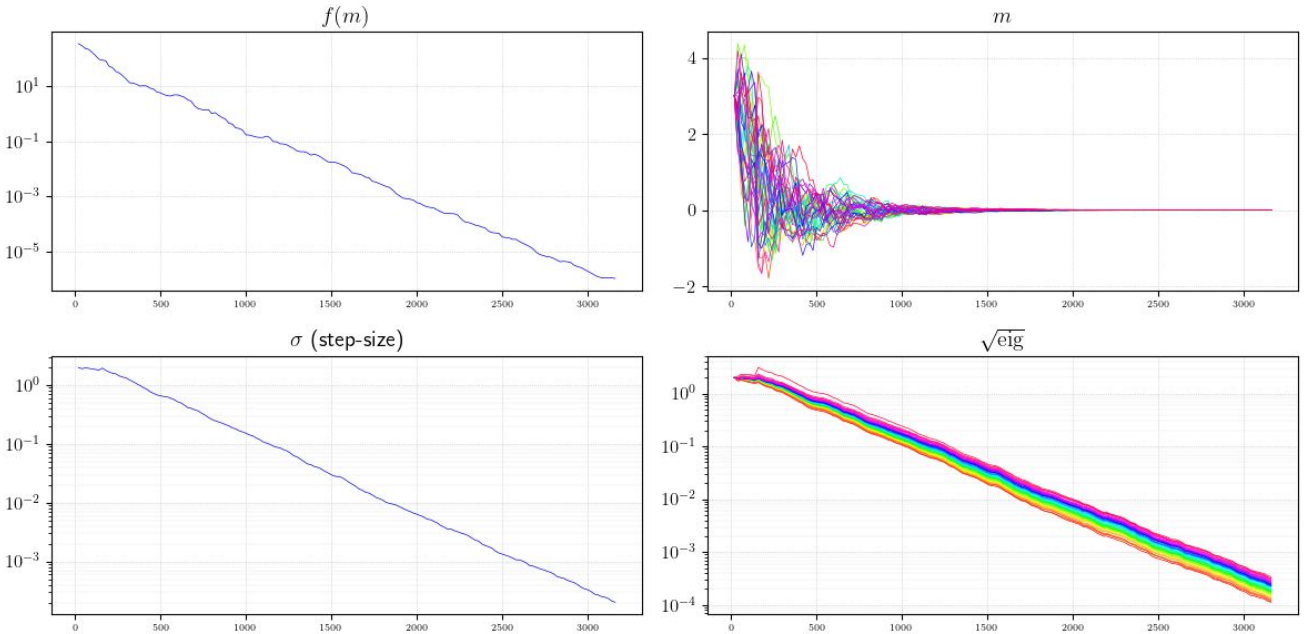
40-Dim. Ellipsoid

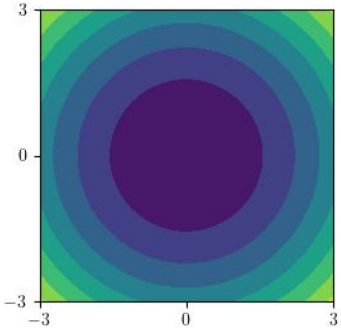


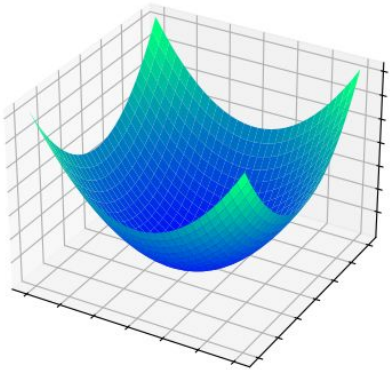
Easy Problem: Sphere Function

$$f_{\text{Sphere}}(x) = \sum_{i=1}^{d} x_i^2$$

importance of each variable is the ***same***



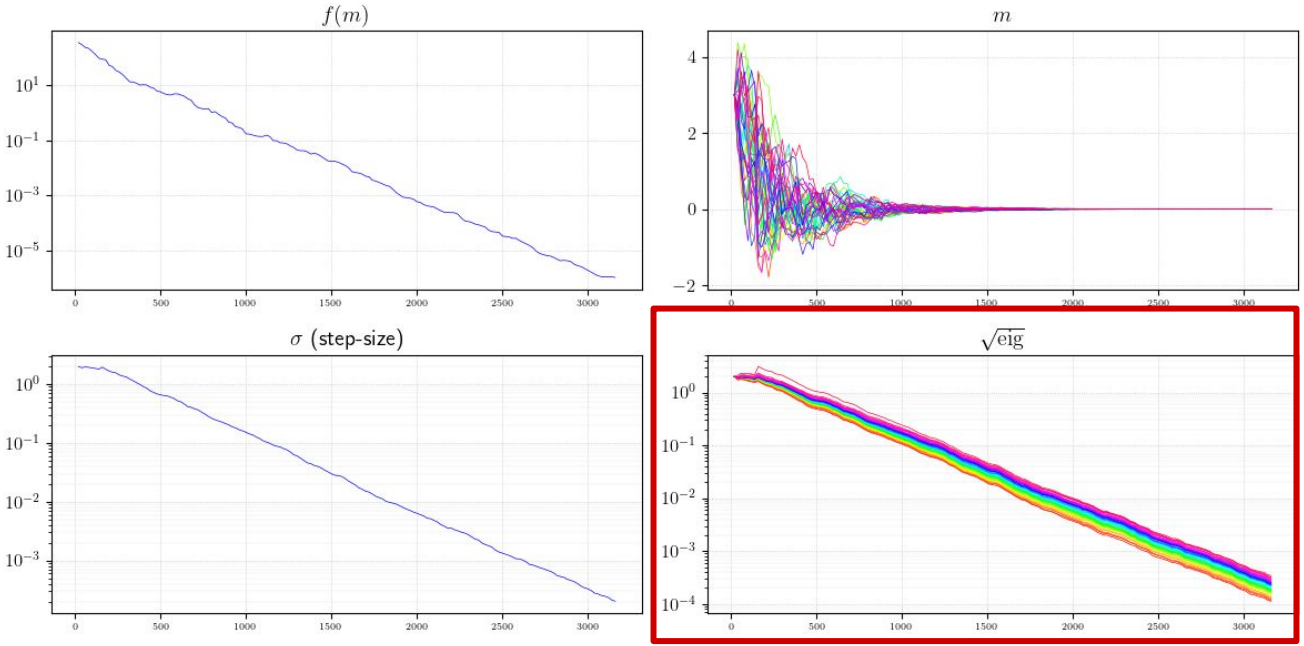


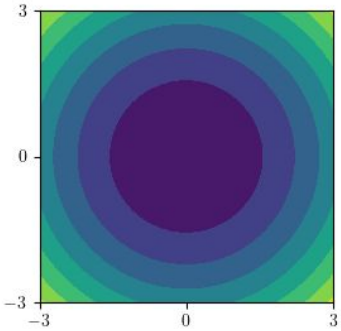


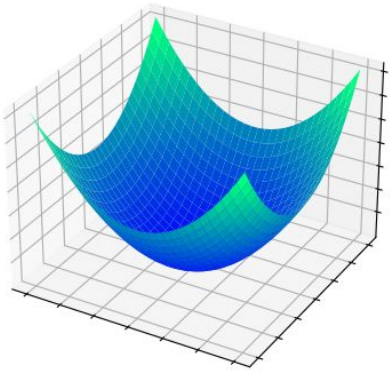
Easy Problem: Sphere Function

$$f_{\mathrm{Sphere}}(x) = \sum_{i=1}^{d} x_i^2$$

importance of each variable is the ***same***



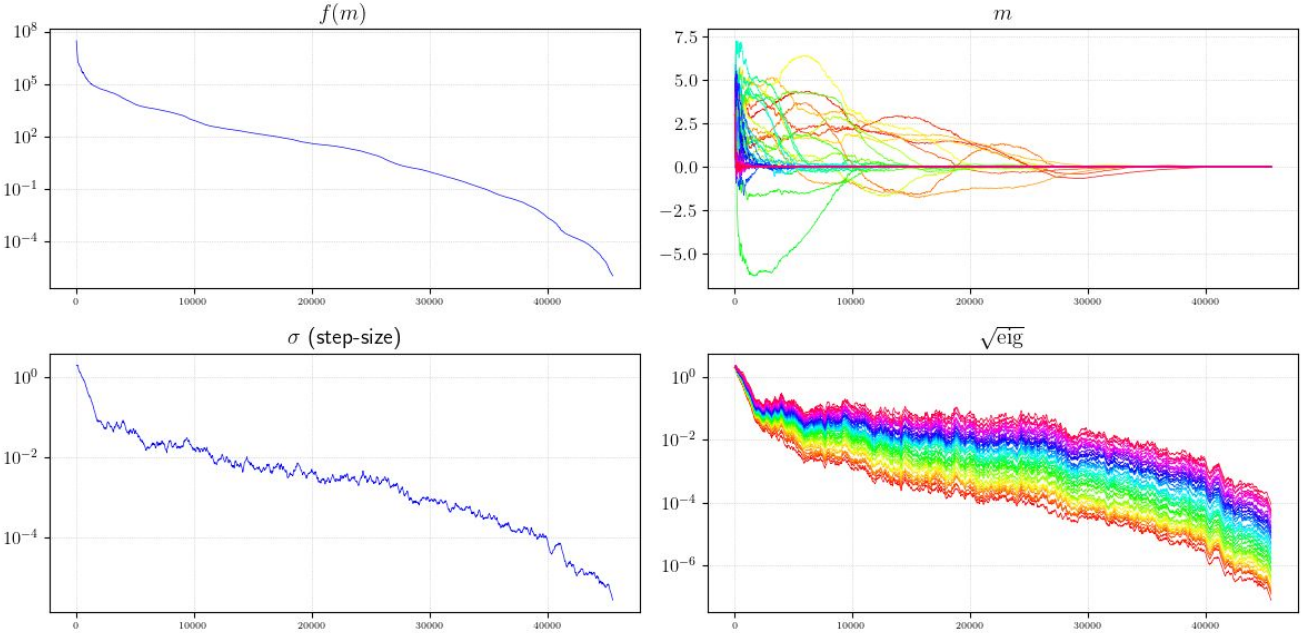


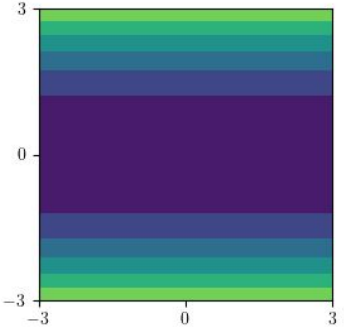


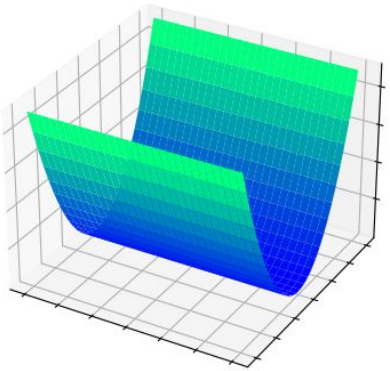
Ill-Conditioned Problem: Ellipsoid Function

$$f_{\text{Ellipsoid}}(x) = \sum_{i=1}^{d} (1000^{\frac{i-1}{d-1}} x_i)^2$$

importance of each variable is ***different***



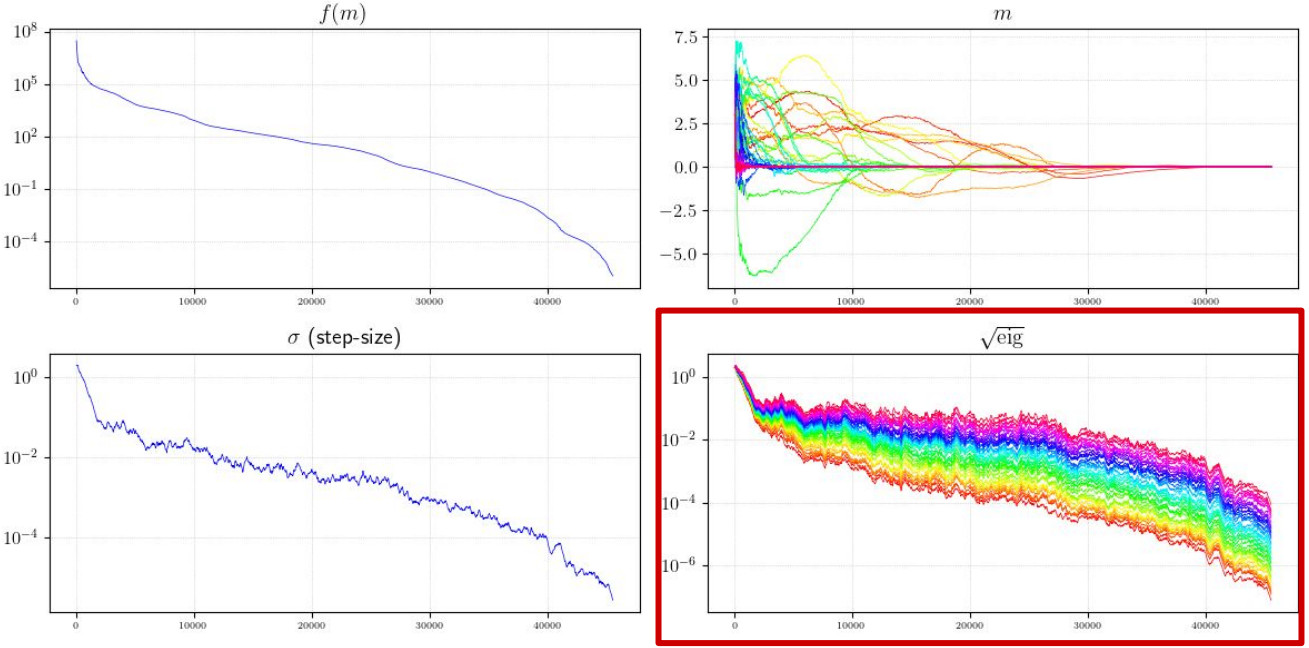


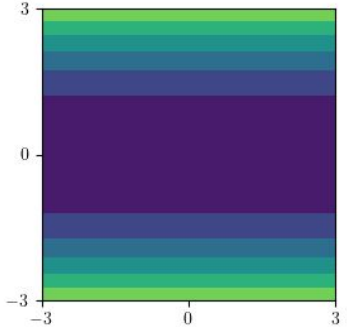


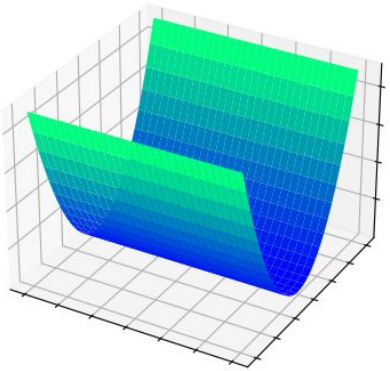
Ill-Conditioned Problem: Ellipsoid Function

$$f_{\text{Ellipsoid}}(x) = \sum_{i=1}^{d} (1000^{\frac{i-1}{d-1}} x_i)^2$$

importance of each variable is ***different***



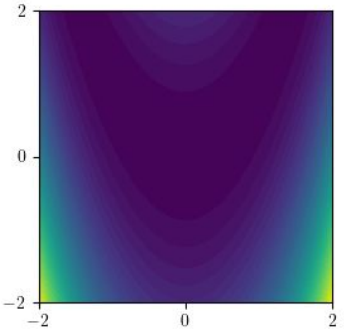


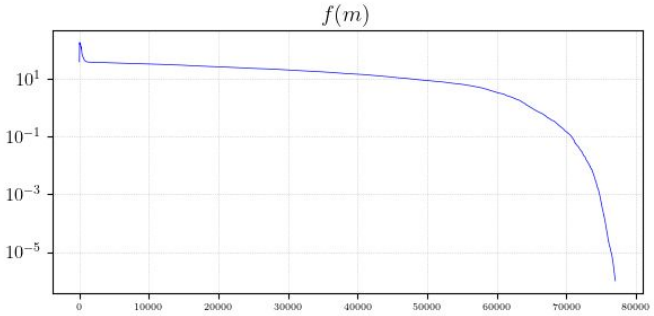


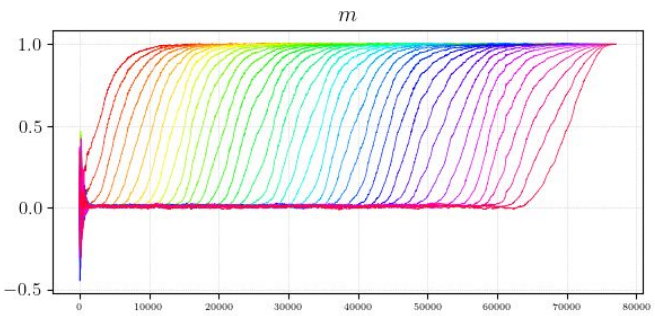
Non-Separable Problem: Rosenbrock Function

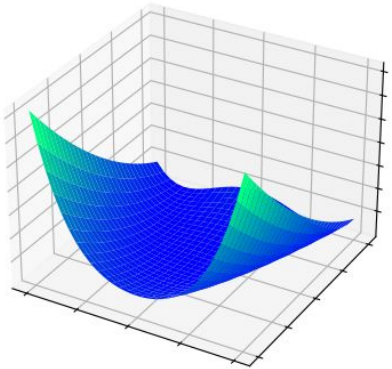
$$f_{\text{Rosenbrock}}(x) = \sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$$

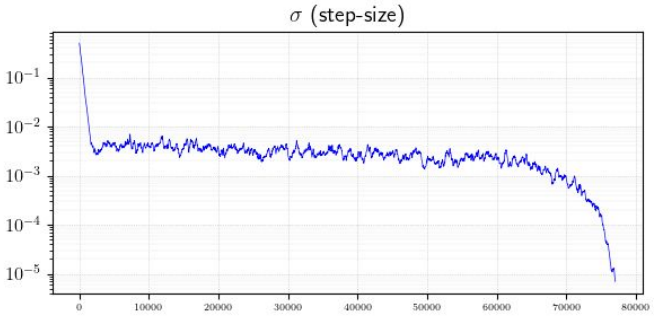
variable dependencies

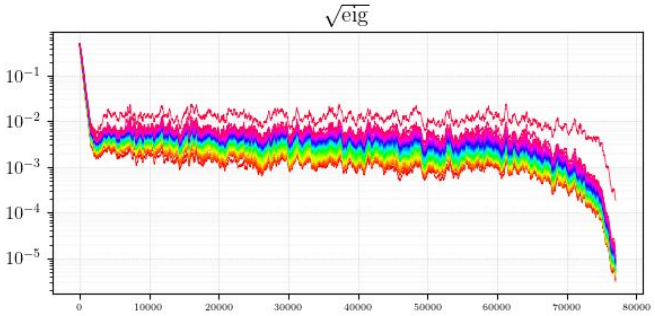








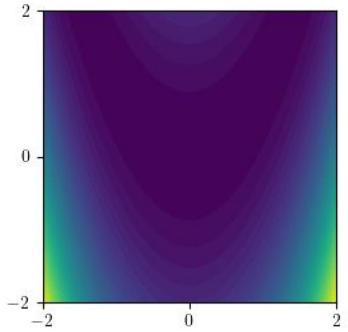


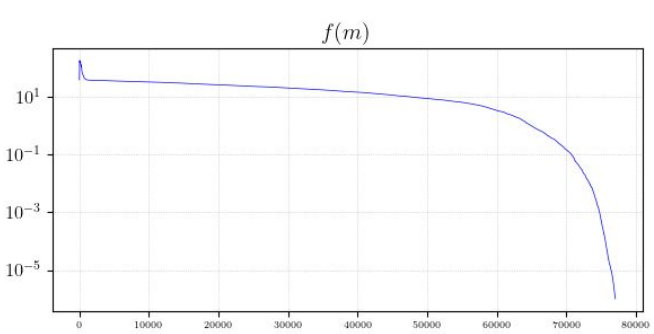


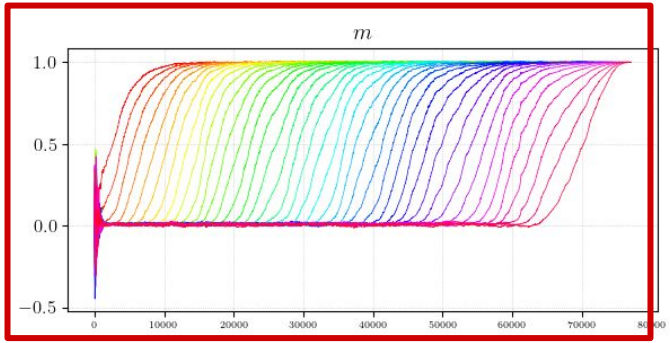
Non-Separable Problem: Rosenbrock Function

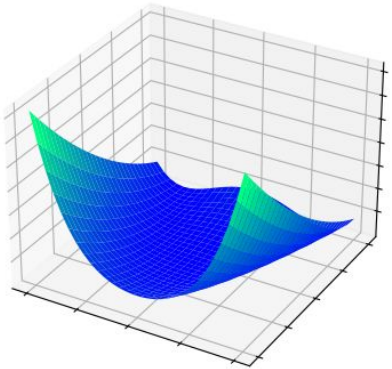
$$f_{\text{Rosenbrock}}(x) = \sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$$

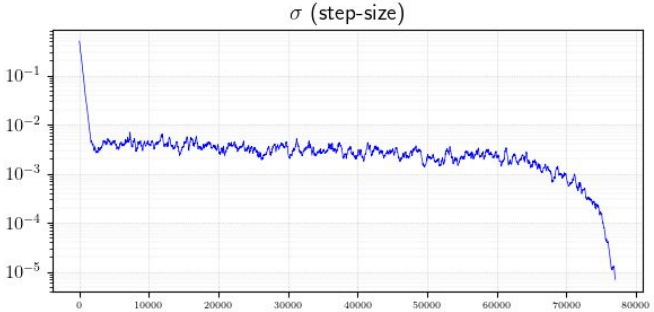
variable dependencies

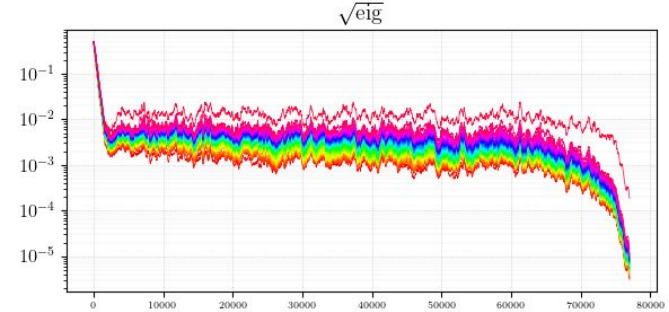






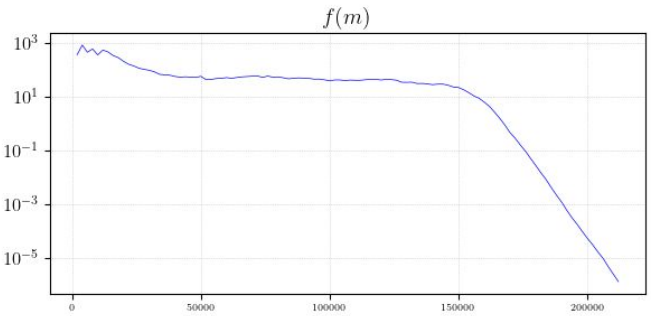


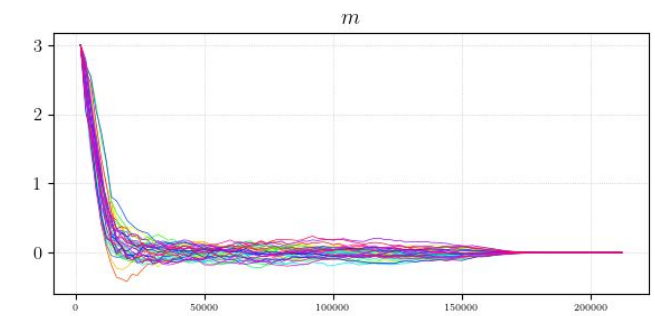


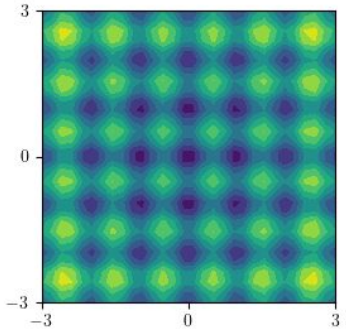


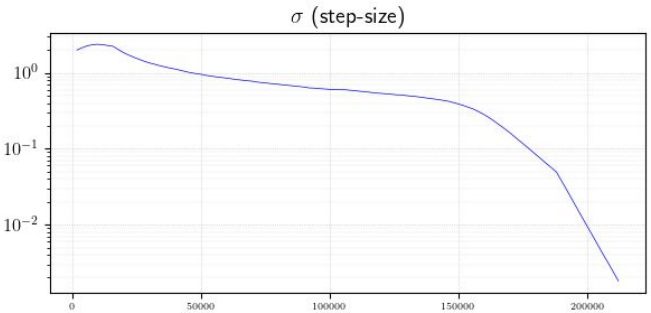
(Well-Structured) Multimodal Problem: Rastrigin Function

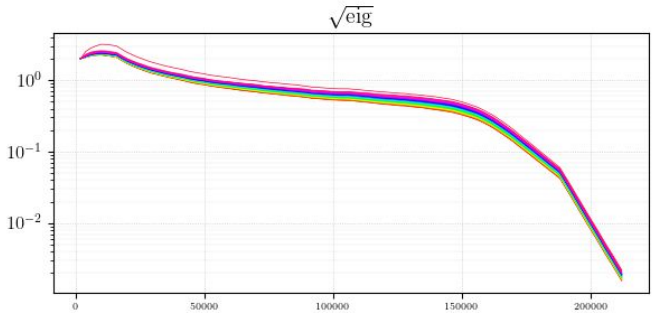
$$f_{\text{Rastrigin}}(x) = 10d + \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i))$$

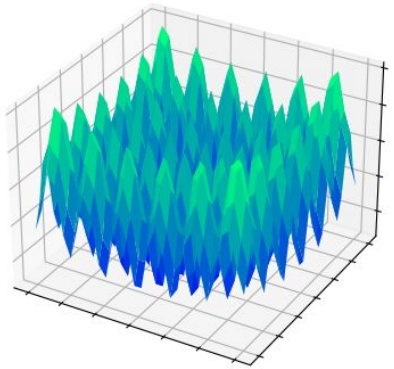








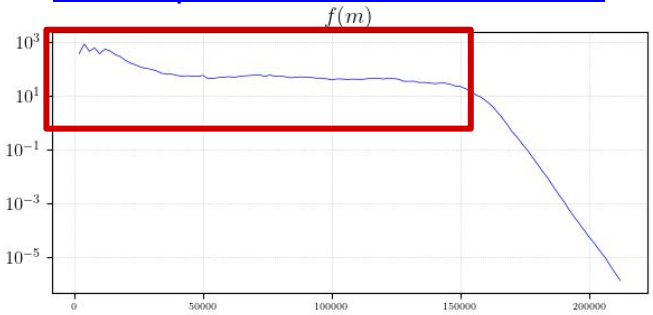


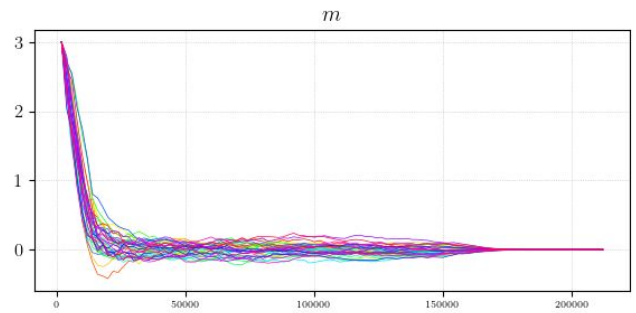


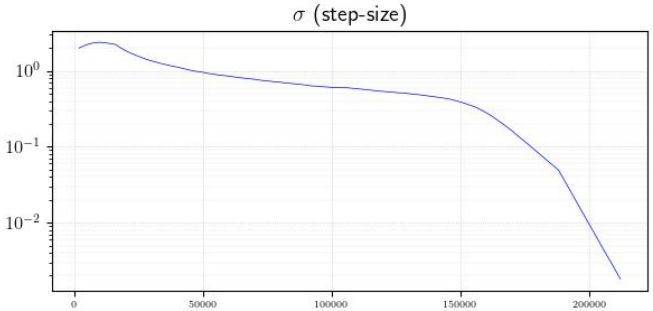
(Well-Structured) Multimodal Problem: Rastrigin Function

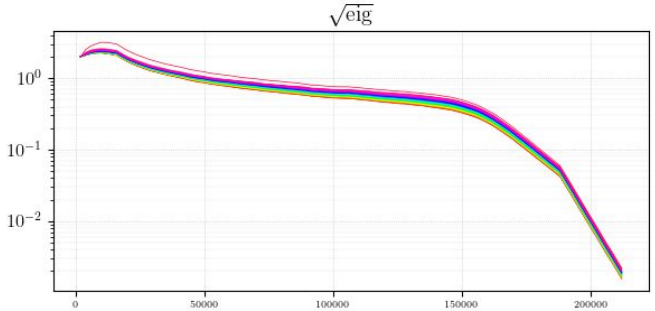
$$f_{\text{Rastrigin}}(x) = 10d + \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i))$$

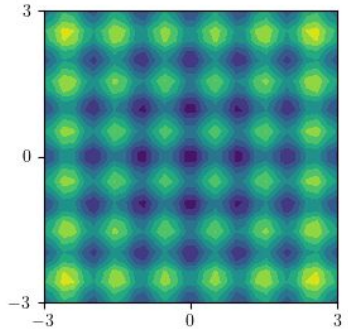
Is the optimization finished? - No!

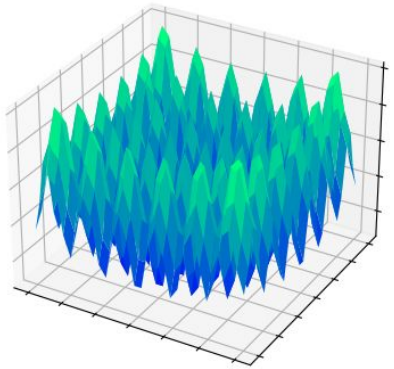






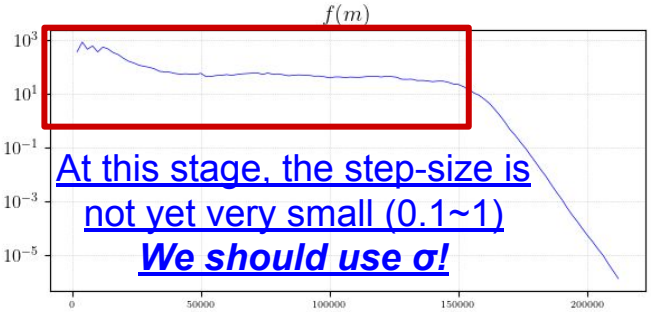






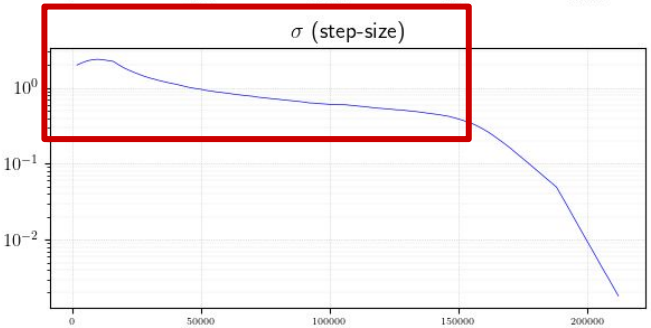
(Well-Structured) Multimodal Problem: Rastrigin Function

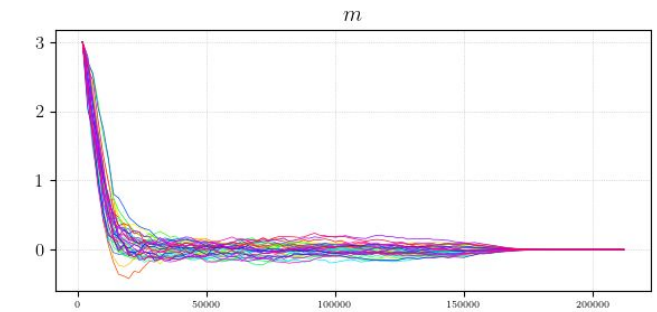
$$f_{\text{Rastrigin}}(x) = 10d + \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i))$$

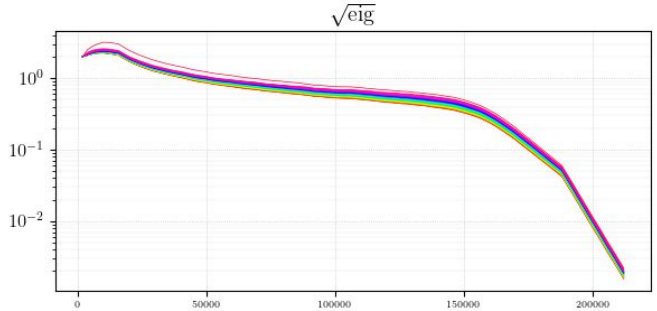


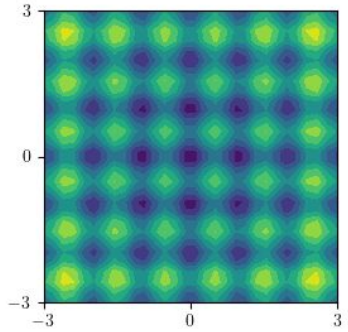
At this stage, the step-size is not yet very small (0.1~1)

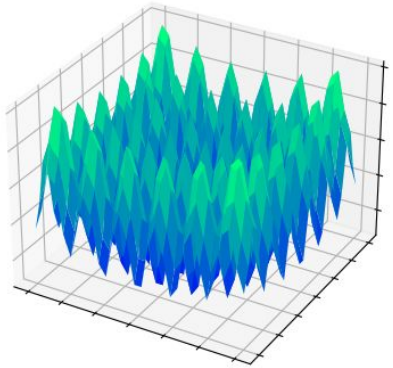
We should use σ!











Python OSS for NES/CMA-ES

- CMA-ES/pycma:
  - Most famous CMA-ES OSS, useful for research and practical purpose
- msu-coinlab/pymoo:
  - Many multi-objective optimization methods including CMA-ES
- google/evojax, RobertTLange/evosax:
  - Jax-based ES implementation
- CyberAgentAILab/cmaes (*focus of this tutorial*):
  - Lightweight implementation, suitable for educational purpose
  - Employed by optuna, which is a popular HPO software

Usage for CyberAgentAILab/cmaes

ask-and-tell interface

```python
import numpy as np
from cmaes import CMA

def quadratic(x1, x2):
    return (x1 - 3) ** 2 + (10 * (x2 + 2)) ** 2

if __name__ == "__main__":
    optimizer = CMA(mean=np.zeros(2), sigma=1.3)

    for generation in range(50):
        solutions = []
        for _ in range(optimizer.population_size):
            x = optimizer.ask()
            value = quadratic(x[0], x[1])
            solutions.append((x, value))
            print(f"#{generation} {value} (x1={x[0]}, x2 = {x[1]})")
        optimizer.tell(solutions)
```

Usage for CyberAgentAILab/cmaes

ask-and-tell interface

```python
import numpy as np
from cmaes import CMA

def quadratic(x1, x2):
    return (x1 - 3) ** 2 + (10 * (x2 + 2)) ** 2

if __name__ == "__main__":
    optimizer = CMA(mean=np.zeros(2), sigma=1.3)

    for generation in range(50):
        solutions = []
        for _ in range(optimizer.population_size):
            x = optimizer.ask()
            value = quadratic(x[0], x[1])
            solutions.append((x, value))
            print(f"#{generation} {value} (x1={x[0]}, x2={x[1]})")
        optimizer.tell(solutions)
```

1: initialize optimizer

2: generate solutions

3: evaluate x on f

4: update distributions

Usage for CyberAgentAILab/cmaes

ask-and-tell interface

```python
import numpy as np
from cmaes import CMA

def quadratic(x1, x2):
    return (x1 - 3) ** 2 + (10 * (x2 + 2)) ** 2

if __name__ == "__main__":
    optimizer = CMA(mean=np.zeros(2), sigma=1.3)

    for generation in range(50):
        solutions = []
        for _ in range(optimizer.population_size):
            x = optimizer.ask()
            value = quadratic(x[0], x[1])
            solutions.append((x, value))
            print(f"#{generation} {value} (x1={x[0]}, x2 = {x[1]})")
        optimizer.tell(solutions)
```

initial distribution
(mean, step-size, and cov.)

How to Set Initial Distribution?: Mean Vector

- If you know promising regions, it is reasonable to initialize the mean there
- E.g. hyperparameter optimization of deep neural networks
  - It is often the case that the paper proposing the network structure provides *recommended values*
    - may be beneficial to set the mean vector to the recommended values
  - However, a certain degree of unimodality is assumed

How to Set Initial Distribution?: Step-Size σ

- If you are somewhat confident that the optimum lies on near mean, it may be possible to reduce step-size

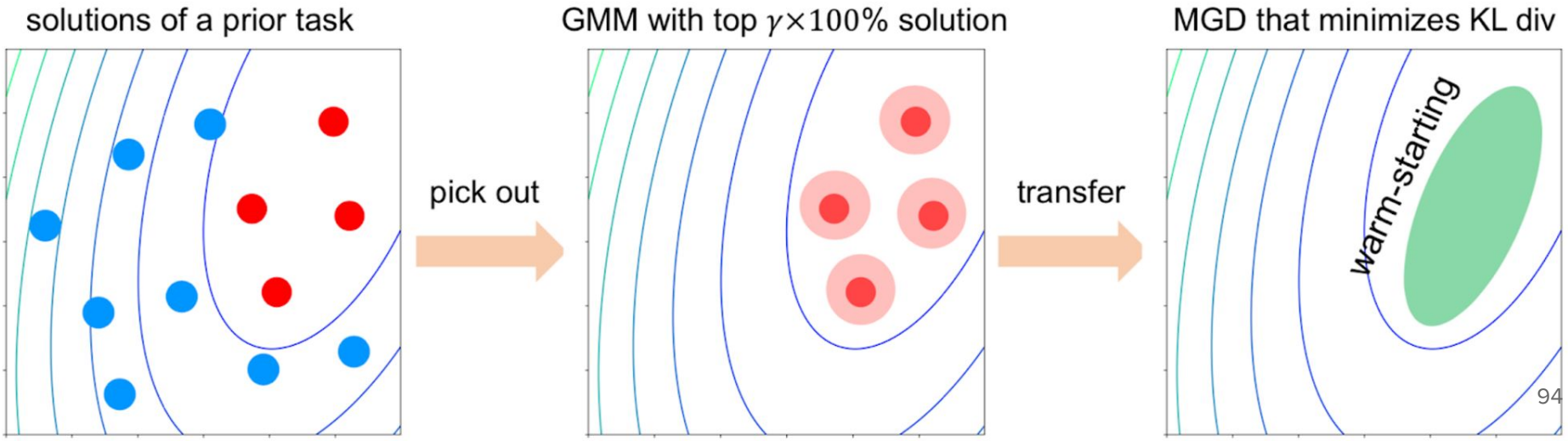
- In well-structured multimodal (i.e. big-valley) problems, setting a relatively large value is a reasonable choice
- However, in weakly-structured multimodal problems, a large step-size can be detrimental
- In BBO scenarios, trial-and-error cannot (or may not) be avoidable

Automatic Determination of Initial Distribution from Data

- In BBO scenarios, is it impossible to utilize information about the problem?
  - Not necessarily the case: we often have results on related (source) task
- Example 1: HPO with periodic execution
  - Updating model and running HPO *whenever* new data becomes available
  - Previous HPO can be considered as the source task
- Example 2: HPO with partial data
  - Before building model with large-scale data, performance are verified using *partial* data
  - HPO with partial data can be considered as the source task
- Example 3: HPO for different dataset
  - Model construction is performed *separately* for each dataset due to accuracy/size
  - HPO for different dataset can be considered as the source task

Warm Starting CMA-ES [NWA+21]: Overview

- Assumption:
  - The source task and target task are similar
- Basic idea:
  - Approaching the initial dist. of MGD to promising regions estimated from source task



Warm Starting CMA-ES [NWA+21]: Implementation

```
if __name__ == "__main__":
    # Generate solutions from a source task
    source_solutions = []
    for _ in range(1000):
        x = np.random.random(2)
        value = source_task(x[0], x[1])
        source_solutions.append((x, value))

    # Estimate a promising distribution of the source task,
    # then generate parameters of the multivariate gaussian distribution.
    ws_mean, ws_sigma, ws_cov = get_warm_start_mgd(
        source_solutions, gamma=0.1, alpha=0.1
    )
    optimizer = CMA(mean=ws_mean, sigma=ws_sigma, cov=ws_cov)
```

result on source task

hyperparameter for Warm Starting CMA: Default is sufficient

initial distribution

How to Set Hyperparameters?: Population Size λ

- Increasing pop. size is beneficial for well-structured multimodal problems
  - However, practitioners often wish to specify # workers as pop. size...
  - You can specify any pop. size by using *learning rate adaptation*
- without LRA (default)

```
optimizer = CMA(mean=np.ones(10) * 3, sigma=2.0)
```

- with LRA [NAO23]

only modification

```
optimizer = CMA(mean=np.ones(10) * 3, sigma=2.0, lr_adapt=True)
```

Part 5: Summary and Future Challenges

NES objective & Natural Gradient

- NES objective: the function of θ, not x

$$J(\theta) = \mathbb{E}_{x \sim p(x;\theta)}[f(x)]$$

- Natural gradient update with Monte Carlo method

$$\theta \leftarrow \theta + \eta \cdot F^{-1} \nabla_\theta J(\theta)$$

$$\approx \theta + \eta \cdot \frac{1}{\lambda} \sum_{i=1}^{\lambda} f(x_i) F^{-1} \nabla_\theta \log p(x_i; \theta)$$

Fitness Shaping

- Fitness shaping: introduce the weights $(w_i \geq \cdots \geq w_\lambda)$
  - rendering invariant under monotonically increasing transformation of f:

$$F^{-1}\nabla_\theta J(\theta) \approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} f(x_i) F^{-1}\nabla_\theta \log p(x_i;\theta)$$

Fitness Shaping

$$\sum_{i=1}^{\lambda} w_i F^{-1}\nabla_\theta \log p(x_{i:\lambda};\theta)$$

$x_{i:\lambda}$: *i*-th best solution

NES uses only ranking rather than function value itself

Information Geometric Optimization (IGO)

- NES with fitness shaping is natural gradient method for another objective
- IGO objective: $J_{\theta^{(t)}}(\theta) = \mathbb{E}_{x \sim p(x;\theta)}[W^f_{\theta^{(t)}}(x)]$

Objective function depends on the current parameter $\theta^{(t)}$

- This framework generalizes some popular stochastic methods
  - continuous: xNES, Rank-μ update CMA-ES
  - binary: PBIL, cGA

Recent Advances

- **Sample Reuse via Importance Sampling**
  - Leveraging discarded ‘old’ samples for improving estimation accuracy
- **Addressing Inefficiency of xNES**
  - Distance weight & emphasis on distribution expansion
- **High-Dimensional Optimization**
  - Restricting representation of covariance matrix
- **Learning Rate Adaptation**
  - Solving difficult problems (e.g. multimodal and noisy) without tuning

Future Challenges (1)

- **Designing Weights (Fitness Shaping) for 'General' Gaussian Distribution**
  - Optimal weights for isotropic Gaussian are derived (quality gain analysis)
    - Not clear for Gaussian distribution with *general* cov. matrix
  - Distance weight addresses the issue, but theoretical validity is not clear
- **Towards Completely Hyperparameter-Free Optimizer**
  - Population size adaptation (PSA) and learning rate adaptation (LRA)
    - But experiments are still not exhaustive (e.g. hyperparameter)
  - Lack of comparison between PSA and LRA

Future Challenges (2)

- **BBO with Safety Constraints**
  - There is a need to optimize while maintaining fval. above a threshold
  - Active in areas such as bayesian optimization [SGBK15],
    but few studies in NES/CMA
- **Mutivariate Discrete BBO via Natural Gradient**
  - While there are BBO methods such as PBIL and cGA that use (*univariate*)
    Bernoulli distributions, there are hardly any that use multivariate ones
  - An example exists of applying the natural gradient method to an RBM
    (Restricted Boltzmann Machine), but empirical evaluation is limited

Thank you for listening!

contact: masahironomura5325@gmail.com

Tutorial materials: https://github.com/nomuramasahir0/cec2023-tutorial

CyberAgentAILab/cmaes: https://github.com/CyberAgentAILab/cmaes

Acknowledgements:

The design of this slide is based on these wonderful slides by Yuta Saito.

References

References (1)

[AAH14] Youhei Akimoto, Anne Auger, and Nikolaus Hansen. Comparison-Based Natural Gradient Optimization in High Dimension. In *Proceedings of the Genetic and Evolutionary Computation*, pages 373– 380, 2014.

[AAH20] Youhei Akimoto, Anne Auger, and Nikolaus Hansen. Quality gain analysis of the weighted recombination evolution strategy on general convex quadratic functions. *Theoretical Computer Science*, 832:42– 67, 2020.

[ABH11] Anne Auger, Dimo Brockhoff, and Nikolaus Hansen. Mirrored Sampling in Evolution Strategies With Weighted Recombination. In *Proceedings of the Genetic and Evolutionary Computation*, pages 861–868, 2011.

[AD98] Shun-Ichi Amari and Scott C Douglas. Why natural gradient? In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 1213–1216. IEEE, 1998.

[AH22] Youhei Akimoto and Nikolaus Hansen. CMA-ES and Advanced Adaptation Mechanisms. In *Proceedings of the Genetic and Evolutionary Computation Companion*, page 1243–1268, 2022. Association for Computing Machinery.

References (2)

[Ama98] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[ANOK10] Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Bidirectional Relation between CMA Evolution Strategies and Natural Evolution Strategies. In *International Conference on Parallel Problem Solving from Nature*, pages 154–163. Springer, 2010.

[Arn05] Dirk V Arnold. Optimal Weighted Recombination. In *Foundations of Genetic Algorithms*, pages 215–237. Springer, 2005.

[ASY+19] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive Stochastic Natural Gradient Method for One-Shot Neural Architecture Search. In *International Conference on Machine Learning*, pages 171–180. PMLR, 2019.

[Bey14] Hans-Georg Beyer. Convergence Analysis of Evolutionary Algorithms That are Based on the Paradigm of Information Geometry. *Evolutionary Computation*, 22(4):679–709, 2014.

References (3)

[FNKO11] Nobusumi Fukushima, Yuichi Nagata, Sigenobu Kobayashi, and Isao Ono. Proposal of distance-weighted exponential natural evolution strategies. In *IEEE Congress of Evolutionary Computation (CEC)*, pages 164–171. IEEE, 2011.

[GK20] Tobias Glasmachers and Oswin Krause. The Hessian Estimation Evolution Strategy. In *International Conference on Parallel Problem Solving from Nature*, pages 597–609. Springer, 2020.

[GSY+10] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and J¨urgen Schmidhuber. Exponential Natural Evolution Strategies. In *Proceedings of the Genetic and Evolutionary Computation*, pages 393–400, 2010.

[Han16] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[HMK03] Nikolaus Hansen, Sibylle D M¨uller, and Petros Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary computation*, 11(1):1–18, 2003.

References (4)

[MA17] Hidekazu Miyazawa and Youhei Akimoto. Effect of the Mean Vector Learning Rate in CMA-ES. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 721–728, 2017.

[NA16] Kouhei Nishida and Youhei Akimoto. Population Size Adaptation for the CMA-ES Based on the Estimation Accuracy of the Natural Gradient. In *Proceedings of the Genetic and Evolutionary Computation*, page 237–244, 2016.

[NA18] Kouhei Nishida and Youhei Akimoto. PSA-CMA-ES: CMA-ES with Population Size Adaptation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 865–872, 2018.

[NAO23] Masahiro Nomura, Youhei Akimoto, and Isao Ono. CMA-ES with Learning Rate Adaptation: Can CMA-ES with Default Population Size Solve Multimodal and Noisy Problems? In *Proceedings of the Genetic and Evolutionary Computation (To Appear)*, 2023.

[NO22] Masahiro Nomura and Isao Ono. Fast Moving Natural Evolution Strategy for High-Dimensional Problems. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2022.

References (5)

[NSFO21] Masahiro Nomura, Nobuyuki Sakai, Nobusumi Fukushima, and Isao Ono. Distance-weighted Exponential Natural Evolution Strategy for Implicitly Constrained Black-Box Function Optimization. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1099– 1106. IEEE, 2021.

[NWA+21] Masahiro Nomura, Shuhei Watanabe, Youhei Akimoto, Yoshihiko Ozaki, and Masaki Onishi. Warm Starting CMA-ES for Hyperparameter Optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9188–9196, 2021.

[OAAH17] Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-Geometric Optimization Algorithms: A Unifying Picture via Invariance Principles. *The Journal of Machine Learning Research*, 18(1):564–628, 2017.

[RH08] Raymond Ros and Nikolaus Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. In *International conference on parallel problem solving from nature*, pages 296–305. Springer, 2008.

[SAOO15] Shinichi Shirakawa, Youhei Akimoto, Kazuki Ouchi, and Kouzou Ohara. Sample Reuse in the Covariance Matrix Adaptation Evolution Strategy Based on Importance Sampling. In *Proceedings of the Genetic and Evolutionary Computation*, pages 305–312, 2015.

References (6)

[SGBK15] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe Exploration for Optimization with Gaussian Processes. In *International conference on machine learning*, pages 997–1005. PMLR, 2015.

[WSG+14] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and J¨urgen Schmidhuber. Natural Evolution Strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.