Stochastic Continuous Optimisation: Adaptivity, Invariances and Surrogate Models

Ilya Loshchilov, Marc Schoenauer and Michèle Sebag
TAO team – INRIA / CNRS / Univ. Paris-Sud

Horizon Maths 2014, IFPEN, 15 dec. 2014
Black-Box Optimization

Find \( \text{Argmin} \{ \mathcal{F} : X \rightarrow \mathbb{R} \} \)

**Context:** ill-posed optimization problems (continuous)
- Function \( \mathcal{F} \) (fitness function) on \( X \subset \mathbb{R}^n \)
- Gradient not available or not useful
- \( \mathcal{F} \) available as an oracle (black box)

\[
x \xrightarrow{\text{Black-box approaches}} f(x)
\]

Build \( \{x_1, x_2, \ldots \} \rightarrow \text{Argmin}(\mathcal{F}) \)

**Black-box approaches**
- Robust
- High computational costs: number of expensive function evaluations (e.g. CFD)
Surrogate-Assisted Optimization

Principle

1. Gather $\mathcal{E} = \{ (x_i, F(x_i)) \}$
2. Learn $\hat{F}$ from $\mathcal{E}$
3. Use surrogate $\hat{F}$ in lieu of $F$ for some time
4. Compute $F(x_i)$ for some new $x_i$
5. Learn new surrogate $\hat{F}$ using some parameters
6. Iterate (goto 3)}
Surrogate-Assisted Optimization (2)

Issues

- Learning
  - Hypothesis space (polynomials, neural nets, Gaussian processes, ...)
  - Selection of training set (prune, update, ...)
  - What is the learning quality indicator?

- Interaction of Learning & Optimization modules
  - Schedule (when to relearn)
    - How to use $\hat{F}$ to support optimization search
    - How to use search results to support learning $\hat{F}$

This study

- Using Covariance-Matrix Estimation within Support Vector Machines
- Self-adaptation of surrogate model hyper-parameters
Content

1. CMA-ES, State-of-the-Art in Continuous Optimization
   - Black-Box Optimization and Search Template
   - On-line Adaptation
   - CMA-ES

2. Support Vector Machines and Rank-SVM
   - Statistical Machine Learning
   - Linear classifiers
   - The kernel trick

3. Rank-based Surrogate Models for CMA-ES
   - Choice of Kernel and Surrogate Lifetime
   - Size of Training Set and SVM Parameters
   - Experimental Results
Content

1. CMA-ES, State-of-the-Art in Continuous Optimization
   - Black-Box Optimization and Search Template
   - On-line Adaptation
   - CMA-ES

2. Support Vector Machines and Rank-SVM
   - Statistical Machine Learning
   - Linear classifiers
   - The kernel trick

3. Rank-based Surrogate Models for CMA-ES
   - Choice of Kernel and Surrogate Lifetime
   - Size of Training Set and SVM Parameters
   - Experimental Results
Black-box (Continuous) Optimization

Context: ill-posed (continuous) optimization problems

- Minimize $\mathcal{F}$ (objective/fitness function) on $X \subset \mathbb{R}^d$
- Gradient not available or not useful
- $\mathcal{F}$ available as an oracle (black box)

Goal: Build $\{x_1, x_2, \ldots\} \rightarrow \text{Argmin}(\mathcal{F})$

Black-box approaches

- Robust thanks to invariance properties
- High computational costs: number of function evaluations
A black box search template to minimize $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Initialize distribution parameters $\theta$, set sample size $\lambda \in \mathbb{N}$

While not terminate

1. Sample distribution $P(x|\theta) \rightarrow x_1, \ldots, x_\lambda \in \mathbb{R}^n$
2. Evaluate $x_1, \ldots, x_\lambda$ on $f$
3. Update parameters $\theta \leftarrow F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda))$

Covers

- Deterministic algorithms,
- Evolutionary Algorithms, PSO, DE
- Estimation of Distribution Algorithms

$P$ implicitly defined by the variation operators
The $(\mu, \lambda)$—Evolution Strategy

Gaussian Mutations

\[ x_i \sim m + \sigma N_i(0, C) \quad \text{for } i = 1, \ldots, \lambda \]

- mean vector $m \in \mathbb{R}^n$ is the current solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the step length
- the covariance matrix $C \in \mathbb{R}^{n \times n}$ determines the shape of the distribution ellipsoid

How to update $m$, $\sigma$, and $C$?

- $m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_i : \lambda$
  - Selection of best $\mu$ individuals

Adaptive $\sigma$ and $C$
Need for adaptation

![Graph showing distance to optimum vs. function evaluations for different search methods: random search, constant step-size, and adaptive step-size. The graph illustrates the performance improvement with adaptive step-size over constant step-size and random search.](image)
Cumulative Step-Size Adaptation (CSA)

Measure the length of the *evolution path*
the pathway of the mean vector \( m \) in the generation sequence

- decrease \( \sigma \)
- increase \( \sigma \)

loosely speaking steps are
- perpendicular under random selection (in expectation)
- perpendicular in the desired situation (to be most efficient)
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i: \lambda, \quad y_i \sim N_i(0, \mathbf{C}) \]

- initial distribution, \( \mathbf{C} = \mathbf{I} \)

- new distribution: \( \mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times y_w y_w^T \)

- ruling principle: the adaptation increases the probability of successful steps, \( y_w \), to appear again
Covariance Matrix Adaptation

Rank-One Update

\[
\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w, \quad \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_i, \quad \mathbf{y}_i \sim \mathcal{N}_i(0, \mathbf{C})
\]

- \(\mathbf{y}_w\), movement of the population mean \(\mathbf{m}\) (disregarding \(\sigma\))
- new distribution: \(\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \mathbf{y}_w \mathbf{y}_w^T\)
- ruling principle: the adaptation increases the probability of successful steps, \(\mathbf{y}_w\), to appear again
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i; \lambda, \quad y_i \sim \mathcal{N}_i(0, C) \]

- mixture of distribution \( C \) and step \( y_w \), \( C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \)
- new distribution: \( C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \)
- ruling principle: the adaptation increases the probability of successful steps, \( y_w \), to appear again
Covariance Matrix Adaptation
Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i; \lambda, \quad y_i \sim \mathcal{N}_i(0, C) \]

new distribution (disregarding \( \sigma \))

- new distribution: \( C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \)
- ruling principle: the adaptation increases the probability of successful steps, \( y_w \), to appear again
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_{i:}\lambda, \quad y_i \sim \mathcal{N}_i(0, C) \]

movement of the population mean \( m \)

- new distribution: \( C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \)
- ruling principle: the adaptation increases the probability of successful steps, \( y_w \), to appear again
Covariance Matrix Adaptation
Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i : \lambda, \quad y_i \sim \mathcal{N}_i (0, C) \]

mixture of distribution \( C \) and step \( y_w \),
\[ C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \]

- new distribution: \( C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \)
- ruling principle: the adaptation increases the probability of successful steps, \( y_w \), to appear again

Loshchilov, Schoenauer & Sebag
Covariance Matrix Adaptation
Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_{i:;} \lambda, \quad y_i \sim \mathcal{N}_i(0, C) \]

- new distribution: \( C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \)
- ruling principle: the adaptation increases the probability of successful steps, \( y_w \), to appear again
Rank-$\mu$ Update

\[
\begin{align*}
    x_i &= m + \sigma y_i, & y_i &\sim \mathcal{N}(0, C), \\
    m &\leftarrow m + \sigma y_w, & y_w &= \sum_{i=1}^{\mu} w_i y_i: \lambda
\end{align*}
\]

Sampling of $\lambda = 150$ solutions where $C = I$ and $\sigma = 1$

Calculating $C$ from $\mu = 50$ points,

\[
\begin{align*}
    \mu &= 50 \\
    w_1 &= \cdots = w_\mu = \frac{1}{\mu}
\end{align*}
\]

Remark: the old (sample) distribution shape has a great influence on the new distribution $\rightarrow$ iterations needed
Summary of Equations
The Covariance Matrix Adaptation Evolution Strategy

Input: \( m \in \mathbb{R}^n, \sigma \in \mathbb{R}_+, \lambda \)
Initialize: \( C = I \), and \( p_c = 0, p_\sigma = 0 \),
Set: \( c_c \approx 4/n, c_\sigma \approx 4/n, c_1 \approx 2/n^2, c_\mu \approx \mu_w/n^2, c_1 + c_\mu \leq 1, \)
\( d_\sigma \approx 1 + \sqrt{\mu_w/n} \), and \( w_i=1\ldots\lambda \) such that \( \mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i} \approx 0.3 \lambda \)

While not terminate

\[ x_i = m + \sigma y_i, \quad y_i \sim N_i(0, C), \quad \text{for } i = 1, \ldots, \lambda \]

\[ m \leftarrow \sum_{i=1}^{\mu} w_i x_i; \lambda = m + \sigma y_w \quad \text{where } y_w = \sum_{i=1}^{\mu} w_i y_i; \lambda \]

\[ p_c \leftarrow (1 - c_c) p_c + 1 \{ \| p_\sigma \| < 1.5\sqrt{n} \} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} y_w \]

\[ p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w} C^{-1/2} y_w \]

\[ C \leftarrow (1 - c_1 - c_\mu) C + c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\mu} w_i y_i; \lambda y_i^T; \lambda \]

\[ \sigma \leftarrow \sigma \times \exp \left( \frac{c_\sigma}{d_\sigma} \left( \frac{\| p_\sigma \|}{E[\| \mathcal{N}(0, I) \|]} - 1 \right) \right) \]

Not covered on this slide:
termination, restarts, active or mirrored sampling, outputs, and boundaries
Invariances: Guarantee for Generalization

Invariance properties of CMA-ES

- **Invariance to order preserving transformations** in function space
  
  like all comparison-based algorithms

- **Translation and rotation invariance**
  
  to *rigid transformations* of the search space

---

CMA-ES is almost **parameterless**

- Tuning of a small set of functions
  
  Hansen & Ostermeier 2001

- Default values generalize to whole classes

- Exception: population size for multi-modal functions
  
  but see IPOP-CMA-ES Auger & Hansen, 05
  
  and BIPOP-CMA-ES Hansen, 09

---

Loshchilov, Schoenauer & Sebag
BBOB – Black-Box Optimization Benchmarking

- ACM-GECCO workshop, in 2009 and 2010 (and 2012)
- Set of 25 benchmark functions, dimensions 2 to 40
- With known difficulties (ill-conditioning, non-separability, ...)
- Noisy and non-noisy versions

Competitors include

- BFGS (Matlab version),
- Fletcher-Powell,
- DFO (Derivative-Free Optimization, Powell 04)
- Differential Evolution
- Particle Swarm Optimization
- and many more
Contents

1. CMA-ES, State-of-the-Art in Continuous Optimization
   - Black-Box Optimization and Search Template
   - On-line Adaptation
   - CMA-ES

2. Support Vector Machines and Rank-SVM
   - Statistical Machine Learning
   - Linear classifiers
   - The kernel trick

3. Rank-based Surrogate Models for CMA-ES
   - Choice of Kernel and Surrogate Lifetime
   - Size of Training Set and SVM Parameters
   - Experimental Results
Supervised Machine Learning

Context

Universe $\rightarrow$ instance $x_i$ $\rightarrow$ Oracle $\downarrow$ $y_i$

**Input:** Training set $\mathcal{E} = \{(x_i, y_i), i = 1 \ldots n, x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$

**Output:** Hypothesis $h : \mathcal{X} \mapsto \mathcal{Y}$

**Criterion:** Quality of $h$
Supervised Machine Learning, 2

The problem

- $\mathcal{E} = \{(x_i, y_i), x_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1 \ldots n\}$
  - Classification: $\mathcal{Y}$ finite
  - Regression: $\mathcal{Y} \subseteq \mathbb{R}$
- Hypothesis space $\mathcal{H}: \mathcal{X} \rightarrow \mathcal{Y}$

The tasks

- Select $\mathcal{H}$ (model selection)
- Find $h^*$ in $\mathcal{H}$ minimizing the error cost in expectation
  
  \[ h^* = \text{Arg min}_{h \in \mathcal{H}} \{ \mathbb{E}[\ell(h(x) \neq y)], h \in \mathcal{H} \} \]

- Under proper hypotheses on $\mathcal{H}$, this amounts to minimizing the empirical error
  
  \[ h^* = \text{Arg min}_{h \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i) \neq y_i) \right\} \]
Linear classification; the noiseless case

\[ H : X \subset \mathbb{R}^d \mapsto \mathbb{R} \]

\[ h(x) = \langle w, x \rangle + b \]

prediction = \( sgn(h(x)) \)
Linear classification; the noiseless case, 2

Example → Constraint

\[ y_i (\langle w, x_i \rangle + b) \geq \text{margin} \geq 0 \]

Maximize minimum margin \( \frac{2}{\|w\|} \)

Formalisation

\[
\begin{cases}
\text{Minimize} & \frac{1}{2} \|w\|^2 \\
\text{subject to} & \forall i, y_i (\langle w, x_i \rangle + b) \geq 1
\end{cases}
\]
Linear classification; the noiseless case, 3

Primal form

\[
\begin{cases}
\text{Minimize} & \frac{1}{2} \|w\|^2 \\
\text{subject to} & \forall i, \ y_i(\langle w, x_i \rangle + b) \geq 1
\end{cases}
\]

Using Lagrange multipliers:

\[
\text{Minimize}_{w,b} \max_\alpha \left\{ \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha_i[y_i(\langle w, x_i \rangle - b) - 1] \right\}
\]

Dual form

\[
\text{Maximize}_\alpha \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \right\}
\]

subject to \( \alpha_i \geq 0, i = 1 \ldots n \)

Optimization: quadratic programming

\[
\mathbf{w}^* = \sum \alpha_i y_i \mathbf{x}_i
\]
Linear classification; the noisy case

Allow constraint violations; introduce slack variables

Primal form

\[
\begin{align*}
\text{Minimize} & \quad \frac{1}{2} \|w\|^2 + C\sum_{i=1}^{n} \xi_i \\
\text{subject to} & \quad \forall \ i, \ y_i (\langle w, x_i \rangle + b) \geq 1 - \xi_i, \quad 0 \leq \xi_i
\end{align*}
\]

Lagrange multipliers:

\[
\begin{align*}
\text{Minimize } w, b, \xi \max_{\alpha, \beta} & \quad \frac{1}{2} \|w\|^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [y_i (\langle w, x_i \rangle - b) - 1 + \xi_i] - \sum_{i=1}^{n} \beta_i \xi_i \\
\text{subject to} & \quad 0 \leq \alpha_i \leq C, i = 1 \ldots n, \sum_{i=1}^{n} \alpha_i y_i = 0
\end{align*}
\]

Dual form

\[
\begin{align*}
\text{Maximize}_{\alpha} & \quad \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \right\}
\end{align*}
\]

Solution

\[
h(x) = \langle w^*, x \rangle = \sum \alpha_i y_i \langle x_i, x \rangle
\]
The kernel trick

Intuition

\[ X \xrightarrow{\Phi : x = (x_1, x_2)} \Omega \xrightarrow{\Omega = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)} \]

Principle: choose \( \Phi, K \) such that

\[ \langle \Phi(x), \Phi(x') \rangle = K(x, x') \]
The kernel trick, 2

SVM only considers the scalar product

\begin{align*}
h(x) &= \sum_i \alpha_i y_i \langle x_i, x \rangle & \text{linear case} \\
h(x) &= \sum_i \alpha_i y_i K(x_i, x) & \text{kernel trick}
\end{align*}

PROS

- A rich hypothesis space
- No computational overhead: no explicit mapping on the feature space
- Open problem: kernel design
The kernel trick, 3

Kernels

- **Polynomial**: \( k(x_i, x_j) = (\langle x_i, x_j \rangle + 1)^d \)
- **Gaussian or Radial Basis Function**: \( k(x_i, x_j) = \exp\left( \frac{||x_i - x_j||^2}{2\sigma^2} \right) \)
- **Hyperbolic tangent**: \( k(x_i, x_j) = \tanh(k \langle x_i, x_j \rangle + c) \)

Examples for Polynomial (left) and Gaussian (right) Kernels:
Regression SVM

Learning to approximate given values

- Labels are real values
- Expert gives a precision threshold $\varepsilon$

$\varepsilon$-insensitive soft constraints

- Primal form, linear

\[
\begin{align*}
\text{Minimize} & \quad \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi'_i) \\
\text{subject to} & \quad y_i - (\langle w, x_i \rangle + b) \leq \varepsilon + \xi_i; \quad 0 \leq \xi_i, \ i = 1, \ldots, N \\
& \quad \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi'_i; \quad 0 \leq \xi'_i, \ i = 1, \ldots, N
\end{align*}
\]

+ Kernel trick
Rank-based SVM

Learning to order things

- On training set $\mathcal{E} = \{x_i, i = 1 \ldots n\}$
- expert gives preferences: $(x_{i_k} \succ x_{j_k}), k = 1 \ldots K$
- *underconstrained regression*

Soft order constraints

- Primal form, linear

\[
\begin{align*}
\text{Minimize} & \quad \frac{1}{2} \| w \|^2 + C \sum_{k=1}^{K} \xi_k \\
\text{subject to} & \quad \langle w, x_{i_k} \rangle - \langle w, x_{j_k} \rangle \geq 1 - \xi_k; \ 0 \leq \xi_k, k = 1, \ldots, K
\end{align*}
\]

+ Kernel trick

Different from ordinal regression  
more degrees of freedom
Contents

1. CMA-ES, State-of-the-Art in Continuous Optimization
   - Black-Box Optimization and Search Template
   - On-line Adaptation
   - CMA-ES

2. Support Vector Machines and Rank-SVM
   - Statistical Machine Learning
   - Linear classifiers
   - The kernel trick

3. Rank-based Surrogate Models for CMA-ES
   - Choice of Kernel and Surrogate Lifetime
   - Size of Training Set and SVM Parameters
   - Experimental Results
Surrogate-Assisted Optimization (reminder)

Principle

- Gather $\mathcal{E} = \{(x_i, F(x_i))\}$
- Learn $\hat{F}$ from $\mathcal{E}$
- Use surrogate $\hat{F}$ in lieu of $F$ for some time
- Compute $F(x_i)$ for some new $x_i$
- Learn new surrogate $\hat{F}$ using some parameters
- Iterate
Surrogate-Assisted Optimization (reminder)

**Principle**

- Gather $\mathcal{E} = \{(x_i, F(x_i))\}$
- Learn $\hat{F}$ from $\mathcal{E}$
- Use surrogate $\hat{F}$ in lieu of $F$ for some time
- Compute $F(x_i)$ for some new $x_i$
- Learn new surrogate $\hat{F}$ using some parameters
- Iterate
Surrogate-Assisted Optimization (reminder)

**Principle**

- Gather $\mathcal{E} = \{(x_i, F(x_i))\}$ \textit{training set}
- Learn $\hat{F}$ from $\mathcal{E}$ \textit{surrogate model in a given class}
- Use surrogate $\hat{F}$ in lieu of $F$ for some time
- Compute $F(x_i)$ for some new $x_i$
- Learn new surrogate $\hat{F}$ using some parameters
- Iterate
Invariances-preserving Surrogate Model for CMA-ES

Rank-SVM

- Build a global model using Rank-SVM

\[ x_i \succ x_j \text{ iff } F(x_i) < F(x_j) \]

+ Preserves order invariance
- Kernel and parameters highly problem-dependent

Adaptive Informed Kernel

- Gaussian kernel using the covariance matrix \( C \) from CMA-ES
+ No computational overhead
+ Preserves coordinate-system invariance

Runarsson, 2006
Loschilov et al., 2010, 2012
About Model Learning

Non-separable Ellipsoid problem

\[ K(x_i, x_j) = e^{-\frac{(x_i - x_j)^T(x_i - x_j)}{2\sigma^2}}; \quad K_C(x_i, x_j) = e^{-\frac{(x_i - x_j)^T C^{-1}(x_i - x_j)}{2\sigma^2}} \]

Invariance to rotations of the search space!
Preliminary Experiments

Sensitivity w.r.t. surrogate lifetime

Simple optimization loop:
optimize $\mathcal{F}$ for 1 generation, then optimize $\hat{\mathcal{F}}$ for $\hat{n}$ generations.

How to choose $\hat{n}$?
Adaptation of Surrogate Lifetime

- Test set: \( \lambda \) recently evaluated points
- Model error: fraction of incorrectly ordered points

\[ n_{\text{max}} \]

\[ T_{\text{err}} \]

**Surrogate lifetime**: inversely proportional to model error
Preliminary Experiments
Sensitivity w.r.t. size of training set

Speed-up vs number of training points
(use most recently evaluated)
Rationale for Self-Adaptive ACM-ES

Sensitive SVM Parameters

- Size of training set
- “Step-size” of SVM Kernel
- Penalty for constraint violation (mantissa and exponent)

Self-adaptation of parameters

- Use an embedded optimizer (e.g., ... CMA-ES!)
- Minimizing model error
- Beware of overfitting → one single iteration
The proposed $s^*a$AAMC algorithm

1. Build Surrogate Model $\hat{f}(x)$ of $f(x)$ using model hyper-parameters $\alpha$
2. Optimize $\hat{f}(x)$ for $\hat{n}$ generations
3. Optimize $f(x)$ for 1 generation
4. Estimate model error $\text{Err}(\alpha)$ using last $\lambda$ evaluated points
5. Adjust number of generations $\hat{n}$
   - optionally
6. Optimize $\text{Err}(\alpha)$ for 1 generation
   - $\alpha = \text{new mean of distribution}$
7. Ranking SVM
8. CMA-ES #1 in space $x$
9. CMA-ES #1 in space $x$
10. CMA-ES #2 in space $\alpha$

Surrogate-assisted CMA-ES with online adaptation of model hyper-parameters.

Loshchilov, Schoenauer & Sebag

[GECCO 2012]
Online adaptation of model hyper-parameters

F8 Rosenbrock 20–D

Number of function evaluations

Value

Fitness

- $N_{\text{training}}$
- $C_{\text{base}}$
- $C_{\text{pow}}$
- $C_{\sigma}$

Loshchilov, Schoenauer & Sebag
Self-adaptation of model hyper-parameters is better than the best offline settings! (IPOP-\(s^*\)aACM vs IPOP-aACM)

Improvements of original CMA lead to improvements of its surrogate-assisted version (IPOP-\(s^*\)aACM vs IPOP-\(s^*\)ACM)
Comparative results
BIPOP-\textsuperscript{s}aACM and IPOP-\textsuperscript{s}aACM (with restarts) on 24 noiseless 20 dimensional functions
Summary and Conclusion

Invariances and Surrogates
- Invariances: guarantee for robustness of parameter tuning
- Surrogates should preserve invariances

Adaptation
- Key to success beyond parameter tuning
- ... with meaningful feedback from current state

Issues
- Parameter tuning vs Parameter Adaptation
- Generic Reward Design?
Rationale

- Time $t = \text{ground truth}$ for time $t - 1$
- Maximize likelihood of ground truth
- Use optimized parameters at time $t$

Proof of concept: other CMA-ES parameters

\[
C_c = \frac{4}{n+4},
C_1 = \frac{2}{(n+1.3)^2 + \mu_w},
C_\mu = \frac{2(\mu_w - 2 + 1/\mu_w)}{(n+2)^2 + \mu_w}
\]

\[0 \leq c_1 + c_\mu + c_c \leq 0.9\]