From Gradient-Based to Evolutionary Optimization

Nikolaus Hansen
Inria
CMAP, CNRS, Ecole Polytechnique, Institut Polytechnique de Paris
Outline

• Why is optimization difficult?

• From gradient descent to evolution strategy

• From first order (gradient descent) to second order (variable metric): CMA-ES

• Practical advice and code examples

…feel free to ask questions…
minimize an objective function

\[ f : \mathbb{R}^n \rightarrow \mathbb{R}, \ x \mapsto f(x) \]

- in theory: convergence to the global optimum

- in practice: find a good solution iteratively as quickly as possible
**Objective: Important Scenarios**

minimize an objective function

\[ f : \mathbb{R}^n \rightarrow \mathbb{R}, \ x \mapsto f(x) \]

- evaluating \( f \) is expensive and/or \textit{dominates} the costs
  hence quick means small number of \( f \) evaluations

- search space \textit{dimension} \( n \) is large

- we can (inexpensively) evaluated the \textit{gradient} of \( f \)
  \[ f(x) \in \mathbb{R}, \ \text{while} \ \nabla f(x) \in \mathbb{R}^n \]

- we can \textit{parallelize} the evaluations of \( f \)
minimize an objective function

\[ f : \mathbb{R}^n \rightarrow \mathbb{R}, \ x \mapsto f(x) \]

- evaluating \( f \) is expensive and/or dominates the costs
  hence quick means small number of \( f \) evaluations

- search space dimension \( n \) is large

- we can (inexpensively) evaluated the gradient of \( f \)
  \[ f(x) \in \mathbb{R}, \text{ while } \nabla f(x) \in \mathbb{R}^n \]

- we can parallelize the evaluations of \( f \)
Objective

minimize an objective function

\[ f : \mathbb{R}^n \rightarrow \mathbb{R}, \ x \mapsto f(x) \]

What Makes an Optimization Problem Difficult?
What Makes an Optimization Problem Difficult?

- non-linear, non-quadratic on linear and quadratic functions
  specialized search policies are available

- non-convexity

- dimensionality (size of search space) and non-separability
  dimension considerably larger than three with dependencies between the variables

- multimodality

- ruggedness
  high frequency modality, non-smooth, discontinuous

- ill-conditioning
  varying sensitivities, worst case: non-smooth concave level sets

In any case, the objective function must be highly regular
dimensionality: On Separable Functions

• Separable functions: for all $x \in \mathbb{R}^n$, for all $i$,

$$\arg \min_{x_i} f(x) \text{ is independent of } x$$

• Additively decomposable functions:

$$f(x) = \sum_{i=1}^{n} g_i(x_i), \quad x = (x_1, \ldots, x_n)$$

are separable

can be solved with $n$ one-dimensional optimizations
What Makes an Optimization Problem Difficult?

- non-linear, non-quadratic
- non-convexity
- dimensionality (size of search space) and non-separability
  - dimension considerably larger than three with dependencies between the variables
- multimodality
- ruggedness
  - high frequency modality, non-smooth, discontinuous
- ill-conditioning
  - varying sensitivities, worst case: non-smooth concave level sets

In any case, the objective function must be highly regular
Section Through a 5-Dimensional Rugged Landscape

\[
f : \mathbb{R}^n \rightarrow \mathbb{R}, \ x \mapsto f(x), \ n = 5
\]
What Makes an Optimization Problem Difficult?

- non-linear, non-quadratic
  on linear and quadratic functions
  specialized search policies are available

- non-convexity

- dimensionality (size of search space) and non-separability
  dimension considerably larger than three with
  dependencies between the variables

- multimodality

- ruggedness
  high frequency modality, non-smooth, discontinuous

- ill-conditioning
  varying sensitivities
  worst case: non-smooth concave level set
  gradient direction Newton direction

In any case, the objective function must be highly regular
Flexible Muscle-Based Locomotion for Bipedal Creatures

SIGGRAPH ASIA 2013

Thomas Geijtenbeek
Michiel van de Panne
Frank van der Stappen
Landscape of Continuous Search Methods

**Gradient-based (Taylor, local)**
- Conjugate gradient methods [Fletcher & Reeves 1964]
- Quasi-Newton methods (BFGS) [Broyden et al 1970]

**Derivative-free optimization (DFO)**
- Trust-region methods (NEWUOA, BOBYQA) [Powell 2006, 2009]
- Simplex downhill [Nelder & Mead 1965]
- Pattern search [Hooke & Jeeves 1961, Audet & Dennis 2006]

**Stochastic (randomized) search methods**
- Evolutionary algorithms (broader sense, continuous domain)
  - Differential Evolution [Storn & Price 1997]
  - Particle Swarm Optimization [Kennedy & Eberhart 1995]
  - Evolution Strategies [Rechenberg 1965, Hansen & Ostermeier 2001]
- Simulated annealing [Kirkpatrick et al 1983]
- Simultaneous perturbation stochastic approximation (SPSA) [Spall 2000]
Basic Approach: Gradient Descent

The gradient is the local direction of the maximal $f$ increase.

\[ f(x) = \text{const} \]

\[ -\nabla f(x) \]

$\mathbb{R}^2$

tangent space

here we
Basic Approach: Gradient Descent

The gradient is the local direction of the maximal $f$ increase
Basic Approach: Gradient Descent

The gradient is the local direction of the maximal $f$ increase

$$\nabla f(x) = - \sum_{i=1}^{n} w_i e_i \quad -w_i = \lim_{\delta \to 0} \frac{f(x + \delta e_i) - f(x)}{\delta}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$= x + \sigma \sum_{i=1}^{n} w_i e_i$$
Basic Approach: Gradient Descent

The *gradient* is the local direction of the maximal $f$ increase

$$\nabla f(x) = - \sum_{i=1}^{n} w_i e_i \quad -w_i = \lim_{\delta \to 0} \frac{f(x + \delta e_i) - f(x)}{\delta}$$

$x \leftarrow x - \sigma \nabla f(x)$

$= x + \sigma \sum_{i=1}^{n} w_i e_i$
Basic Approach: Gradient Descent

The \textit{gradient} is the local direction of the maximal \( f \) increase.

\[
\nabla f(x) \approx - \sum_{i=1}^{n} w_i e_i \quad - w_i = \lim_{\delta \to 0} \frac{f(x + \delta e_i) - f(x)}{\delta}
\]

\[
x \leftarrow x - \sigma \nabla f(x)
\]

\[
\approx x + \sigma \sum_{i=1}^{n} w_i e_i
\]
Basic Approach: Approximated Gradient Descent

We modify the gradient equation...

\[ \nabla f(x) \approx - \sum_{i=1}^{m} w_i y_i \]

\[ -w_i = \lim_{\delta \to 0} \frac{f(x + \delta y_i) - f(x)}{\delta} \]

\[ x \leftarrow x - \sigma \nabla f(x) \]

\[ \approx x + \sigma \sum_{i=1}^{m} w_i y_i \]
Basic Approach: Approximated Gradient Descent

We modify the gradient equation...

\[ y_i \sim N(0, I) \]

\[ -w_i = \lim_{\delta \to 0} \frac{f(x + \delta y_i) - f(x)}{\delta} \]

\[ x \leftarrow x - \sigma \nabla f(x) \]

\[ \approx x + \sigma \sum_{i=1}^{m} w_i y_i \]

Evolutionary Gradient Search (EGS) [Salmon 1998, Arnold & Salomon 2007]
Rank-Based Approximated Gradient Descent

Using ranks introduces invariance to order-preserving $f$-transformations.

\[ y_i \sim \mathcal{N}(0,I) \quad -w_i \propto \text{rank}_i(f(x + \delta y_i)) - \frac{m + 1}{2} \]

\[ x \leftarrow x - \sigma \nabla f(x) \]

\[ \approx x + \sigma \sum_{i=1}^{m} w_i y_i \]

Rank-Based Approximated Gradient Descent

Using ranks introduces invariance to order-preserving $f$-transformations.

$$y_i \sim \mathcal{N}(0, I)$$

$$-w_i = \frac{\ln \left( \text{rank}_i(f(x + \delta y_i)) \right) - \ln \frac{m + 1}{2}}{\frac{1}{2} \sum_{j=1}^{m} \ln(j)}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{m} w_i y_i$$

Invariance from Rank-Based Weights

Three functions belonging to the same equivalence class

A rank-based search algorithm is invariant under the transformation with any order preserving (strictly increasing) \( g \).

Invariances make

- observations meaningful as a rigorous notion of generalization
- algorithms predictable and/or "robust"
Comparison to BFGS, NEWUOA, PSO and DE

$f$ convex quadratic, separable with varying condition number $\alpha$

Ellipsoid dimension 20, 21 trials, tolerance $1e^{-09}$, eval max $1e+07$

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFGS</td>
<td>(Broyden et al 1970)</td>
</tr>
<tr>
<td>NEWUOA</td>
<td>(Powell 2004)</td>
</tr>
<tr>
<td>DE</td>
<td>(Storn &amp; Price 1996)</td>
</tr>
<tr>
<td>PSO</td>
<td>(Kennedy &amp; Eberhart 1995)</td>
</tr>
<tr>
<td>CMA-ES</td>
<td>(Hansen &amp; Ostermeier 2001)</td>
</tr>
</tbody>
</table>

$f(x) = g(x^T H x)$ with

- $H$ diagonal
- $g$ identity (for BFGS and NEWUOA)
- $g$ any order-preserving = strictly increasing function (for all other)

$SP1 = \text{average number of objective function evaluations}^{14}$ to reach the target function value of $g^{-1}(10^{-9})$

---

$^{14}$ Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA
Comparison to BFGS, NEWUOA, PSO and DE

\( f \) convex quadratic, non-separable (rotated) with varying condition number \( \alpha \)

Rotated Ellipsoid dimension 20, 21 trials, tolerance 1e−09, eval max 1e+07

\[ f(x) = g(x^T H x) \] with 

\( H \) full 

\( g \) identity (for BFGS and NEWUOA) 

\( g \) any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations\(^{15}\) to reach the target function value of \( g^{-1}(10^{-9}) \)

\(^{15}\) Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA
Comparing Experiments

Comparison to BFGS, NEWUOA, PSO and DE

\( f \) non-convex, non-separable (rotated) with varying condition number \( \alpha \)

Sqrt of sqrt of rotated ellipsoid dimension 20, 21 trials, tolerance 1e-09, eval max 1e+07

---

\[ f(x) = g(x^T H x) \] with

- \( H \) full
- \( g : x \mapsto x^{1/4} \) (for BFGS and NEWUOA)
- \( g \) any order-preserving = strictly increasing function (for all other)

---

**SP1** = average number of objective function evaluations\(^{16}\) to reach the target function value of \( g^{-1}(10^{-9}) \)

---

\(^{16}\) Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA

---
# From Gradient Search to Evolution Strategies

<table>
<thead>
<tr>
<th></th>
<th>Gradient Search</th>
<th>Evolution Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Steps:</strong></td>
<td>unit vectors</td>
<td>random vectors</td>
</tr>
<tr>
<td></td>
<td>dimension $n$</td>
<td>any number $&gt; 1$</td>
</tr>
<tr>
<td></td>
<td>small</td>
<td>large</td>
</tr>
<tr>
<td><strong>Weights:</strong></td>
<td>partial derivatives</td>
<td>fixed rank-based</td>
</tr>
<tr>
<td><strong>Realized Step Length:</strong></td>
<td>line search</td>
<td>step-size control (non-trivial)</td>
</tr>
</tbody>
</table>
What Makes an Optimization Problem Difficult?

- non-linear, non-quadratic
  on linear and quadratic functions
  specialized search policies are available
- non-convexity
- dimensionality (size of search space) and non-separability
  dimension considerably larger than three with
  dependencies between the variables
- multimodality
- ruggedness
  high frequency modality, non-smooth, discontinuous
- ill-conditioning
  varying sensitivities,
  worst case: non-smooth concave level sets

In any case, the objective function must be highly regular
Ill-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function
\[ f(x) = \frac{1}{2} (x - x^*)^T H (x - x^*) = \frac{1}{2} \sum_i h_{i,i} (x_i - x^*_i)^2 + \frac{1}{2} \sum_{i \neq j} h_{i,j} (x_i - x^*_i) (x_j - x^*_j) \]

\( H \) is Hessian matrix of \( f \) and symmetric positive definite

Ill-conditioning means squeezed level sets (high curvature).
Condition number equals nine here. Condition numbers up to \( 10^{10} \) are not unusual in real world problems.

If \( H \approx I \) (small condition number of \( H \)) first order information (e.g. the gradient) is sufficient. Otherwise second order information (estimation of \( H^{-1} \)) is necessary.
Rank-Based Approximated Gradient Descent

Using ranks introduces **invariance** to order-preserving $f$-transformations.

$$y_i \sim \mathcal{N}(0, I)$$

$$-w_i = \frac{\ln (\text{rank}_i(f(x + \delta y_i))) - \ln \frac{m+1}{2}}{\frac{1}{2} \sum_{j=1}^{m} \ln(j)}$$

$$x \leftarrow x - \sigma \nabla f(x)$$

$$\approx x + \sigma \sum_{i=1}^{m} w_i y_i$$

Rank-Based Approximated Gradient Descent

\[ y_i \sim \mathcal{N}(0, C) \]

\[ -w_i = \frac{\ln(\text{rank}_i(f(x + \delta y_i))) - \ln \frac{m + 1}{2}}{\frac{1}{2} \sum_{j=1}^{m} \ln(j)} \]

\[ x \leftarrow x - \sigma \nabla f(x) \]

\[ \approx x + \sigma \sum_{i=1}^{m} w_i y_i \]

variable metric, updated to estimate \( H^{-1} \)

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen & Ostermeier 2001, Hansen et al 2003]

From Gradient-Based to Evolutionary Optimization
Let \( m \in \mathbb{R}^n, \sigma > 0, C = I_n, y_0 = 0 \)

\[
x_k \sim \mathcal{N}(m, \sigma^2 C) = m + \sigma \mathcal{N}(0, C) \in \mathbb{R}^n, \quad k = 1 \ldots \lambda
\]

\[
y_k = \frac{x_{\text{permute}_i(k)} - m}{\sigma}
\]

sorted by fitness \( y_k \sim \mathcal{N}(0, C) \)

\[
m \leftarrow m + c_m \sigma \sum_{k=1}^{\mu} w_k y_k, \quad c_m \approx \sum_{k=1}^{\mu} w_k \approx 1, \quad \mu \approx \lambda / 2
\]

\[
y_0 \leftarrow (1 - c_c) y_0 + \sqrt{c_c (2 - c_c) \mu_w} \sum_{k=1}^{\mu} w_k y_k, \quad \mu_w = \frac{(\sum_{i=1}^{\mu} w_k)^2}{\sum_{i=1}^{\mu} w_k^2}
\]

\[
C \leftarrow C + c_\mu \sum_{k=0}^{\lambda} w_k (y_k y_k^\top - C), \quad c_\mu \approx \lambda / n^2, \quad \sum_{k=0}^{\lambda} w_k \approx 0
\]

\[
\sigma \leftarrow \sigma \times \exp(\ldots)
\]
http://blog.otoro.net/2017/10/29/visual-evolution-strategies/
CMA-ES
Covariance Matrix Adaptation Evolution Strategy

• Strive to sample the optimal (multi-variate) Gaussian distribution at any given iteration

• “optimal” mean (best estimate of the optimum) given the available information

• optimal covariance matrix $C$ given the available information

• optimal step-size $\sigma$ given the covariance matrix

• A natural gradient update of mean and covariance matrix provides a theoretical framework/justification [JMLR 18(18), 2017]

\[
\theta_{t+1} = \theta_t + \eta \frac{1}{Z(\lambda)} \sum_{k=1}^{\lambda} \left( \frac{\lambda}{2} - \text{rank}(f(x_k)) \right) \nabla_{\theta} \ln p(x_k | \theta) \bigg|_{\theta=\theta_t}
\]

• Convergence speed is almost independent of the number of samples (not $\gg n$) property of multi-recombinative Evolution Strategies
Practical Advice
Approaching an Unknown Optimization Problem

- **Objective formulation**
  
  for example $\sum_i x_i^2$ and $\sum_i |x_i|$ have the same optimal (minimal) solution but may be very differently “optimizable”

- **Problem/variable encoding**
  
  for example log scale vs linear scale vs quadratic transformation

- **Create section plots ($f(x)$ vs $x$ on a line)**
  
  one-dimensional grid search is cheap, may reveal ill-conditioning or multi-modality

- **Try to locally improve a given (good) solution**

- **Start local search from different initial solutions**
  
  Ending up always in different solutions? Or always in the same?

- **Apply “global search” setting**

- **see also** [http://cma.gforge.inria.fr/cmaes_sourcecode_page.html#practical](http://cma.gforge.inria.fr/cmaes_sourcecode_page.html#practical)
Project description

A stochastic numerical optimization algorithm for difficult (non-convex, ill-conditioned, multi-modal, rugged, noisy) optimization problems in continuous search spaces, implemented in Python.

Typical domain of application are bound-constrained or unconstrained objective functions with:

- search space dimension between, say, 5 and (a few) 100,
- no gradients available,
- at least, say, 100 times dimension function evaluations needed to get satisfactory solutions,
- non-separable, ill-conditioned, or rugged/multi-modal landscapes.

The CMA-ES is quite reliable, however for small budgets (fewer function evaluations than, say, 100 times dimension) or in very small dimensions better (i.e. faster) methods are available.

The `pycma` module provides two independent implementations of the CMA-ES algorithm in the classes `cma.CMAEvolutionStrategy` and `cma.purecma.CMAES`.

Installation

There are several ways of installation:

- In the terminal command line type:
Python Example in Jupyter-Lab

```python
# download & install anaconda python
# optional: "conda create" in case a different Python version is needed
# shell cmd "pip install cma" to install a CMA-ES module (or see github)
# shell cmd "jupyter-notebook" or "jupyter-lab"

%pylab ipympl
import cma

Populating the interactive namespace from numpy and matplotlib

```
Figure 326

\[ f(x) = \sum_{i=1}^{n} \alpha_i x_i^2 \]

\[ f(x) = n \sum_{i=1}^{n} \alpha_i x_i^2 \]
function evaluations

1 $x, es = \text{cma.fmin2}(\text{cma.ff.elli}, 11 \times [1], 1e-5)$

(5_w,11)-aCMA-ES (mu_w=3.4,w_1=42%) in dimension 11 (seed=909918, Thu Jul 11 10:51:45 2019)
Iterat #Fevals function value axis ratio sigma min\$\text{\&}\$ max std t[m:s]
1 11 1.335426651544717e+06 1.0e+00 9.41e-06 9e-06 9e-06 0:00.0
2 22 1.335403970874783e+06 1.2e+00 1.02e-05 1e-05 1e-05 0:00.0
3 33 1.335381895674725e+06 1.3e+00 1.13e-05 1e-05 1e-05 0:00.0
41 451 1.23641361768105e+06 3.4e+00 7.70e-03 6e-03 1e-02 0:00.1
termination on tolfacupx=1000.0 (Thu Jul 11 10:51:45 2019)
final/bestever f-value = 1.239692e+06 1.236413e+06
incumbent solution: [1.01162517 0.99668741 0.98913604 1.01240495 0.99246088 0.98896483
  0.9867979 0.98844654 ...]
std deviations: [0.00708466 0.0063847 0.00740649 0.00944938 0.00763342 0.00751928
  0.00737737 0.00689999 ...]

1 $x, es = \text{cma.fmin2}(\text{cma.ff.elli}, 11 \times [1], 1e-5, \{'tolfacupx': 1e6\})$

(5_w,11)-aCMA-ES (mu_w=3.4,w_1=42%) in dimension 11 (seed=942830, Thu Jul 11 10:52:23 2019)
Iterat #Fevals function value axis ratio sigma min\$\text{\&}\$ max std t[m:s]
1 11 1.335419049661035e+06 1.0e+00 1.03e-05 1e-05 1e-05 0:00.0
2 22 1.335398380223496e+06 1.2e+00 1.19e-05 1e-05 1e-05 0:00.0
3 33 1.3353793953540697e+06 1.4e+00 1.38e-05 1e-05 1e-05 0:00.0
100 1100 4.99925215133446e+03 6.5e+00 2.01e-01 5e-02 2e-01 0:00.1
200 2200 1.030349103924029e+02 9.9e+01 1.24e-01 4e-03 3e-01 0:00.2
300 3300 9.939855671544342e+00 3.0e+02 3.53e-02 4e-04 9e-02 0:00.3
400 4400 2.862006949075298e-04 1.1e+03 1.55e-03 6e-06 6e-03 0:00.4
500 5500 2.373014204647906e-10 1.1e+03 3.30e-06 6e-09 6e-06 0:00.6
560 6160 1.84063507390707e-14 1.0e+03 5.18e-08 6e-11 5e-08 0:00.7
termination on tolfun=1e-11 (Thu Jul 11 10:52:24 2019)
final/bestever f-value = 9.862934e-15 9.862934e-15
incumbent solution: [-1.83502574e-08 -2.46920059e-08 -2.79112525e-09 1.76731134e-09
  -1.63323394e-09 -3.73512722e-10 4.61670724e-10 -2.78078340e-10 ...]
Figure 327

Object Variables (curr best, 11-D, popsize=11)

Principle Axes Lengths

Standard Deviations × σ⁻¹ in All Coordinates

Nikolaus Hansen, Inria, IP Paris

<cmagrammer.CMADatLogger at 0x7f90c02fd828>

Gradient-Based to Evolutionary Optimization
A Transparent Interface

```
es = cma.CMAEvolutionStrategy([0], 1.0)
while not es.stop():
    X = es.ask()
    es.tell(X, [cma.ff.rosen(x) for x in X])
    es.logger.add()
    es.disp()
es.result_pretty()
```

(5_w,10)—aCMA-ES (mu_w=3.2, w_1=45%) in dimension 8 (seed=610691, Sat Jul 6 20:59:48 2019)
Iterat #Evals function value axis ratio sigma min&max std t[m:s]
1 10 2.655345399152838e+02 1.0e+00 8.69e-01 8e-01 9e-01 0:00.0
2 20 4.50584260898847e+02 1.1e+00 7.90e-01 7e-01 8e-01 0:00.0
3 30 4.555527678670772e+02 1.2e+00 6.54e-01 6e-01 6e-01 0:00.0
100 1000 4.338593958650433e+00 8.1e+00 5.45e-02 2e-02 5e-02 0:00.1
200 2000 4.18533422985505e-01 1.5e+01 4.29e-02 4e-03 2e-02 0:00.2
300 3000 7.70046946188576e-06 6.3e+01 1.78e-03 6e-05 1e-03 0:00.2
400 4000 5.768459312593045e-13 6.5e+01 2.38e-06 2e-08 7e-07 0:00.3
417 4170 3.324045089491747e-14 7.4e+01 5.66e-07 4e-09 1e-07 0:00.3

termination on tolfun=1e-11
final/bestever f-value = 3.324045e-14 3.324045e-14
incumbent solution: [1.000000049914348, 1.0000000000632427, 1.0000000033696235, 1.0000000043215198, 1.00000000018643387, 1.00000000034016863, 1.0000000062489975, 1.000000017143993]
std deviation: [3.6509133595847e-09, 4.633054743103208e-09, 5.686358351473224e-09, 8.13637244689455e-09, 1.69283467913827e-08, 3.289709422562294e-08, 6.4072872918568e-08, 1.282870272399136e-07]

```
cma.plot()
```

Figure 328

Object Variables (curr best, 8-D, popsize~10)
Figure 328

Object Variables (curr best, 8-D, popsize~10)

Principle Axes Lengths

Standard Deviations x $\sigma^{-1}$ in All Coordinates

$\min(f)=3.3280450094917473e-14$

$cma.plot()$
Thank You