Optimization and Data Science

Lecture 19: Evolution Strategies for Optimization

Prof. Dr. Thomas Slawig

Kiel University - CAU Kiel
Dep. of Computer Science

Summer 2020

Contents

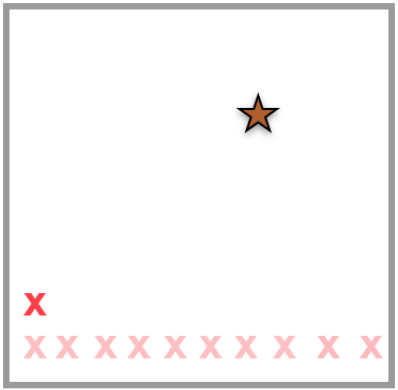
Evolution Strategies for Optimization

- What is it?
  Evolutionary strategies (or evolutionary algorithms, also genetic algorithms)
  Mimic the biological evolution for optimization
  Iterative algorithms including stochastic parts
  Sometimes termed "global" methods, but this is debatable
- Why are we studying this?
  Main class of optimization algorithms
  Need only function evaluations, no gradients, easily parallelizable
- How does it work?
  Evaluate function at a number of points ("population"), changing them randomly
  "Survival of the fittest"
- What if we can use it?
  Have an alternative to gradient-based method
  Combination (hybrid methods) possible

Contents

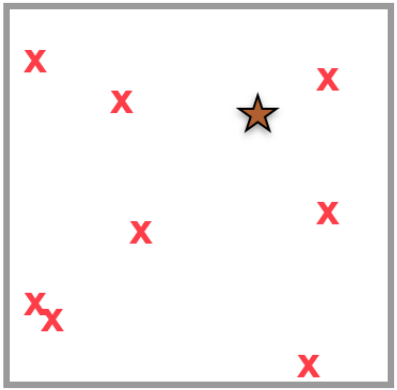
Global deterministic search

- Search for minimizer ★ checking all points on a grid row by row (or column by column) in the feasible set.



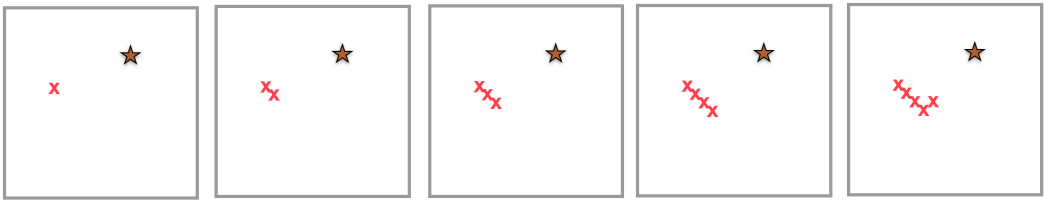
Global stochastic search

- Search for minimizer ★ checking randomly points in the feasible set.



Local deterministic search

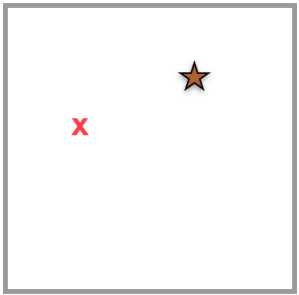
- Searching for the minimizer ★ following a given rule (e.g., negative gradient direction).



etc.

Local stochastic search

- Search at a number of randomly chosen points in the vicinity of the last point.



randomly chosen initial point

Comparison the effort: Global deterministic search

- Assumption: desired accuracy in parameter $x_i \in [a_i, b_i]$:

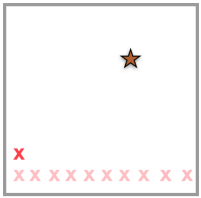
$$\frac{b_i - a_i}{10^k} \text{ mit } k \in \mathbb{N} \text{ arbitrary.}$$

- Example: $k = 2 \rightsquigarrow$ Accuracy 1 % of interval size $b_i - a_i$.
- $\rightsquigarrow$ we need 10^k points in every dimension.
- For n parameters:

$$10^{k \cdot n} \text{ points} = \text{function evaluations}$$

- Example: $k = 2, n = 7 \rightsquigarrow 10^{14}$ points/evaluations.
- Example: $k = 2, n = 40 \rightsquigarrow 10^{80} \approx \#$ atoms in the universe.

$\rightsquigarrow$ Not feasible.



Comparison effort: Global stochastic search (1)

- With accuracy as above:

$$10^{k \cdot n} \text{ grid points in total}$$

- Now: random choice. Probability to **find** minimizer in 1st guess:

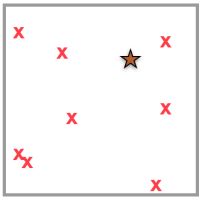
$$P = \frac{1}{10^{k \cdot n}}$$

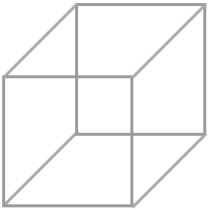
- Probability to **not find** minimizer in 1st guess:

$$P = 1 - \frac{1}{10^{k \cdot n}}$$

- Probability to **not find** minimizer after m guesses:

$$P = \left(1 - \frac{1}{10^{k \cdot n}}\right)^m$$





Comparison effort: Global stochastic search (2)

- Probability to **find** minimizer after m guesses

$$P = 1 - \left(1 - \frac{1}{10^{k \cdot n}}\right)^m \Leftrightarrow 1 - P = \left(1 - \frac{1}{10^{k \cdot n}}\right)^m \Leftrightarrow \log(1 - P) = m \log\left(1 - \frac{1}{10^{k \cdot n}}\right)$$

- Number m of steps to find minimizer with $P = 0.95$:

$$m = \frac{\log(1 - P)}{\log\left(1 - \frac{1}{10^{k \cdot n}}\right)} \approx \frac{\log(1 - P)}{-\frac{1}{10^{k \cdot n}}} = -10^{k \cdot n} \log(1 - P)$$

- Example: $P = 0.95 \Rightarrow \log(1 - P) \approx -3 \Rightarrow m \approx 3 \times 10^{k \cdot n}$.

$\rightsquigarrow$ 3 times the number for the deterministic search.

$\rightsquigarrow$ Also unfeasible.

$\rightsquigarrow$ Need local methods.

Contents

Basics

- Evolutionary Strategies (ES)
- ... or Evolutionary Algorithms
- ... or Genetic Algorithms (GA).
- Pioneer: Ingo Rechenberg (*1934, Prof. at TU Berlin)
- Emulation of Darwin's theory of evolution:
  - Inheritance
  - Mutations
  - Recombination of genes
  - Survival of the fittest.
- No derivatives needed.
- Parallelizable.
- Still a local method.

$(\mu + \lambda)$-Evolution Strategy

1. Define $\mu \in \mathbb{N}_{>0}$: # parents, $\lambda \in \mathbb{N}_{>0}$: # offspring.
2. Choose initial population $P = \{x_1, \ldots, x_\mu\}$ and mutability $\delta > 0$.
3. For every generation $k = 0, 1, \ldots$:
   1. Variation/Mutation: For $i = 1, \ldots, \lambda$:
      - choose parent randomly, i.e., choose $j \in \{1, \ldots, \mu\}$,
      - generate offspring: $\hat{x}_i := x_j + \delta z$,
      - using a random vector $z \in \mathcal{N}(0, \frac{1}{n})^n$.
   2. Selection: Choose $P \subset \hat{P} := \{x_1, \ldots, x_\mu, \hat{x}_1, \ldots, \hat{x}_\lambda\}, |P| = \mu$, such that
   $$\max\{f(x) : x \in P\} \leq \min\{f(x) : x \in \hat{P} \setminus P\}.$$
   3. If a stopping criterion is satisfied: stop.

$(\mu + \lambda)$-ES with different mutabilities

1. Define $\mu \in \mathbb{N}_{>0}$: # parents, $\lambda \in \mathbb{N}_{>0}$: # offspring.
2. Choose initial population $P = \{x_1, \ldots, x_\mu\}$ and mutabilities $\delta_j > 0, j = 1, \ldots, \mu$.
3. For every generation $k = 0, 1, \ldots$:
   1. Variation/Mutation: For $i = 1, \ldots, \lambda$:
      - choose parent randomly, i.e., choose $j \in \{1, \ldots, \mu\}$,
      - generate offspring: $\hat{x}_i := x_j + \delta_j z$,
      - using a random vector $z \in \mathcal{N}(0, \frac{1}{n})^n$.
   2. Selection: Choose $P \subset \hat{P} := \{x_1, \ldots, x_\mu, \hat{x}_1, \ldots, \hat{x}_\lambda\}, |P| = \mu$, such that
   $$\max\{f(x) : x \in P\} \leq \min\{f(x) : x \in \hat{P} \setminus P\}.$$
   3. If a stopping criterion is satisfied: stop.

$(\mu + \lambda)$-ES with mutated mutabilities

1. Define $\mu \in \mathbb{N}_{>0}$: # parents, $\lambda \in \mathbb{N}_{>0}$: # offspring, parameter $\alpha > 0$.
2. Choose initial population $P = \{x_1, \ldots, x_\mu\}$ and mutabilities $\delta_j > 0, j = 1, \ldots, \mu$.
3. For every generation $k = 0, 1, \ldots$:
   1. Variation/Mutation: For $i = 1, \ldots, \lambda$:
      - choose parent randomly, i.e., choose $j \in \{1, \ldots, \mu\}$,
      - generate offspring: $\hat{x}_i := x_j + \hat{\delta}_i z$,
      - using a random vector $z \in \mathcal{N}(0, \frac{1}{n})^n$
      - and a mutated mutability: $\hat{\delta}_i := \xi \delta_j$
      - where $\xi \in \{\alpha, \frac{1}{\alpha}\}$ is chosen randomly. (Rechenberg: $\alpha = 1.3$)
   2. Selection: Choose $P \subset \hat{P} := \{x_1, \ldots, x_\mu, \hat{x}_1, \ldots, \hat{x}_\lambda\}, |P| = \mu$, such that

$$\max\{f(x) : x \in P\} \leq \min\{f(x) : x \in \hat{P} \setminus P\}.$$

   3. If a stopping criterion is satisfied: stop.

$(\mu + \lambda)$-ES with Mutation Strategy Parameter Control (MSC): Inheritance of mutabilities of best individuals

1. Define $\mu \in \mathbb{N}_{>0}$: # parents, $\lambda \in \mathbb{N}_{>0}$: # offspring, parameter $\alpha > 0$.
2. Choose initial population $P = \{x_1, \ldots, x_\mu\}$ and mutabilities $\delta_j > 0, j = 1, \ldots, \mu$.
3. For every generation $k = 0, 1, \ldots$:
   1. Variation/Mutation: For $i = 1, \ldots, \lambda$:
      - choose parent randomly, i.e., choose $j \in \{1, \ldots, \mu\}$,
      - generate offspring: $\hat{x}_i := x_j + \hat{\delta}_i z$,
      - using a random vector $z \in \mathcal{N}(0, \frac{1}{n})^n$
      - and a mutated mutability: $\hat{\delta}_i := \xi \delta_j$
      - where $\xi \in \{\alpha, \frac{1}{\alpha}\}$ is chosen randomly.
   2. Selection: Choose $P \subset \hat{P} := \{x_1, \ldots, x_\mu, \hat{x}_1, \ldots, \hat{x}_\lambda\}, |P| = \mu$, such that

$$\max\{f(x) : x \in P\} \leq \min\left\{f(x) : x \in \hat{P} \setminus P\right\}.$$

   3. Save corresponding mutabilities: $\delta_i := \hat{\delta}(x_i)$ für $x_i \in P, i = 1, \ldots, \mu$.
   4. If a stopping criterion is satisfied: stop.

Variant: (μ, λ)-ES with MSC

- Parents are not included in new generation.
- Changes in the algorithm:

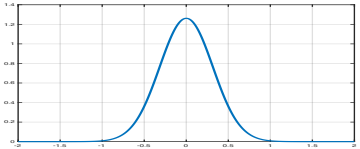
1. ... $\mu < \lambda$.
2. ...
3. For every generation $k = 0, 1, \ldots$:
   1. Variation/Mutation: ...
   2. Selection: Choose $P \subset \hat{P} := \{\hat{x}_1, \ldots, \hat{x}_\lambda\}, |P| = \mu$, such that

$$\max\{f(x) : x \in P\} \leq \min\left\{f(x) : x \in \hat{P} \setminus P\right\}.$$

   3. Save corresponding mutabilities: ...
   4. If a stopping criterion is satisfied: stop.

Random numbers in the ES

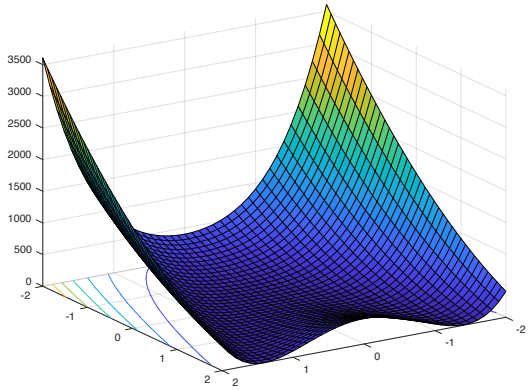
1. ....
2. Choose initial population ...
3. For every generation $k = 0, 1, \ldots$:
   1. Variation/Mutation: For $i = 1, \ldots, \lambda$:
      - choose parent ... $j \in \{1, \ldots, \mu\}$, ← uniform distribution, same probability for every j
      - generate offspring: $\hat{x}_i := x_j + \delta_i z$,
      - using a random vector $z \in \mathcal{N}(0, \frac{1}{n})^n$ ← normal distribution with variance $\frac{1}{n}$

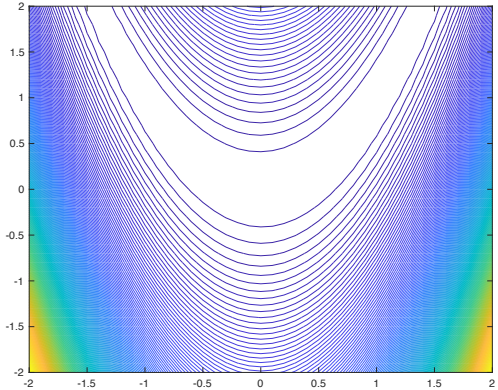


      - and a mutated mutability: $\delta_i := \xi \delta_j$
      - where $\xi \in \{\alpha, \frac{1}{\alpha}\}$ is chosen randomly. ← uniform distribution, same probability for both values
   2. Selection: ...

Example: Optimization of Rosenbrock function

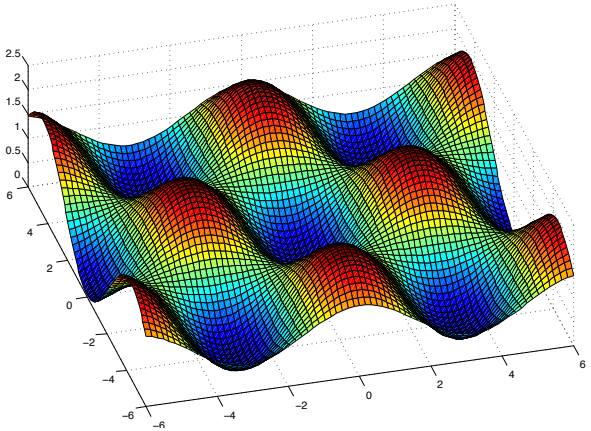
$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x_1, x_2) = 100 \left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

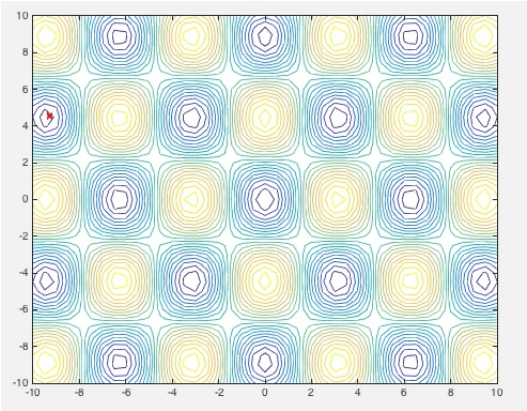




Example: Optimization of Griewank function

$$f : \mathbb{R}^n \to \mathbb{R}, \quad f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$





Contents

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

- Extension of the original idea, also inspired by Rechenberg.
- Elaborated by two of his students: see Hansen/Ostermeier: *Completely Derandomized Self-Adaptation in Evolution Strategies*, in Evolutionary Computation 9(2), 2001.
- Main differences:

  (1) Offspring not generated by mutation of single individual:

  $$\text{choose } j \text{ randomly, ... } \hat{x}_i := x_j + \delta_i z,$$

  but from weighted mean of the current population:

  $$\hat{x}_i := \bar{x} + \delta_i z, \quad \text{where } \bar{x} := \frac{1}{\sum_{j=1}^{\mu} w_j} \sum_{j=1}^{\mu} w_j x_j, \quad w_j > 0.$$

  - This can be seen as recombination of the best individuals.
  - Best individuals may get highest weights w_j.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

(2) Random perturbations z are not chosen to be independent for each i,

$$\hat{x}_i := \bar{x} + \delta_i z,$$

but chosen such that

$$z \sim \mathcal{N}(0, C),$$

where C is a given covariance matrix.

- This can be realized by
  - generating a random vector $\bar{z} \sim \mathcal{N}(0, I) = \mathcal{N}(0,1)^n$,
  - computing the singular value decomposition (svd) of

$$C = U\Sigma U^\top, \quad \text{(note: } C \text{ is symmetric)}$$

  - and setting

$$\hat{x}_i := \bar{x} + \delta_i U \Sigma^{\frac{1}{2}} \bar{z}, \quad \Sigma^{\frac{1}{2}} = \operatorname{diag}\left(\left(\sigma_i^{\frac{1}{2}}\right)_{i=1}^{n}\right).$$

Covariance adaptation in CMA-ES

(3) The covariance matrix C is updated in every iteration using a rank-1-update:

$$\hat{C} := (1-\alpha)C + \alpha \hat{p}\hat{p}^\top, \quad \alpha \in (0,1].$$

using

$$\hat{p} := (1-\beta)p + \gamma U \Sigma^{\frac{1}{2}} \left(\bar{\hat{x}} - \bar{x}\right).$$

The parameters α, β, γ are chosen appropriately.

(4) Also the step-size δ is chosen differently.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

- Offspring generated by mutation of weighted mean of the current population:

$$\hat{x}_i := \bar{x} + \delta_i U \Sigma^{\frac{1}{2}} \bar{z}, \quad \bar{z} \sim \mathcal{N}(0, I).$$

- With $Cov(AX) = A\, Cov(X) A^\top$ we get:

$$Cov(U\Sigma^{\frac{1}{2}}\bar{z}) = U\Sigma^{\frac{1}{2}}\, Cov(\bar{z})(U\Sigma^{\frac{1}{2}})^\top = U\Sigma^{\frac{1}{2}} I (U\Sigma^{\frac{1}{2}})^\top = U\Sigma^{\frac{1}{2}}\Sigma^{\frac{1}{2}}U^\top = U\Sigma U^\top = C.$$

- We thus get

$$z := U\Sigma^{\frac{1}{2}}\bar{z} \sim \mathcal{N}(0, C),$$

if (U, Σ) is computed as the singular value decomposition (svd) of C.

Contents

Sparse Evolutionary Training (SET) Algorithm

Mocanu, Mocanu, Stone, Nguyen, Gibescu, Liotta: Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science, in Nature Communications 9, 2018.

1. Initialize fully connected ANN, i.e., define # layers, # nodes per layer.
2. Set parameters $\varepsilon \in \mathbb{R}_{>0}, \zeta \in (0,1)$.
3. Replace every layer ℓ by sparse connected layer:
   - Define a uniformly distributed random matrix $W^\ell := (w_{\ell ij})_{ij} \in [0,1]^{N_\ell \times N_{\ell-1}}$
   - If $w_{\ell ij} > P_\ell := \dfrac{\varepsilon(N_\ell + N_{\ell-1})}{N_\ell N_{\ell-1}}$: remove edge (i,j).
4. Initialize training parameters.
5. For each training epoch ($\hat{=}$ iteration k in the optimization algorithm):
   1. Perform iteration and update parameters.
   2. For each layer:
      1. Remove a fraction ζ of weights with smallest absolute values (and corresponding edges).
      2. Add the same number of new weights (and edges) randomly (except in the last epoche).

What is important

- Evolutionary Strategies are optimization methods that mimic the natural biological evolution.
- They build a population of parameter vectors (=individuals), change them randomly in a given mutation radius and chose the best individuals to survive in the next iteration.
- They are still local methods.
- There are different options for the choice of population size, number of surviving individuals and mutation radius.
- The methods need an adaptive setting of the mutation radius to become efficient.
- A sophisticated version is the covariance-adapted method (CMAES) that can be found as library.
- The methods are easily parallelizable and especially good for cost functions that show a non-smooth behavior, maybe due to data uncertainties.