technische universität
dortmund

Computational Intelligence

Winter Term 2018/19

Prof. Dr. Günter Rudolph

Lehrstuhl für Algorithm Engineering (LS 11)

Fakultät für Informatik

TU Dortmund

mutation: Y = X + Z Z ~ N(0, C) multinormal distribution

↓

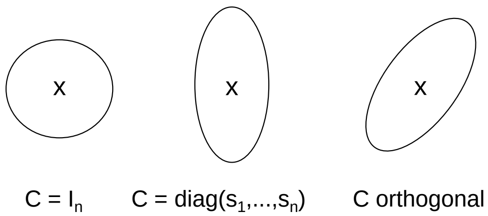
maximum entropy distribution for support $\mathbb{R}^n$, given expectation vector and covariance matrix

how should we choose covariance matrix C?

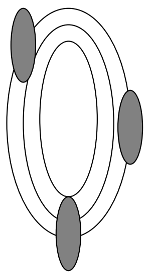
unless we have not learned something about the problem during search

⇒ don‘t prefer any direction!

⇒ covariance matrix C = I_n (unit matrix)



claim: mutations should be aligned to isolines of problem (Schwefel 1981)



if true then covariance matrix should be inverse of Hessian matrix!

⇒ assume f(x) ≈ ½ x‘Ax + b‘x + c ⇒ H = A

Z ~ N(0, C) with density

$$f_Z(x) = \frac{1}{(2\pi)^{n/2}\,|C|^{1/2}} \exp\left(-\frac{1}{2}x'C^{-1}x\right)$$

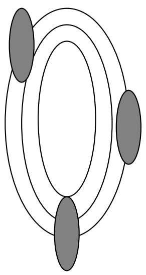
since then many proposals how to adapt the covariance matrix

⇒ extreme case: use n+1 pairs (x, f(x)),

apply multiple linear regression to obtain estimators for A, b, c

invert estimated matrix A! OK, **but:** O(n^6)! (Rudolph 1992)

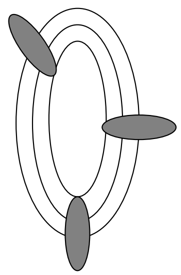
doubts: are equi-aligned isolines really optimal?



principal axis →

should point into negative gradient direction!

(proof next slide)



most (effective) algorithms behave like this:

run roughly into negative gradient direction,
sooner or later we approach longest main principal axis of Hessian,

now negative gradient direction coincidences with direction to optimum,
which is parallel to longest main principal axis of Hessian,
which is parallel to the longest main principal axis of the inverse covariance matrix

(Schwefel OK in this situation)

$Z = rQu$, $A = B'B$, $B = Q^{-1}$

$$
\begin{aligned}
f(x + rQu) &= \tfrac{1}{2}\,(x + rQu)'A\,(x + rQu) + b'(x + rQu) + c \\
&= \tfrac{1}{2}\,(x'Ax + 2\,rx'AQu + r^2u'Q'AQu) + b'x + rb'Qu + c \\
&= f(x) + rx'AQu + rb'Qu + \tfrac{1}{2}\,r^2u'Q'AQu \\
&= f(x) + r(Ax + b + \tfrac{r}{2}AQu)'Qu \\
&= f(x) + r(\nabla f(x) + \tfrac{r}{2}AQu)'Qu \\
&= f(x) + r\,\nabla f(x)'Qu + \tfrac{r^2}{2}\,u'Q'AQu \\
&= f(x) + r\,\nabla f(x)'Qu + \tfrac{r^2}{2}
\end{aligned}
$$

if Qu were deterministic ...

⇒ set $Qu = -\nabla f(x)$ (direction of steepest descent)

Apart from (inefficient) regression, how can we get matrix elements of Q?

⇒ iteratively: $C^{(k+1)} = \text{update}(\, C^{(k)}, \text{Population}^{(k)}\,)$

basic constraint: $C^{(k)}$ must be positive definite (p.d.) and symmetric for all $k \geq 0$, otherwise Cholesky decomposition impossible: $C = Q'Q$

Lemma

Let A and B be quadratic matrices and $\alpha, \beta > 0$.

a) A, B symmetric ⇒ $\alpha A + \beta B$ symmetric.

b) A positive definite and B positive semidefinite ⇒ $\alpha A + \beta B$ positive definite

Proof:

ad a) $C = \alpha A + \beta B$ symmetric, since $c_{ij} = \alpha\, a_{ij} + \beta\, b_{ij} = \alpha\, a_{ji} + \beta\, b_{ji} = c_{ji}$

ad b) $\forall x \in \mathbb{R}^n \setminus \{0\}$: $x'(\alpha A + \beta B)\,x = \underbrace{\alpha\, x'Ax}_{>0} + \underbrace{\beta\, x'Bx}_{\geq 0} > 0$ ■

Theorem

A quadratic matrix $C^{(k)}$ is symmetric and positive definite for all $k \geq 0$,

if it is built via the iterative formula $\mathbf{C^{(k+1)} = \alpha_k\, C^{(k)} + \beta_k\, v_k\, v'_k}$

where $C^{(0)} = I_n$, $v_k \neq 0$, $\alpha_k > 0$ and liminf $\beta_k > 0$.

Proof:

If $v \neq 0$, then matrix $V = vv'$ is symmetric and positive semidefinite, since

- as per definition of the dyadic product $v_{ij} = v_i \cdot v_j = v_j \cdot v_i = v_{ji}$ for all i, j and
- for all $x \in \mathbb{R}^n$: $x'\,(vv')\,x = (x'v) \cdot (v'x) = (x'v)^2 \geq 0$.

Thus, the sequence of matrices $v_k v'_k$ is symmetric and p.s.d. for $k \geq 0$.

Owing to the previous lemma matrix $C^{(k+1)}$ is symmetric and p.d., if

$C^{(k)}$ is symmetric as well as p.d. and matrix $v_k v'_k$ is symmetric and p.s.d.

Since $C^{(0)} = I_n$ symmetric and p.d. it follows that $C^{(1)}$ is symmetric and p.d.

Repetition of these arguments leads to the statement of the theorem. ■

Idea: Don't estimate matrix **C** in each iteration! Instead, approximate iteratively!

(Hansen, Ostermeier et al. 1996ff.)

→ **C**ovariance **M**atrix **A**daptation **E**volutionary **A**lgorithm (CMA-EA)

Set initial covariance matrix to $C^{(0)} = I_n$

$$C^{(t+1)} = (1-\eta)\, C^{(t)} + \eta \sum_{i=1}^{\mu} w_i\, (x_{i:\lambda} - m^{(t)})\, (x_{i:\lambda} - m^{(t)})'$$

η : "learning rate" $\in (0,1)$

w_i : weights; mostly $1/\mu$

$$m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{i:\lambda}$$

mean of all selected parents

complexity: $\mathcal{O}(\mu n^2 + n^3)$

sorting: $f(x_{1:\lambda}) \leq f(x_{2:\lambda}) \leq \ldots \leq f(x_{\lambda:\lambda})$

Caution: must use mean $m^{(t)}$ of "old" selected parents; not „new" mean $m^{(t+1)}$!

⇒ Seeking covariance matrix of fictitious distribution pointing in gradient direction!

State-of-the-art: **CMA-EA** (currently many variants)

→ many successful applications in practice

available in WWW:

- http://www.lri.fr/~hansen/cmaes_inmatlab.html → **C, C++, Java Fortran, Python, Matlab, R, Scilab**
- http://shark-project.sourceforge.net/ (EAlib, C++)
- …

advice:

before designing your own new method
or grabbing another method with some fancy name ...
try CMA-ES – it is available in most software libraries and often does the job!