Advanced computational statistics, lecture 3

Frank Miller, Department of Computer and Information Science, Linköping University

Department of Statistics; Stockholm University

April 4, 2023

LINKÖPING UNIVERSITY

Course schedule

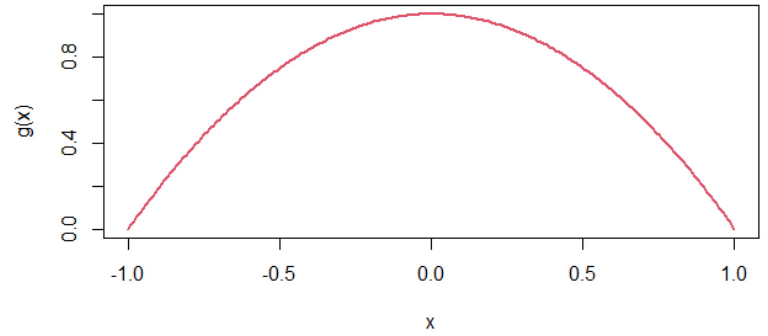
- Topic 1: **Gradient based optimisation**
- Topic 2: **Stochastic gradient based optimisation**
- Topic 3: **Gradient free optimisation**
- Topic 4: **Optimisation with constraints**
- Topic 5: **EM algorithm and bootstrap**
- Topic 6: **Simulation of random variables**
- Topic 7: **Importance sampling**

Course homepage:
http://www.adoptdesign.de/frankmillereu/adcompstat2023.html

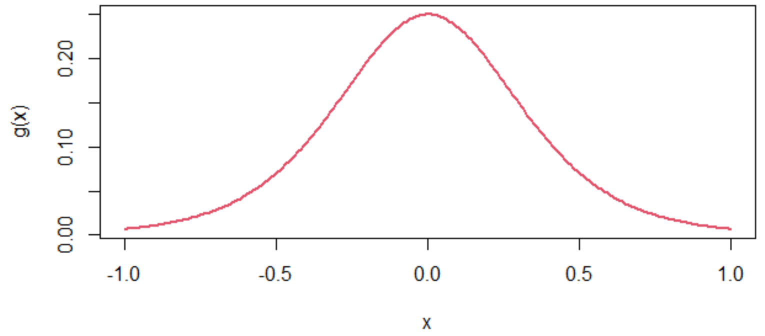
Includes schedule, reading material, lecture notes, assignments

Convexity / Concavity and log likelihood

- Function g concave, if $g((\mathbf{x}+\mathbf{y})/2) \geq (g(\mathbf{x})+g(\mathbf{y}))/2$ for all $\mathbf{x},\mathbf{y}$



concave



non-concave

- If g is concave, a local maximum is a global maximum
- Log likelihood for exponential families is concave
- Log likelihoods can be non-concave (e.g. Cauchy-distribution in Problem 1.1)
- Deep learning optimisation problems are often non-concave / non-convex and have multiple local extrema

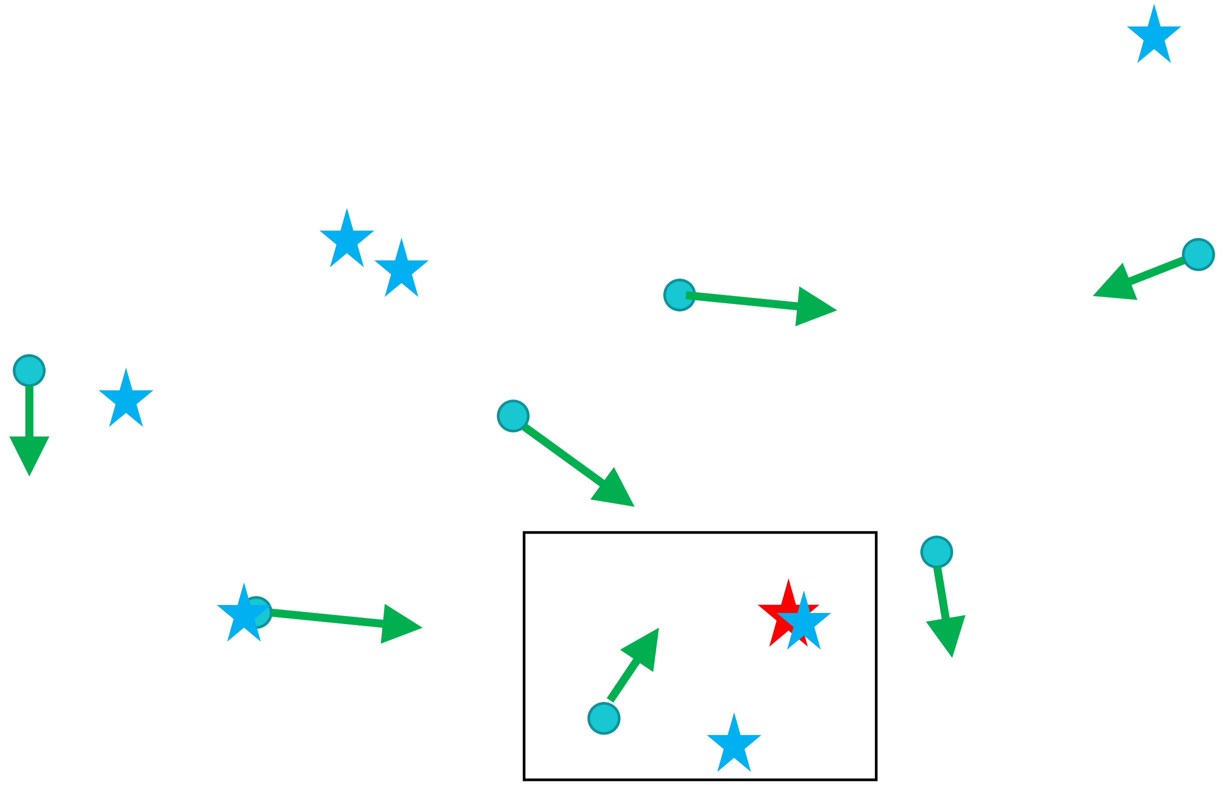
Today’s schedule: gradient free methods

- **Particle swarm optimisation** (PSO)
  - Idea
  - Different versions
  - Theoretical investigations
- **Simulated annealing**
  - Idea (for the generic optimisation problem)
  - Simulated annealing for combinatorial optimisation
  - Theoretical basis
- To compare algorithms or hyperparameter choices by empirical studies
- **Nelder-Mead algorithm**

Particle swarm optimization (PSO)

Particle swarm optimisation

- Swarm of N particles
  - ● Position of particle i at iteration t+1: $\boldsymbol{x}_i^{(t+1)}$
  - Velocity of particle i at iteration t+1: $\boldsymbol{v}_i^{(t+1)}$
- Best positions found so far:
  - ★ Best location found by particle i: $\boldsymbol{p}_{\text{best, }i}^{(t)}$
  - ★ Global best solution found: $\boldsymbol{g}_{\text{best}}^{(t)}$

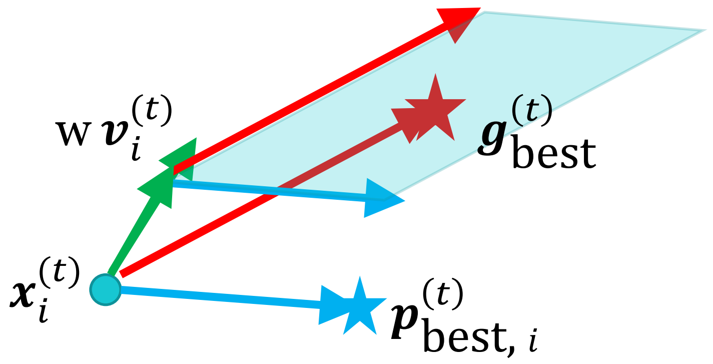


Particle swarm optimisation

- Movement of particle i at iteration t+1:
  - $\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \boldsymbol{v}_i^{(t+1)}$
  - $\boldsymbol{v}_i^{(t+1)} = \mathrm{w}\boldsymbol{v}_i^{(t)} + c_1 R_1^{(t+1)} (\boldsymbol{p}_{\text{best, }i}^{(t)} - \boldsymbol{x}_i^{(t)}) + c_2 R_2^{(t+1)} (\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}_i^{(t)})$

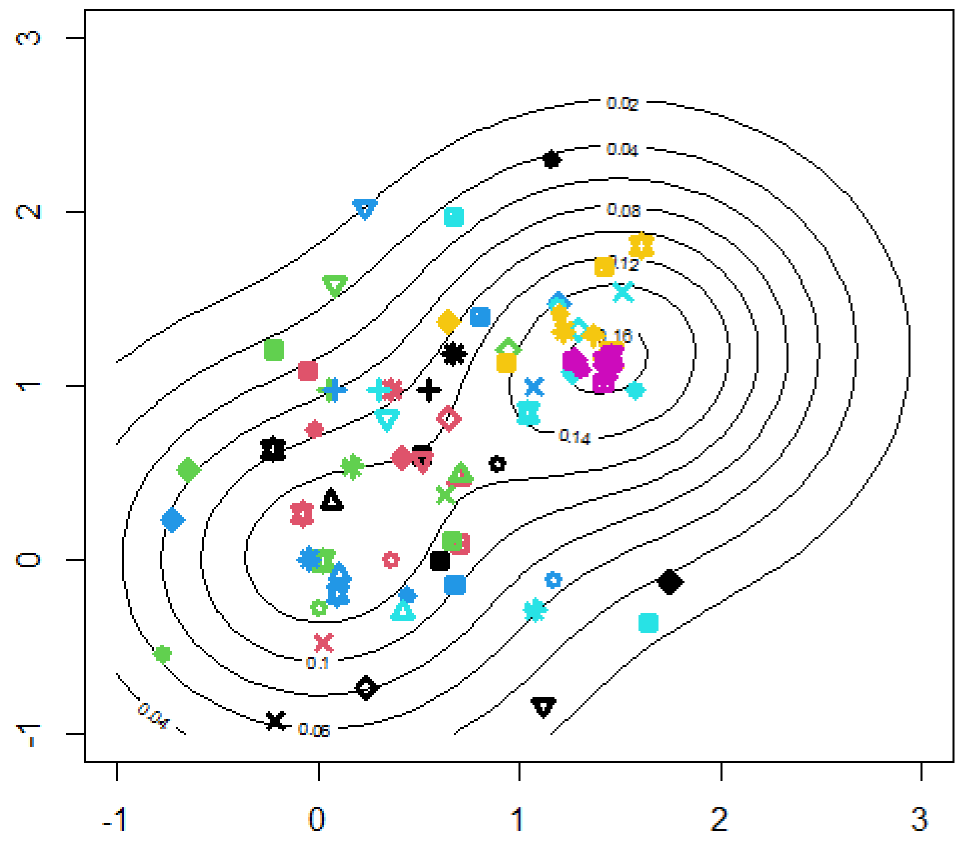
inertia weight cognitive component social component

- $R_1^{(t+1)}$ and $R_2^{(t+1)}$ are uniformly distributed, `runif()`


$\mathrm{w}\,\boldsymbol{v}_i^{(t)}$
$\boldsymbol{g}_{\text{best}}^{(t)}$
$\boldsymbol{x}_i^{(t)}$
$\boldsymbol{p}_{\text{best, }i}^{(t)}$

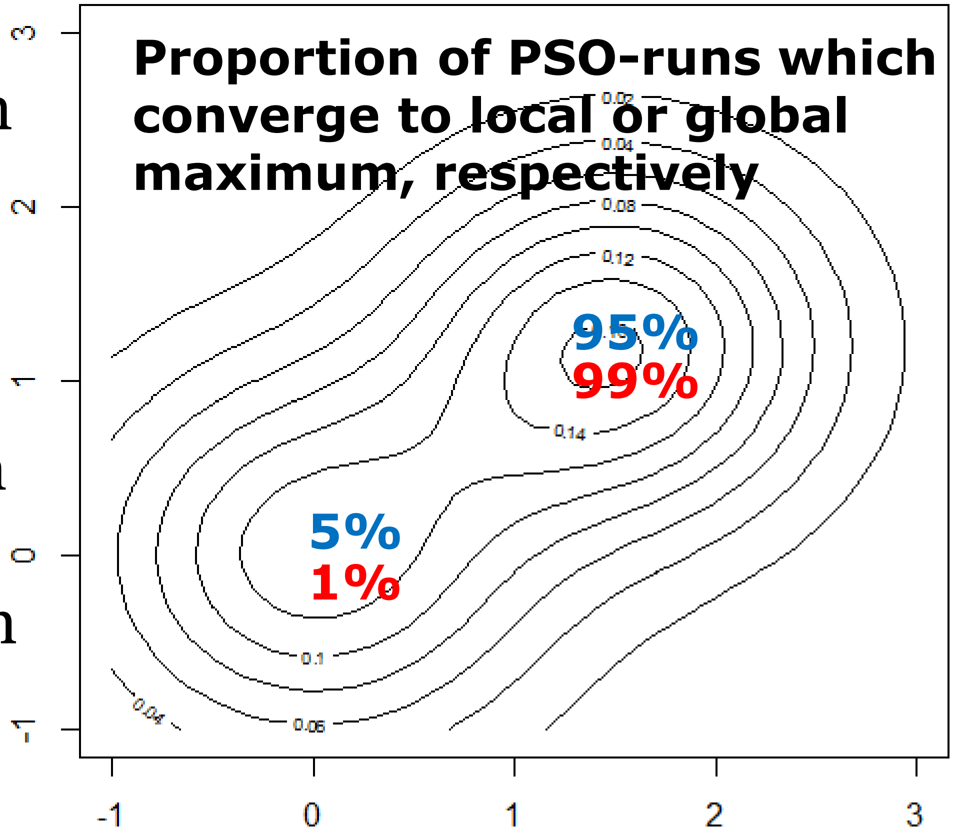
Particle swarm optimisation

- Bimodal normal mixture example from Lecture 1
- PSO with s=12 particles using **`psoptim`** (in **`R`**-package **`pso`**)
    - Iteration 1 (black)
    - Iteration 2 (red)
    - Iteration 3 (green)
    - Iteration 4 (blue)
    - Iteration 5 (light blue)
    - Iteration 20 (yellow)
    - Iteration 40 (pink)



Particle swarm optimisation

- Bimodal normal mixture example from Lecture 1
- In some runs, the local maximum is identified as global maximum
- Risk to remain at a local maximum can be reduced if not all particles are informed about the global best solution
- Option `control=list(p= )` controls proportion informed; default `1-(11/12)^3=0.23`.



All informed (p=1)

23% informed (default)

Particle swarm optimisation

- Example call:
-

```
pso <- psoptim(par=rep(NA,2),
               fn=g,
               lower=-1, upper=3,
               control=list(
                 fnscale=-1,
                 maxit=1000,
                 p=0.23,
                 s=12
               )).
```

Dimension of problem

Function to optimise

Search space
(using vectors as limits enables different limits for the dimensions)

For maximisation

Running time roughly linear in each of these two parameters

Iteration number; *default can be too large in many situations*

Proportion informed

Swarm size; *default can be too low in some situations*

- Some further options: **`c.p=`** c_1 (cognitive comp.), **`c.g=`** c_2 (social comp.), **`w=`** w (inertia weight/exploitation const.), **`trace=1`** (output of tracing info)

Particle swarm optimisation – versions

- PSO first suggested: 1995 by Kennedy and Eberhart
- Clerc (2016) distinguishes following (main) versions:
    - 1998. A basic version
    - SPSO 2007 (”Standard PSO”)
    - SPSO 2011

Particle swarm optimisation – inertia weight

- Movement of particle i at iteration t+1:
  - $\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \boldsymbol{v}_i^{(t+1)}$
  - $\boldsymbol{v}_i^{(t+1)} = \mathrm{w}\boldsymbol{v}_i^{(t)} + c_1 R_1^{(t+1)} (\boldsymbol{p}_{\text{best, } i}^{(t)} - \boldsymbol{x}_i^{(t)}) + c_2 R_2^{(t+1)} (\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}_i^{(t)})$
- In the first version from 1995, the inertia weight w was not included
- Particle swarm might “explode”
- Explosion can be prevented by introducing maximum velocity
- Alternatively, inertia weight w < 1 can prevent explosion
- Included in basic version from 1998

Particle swarm optimisation – dimensions

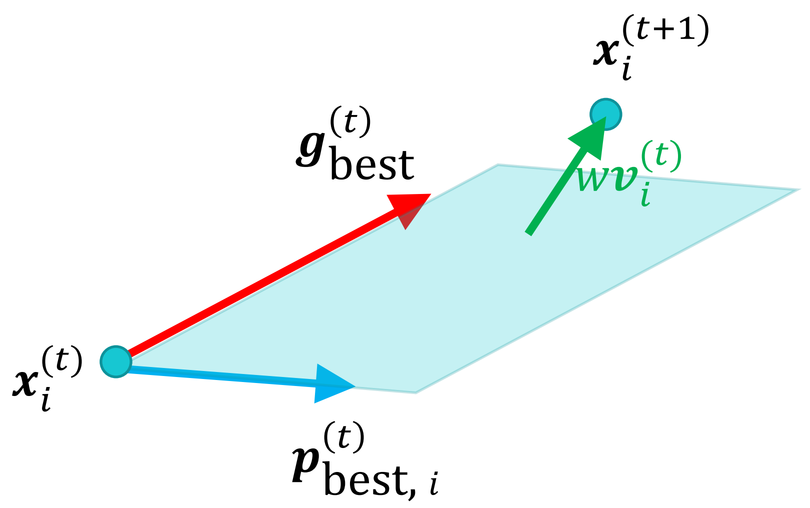
- In first versions including 1998-basic version and SPSO 2007, random variables applied for each dimension separately:
  - $\boldsymbol{v}_i^{(t+1)} = \mathrm{w}\boldsymbol{v}_i^{(t)} + c_1\boldsymbol{R}_1^{(t+1)} \otimes \left(\boldsymbol{p}_{\text{best, }i}^{(t)} - \boldsymbol{x}_i^{(t)}\right) + c_2\boldsymbol{R}_2^{(t+1)} \otimes \left(\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}_i^{(t)}\right)$

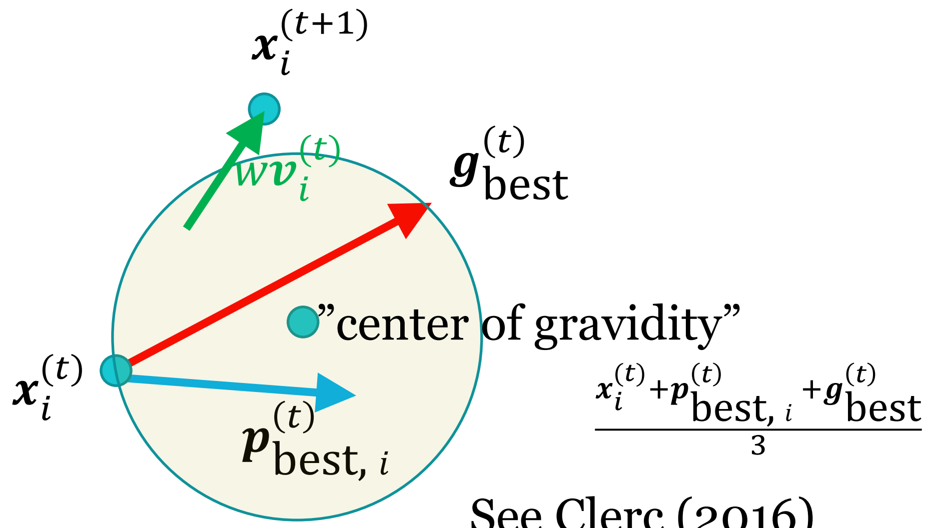
    where $\otimes$ is componentwise multiplication and $\boldsymbol{R}_k^{(t+1)}$ are vectors
- `v[i] <- w*v[i] + c1*runif(p)*(pbest[i]-x[i]) + c2*runif(p)*(gbest-x[i])`

  where `v[i]`, `x[i]`, `pbest[i]`, `gbest` vectors for each particle `i`
- In SPSO 2011, same random variable used for all dimensions leading to movement in hyperspheres:
  - $\boldsymbol{v}_i^{(t+1)} = \mathrm{w}\boldsymbol{v}_i^{(t)} + c_1 R_1^{(t+1)} (\boldsymbol{p}_{\text{best, }i}^{(t)} - \boldsymbol{x}_i^{(t)}) + c_2 R_2^{(t+1)} (\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}_i^{(t)})$
- `v[i] <- w*v[i] + c1*runif(1)*(pbest[i]-x[i]) + c2*runif(1)*(gbest-x[i])`

Particle swarm optimisation – dimensions

- Velocity of particle i at iteration t+1:
  - $\boldsymbol{v}_i^{(t+1)} = \mathrm{w}\boldsymbol{v}_i^{(t)} + c_1 R_1^{(t+1)} \left(\boldsymbol{p}_{\text{best, }i}^{(t)} - \boldsymbol{x}_i^{(t)}\right) + c_2 R_2^{(t+1)} \left(\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}_i^{(t)}\right)$
- In SPSO 2011, same random variable used for all dimensions leading to movement in hyperspheres


$\boldsymbol{x}_i^{(t+1)}$
$\boldsymbol{g}_{\text{best}}^{(t)}$
$w\boldsymbol{v}_i^{(t)}$
$\boldsymbol{x}_i^{(t)}$
$\boldsymbol{p}_{\text{best, }i}^{(t)}$


$\boldsymbol{x}_i^{(t+1)}$
$w\boldsymbol{v}_i^{(t)}$
$\boldsymbol{g}_{\text{best}}^{(t)}$
"center of gravidity"
$\frac{\boldsymbol{x}_i^{(t)}+\boldsymbol{p}_{\text{best, }i}^{(t)}+\boldsymbol{g}_{\text{best}}^{(t)}}{3}$
$\boldsymbol{x}_i^{(t)}$
$\boldsymbol{p}_{\text{best, }i}^{(t)}$

See Clerc (2016)

Particle swarm optimisation – dimensions

- In version SPSO 2011, particles can move only in hyperspace spanned by starting particles
- Disadvantages:
  - If dimension of problem p is large in relation to swarm size s, e.g. p>s, optimisation done only in a subspace and high risk that optimum is missed
  - Even if starting particles well distributed, they might become close to a hyperspace after some iterations
- Advantages:
  - Problem with dependence on coordinate system and with “biased search” is reduced; finds optima along axes and diagonal easier (Clerc, 2016)
  - Linearly constrained problems can easily be handled (see L4)

PSO – choice of hyperparameters

- Velocity of particle i at iteration t+1:
  - $\boldsymbol{v}_i^{(t+1)} = \mathrm{w}\boldsymbol{v}_i^{(t)} + c_1 R_1^{(t+1)} \left(\boldsymbol{p}_{\text{best, } i}^{(t)} - \boldsymbol{x}_i^{(t)}\right) + c_2 R_2^{(t+1)}\left(\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}_i^{(t)}\right)$
- Hyperparameters to choose: w, c_1, c_2
- Particles should not diverge
- “Stability analyses” had been done – these are simplified analytical computations, **for example**:
  - Assume one dimensional case,
  - Assume static $\boldsymbol{p}_{\text{best, } i}^{(t)} = \boldsymbol{p}_{\text{best, } i}$ and $\boldsymbol{g}_{\text{best}}^{(t)} = \boldsymbol{g}_{\text{best}}$ (“stagnation assumption”)
  - Ignore randomness (replace $R_k^{(t+1)}$ by expected value ½)
- Derive requirements for w, c_1, c_2 such that $\boldsymbol{x}_i^{(t)}$ “converges”

PSO – choice of hyperparameters

- Velocity of particle i at iteration t+1:
  - $\boldsymbol{v}_i^{(t+1)} = \mathrm{w}\boldsymbol{v}_i^{(t)} + c_1 R_1^{(t+1)} \left(\boldsymbol{p}_{\text{best}, i}^{(t)} - \boldsymbol{x}_i^{(t)}\right) + c_2 R_2^{(t+1)} \left(\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}_i^{(t)}\right)$
- Standard choice in SPSO 2007, based originally on stability analyses from Clerc and Kennedy (2002):
  - $w = \frac{1}{2\ln(2)} = 0.721,$
  - $c_1 = c_2 = \frac{1}{2} + \ln(2) = 1.193$

- Since deterministic $R_k^{(t+1)} = \frac{1}{2}$ and static $\boldsymbol{p}_{\text{best}}, \boldsymbol{g}_{\text{best}}$ are used in stability analyses, no distinctive requirements for c_1 and c_2 are obtained and a default is often just $c_1 = c_2$
- Write now $C_k^{(t+1)} = c_k R_k^{(t+1)} \sim Unif[0, c_k], k = 1,2.$

Particle swarm optimisation – stability analyses

- Movement of specific particle at iteration t+1 (drop index i):
  - $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \boldsymbol{v}^{(t+1)}$
  - $\boldsymbol{v}^{(t+1)} = \mathrm{w}\boldsymbol{v}^{(t)} + c_1^{(t+1)}\,(\boldsymbol{p}_{\text{best}}^{(t)} - \boldsymbol{x}^{(t)}) + c_2^{(t+1)}(\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}^{(t)})$
- Focusing on particle locations, we can describe PSO as:

$$\begin{aligned}
&\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \boldsymbol{v}^{(t+1)} \\
&= \boldsymbol{x}^{(t)} + \mathrm{w}\boldsymbol{v}^{(t)} + c_1^{(t+1)}\left(\boldsymbol{p}_{\text{best}}^{(t)} - \boldsymbol{x}^{(t)}\right) + c_2^{(t+1)}\left(\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}^{(t)}\right) \\
&= \boldsymbol{x}^{(t)} + \mathrm{w}\left(\boldsymbol{x}^{(t)} - \boldsymbol{x}^{(t-1)}\right) + c_1^{(t+1)}\left(\boldsymbol{p}_{\text{best}}^{(t)} - \boldsymbol{x}^{(t)}\right) + c_2^{(t+1)}\left(\boldsymbol{g}_{\text{best}}^{(t)} - \boldsymbol{x}^{(t)}\right) \\
&= \boldsymbol{x}^{(t)}\left(1 + w - c_1^{(t+1)} - c_2^{(t+1)}\right) - \mathrm{w}\boldsymbol{x}^{(t-1)} + c_1^{(t+1)}\boldsymbol{p}_{\text{best}}^{(t)} + c_2^{(t+1)}\boldsymbol{g}_{\text{best}}^{(t)}
\end{aligned}$$

$$\boxed{\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} + \boldsymbol{v}^{(t)}}$$

- Therefore, a single equation is sufficient to describe the PSO iterations ($\boldsymbol{x}^{(t+1)}$ depends then on both $\boldsymbol{x}^{(t)}$ and $\boldsymbol{x}^{(t-1)}$)

Particle swarm optimisation – stability analyses

- Movement of specific particle at iteration t+1 with PSO:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}\left(1 + w - C_1^{(t+1)} - C_2^{(t+1)}\right) - \mathrm{w}\boldsymbol{x}^{(t-1)} + C_1^{(t+1)}\boldsymbol{p}_{\text{best}}^{(t)} + C_2^{(t+1)}\boldsymbol{g}_{\text{best}}^{(t)}$$

- Stability analyses were improved during the two previous decades, see Bonyadi and Michalewicz (2016) and Cleghorn and Engelbrecht (2018); definitions below follow the latter
- Order-1 stability
  A sequence $(\boldsymbol{x}^{(t)})$ of p-dimensional random variables is called *order-1 stable* if $\mathrm{E}\left[\boldsymbol{x}^{(t)}\right] \rightarrow \boldsymbol{x}_E$ for some $\boldsymbol{x}_E$
- Order-2 stability
  A sequence $(\boldsymbol{x}^{(t)})$ of p-dimensional random variables is called *order-2 stable* if $\mathrm{Var}\left[\boldsymbol{x}^{(t)}\right] \rightarrow \boldsymbol{x}_V$ for some $\boldsymbol{x}_V$

Particle swarm optimisation – stability analyses

- Movement of specific particle at iteration t+1 with PSO:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}\left(1 + w - C_1^{(t+1)} - C_2^{(t+1)}\right) - \mathrm{w}\boldsymbol{x}^{(t-1)} + C_1^{(t+1)}\boldsymbol{p}_{\text{best}}^{(t)} + C_2^{(t+1)}\boldsymbol{g}_{\text{best}}^{(t)}$$

- Bonyadi and Michalewicz (2016) interpret each of $C_1^{(t+1)}, C_2^{(t+1)}$, $\boldsymbol{p}_{\text{best}}^{(t)}, \boldsymbol{g}_{\text{best}}^{(t)}$ as iid random variables
- This generalises assumptions that these values are fixed values; it weakens the stagnation assumption
- The iid assumption for $\boldsymbol{p}_{\text{best}}^{(t)}, t = 1, \ldots$ and for $\boldsymbol{g}_{\text{best}}^{(t)}, t = 1, \ldots$ still need to be seen as approximations

Particle swarm optimisation – stability analyses

- We consider the one-dimensional case (p=1) now
- Movement of specific particle at iteration t+1 with PSO:

$$x^{(t+1)} = x^{(t)}\left(1 + w - C_1^{(t+1)} - C_2^{(t+1)}\right) - \mathrm{w}x^{(t-1)} + C_1^{(t+1)} p_{\text{best}}^{(t)} + C_2^{(t+1)} g_{\text{best}}^{(t)}$$

- To write the iterations as a linear one-step relation, we write

$$\boldsymbol{z}^{(t+1)} = (x^{(t+1)}, x^{(t)})^T, \qquad U = 1 + w - C_1^{(t+1)} - C_2^{(t+1)},$$

and

$$\boldsymbol{z}^{(t+1)} = \begin{pmatrix} U & -w \\ 1 & 0 \end{pmatrix} \boldsymbol{z}^{(t)} + \begin{pmatrix} C_1^{(t+1)} p_{\text{best}}^{(t)} + C_2^{(t+1)} g_{\text{best}}^{(t)} \\ 0 \end{pmatrix}$$

- Since U and $\boldsymbol{z}^{(t)}$ are independent, we have

$$E\boldsymbol{z}^{(t+1)} = \begin{pmatrix} EU & -w \\ 1 & 0 \end{pmatrix} E\boldsymbol{z}^{(t)} + \begin{pmatrix} E\left[C_1^{(t+1)} p_{\text{best}}^{(t)}\right] + E\left[C_2^{(t+1)} g_{\text{best}}^{(t)}\right] \\ 0 \end{pmatrix}$$

➢Sequence $E\boldsymbol{z}^{(t+1)}$ is of form $E\boldsymbol{z}^{(t+1)} = \boldsymbol{M}E\boldsymbol{z}^{(t)} + \boldsymbol{b}$

Particle swarm optimisation – stability analyses

- Sequence $E\boldsymbol{z}^{(t+1)}$ is of form $E\boldsymbol{z}^{(t+1)} = \boldsymbol{M}E\boldsymbol{z}^{(t)} + \boldsymbol{b}$
- Functional analysis says that $E\boldsymbol{z}^{(t)}$ converges if the spectral radius of $\boldsymbol{M}$ is <1, see Bonyadi and Michalewicz (2016)'s Lemma 1
- Spectral radius $\rho(\boldsymbol{M})$ of $\boldsymbol{M} \in \mathbb{R}^{p\times p}$ is $\rho(\boldsymbol{M}) = \max\{|\lambda_1|, \dots, |\lambda_p|\}$ where λ_j are the p (real or complex) eigenvalues of $\boldsymbol{M}$
- Recall that a non-symmetric $\mathbb{R}^{p\times p}$ matrix still has p eigenvalues as long as we allow for complex eigenvalues
- If $\lambda = r + c\mathrm{i}$ then $|\lambda| = \sqrt{r^2 + c^2}$; **R** can cope with this easily:
-
```
> M <- matrix(c(-0.66, 1, -0.72, 0), ncol=2)
> eigen(M)$values
[1] -0.33+0.7817289i -0.33-0.7817289i
> max(abs(eigen(M)$values))  # spectral radius
[1] 0.8485281
```

Particle swarm optimisation – stability analyses

- We have

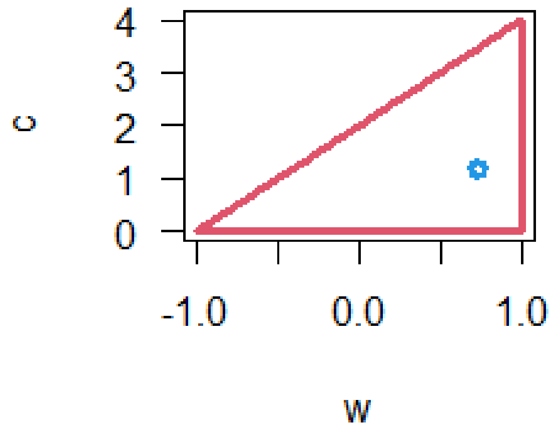
$$Ez^{(t+1)} = \begin{pmatrix} EU & -w \\ 1 & 0 \end{pmatrix} Ez^{(t)} + \begin{pmatrix} E\left[C_1^{(t+1)} p_{\text{best}}^{(t)}\right] + E\left[C_2^{(t+1)} g_{\text{best}}^{(t)}\right] \\ 0 \end{pmatrix}$$

- Compute spectral radius of $\begin{pmatrix} EU & -w \\ 1 & 0 \end{pmatrix}$
- Eigenvalues: $0 = \det\begin{pmatrix} \lambda - EU & w \\ -1 & \lambda \end{pmatrix} = \lambda^2 - \lambda EU + w \quad \Rightarrow \quad \lambda_{1,2} = \frac{EU \pm \sqrt{EU^2 - 4w}}{2}$
- $EU = 1 + w - EC_1^{(t+1)} - EC_2^{(t+1)} = 1 + w - \frac{c_1 + c_2}{2}$
- One can show:

$$\rho(M) = \max\left\{\frac{\left|EU + \sqrt{EU^2 - 4w}\right|}{2}, \frac{\left|EU - \sqrt{EU^2 - 4w}\right|}{2}\right\} < 1 \text{ iff}$$

$-1 < w < 1 \text{ and } 0 < \frac{c_1 + c_2}{2} < 2(w + 1)$

- Assume $c = c_1 = c_2$



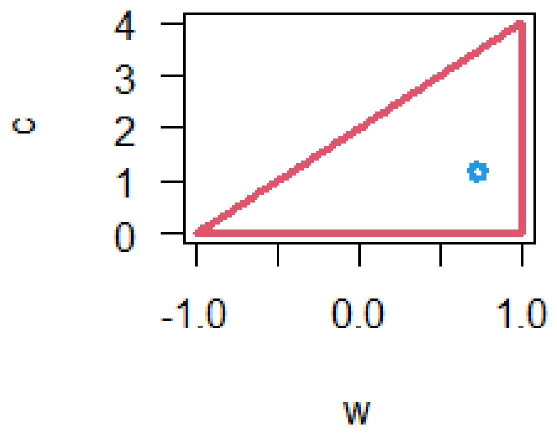
Particle swarm optimisation - stability analyses

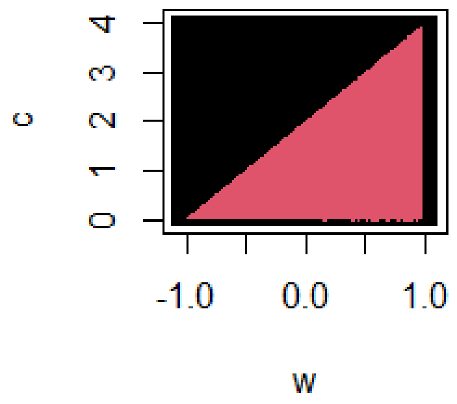
- Assume $c = c_1 = c_2$. $EU = 1 + w - c$
- One can show:

$$\rho(M) = \max\left\{\frac{\left|EU+\sqrt{EU^2-4w}\right|}{2}, \frac{\left|EU-\sqrt{EU^2-4w}\right|}{2}\right\} < 1 \text{ iff}$$

$$-1 < w < 1 \text{ and } 0 < c < 2(w+1)$$

- If it would be too difficult to show the above, one could calculate the maximum eigenvalue for a grid of (w, c)-pairs and plot the cases when it is <1 (see **R** code on homepage)





Particle swarm optimisation – stability analyses

- To do stability analyses for order-2 stability (about the limit of the variance $\mathrm{Var}(\boldsymbol{z}^{(t+1)})$), we can investigate

$$\boldsymbol{z}^{(t+1)} = (x^{(t+1)}, x^{(t)}, \left(x^{(t+1)}\right)^2, \left(x^{(t)}\right)^2, x^{(t+1)}x^{(t)})^T$$

- The iterations can be written as system

$$E\boldsymbol{z}^{(t+1)} = \begin{pmatrix} EU & -w & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 2E[UP] & -2wEP & E[U^2] & w^2 & -2wEU \\ 0 & 0 & 1 & 0 & 0 \\ EP & 0 & EU & 0 & -w \end{pmatrix} E\boldsymbol{z}^{(t)} + \boldsymbol{b}$$

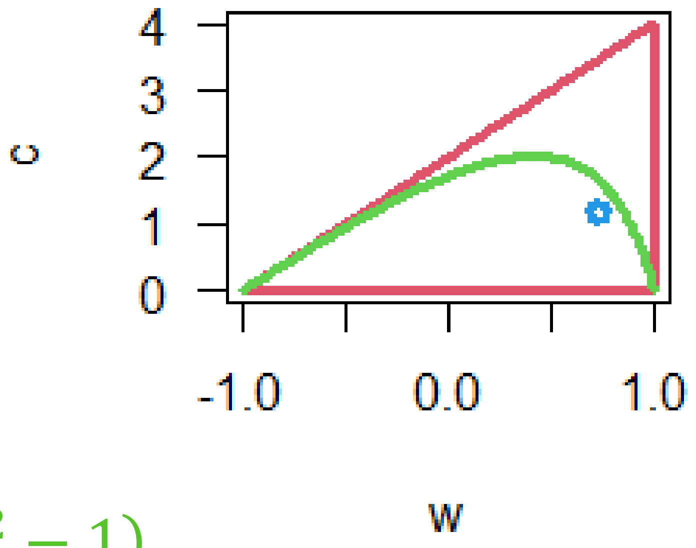
where $P = C_1^{(t+1)} p_{\text{best}}^{(t)} + C_2^{(t+1)} g_{\text{best}}^{(t)}$

Particle swarm optimisation – stability analyses

- $c = c_1 = c_2$
- $-1 < w < 1$ and $0 < c < 2(w+1)$
- Sequence $\left(\mathbf{z}^{(t+1)}\right)$ is order-2 stable if: $-1 < w < 1$ and

$$0 < c < \frac{12(w^2-1)}{5w-7}$$

- Default in **R**–package **pso** based on Clerc and Kennedy (2002): $w = \frac{1}{2\ln(2)} = 0.721,\ c = c_1 = c_2 = \frac{1}{2} + \ln(2) = 1.193$



PSO – choice of hyperparameters

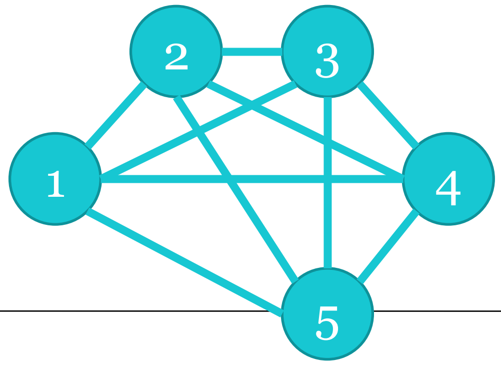
- Based on stability analysis, choose w, c_1, c_2 respecting $-1 < w < 1$ and $0 < c_1 + c_2 < \frac{24(w^2-1)}{5w-7}$
- w>0 is in spirit of the algorithm's idea

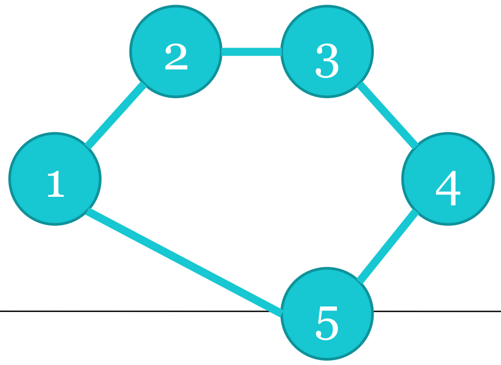
- Another hyperparameter to be chosen: swarm size
- Swarm size motivated by empirical studies based on standard optimisation problems
- SPSO 2007: $10 + \lfloor 2\sqrt{p} \rfloor$
- Clerc (2012) shows with 12 standard optimisation problems:
  - usually swarm sizes larger than $10 + \lfloor 2\sqrt{p} \rfloor$ better,
  - dependence on dimension p is weak
- SPSO 2011: choice of user; suggested: 40

PSO – topologies for particles

- Particles ”inform” other particles about their results
- In the original PSO, each particle informs all others
- To ensure that not all particles are attracted prematurely by particle at a local optimum, do not inform all particles
- The structure how information flows is specified in ”topologies”
- Global top. (all inform all)

Ring top. (all inform their two ”neighbours”)

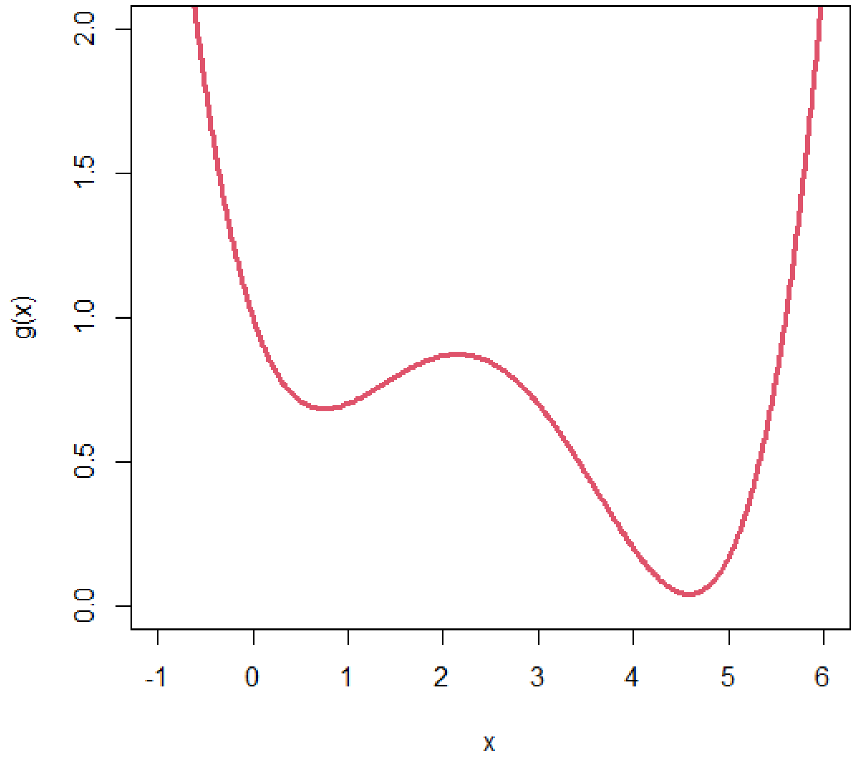




PSO – exploration versus exploitation

- Exploration of the search space
- Exploitation around a promising position
- The topology: A sparce topology (e.g. ring top.) ensures more exploration compared to a dense one (e.g. global top.)
- Parameter w: Larger w leads to more exploration
- Parameters c_1 and c_2: Smaller c_2 (and c_1) lead to more exploration
- Clerc (2016; Section 8.6.4.1): The experimental evidence for such dependencies [on w, c_1, c_2] is weak

Simulated annealing



AlphaOpt (2017). Introduction To Optimization: Gradient Free Algorithms (2/2) – Simulated Annealing, Nelder-Mead (0:15-1:35)

Simulated annealing

- Start value $x^{(0)}$; Stage j=0,1,2,... has m_j iterations; set j=0
- Given iteration $x^{(t)}$, generate $x^{(t+1)}$ as follows:

1. Sample a candidate x^* from a proposal distribution $p(\cdot|x^{(t)})$
2. Compute $h\left(x^{(t)}, x^*\right) = \exp\left(\frac{g(x^*)-g\left(x^{(t)}\right)}{\tau_j}\right)$

$g\left(x^{(t)}\right) - g(x^*)$ for minimisation

3. Define next iteration $x^{(t+1)}$ according to

$$x^{(t+1)} = \begin{cases} x^*, \text{with probability} \min\{h\left(x^{(t)}, x^*\right), 1\} \\ x^{(t)}, \text{otherwise} \end{cases}$$

4. Set **`t <- t+1`** and repeat 1.-3. m_j times
5. Update $\tau_j = \alpha(\tau_{j-1})$ and $m_j = \beta(m_{j-1})$; set **`j <- j+1`**; go to 1

τ_j is temperature; function α should slowly decrease it; function β should be increasing

Simulated annealing

- Initially, also “bad” proposals are accepted
- With decreasing temperature, accept only improvements
- This helps to explore first and avoids convergence to a local maximum too early
- Algorithm which has therefore chances to find the global optimum in presence of multiple local optima

- **`method=“SANN”`** of R function **`optim`** is “a variant of simulated annealing” (documentation of **`optim`**)
  - Initial temperature seems to be important choice (can be changed e.g. by **`control=list(temp=0.01)`**; default 10 might be bad)

Simulated annealing: proposal distribution

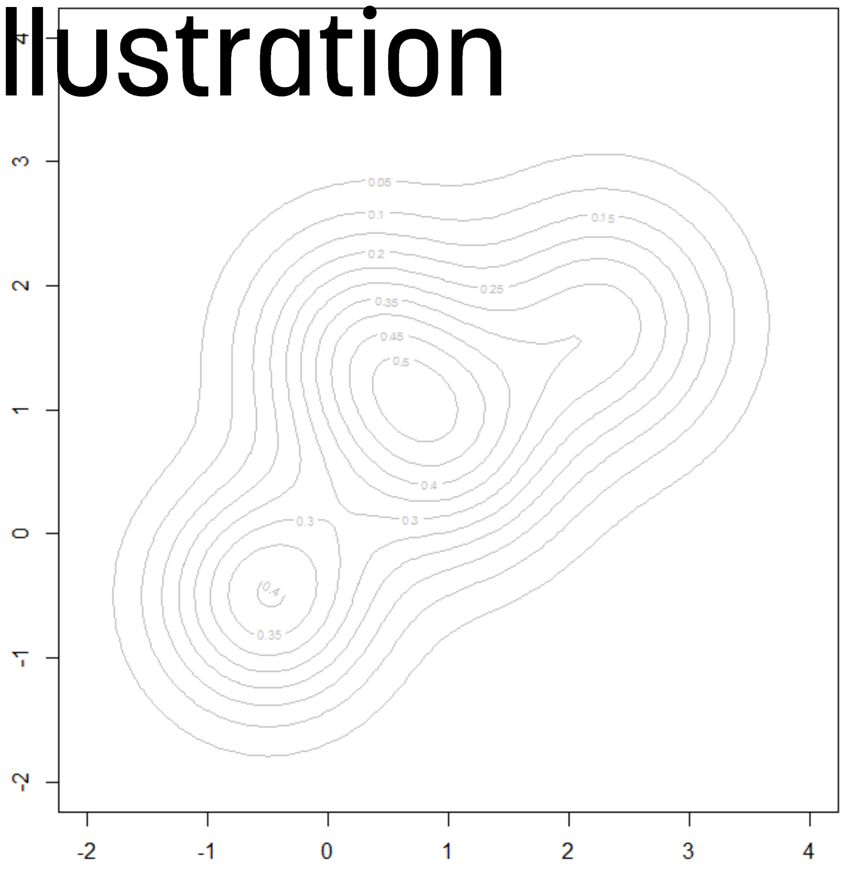
- Step 1 in simulated annealing iteration rule:

1. Sample a candidate x* from a proposal distribution $p(\cdot|x^{(t)})$

- Proposal distribution could be uniform distribution on a **neighbourhood** of $x^{(t)}$; for a unidimensional optimisation problem:
  ```
  xs <- xt + runif(n=1, min=-1, max=1)
  ```
- Instead of Unif[-1,1], a distribution on a smaller or larger neighbourhood could be used
- But also, normal distribution $N(0, \sigma^2)$ or other **symmetric** distribution around 0 might be added to $x^{(t)}$ instead
- For multidimensional cases, one could use iid components, a uniform distribution on a ball around $\mathbf{x}^{(t)}$ or a multivariate normal distribution with mean $\mathbf{x}^{(t)}$

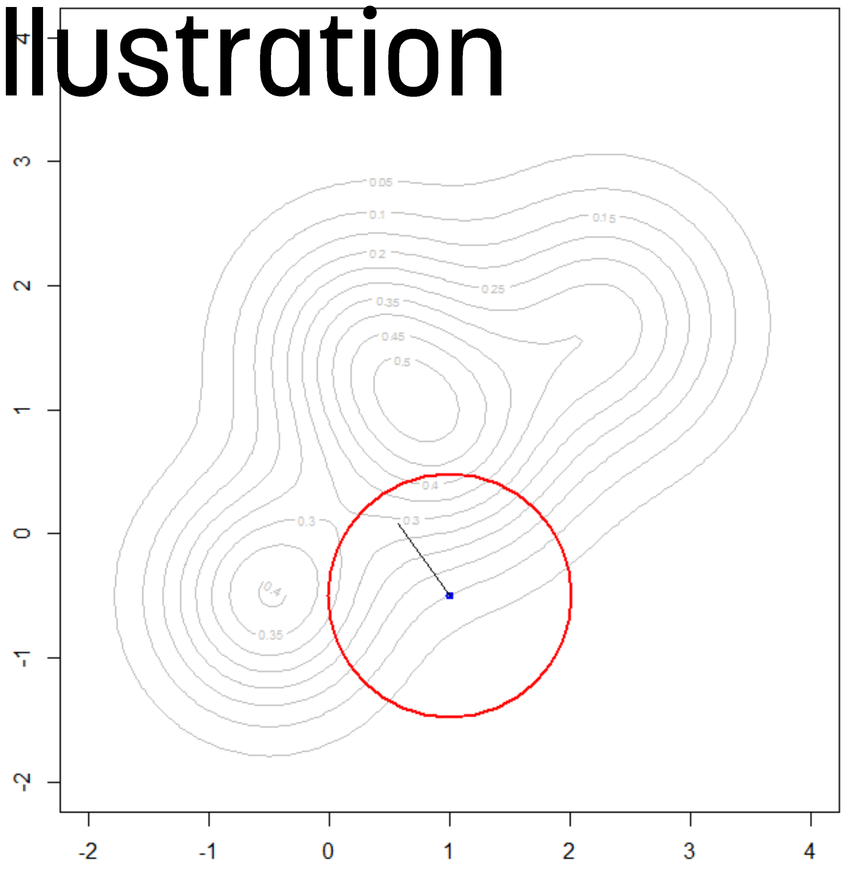
Simulated annealing – Illustration

- For illustration, we consider two-dimensional function g according to contour lines in figure (one global and one local maximum) and fixed temperature τ
- Proposal distribution $p(x^*|x^{(t)}) = p(x^{(t)}|x^*)$ $=\frac{1}{\pi r^2}\mathbf{1}\{\|x^{(t)} - x^*\| < r\}$ for some constant r (here=1)

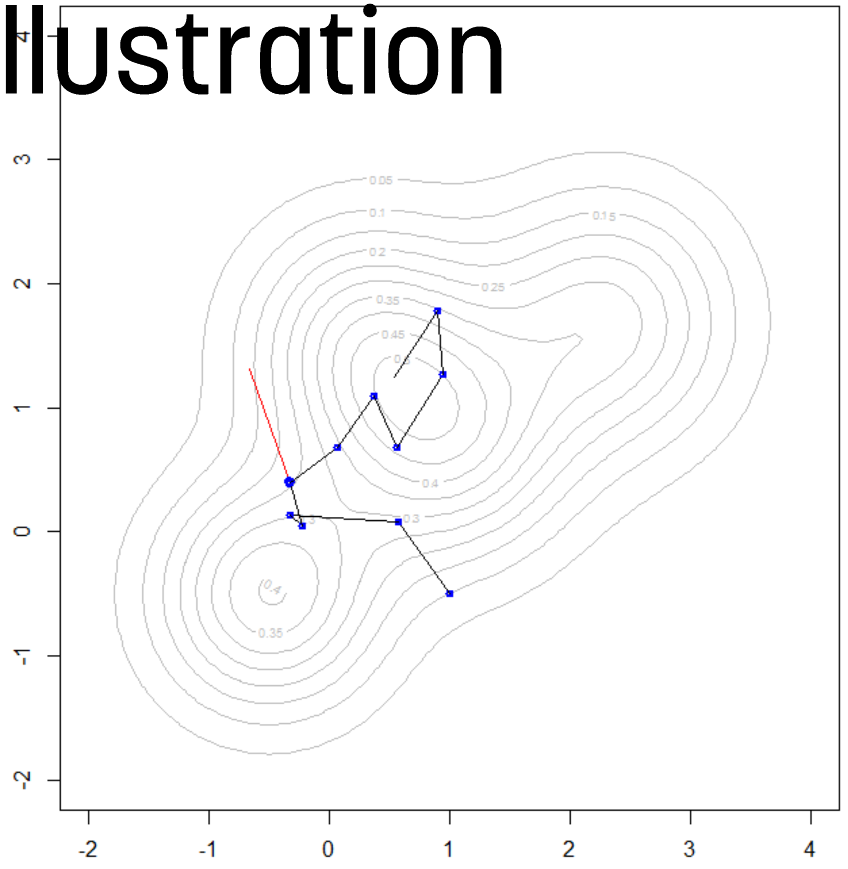


Simulated annealing – Illustration

- Proposal distribution
  $p(x^*|x^{(t)}) = p(x^{(t)}|x^*)$
  $= \frac{1}{\pi r^2}\mathbf{1}\{\|x^{(t)} - x^*\| < r\}$
  for some constant r (here=1)
- Start here with $x^{(0)}=(1,-0.5)$
- Randomize uniformly on unit circle around $x^{(0)}$ (proposal distribution); result $x^*=(0.58,0.08)$
- $g(x^*)=0.296 > g(x^{(0)}) = 0.098$; so this was an uphill step and is automatically accepted $(h(x^{(t)}, x^*) > 1)$



Simulated annealing – Illustration



- $x^{(0)}=(1,-0.5)$
- Uphill steps: $x^{(1)}=(0.58,0.08)$
- $x^{(2)}=(-0.33,0.13)$
- $x^{(3)}=(-0.23,0.05)$
- Then downhill step proposed: $x^{*}=(-0.32,0.4)$, $h\left(x^{(t)},x^{*}\right)=0.774$
- Random Unif(0,1) generated: 0.573 and since this is smaller than h=0.774, $x^{(4)}=x^{*}=(-0.32,0.4)$ is accepted
- Again downhill step proposed: $x^{*}=(-0.67,1.31)$, $h\left(x^{(t)},x^{*}\right)=0.560$; random Unif(0,1): 0.890 and rejection of x^{*}
- $x^{(5)}=x^{(4)}=(-0.32,0.4)$

Combinatorial optimisation

- Generic optimisation problem:
  - $\boldsymbol{x}$ p-dimensional vector, $g\colon \mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \min g(\boldsymbol{x})$

- Now, we consider also optimisation problems which cannot exactly be formulated according to the generic one
- Especially, function g might be defined on another space than $\mathbb{R}^p$

- Generalized optimisation problem:
  - $\boldsymbol{x}$ p-dimensional vector, $g\colon \mathbb{S} \to \mathbb{R}$ function for some set $\mathbb{S}$
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \min g(\boldsymbol{x})$

Example: Multiple linear regression

- Generalized optimisation problem:
  - $\boldsymbol{x}$ p-dimensional vector, $g\colon\ \mathbb{S} \to \mathbb{R}$ function for some set $\mathbb{S}$
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \min g(\boldsymbol{x})$

- Multiple linear regression with q predictors
- Desired to choose best model based on criterion like AIC
- There are 2^q possible models
- If q small, AIC of all models can be computed (exhaustive search); for q larger, this is impossible (e.g. q=50, 1*ms* to compute an AIC $\to$ more than 35 000 years needed!)
- One model can be represented as element of $\mathbb{S} = \{0, 1\}^q$ (1=predictor included in model, 0 otherwise)

Example: Multiple linear regression

- Generalized optimisation problem:
  - $\boldsymbol{x}$ p-dimensional vector, $g\colon\ \mathbb{S} \to \mathbb{R}$ function for some set $\mathbb{S}$
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \min g(\boldsymbol{x})$
- Optimisation problem: Which model gives best AIC?
- Model 1: (1, 0, 0, 0, 1, 1, 0, 1, ...)
  Model 2: (1, 1, 1, 0, 1, 1, 0, 0, ...)
- Which models are ”close” to each other? (Need metric on $\mathbb{S} = \{0, 1\}^q$)
  What is a neighbourhood of a model?

- Apply simulated annealing e.g. with neighbourhood being all models which differ by one predictor (for proposal dist.)
- Uniform distribution on neighbourhood can be used

Example: Multiple linear regression

- Generalized optimisation problem:
  - $\boldsymbol{x}$ p-dimensional vector, $g\colon\ \mathbb{S} \to \mathbb{R}$ function for some set $\mathbb{S}$
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \min g(\boldsymbol{x})$

- Arbitrary starting model generated (e.g. uniform distribution on $\mathbb{S} = \{0, 1\}^q$, **`xs <- rbinom(q, size=1, prob=0.5)`**)

- See example in Givens and Hoeting (2013), Section 3.3, with 27 predictors

Recall from L1: Maximising information of experimental designs

- Regression model $\mathbf{y}=\mathbf{X}\,\boldsymbol{\beta}+\boldsymbol{\varepsilon}$ (where $\boldsymbol{\varepsilon}$ has iid components)
- $\mathbf{X}$ design matrix (depends on choice of observational points)
- Covariance matrix of Least Squares estimate $\widehat{\boldsymbol{\beta}}$ is
$$\mathrm{Cov}\big(\widehat{\boldsymbol{\beta}}\big) = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-\mathbf{1}}\cdot const$$
- Choose design of an experiment such that $\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$ “large”
- D-optimality: $g(\text{"}\mathbf{design}\text{"}) = \det(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})$
- We search $\mathbf{design}^*$ with $g(\mathbf{design}^*) = \max g(\mathbf{design})$

Ex: Maximising information of experimental designs

- Regression model $\mathbf{y}=\mathbf{X}\,\boldsymbol{\beta}+\boldsymbol{\varepsilon}$, $\mathrm{Cov}(\hat{\boldsymbol{\beta}}) = (\boldsymbol{X}^T\boldsymbol{X})^{-\mathbf{1}} \cdot const$
- We search **design*** with $g(\textbf{design}^*) = \max g(\textbf{design})$
- Example: cubic regression, $y = \beta_0 + \beta_1 w + \beta_2 w^2 + \beta_3 w^3 + \varepsilon$,
  w can be chosen in [-1, 1], but practical circumstances require here a distance between design points of 0.05
- Therefore, we allow design points {-1, -0.95, -0.9, …, 1} and at most one observation can be done at each point
- Each observation has a cost; and we want to minimise the penalized D-optimality
  $\#\text{observations} * 0.2 - \log\left(\det(\boldsymbol{X}^T\boldsymbol{X})\right)$
- $$\boldsymbol{X} = \begin{pmatrix} 1 & w_1 & w_1^2 & w_1^3 \\ 1 & w_2 & w_2^2 & w_2^3 \\ \dots & \dots & \dots & \dots \\ 1 & w_n & w_n^2 & w_n^3 \end{pmatrix}$$

Ex: Maximising information of experimental designs

- Example: cubic regression, $y = \beta_0 + \beta_1 w + \beta_2 w^2 + \beta_3 w^3 + \varepsilon$,
  w can be chosen in [-1, 1], but practical circumstances require here a distance between design points of 0.05
- Therefore, we allow design points {-1, -0.95, -0.9, …, 1} and at most one observation can be done at each point
- A design can be represented by a vector in $\mathbb{S} = \{0, 1\}^{41}$ where 0 means that no observation is done at a design point and 1 means that one observation is made there
- How can a reasonable neighbourhood on $\mathbb{S}$ look like here?

Simulated annealing

- Start value $x^{(0)}$; Stage j=0,1,2,... has m_j iterations; set j=0
- Given iteration $x^{(t)}$, generate $x^{(t+1)}$ as follows:

1. Sample a candidate x^* from a proposal distribution $p(\cdot|x^{(t)})$
2. Compute $h\left(x^{(t)}, x^*\right) = \exp\left(\frac{g(x^*)-g\left(x^{(t)}\right)}{\tau_j}\right)$

   $g\left(x^{(t)}\right) - g(x^*)$ for minimisation

3. Define next iteration $x^{(t+1)}$ according to

$$x^{(t+1)} = \begin{cases} x^*, \text{with probability} \min\{h\left(x^{(t)}, x^*\right), 1\} \\ x^{(t)}, \text{otherwise} \end{cases}$$

4. Set `t <- t+1` and repeat 1.-3. m_j times
5. Update $\tau_j = \alpha(\tau_{j-1})$ and $m_j = \beta(m_{j-1})$; set `j <- j+1`; go to 1

τ_j is temperature; function α should slowly decrease it; function β should be increasing

Markov Chain Monte Carlo – Metropolis algorithm

(Metropolis et al., 1953)

- Given a density $f(x)$ and aim is to generate a sample following f
- A starting value $x^{(0)}$ is generated from some starting distribution
- Given observation $x^{(t)}$, generate $x^{(t+1)}$ as follows:

1. Sample candidate x^* from symmetric proposal dist. $p(\cdot|x^{(t)})$
2. Compute ratio $R\left(x^{(t)}, x^*\right) = \frac{f(x^*)}{f\left(x^{(t)}\right)}$
3. Sample $x^{(t+1)}$ according to

$$x^{(t+1)} = \begin{cases} x^*, \text{with probability} \min\{R\left(x^{(t)}, x^*\right), 1\} \\ x^{(t)}, \text{otherwise} \end{cases}$$

4. If more observations needed, set `t <- t+1`; go to 1

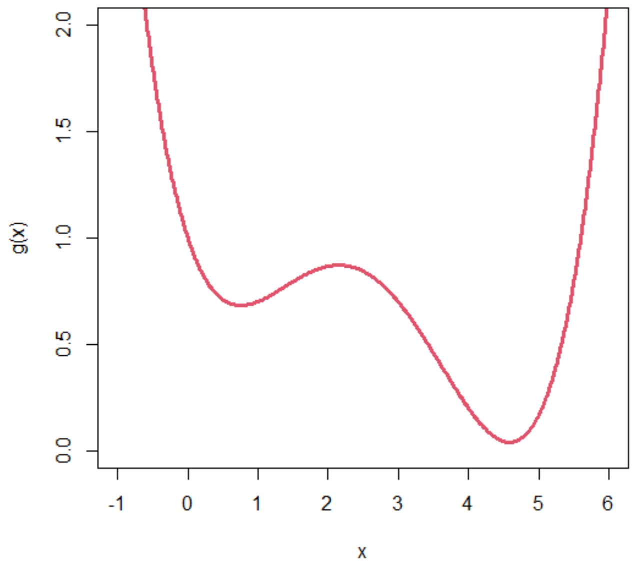
symmetric proposal:
$p\left(x^{(t)}\middle|x^*\right) = p\left(x^*\middle|x^{(t)}\right)$

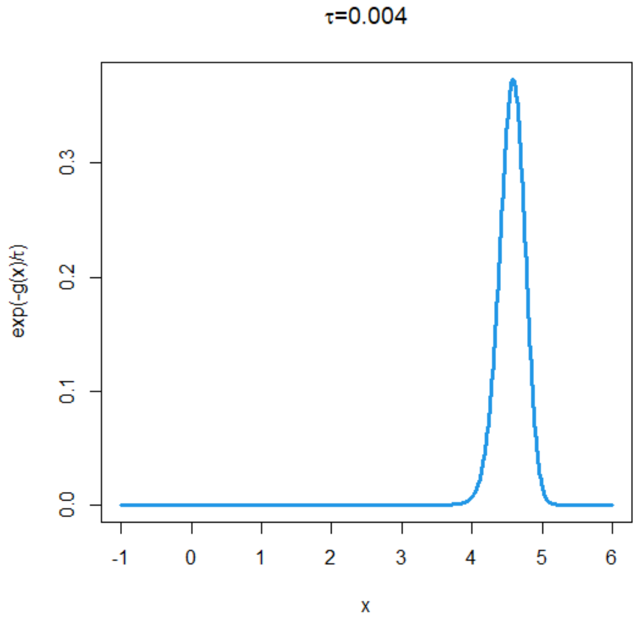
Simulated annealing and Metropolis algorithm

- For fixed temperature τ, simulated annealing algorithm is a Metropolis algorithm
- Kirkpatrick et al. (1983) proposed name simulated annealing for using it as optimisation method
- $h\left(x^{(t)}, x^*\right) = \exp\left(\frac{g\left(x^{(t)}\right) - g(x^*)}{\tau_j}\right) = \frac{\exp\left(-\frac{g(x^*)}{\tau_j}\right)}{\exp\left(-\frac{g\left(x^{(t)}\right)}{\tau_j}\right)} = \frac{f(x^*)}{f\left(x^{(t)}\right)} = R(x^{(t)}, x^*)$
- Key ingredient of Metropolis and simulated annealing alg.: Markov chain $\boldsymbol{x}^{(t)}$ **has limiting stationary distribution $\boldsymbol{f}$**; for a proof see e.g. Koski (2009)
- Requirement for all: $x^{(t)}$ irreducible and aperiodic chain

Simulated annealing: stationary distribution for fixed temperature τ

- Fixed temperature τ: Markov chain $x^{(t)}$ has limiting stationary distribution with density proportional to $f(x) = \exp\left(-\frac{g(x)}{\tau}\right)$





Convergence of simulated annealing

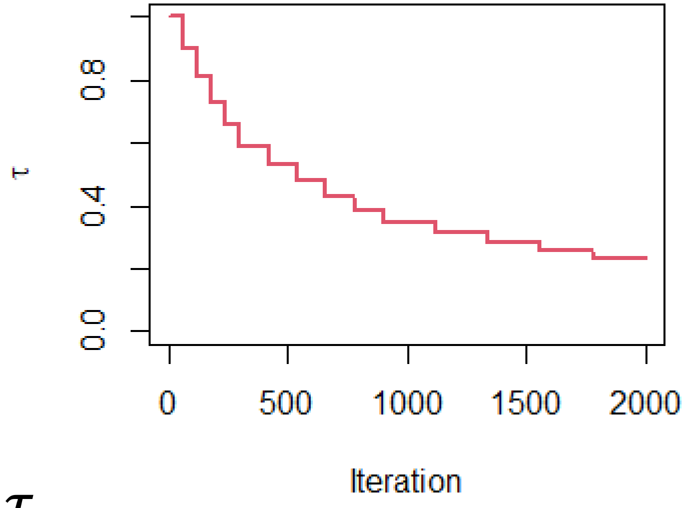
- Convergence proofs see generated sequence either as sequence of homogeneous Markov chains (one for each τ) or as one inhomogeneous Markov chain
- For discrete $\mathbb{S} = \{x_1, x_2, x_3, \ldots\}$ and g having a finite set M of global minima, simulated annealing converges with probability 1/|M| to each of the M global minima (references for proofs in Givens and Hoeting, 2013); main idea:
- Stationary distribution proportional to: $\exp\left(-\frac{g(x)}{\tau}\right)$ or to $\exp\left(-\frac{g(x)-g_{min}}{\tau}\right)$ with $g_{min} = \min\{g(x)\}$
- Therefore, if P is distribution according to stationary distribution,

$$P(x_i) = \underbrace{\exp\left(-\frac{g(x_i)-g_{min}}{\tau}\right)}_{\substack{\to 0 \text{ for } x_i \notin M, \\ \to 1 \text{ for } x_i \in M}} / \{|M| + \underbrace{\textstyle\sum_{x_j \notin M} \exp\left(-\frac{g(x_j)-g_{min}}{\tau}\right)}_{\to 0}\} \to \frac{1}{|M|}\ (x_i \in M)$$

$\boxed{\tau \to 0:}$

Convergence of simulated annealing

- To achieve convergence to a global minimum (possibly in presence of local minima) in practise, one needs:
  - Run iterations for each fixed temperature long enough such that convergence to stationary distribution achieved
  - Cool temperature slowly enough such that iterations have time to escape from local minima
- Example from Givens and Hoeting (2013; p.73):
  - 5 stages with 60 iterations, then
  - 5 stages with 120 iterations, then
  - 5 stages with 220 iterations
  - From one stage to the next, τ is decreased by 10%, **`tau <- 0.9*tau`**; final τ is $0.9^{15} = 0.206$*initial τ



Simulated annealing: + and –

+ Very easy to implement
+ Theoretical property is good: theoretically, we can guarantee convergence to a global optimum even in the presence of local optima
+ Can even handle some non-standard optimisation problems

– In practice, convergence can be "maddeningly slow"
– One needs to play around with cooling schedule to ensure convergence in practice
  - We need to run the algorithm "long enough" at each temperature (to ensure stationary distribution)
  - We need to cool the temperature slowly enough (to allow escaping from local optima)

Comparisons of algorithms or hyperparameter choices based on empirical studies

- We have several options for optimisation algorithms
- Or – within one algorithm – we can choose some hyperparameters
- A possibility is to compare the options by running them on an example problem. Better, one might want to compare options for a set of easy and difficult optimisation problems
- For comparability, often ”standard optimisation problems” used; see e.g. Liang et al. (2013)
- Can be mathematical functions or statistical optimisation problems

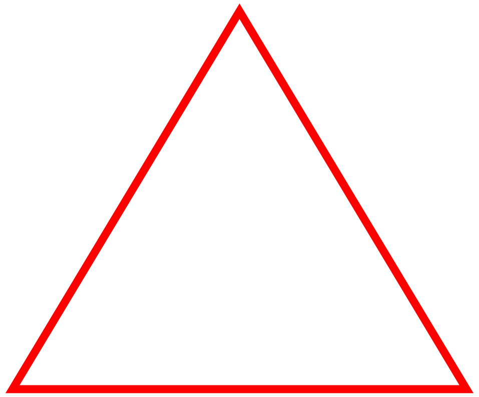
Comparisons of algorithms or hyperparameter choices based on empirical studies

- After choosing some standard optimisation problems, one needs to define a success criterion (example in Clerk, 2016)
- Possibility: count runs of algorithm leading to a solution x_s with $g(x_s) < g(x^*) + \delta$; here x^* true position of global minimum, and δ small (ideally $\delta < g(x_L) - g(x^*)$ for any local minimum x_L)
- If true success rate for an algorithm is p, we observe a $\mathrm{Bin}(1, p)$-random variable in each run

➢Success rate has sd $\sqrt{\frac{p(1-p)}{n}}$ when doing n runs and you can do informed choice of n

- E.g. $p = 0.8, n = 100 \rightarrow \text{ sd} = 0.04$.

Nelder-Mead algorithm



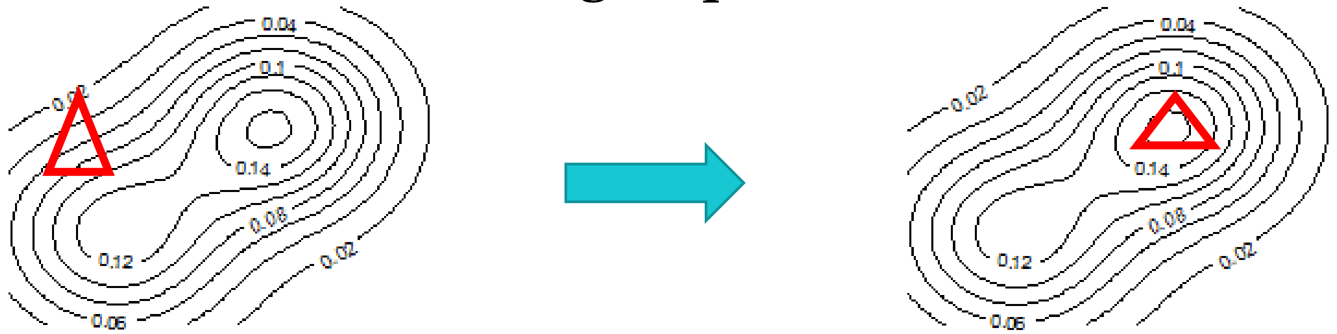
Nelder-Mead

- $\boldsymbol{x}$ p–dimensional vector, $g\colon \mathbb{R}^p \to \mathbb{R}$ function
- We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$

- Nelder-Mead method is heuristic method for p-dimensional optimisation problem (default in **`R`**-function **`optim`**)
- Positive:
  - +No computation of derivatives necessary
- Negative:
  - No theoretical guarantee for converge (counter examples exist)
  - Might be slow
- Works often well, especially if p not too large

Nelder-Mead

- Idea: Work with simplex of *p*+1 points; i.e. for two-dimensional optimisation: work with triangle
- Aim that triangle includes maximum
- Choose arbitrary starting triangle
- Change vertices to ”move the triangle upwards”



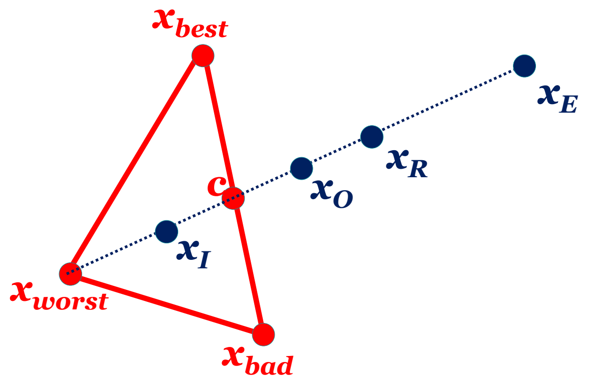
- Two animations:
    - https://www.youtube.com/watch?v=HUqLxHfxWqU
    - https://www.youtube.com/watch?v=KEGSLQ6TlBM

Nelder-Mead

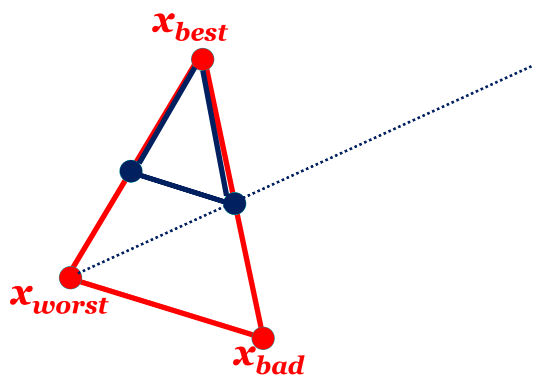
- Identify worst vertex $\boldsymbol{x}_{\boldsymbol{worst}}$ ($g(\boldsymbol{x}_{\boldsymbol{worst}})$ minimal among all vertices) and compute average $\boldsymbol{c}$ of remaining vertices
- Let $\boldsymbol{x}_{\boldsymbol{best}}$ be best and $\boldsymbol{x}_{\boldsymbol{bad}}$ be second worst vertex
- Rules for
  - Reflection
  - Expansion
  - Outer contraction
  - Inner contraction
  - Shrinkage

Nelder-Mead

- Replace $\boldsymbol{x}_{worst}$ with one of $\boldsymbol{x}_I$, $\boldsymbol{x}_O$, $\boldsymbol{x}_R$, $\boldsymbol{x}_E$ (rule depends on values for $g(\boldsymbol{x}_{worst})$, $g(\boldsymbol{x}_{bad})$, $g(\boldsymbol{x}_{best})$, $g(\boldsymbol{x}_I)$, $g(\boldsymbol{x}_O)$, $g(\boldsymbol{x}_R)$, $g(\boldsymbol{x}_E)$; see Givens and Hoeting, page 47-48) and create new simplex/triangle



- Or in specific cases: Shrink (keep $\boldsymbol{x}_{best}$ and move all other vertices towards it)



Nelder-Mead

- Nelder-Mead algorithm is quite old, but still popular
- Research is ongoing e.g. about convergence results and variants of Nelder-Mead

- Note that Nelder-Mead can be used for dimension p=1 as well
- However, there exist better gradient free algorithms for p=1
  - **`R`**-function **`optimize`** uses gradient free algorithm with convergence order q=1.324 (some requirements to function *g* necessary)

Solution x of $0 = x^3 - x - 1$; (Brent, 1973)

References

- Brent, 1973. Algorithms for minimization without derivatives. Englewood Cliffs N.J.: Prentice-Hall.
- Bonyadi MR, Michalewicz Z (2016). Stability analysis of the particle swarm optimization without stagnation assumption. *IEEE transactions on evolutionary computation*, 20(5):814-819.
- Cleghorn CW, Engelbrecht AP (2018). Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption. *Swarm Intelligence*, 12(1):1-22. An author version is available here.
- Clerc M (2012). Standard particle swarm optimization. Preprint, HAL open science.
- Clerc M (2016). Chapter 8: Particle swarms. In: *Metaheuristics*. (Siarry P ed.).
- Clerc M, Kennedy J (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, 6(1), 58-73.
- Givens GH, Hoeting JA (2013). *Computational Statistics*, 2nd edition. John Wiley & Sons, Inc., Hoboken, New Jersey.
- Kennedy J, Eberhart R (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). IEEE.
- Kirkpatrick S, Gelatt CD, Vecchi MP (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.
- Koski T (2009). *Sf2955 Computer intensive methods MCMC – On the Discrete Metropolis-Hastings Algorithm*. KTH, Stockholm. https://www.math.kth.se/matstat/gru/sf2955/metropolis3.pdf
- Liang, J. J., Qu, B. Y., & Suganthan, P. N. (2013). Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University*, Singapore, 635(2).
- Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21, 1087–1092.