Metaheuristic Optimization: Nelder-Mead Simplex Algorithm

Adaptive and Cooperative Algorithms (ECE 457A)

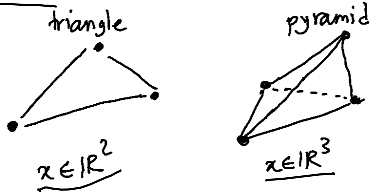
ECE, MME, and MSCI Departments,
University of Waterloo, ON, Canada

Course Instructor: Benyamin Ghojogh

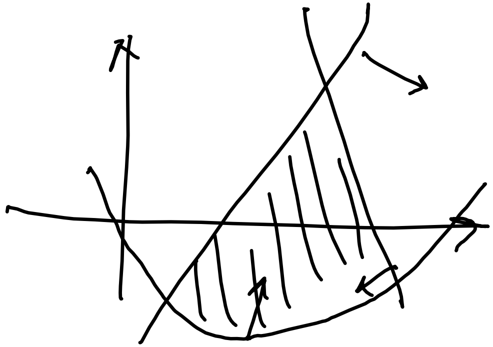
Fall 2023

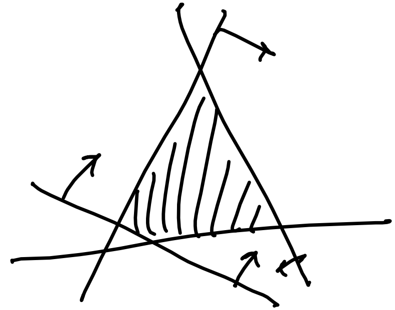
Nelder-Mead Simplex Algorithm

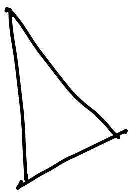
- The **Nelder-Mead simplex algorithm**, also called the **Nelder-Mead method**, was proposed by John A. Nelder and Roger Mead in 1965 [1].
- Used in "**fminsearch**" function of **MATLAB**: https://www.mathworks.com/help/matlab/ref/fminsearch.html
- Its idea:
  - If the dimensionality of optimization variable is d, choose $d+1$ random points in the feasible set to make a **simplex**.

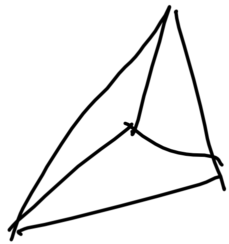


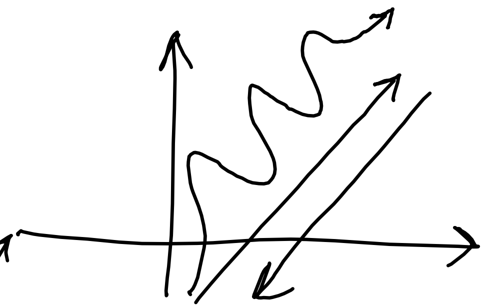
  - Update this simplex iteratively until it converges to the optimal solution (it gradually moves toward the solution and shrinks to the solution.)

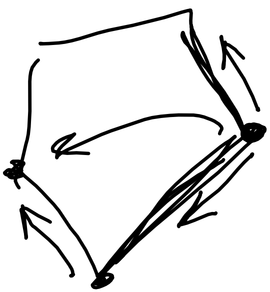


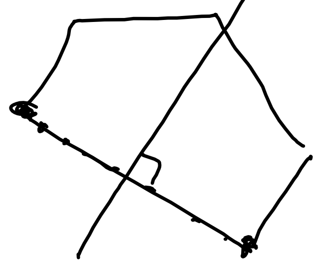


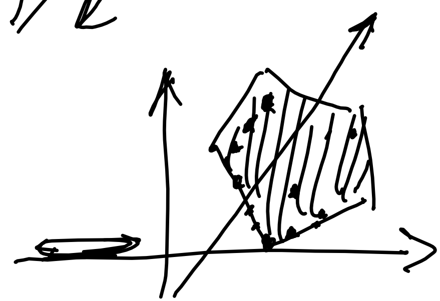








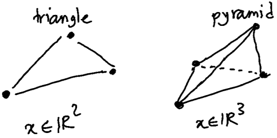




2)

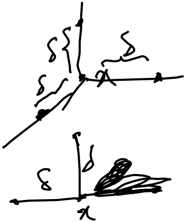
Nelder-Mead Method: initial simplex

- If the dimensionality of optimization variable is d, choose $d+1$ random points in the feasible set to make a **simplex**.



- This initial simplex is important. A too small simplex may get stuck in a local optimum (cannot do enough exploration).
- It is suggested in [1] to select the initial simplex as the following:
  - a random point for $\boldsymbol{x}_1 = [x_{11}, x_{12}, \ldots, x_{1d}]^\top$
  - each of $\{\boldsymbol{x}_2, \ldots, \boldsymbol{x}_{d+1}\}$ is a fixed step along each dimension in turn:

$$\boldsymbol{x}_2 = [x_{11} + \delta, x_{12}, \ldots, x_{1d}]^\top,$$
$$\boldsymbol{x}_3 = [x_{11}, x_{12} + \delta, \ldots, x_{1d}]^\top,$$
$$\vdots$$
$$\boldsymbol{x}_{d+1} = [x_{11}, x_{12}, \ldots, x_{1d} + \delta]^\top,$$



where $\delta > 0$ is a not-too-small number.

Nelder-Mead Method: order

- We want to **minimize** the function $f(.)$. In the feasibility set, we make some simplex and change it in the feasibility set iteratively to converge to the solution.
- **order**: at the start of every iteration, order (sort) the corners of simplex:

$$f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_2) \leq \cdots \leq f(\boldsymbol{x}_{d+1}),$$

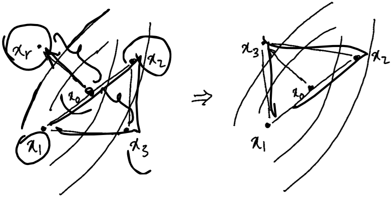
where we denote the best and worst corners by $\boldsymbol{x}_1$ and $\boldsymbol{x}_{d+1}$, respectively.

Nelder-Mead Method: order and reflection

- **centroid**: the centroid of all points except $\boldsymbol{x}_{d+1}$ as: $\boldsymbol{x}_o = \frac{1}{d}\sum_{i=1}^{d} \boldsymbol{x}_i$.
- **reflection**:
  - the reflected point:

$$\boxed{\boldsymbol{x}_r = \boldsymbol{x}_o + \alpha(\boldsymbol{x}_o - \boldsymbol{x}_{d+1}).}$$

  - $\alpha > 0$, usually $\alpha = 1$.
  - if $f(\boldsymbol{x}_1) \le f(\boldsymbol{x}_r) < f(\boldsymbol{x}_d)$:
    - ★ replace the worst point with the reflected point, $\boldsymbol{x}_{d+1} := \boldsymbol{x}_r$.
    - ★ go to the next iteration and order the points again.



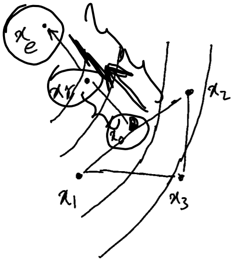
- It has some connection with **opposition learning** proposed in 2005 [2] and used in metaheuristic optimization in 2008 [3].

Nelder-Mead Method: expansion

- **expansion**: if $f(\boldsymbol{x}_r) \leq f(\boldsymbol{x}_1)$:
  - the expanded point:

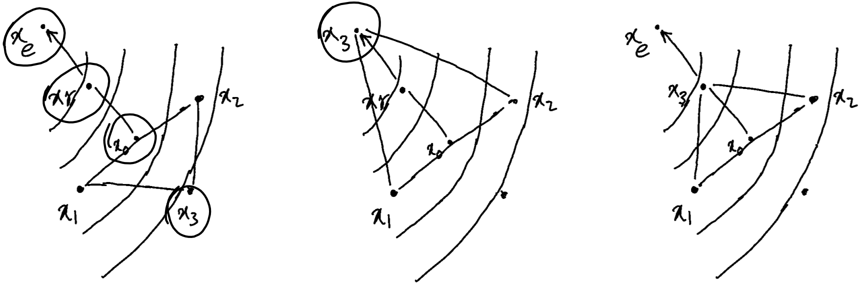
$$\boldsymbol{x}_e = \boldsymbol{x}_o + \gamma(\boldsymbol{x}_r - \boldsymbol{x}_o).$$

  - $\gamma > 1$, usually $\gamma = 2$.



Nelder-Mead Method: expansion

- if the expanded point is better than the reflected point, $f(\boldsymbol{x}_e) < f(\boldsymbol{x}_r)$:
  - replace the worst point with the expanded point, $\boldsymbol{x}_{d+1} := \boldsymbol{x}_e$.
- else, $f(\boldsymbol{x}_e) \geq f(\boldsymbol{x}_r)$:
  - replace the worst point with the reflected point, $\boldsymbol{x}_{d+1} := \boldsymbol{x}_r$.
- go to the next iteration and order the points again.

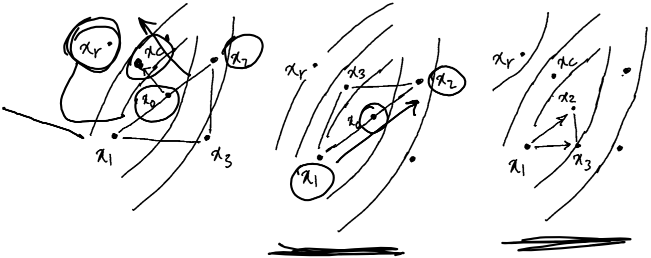


Nelder-Mead Method: contraction & shrinking

- **contraction**: if $f(\boldsymbol{x}_r) \geq f(\boldsymbol{x}_d)$:
  - if $f(\boldsymbol{x}_r) < f(\boldsymbol{x}_{d+1})$:
    - the contracted point **outside**:

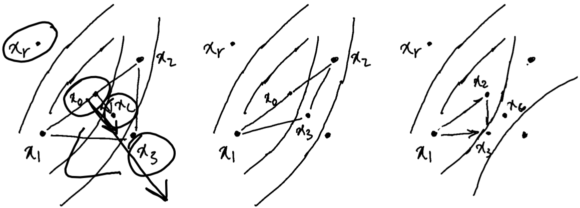
$$\boxed{\boldsymbol{x}_c = \boldsymbol{x}_o + \rho(\boldsymbol{x}_r - \boldsymbol{x}_o).}$$

    - $0 < \rho \leq 0.5$, usually $\rho = 0.5$.
    - if the contracted point is better than the reflected point, $f(\boldsymbol{x}_c) < f(\boldsymbol{x}_r)$:
      replace the worst point with the contracted point, $\boldsymbol{x}_{d+1} := \boldsymbol{x}_c$.
    - else, $f(\boldsymbol{x}_c) \geq f(\boldsymbol{x}_r)$:
      shrinking: replace all points (except the best point $\boldsymbol{x}_1$) with the reflected point, $\boldsymbol{x}_i := \boldsymbol{x}_1 + \sigma(\boldsymbol{x}_i - \boldsymbol{x}_1)$, where $\sigma = 0.5$.
    - go to the next iteration and order the points again.



Nelder-Mead Method: contraction & shrinking

- **contraction**: if $f(\boldsymbol{x}_r) \geq f(\boldsymbol{x}_d)$:
  - if $f(\boldsymbol{x}_r) \geq f(\boldsymbol{x}_{d+1})$:
    - ★ the contracted point **inside**:
    $$\boxed{\boldsymbol{x}_c = \boldsymbol{x}_o + \rho(\boldsymbol{x}_{d+1} - \boldsymbol{x}_o).}$$
    - ★ $0 < \rho \leq 0.5$, usually $\rho = 0.5$.
    - ★ if the contracted point is better than the worst point, $f(\boldsymbol{x}_c) < f(\boldsymbol{x}_{d+1})$:
      replace the worst point with the contracted point, $\boldsymbol{x}_{d+1} := \boldsymbol{x}_c$.
    - ★ else, $f(\boldsymbol{x}_c) \geq f(\boldsymbol{x}_{d+1})$:
      shrinking: replace all points (except the best point $\boldsymbol{x}_1$) with the reflected point, $\boldsymbol{x}_i := \boldsymbol{x}_1 + \sigma(\boldsymbol{x}_i - \boldsymbol{x}_1)$, where $\sigma = 0.5$.
    - ★ go to the next iteration and order the points again.



Nelder-Mead Method: summary

initialize simplex with $(d+1)$ points in feasibility set

loop until convergence:

calculate centroid: x_0

calculate reflected point: x_r

if $f(x_1) \leqslant f(x_r) \leqslant f(x_d)$: 1

$x_{d+1} \leftarrow x_r$

continue

elseif $f(x_r) \leqslant f(x_1)$: 2

$x_e = x_0 + \gamma(x_r - x_0)$

if $f(x_e) < f(x_r)$:

$x_{d+1} \leftarrow x_e$

else:

$x_{d+1} \leftarrow x_r$

continue

Nelder-Mead Method: summary

loop until convergence:

elseif $f(x_r) \geq f(x_d)$: 3

if $f(x_r) < f(x_{d+1})$:

$x_c = x_0 + \rho(x_r - x_0)$

if $f(x_c) < f(x_r)$:

$x_{d+1} \leftarrow x_c$

else:

$x_i \leftarrow x_1 + \sigma(x_i - x_1) \quad \forall i$

continue

else:

$x_c = x_0 + \rho(x_{d+1} - x_0)$

if $f(x_c) < f(x_{d+1})$:

$x_{d+1} \leftarrow x_c$

else:

$x_i \leftarrow x_1 + \sigma(x_i - x_1) \quad \forall i$

continue

Acknowledgment

- Another YouTube video of mine about this algorithm: https://www.youtube.com/watch?v=s-6rV1nMw0w

References

[1] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[2] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, vol. 1, pp. 695–701, IEEE, 2005.

[3] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary computation*, vol. 12, no. 1, pp. 64–79, 2008.