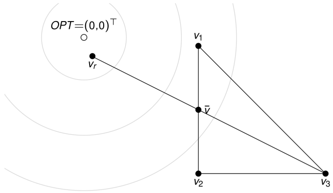
Optimization in Machine Learning

Nelder-Mead method



Learning goals

- General idea
- Reflection, expansion, contraction
- Advantages & disadvantages
- Examples

NELDER-MEAD METHOD

Nelder-Mead is a robust procedure, which also works without derivatives.

Generalization of bisection in d-dimensional space.

Instead of an interval, a simplex is used, a geometric figure defined by $d + 1$ points:

- $d =$1 interval
- $d =$2 triangle
- $d =$3 tetrahedron ...

NELDER-MEAD METHOD

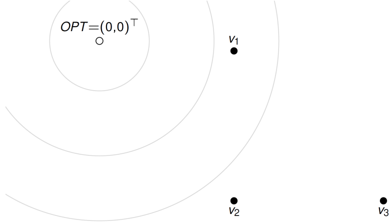
A version of the **Nelder-Mead** method:

Initialization: Choose $d + 1$ random, linearly independent points $\mathbf{v}_i$ ($\mathbf{v}_i$ are vertices: corner points of the simplex/polytope):

1. **Order**: Order points according to ascending function values

$$f(\mathbf{v}_1) \leq f(\mathbf{v}_2) \leq \ldots \leq f(\mathbf{v}_d) \leq f(\mathbf{v}_{d+1}).$$

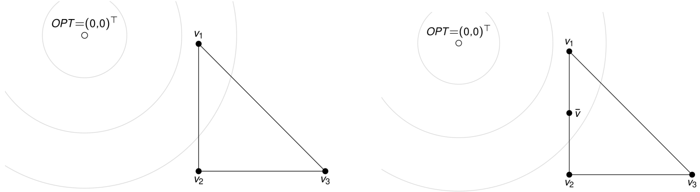
   with $\mathbf{v}_1$ best point, $\mathbf{v}_{d+1}$ worst point.



NELDER-MEAD METHOD

2. Calculate **centroid** without worst point

$$\bar{\mathbf{v}} = \frac{1}{d}\sum_{i=1}^{d} \mathbf{v}_i .$$

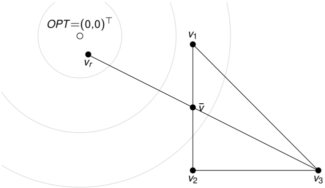


NELDER-MEAD METHOD

3. **Reflection:** calculate reflection point

$$\mathbf{v}_r = \bar{\mathbf{v}} + \rho(\bar{\mathbf{v}} - \mathbf{v}_{d+1}),$$

with $\rho > 0$. Calculate $f(\mathbf{v}_r)$.



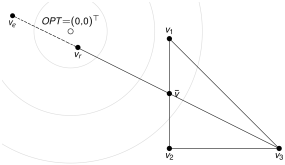
Note: the standard value for the reflection coefficient is $\rho = 1$.

NELDER-MEAD METHOD

We now distinguish three cases:

- **Case 1**: $f(\mathbf{v}_1) \leq f(\mathbf{v}_r) < f(\mathbf{v}_d)$
  If the reflection point is better than the second worst corner, but not better than the best corner, we accept $\mathbf{v}_r$ and discard $\mathbf{v}_{d+1}$.

- **Case 2**: $f(\mathbf{v}_r) < f(\mathbf{v}_1)$
  If the reflection point is better than the best corner so far, we "expand" the current point (**Expansion**) to find out if we could get even better in the direction of $\boldsymbol{v}_r$:



This is **case 2**: The reflection point $\boldsymbol{v}_r$ is better than the best point $\boldsymbol{v}_1$.
If the **expansion** does not return a better point than $\boldsymbol{v}_r$, accept $\boldsymbol{v}_r$ and reject $\boldsymbol{v}_3$.

$$\mathbf{v}_e = \bar{\mathbf{v}} + \chi(\mathbf{v}_r - \bar{\mathbf{v}}), \quad \chi > 1.$$

  We discard $\boldsymbol{v}_{d+1}$ in favor of the better of the two corners $\boldsymbol{v}_r, \boldsymbol{v}_e$.

Note: the standard value for the expansion coefficient is $\chi = 2$.

NELDER-MEAD METHOD

- **Case 3**: $f(\mathbf{v}_r) \geq f(\mathbf{v}_d)$ we find that running toward $\boldsymbol{v}_r$ was not purposeful. We calculate a **contraction** point:

$$\mathbf{v}_c = \bar{\mathbf{v}} + \gamma(\mathbf{v}_{d+1} - \bar{\mathbf{v}})$$

with $0 < \gamma \leq 0.5$.

    - If $\mathbf{v}_c$ is better than the worst point, we accept $\mathbf{v}_c$.
    - Otherwise, we shrink the **entire** Simplex (**Shrinking**):

$$\mathbf{v}_i = \mathbf{v}_1 + \sigma(\mathbf{v}_i - \mathbf{v}_1) \quad \text{for all } i$$

In each of the three cases, we then continue with step 1 until a termination criterion is met.

Note: standard values for the contraction and shrinkage coefficient are $\gamma = 0.5$ and $\sigma = 0.5$.

NELDER-MEAD

Advantages:

- Nelder-Mead only needs function values (no gradients).
- Very robust, often works well for non-differentiable functions.

Drawbacks:

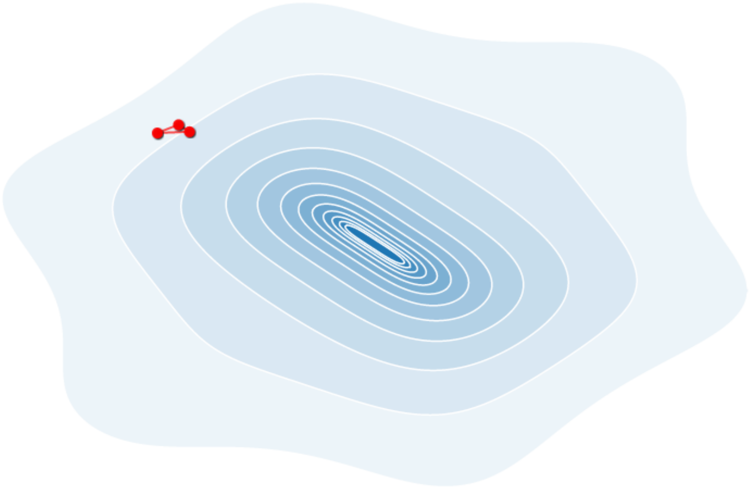
- Relatively slow.
- Not every step leads to an improvement of the solution, only the mean over the points in the simplex is reduced.
- No guarantee for convergence in local optimum.

Visualization:
a good illustration of the Nelder-Mead algorithm for one- and higher-dimensional optimization problems can be found at the following link:
`http://www.benfrederickson.com/numerical-optimization/`

Attention: Nelder-Mead is default method of **R** function **optim()**. If gradient is easy to calculate, BFGS is preferred.

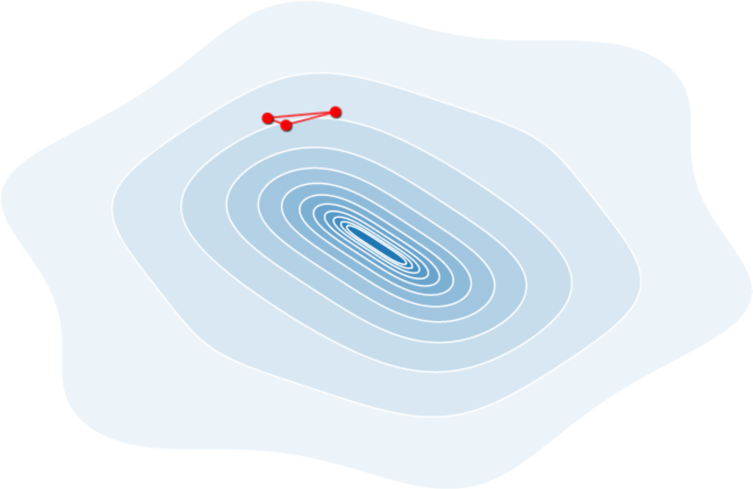
NELDER-MEAD VISUALIZATION IN 2D

$$\min_{\mathbf{x}} f(x_1, x_2) = x_1^2 + x_2^2 + x_1 \cdot \sin x_2 + x_2 \cdot \sin x_1$$



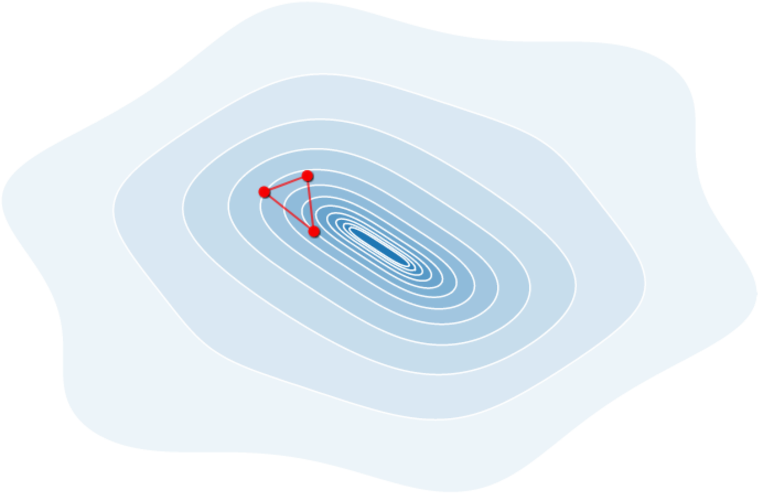
NELDER-MEAD VISUALIZATION IN 2D

$$\min_{\mathbf{x}} f(x_1, x_2) = x_1^2 + x_2^2 + x_1 \cdot \sin x_2 + x_2 \cdot \sin x_1$$



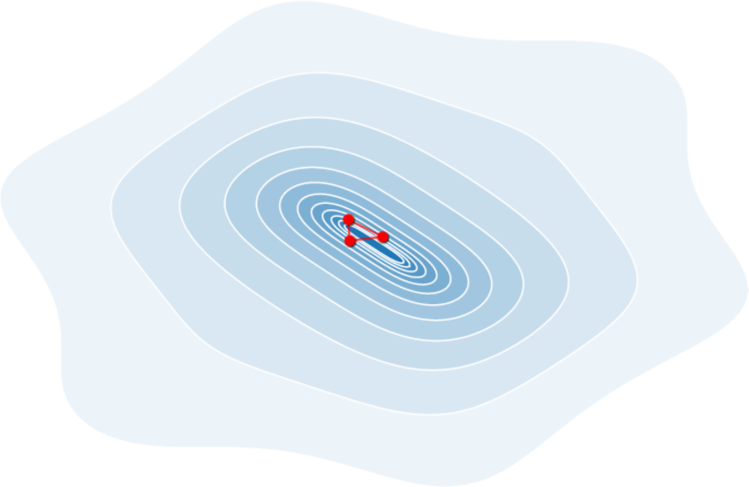
NELDER-MEAD VISUALIZATION IN 2D

$$\min_{\mathbf{x}} f(x_1, x_2) = x_1^2 + x_2^2 + x_1 \cdot \sin x_2 + x_2 \cdot \sin x_1$$

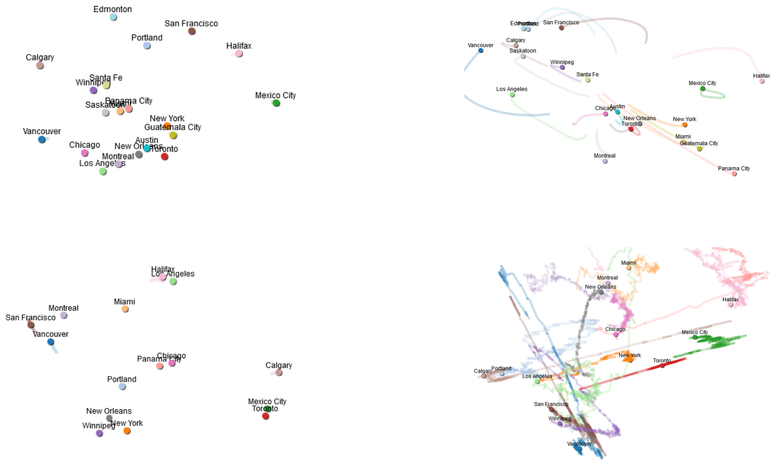


NELDER-MEAD VISUALIZATION IN 2D

$$\min_{\mathbf{x}} f(x_1, x_2) = x_1^2 + x_2^2 + x_1 \cdot \sin x_2 + x_2 \cdot \sin x_1$$

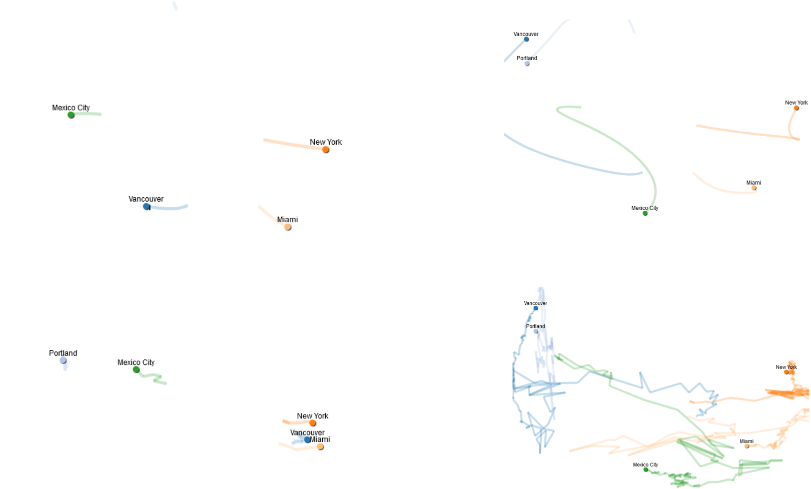


NELDER-MEAD VS. GD



NM for multidim. scaling. Convert a matrix of distances to 2D coords, so the distances approximately stay. For >10 cities, GD (top) converges well for an appropriate learning rate. NM (bottom) completely fails to converge, even after many iterations.

NELDER-MEAD VS. GD



Even for only 5 cities, NM (bottom) struggles. GD (top) again works well.