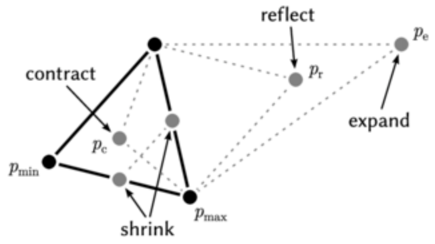
WIKIPEDIA

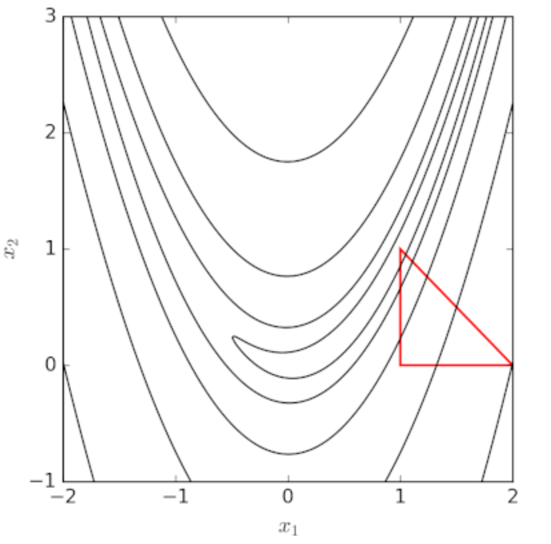
Nelder–Mead method

The **Nelder–Mead method** (also **downhill simplex method**, **amoeba method**, or **polytope method**) is a numerical method used to find the minimum or maximum of an objective function in a multidimensional space. It is a direct search method (based on function comparison) and is often applied to nonlinear optimization problems for which derivatives may not be known. However, the Nelder–Mead technique is a heuristic search method that can converge to non-stationary points[1] on problems that can be solved by alternative methods.[2]



An iteration of the Nelder-Mead method over two-dimensional space.

The Nelder–Mead technique was proposed by John Nelder and Roger Mead in 1965,[3] as a development of the method of Spendley et al.[4]



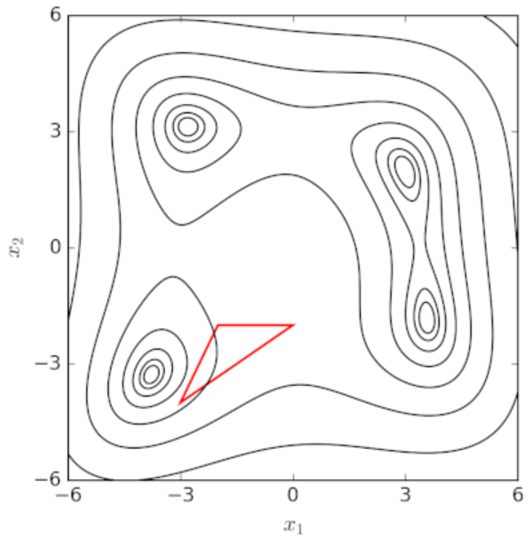
Search over the Rosenbrock banana function

Overview

The method uses the concept of a simplex, which is a special polytope of $n + 1$ vertices in n dimensions. Examples of simplices include a line segment on a line, a triangle on a plane, a tetrahedron in three-dimensional space and so forth.

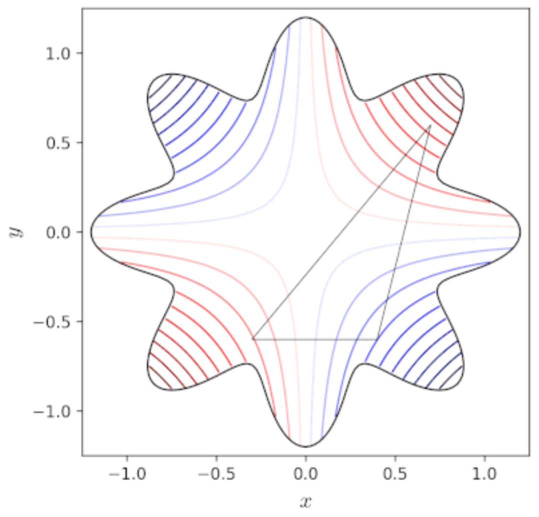
The method approximates a local optimum of a problem with n variables when the objective function varies smoothly and is unimodal. Typical implementations minimize functions, and we maximize $f(\mathbf{x})$ by minimizing $-f(\mathbf{x})$.

For example, a suspension bridge engineer has to choose how thick each strut, cable, and pier must be. These elements are interdependent, but it is not easy to visualize the impact of changing any specific element. Simulation of such complicated structures is often extremely computationally expensive to run, possibly taking upwards of hours per execution. The Nelder–Mead method requires, in the original variant, no more than two evaluations per iteration, except for the *shrink* operation described later, which is attractive compared to some other direct-search optimization methods. However, the overall number of iterations to proposed optimum may be high.



Search over Himmelblau's function

Nelder–Mead in n dimensions maintains a set of $n + 1$ test points arranged as a simplex. It then extrapolates the behavior of the objective function measured at each test point in order to find a new test point and to replace one of the old test points with the new one, and so the technique progresses. The simplest approach is to replace the worst point with a point reflected through the centroid of the remaining n points. If this point is better than the best current point, then we can try stretching exponentially out along this line. On the other hand, if this new point isn't much better than the previous value, then we are stepping across a valley, so we shrink the simplex towards a better point. An intuitive explanation of the algorithm from "Numerical Recipes": [5]



Nelder–Mead minimum search of Simionescu's function. Simplex vertices are ordered by their value, with 1 having the lowest (best) value.

> The downhill simplex method now takes a series of steps, most steps just moving the point of the simplex where the function is largest (“highest point”) through the opposite face of the simplex to a lower point. These steps are called reflections, and they are constructed to conserve the volume of the simplex (and hence maintain its nondegeneracy). When it can do so, the method expands the simplex in one or another direction to take larger steps. When it reaches a “valley floor”, the method contracts itself in the transverse direction and tries to ooze down the valley. If there is a situation where the simplex is trying to “pass through the eye of a needle”, it contracts itself in all directions, pulling itself in around its lowest (best) point.

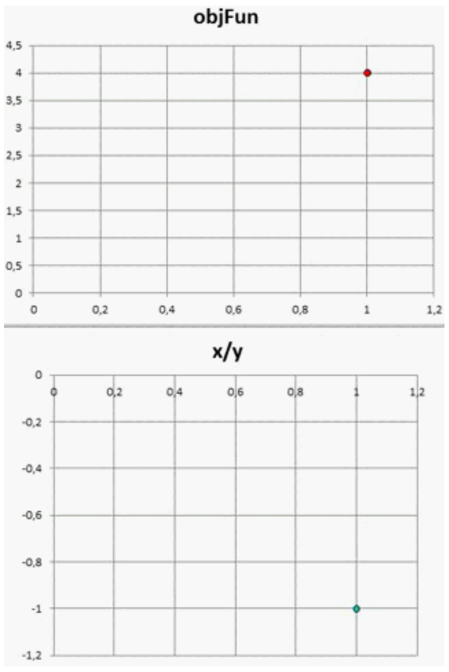
Unlike modern optimization methods, the Nelder–Mead heuristic can converge to a non-stationary point, unless the problem satisfies stronger conditions than are necessary for modern methods.[1] Modern improvements over the Nelder–Mead heuristic have been known since 1979.[2]

Many variations exist depending on the actual nature of the problem being solved. A common variant uses a constant-size, small simplex that roughly follows the gradient direction (which gives steepest descent). Visualize a small triangle on an elevation map flip-flopping its way down a valley to a local bottom. This method is also known as the **flexible polyhedron method**. This, however, tends to perform poorly against the method described in this article because it makes small, unnecessary steps in areas of little interest.

One possible variation of the NM algorithm

(This approximates the procedure in the original Nelder–Mead article.)

We are trying to minimize the function $f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^n$. Our current test points are $\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}$.



Nelder–Mead method applied to the Rosenbrock function

1. Order according to the values at the vertices:

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_{n+1}).$$

Check whether method should stop. See **Termination** below. Sometimes inappropriately called "convergence".

2. Calculate $\mathbf{x}_o$, the centroid of all points except $\mathbf{x}_{n+1}$.

3. Reflection

Compute reflected point $\mathbf{x}_r = \mathbf{x}_o + \alpha(\mathbf{x}_o - \mathbf{x}_{n+1})$ with $\alpha > 0$.
If the reflected point is better than the second worst, but not better than the best, i.e. $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$,
then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the reflected point $\mathbf{x}_r$, and go to step 1.

4. Expansion

If the reflected point is the best point so far, $f(\mathbf{x}_r) < f(\mathbf{x}_1)$,
then compute the expanded point $\mathbf{x}_e = \mathbf{x}_o + \gamma(\mathbf{x}_r - \mathbf{x}_o)$ with $\gamma > 1$.
If the expanded point is better than the reflected point, $f(\mathbf{x}_e) < f(\mathbf{x}_r)$,
then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the expanded point $\mathbf{x}_e$ and go to step 1;
else obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the reflected point $\mathbf{x}_r$ and go to step 1.

5. Contraction

Here it is certain that $f(\mathbf{x}_r) \geq f(\mathbf{x}_n)$. (Note that $\mathbf{x}_n$ is second or "next" to highest.)
If $f(\mathbf{x}_r) < f(\mathbf{x}_{n+1})$,
then compute the contracted point on the outside $\mathbf{x}_c = \mathbf{x}_o + \rho(\mathbf{x}_r - \mathbf{x}_o)$ with $0 < \rho \leq 0.5$.

```
        If the contracted point is better than the reflected point, i.e. $f(\mathbf{x}_c) < f(\mathbf{x}_r)$,
            then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the contracted point $\mathbf{x}_c$ and go to step 1;
        Else go to step 6;
    If $f(\mathbf{x}_r) \geq f(\mathbf{x}_{n+1})$,
        then compute the contracted point on the inside $\mathbf{x}_c = \mathbf{x}_o + \rho(\mathbf{x}_{n+1} - \mathbf{x}_o)$ with $0 < \rho \leq 0.5$.
        If the contracted point is better than the worst point, i.e. $f(\mathbf{x}_c) < f(\mathbf{x}_{n+1})$,
            then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the contracted point $\mathbf{x}_c$ and go to step 1;
        Else go to step 6;
```

6. Shrink

```
    Replace all points except the best ($\mathbf{x}_1$) with
    $\mathbf{x}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_i - \mathbf{x}_1)$ and go to step 1.
```

Note: α, γ, ρ and σ are respectively the reflection, expansion, contraction and shrink coefficients. Standard values are $\alpha = 1$, $\gamma = 2$, $\rho = 1/2$ and $\sigma = 1/2$.

For the **reflection**, since $\mathbf{x}_{n+1}$ is the vertex with the higher associated value among the vertices, we can expect to find a lower value at the reflection of $\mathbf{x}_{n+1}$ in the opposite face formed by all vertices $\mathbf{x}_i$ except $\mathbf{x}_{n+1}$.

For the **expansion**, if the reflection point $\mathbf{x}_r$ is the new minimum along the vertices, we can expect to find interesting values along the direction from $\mathbf{x}_o$ to $\mathbf{x}_r$.

Concerning the **contraction**, if $f(\mathbf{x}_r) > f(\mathbf{x}_n)$, we can expect that a better value will be inside the simplex formed by all the vertices $\mathbf{x}_i$.

Finally, the **shrink** handles the rare case that contracting away from the largest point increases f, something that cannot happen sufficiently close to a non-singular minimum. In that case we contract towards the lowest point in the expectation of finding a simpler landscape. However, Nash notes that finite-precision arithmetic can sometimes fail to actually shrink the simplex, and implemented a check that the size is actually reduced.[6]

Initial simplex

The initial simplex is important. Indeed, a too small initial simplex can lead to a local search, consequently the NM can get more easily stuck. So this simplex should depend on the nature of the problem. However, the original article suggested a simplex where an initial point is given as $\mathbf{x}_1$, with the others generated with a fixed step along each dimension in turn. Thus the method is sensitive to scaling of the variables that make up $\mathbf{x}$.

Termination

Criteria are needed to break the iterative cycle. Nelder and Mead used the sample standard deviation of the function values of the current simplex. If these fall below some tolerance, then the cycle is stopped and the lowest point in the simplex returned as a proposed optimum. Note that a very "flat" function may have almost equal function values over a large domain, so that the solution will be sensitive to the tolerance. Nash adds the test for shrinkage as another termination criterion.[6] Note that programs terminate, while iterations may converge.

See also

- Derivative-free optimization
- COBYLA
- NEWUOA
- LINCOA
- Nonlinear conjugate gradient method
- Levenberg–Marquardt algorithm
- Broyden–Fletcher–Goldfarb–Shanno or BFGS method
- Differential evolution
- Pattern search (optimization)
- CMA-ES

References

1.
    - Powell, Michael J. D. (1973). "On Search Directions for Minimization Algorithms". *Mathematical Programming*. **4**: 193–201. doi:10.1007/bf01584660 (https://doi.org/10.1007%2Fbf01584660). S2CID 45909653 (https://api.semanticscholar.org/CorpusID:45909653).
    - McKinnon, K. I. M. (1999). "Convergence of the Nelder–Mead simplex method to a non-stationary point". *SIAM Journal on Optimization*. **9**: 148–158. CiteSeerX 10.1.1.52.3900 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.3900). doi:10.1137/S1052623496303482 (https://doi.org/10.1137%2FS1052623496303482). (algorithm summary online).
2.
    - Yu, Wen Ci. 1979. "Positive basis and a class of direct search techniques". *Scientia Sinica* [*Zhongguo Kexue*]: 53—68.
    - Yu, Wen Ci. 1979. "The convergent property of the simplex evolutionary technique". *Scientia Sinica* [*Zhongguo Kexue*]: 69–77.
    - Kolda, Tamara G.; Lewis, Robert Michael; Torczon, Virginia (2003). "Optimization by direct search: new perspectives on some classical and modern methods". *SIAM Rev*. **45** (3): 385–482. CiteSeerX 10.1.1.96.8672 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.96.8672). doi:10.1137/S003614450242889 (https://doi.org/10.1137%2FS003614450242889).
    - Lewis, Robert Michael; Shepherd, Anne; Torczon, Virginia (2007). "Implementing generating set search methods for linearly constrained minimization". *SIAM J. Sci. Comput*. **29** (6): 2507–2530. CiteSeerX 10.1.1.62.8771 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.8771). doi:10.1137/050635432 (https://doi.org/10.1137%2F050635432).
3. Nelder, John A.; R. Mead (1965). "A simplex method for function minimization". *Computer Journal*. **7** (4): 308–313. doi:10.1093/comjnl/7.4.308 (https://doi.org/10.1093%2Fcomjnl%2F7.4.308).
4. Spendley, W.; Hext, G. R.; Himsworth, F. R. (1962). "Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation". *Technometrics*. **4** (4): 441–461. doi:10.1080/00401706.1962.10490033 (https://doi.org/10.1080%2F00401706.1962.10490033).

5.
   - Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. (2007). "Section 10.5. Downhill Simplex Method in Multidimensions" (http://apps.nrbook.com/empanel/index.html#pg=502). *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

6. Nash, J. C. (1979). *Compact Numerical Methods: Linear Algebra and Function Minimisation*. Bristol: Adam Hilger. ISBN 978-0-85274-330-0.

Further reading

- Avriel, Mordecai (2003). *Nonlinear Programming: Analysis and Methods*. Dover Publishing. ISBN 978-0-486-43227-4.
- Coope, I. D.; Price, C. J. (2002). "Positive Bases in Numerical Optimization". *Computational Optimization & Applications*. **21** (2): 169–176. doi:10.1023/A:1013760716801 (https://doi.org/10.1023%2FA%3A1013760716801). S2CID 15947440 (https://api.semanticscholar.org/CorpusID:15947440).
- Gill, Philip E.; Murray, Walter; Wright, Margaret H. (1981). "Methods for Multivariate Non-Smooth Functions". *Practical Optimization* (https://archive.org/details/practicaloptimiz00gill). New York: Academic Press. pp. 93 (https://archive.org/details/practicaloptimiz00gill/page/n110)–96. ISBN 978-0-12-283950-4.
- Kowalik, J.; Osborne, M. R. (1968). *Methods for Unconstrained Optimization Problems* (https://archive.org/details/methodsforuncons0000kowa/page/24). New York: Elsevier. pp. 24–27 (https://archive.org/details/methodsforuncons0000kowa/page/24). ISBN 0-444-00041-0.
- Swann, W. H. (1972). "Direct Search Methods". In Murray, W. (ed.). *Numerical Methods for Unconstrained Optimization*. New York: Academic Press. pp. 13–28. ISBN 978-0-12-512250-4.

External links

- Nelder–Mead (Downhill Simplex) explanation and visualization with the Rosenbrock banana function (http://www.brnt.eu/phd/node10.html#SECTION00622200000000000000)
- John Burkardt: Nelder–Mead code in Matlab (http://people.sc.fsu.edu/~burkardt/m_src/asa047/nelmin.m) - note that a variation of the Nelder–Mead method is also implemented by the Matlab function fminsearch.
- Nelder-Mead optimization in Python in the SciPy library. (https://docs.scipy.org/doc/scipy/reference/optimize.minimize-neldermead.html#optimize-minimize-neldermead)
- nelder-mead (https://github.com/fchollet/nelder-mead) - A Python implementation of the Nelder–Mead method
- SOVA 1.0 (freeware) (http://people.fsv.cvut.cz/~svobodal/sova/) - Simplex Optimization for Various Applications
- [1] (https://web.archive.org/web/20190512002351/http://www.hillstormer.es/) - HillStormer, a practical tool for nonlinear, multivariate and linear constrained Simplex Optimization by Nelder Mead.