Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

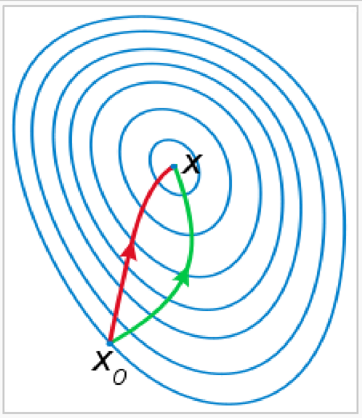
Print/export
Create a book
Download as PDF
Printable version

Languages
Français
Polski
Edit links

Newton's method in optimization

From Wikipedia, the free encyclopedia

In calculus, Newton's method is an iterative method for finding the roots of a differentiable function f (i.e. solutions to the equation $f(x) = 0$). In optimization, Newton's method is applied to the derivative f' of a twice-differentiable function f to find the roots of the derivative (solutions to $f'(x) = 0$), also known as the stationary points of f.



A comparison of gradient descent (green) and Newton's method (red) for minimizing a function (with small step sizes). Newton's method uses curvature information to take a more direct route.

Contents [hide]

Method [edit]

In the one-dimensional problem, Newton's method attempts to construct a sequence x_n from an initial guess x_0 that converges towards some value x^* satisfying $f'(x^*)=0$. This x^* is a stationary point of f.

The second order Taylor expansion $f_T(x)$ of f around x_n is:

$$f_T(x) = f_T(x_n + \Delta x) \approx f(x_n) + f'(x_n)\Delta x + \frac{1}{2} f''(x_n)\Delta x^2.$$

The last expression attains its extremum with respect to Δx when its derivative is equal to zero, i.e. when:

$$0 = \frac{d}{d\Delta x}\left(f(x_n) + f'(x_n)\Delta x + \frac{1}{2} f''(x_n)\Delta x^2\right) = f'(x_n) + f''(x_n)\Delta x.$$

For the value of $\Delta x = -\frac{f'(x_n)}{f''(x_n)}$, which satisfies this equation, it can be hoped that

$x_{n+1} = x_n + \Delta x = x_n - \frac{f'(x_n)}{f''(x_n)}$ will be closer to a stationary point x^*. This is the case provided that f is a twice-differentiable function and other technical conditions are satisfied; the sequence $x_0, x_1, x_2, \ldots$ converges to x^*.

Geometric interpretation [edit]

The geometric interpretation of Newton's method is that at each iteration one approximates $f(\mathbf{x})$ by a quadratic function around $\mathbf{x}_n$, and then takes a step towards the maximum/minimum of that quadratic function (in higher dimensions, this may also be a saddle point). Note that if $f(\mathbf{x})$ happens to *be* a quadratic function, then the exact extremum is found in one step.

Higher dimensions [edit]

The above iterative scheme can be generalized to several dimensions by replacing the derivative with the gradient, $\nabla f(\mathbf{x})$, and the reciprocal of the second derivative with the inverse of the Hessian matrix, $Hf(\mathbf{x})$. One obtains the iterative scheme

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [Hf(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n), \ n \geq 0.$$

Often Newton's method is modified to include a small step size $\gamma \in (0,1)$ instead of $\gamma = 1$

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma[Hf(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n).$$

This is often done to ensure that the Wolfe conditions are satisfied at each step $\mathbf{x}_n \to \mathbf{x}_{n+1}$ of the iteration.

Where applicable, Newton's method converges much faster towards a local maximum or minimum than gradient descent. In fact, every local minimum has a neighborhood N such that, if we start with $\mathbf{x}_0 \in N$, Newton's method with step size $\gamma = 1$ converges quadratically (if the Hessian is invertible and a Lipschitz continuous function of $\mathbf{x}$ in that neighborhood).

Finding the inverse of the Hessian in high dimensions can be an expensive operation. In such cases, instead of directly inverting the Hessian it's better to calculate the vector $\mathbf{p}_n = [Hf(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n)$ as the solution to the system of linear equations

$$[Hf(\mathbf{x}_n)]\mathbf{p}_n = \nabla f(\mathbf{x}_n)$$

which may be solved by various factorizations or approximately (but to great accuracy) using iterative methods. Many of these methods are only applicable to certain types of equations, for example the Cholesky factorization and conjugate gradient will only work if $[Hf(\mathbf{x}_n)]$ is a positive definite matrix. While this may seem like a limitation, it's often useful indicator of something gone wrong, for example if a minimization problem is being approached and $[Hf(\mathbf{x}_n)]$ is not positive definite, then the iterations are converging to a saddle point and not a minimum.

On the other hand, if a constrained optimization is done (for example, with Lagrange multipliers), the problem may become one of saddle point finding, in which case the Hessian will be symmetric indefinite and the solution of $\mathbf{p}_n$ will need to be done with a method that will work for such, such as the **LDL**$^{\mathrm{T}}$ variant of Cholesky factorization or the conjugate residual method.

There also exist various quasi-Newton methods, where an approximation for the Hessian (or its inverse directly) is built up from changes in the gradient.

If the Hessian is close to a non-invertible matrix, the inverted Hessian can be numerically unstable and the solution may diverge. In this case, certain workarounds have been tried in the past, which have varied success with certain problems. One can, for example, modify the Hessian by adding a correction matrix B_n so as to make $H_f(\mathbf{x}_n) + B_n$ positive definite. One approach is to diagonalize H_f and choose B_n so that $H_f(\mathbf{x}_n) + B_n$ has the same eigenvectors as H_f, but with each negative eigenvalue replaced by $\epsilon > 0$.

An approach exploited in the Levenberg–Marquardt algorithm (which uses an approximate Hessian) is to add a scaled identity matrix to the Hessian, $\mu\mathbf{I}$, with the scale adjusted at every iteration as needed. For large μ and small Hessian, the iterations will behave like gradient descent with step size $\frac{1}{\mu}$. This results in slower but more reliable convergence where the Hessian doesn't provide useful information.

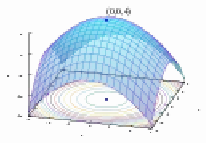
See also [edit]

- Quasi-Newton method
- Gradient descent
- Gauss–Newton algorithm
- Levenberg–Marquardt algorithm
- Trust region
- Optimization
- Nelder–Mead method

Notes [edit]

References [edit]

- Avriel, Mordecai (2003). *Nonlinear Programming: Analysis and Methods*. Dover Publishing. ISBN 0-486-43227-0.
- Bonnans, J. Frédéric; Gilbert, J. Charles; Lemaréchal, Claude; Sagastizábal, Claudia A. (2006). *Numerical optimization: Theoretical and practical aspects*. Universitext (Second revised ed. of translation of 1997 French ed.). Berlin: Springer-Verlag. pp. xiv+490. doi:10.1007/978-3-540-35447-5. ISBN 3-540-35445-X. MR 2265882.
- Fletcher, Roger (1987). *Practical methods of optimization* (2nd ed.). New York: John Wiley & Sons. ISBN 978-0-471-91547-8..
- Nocedal, Jorge & Wright, Stephen J. (1999). *Numerical Optimization*. Springer-Verlag. ISBN 0-387-98793-2.
- "Newton-Raphson visualization (1D)"..

Optimization: Algorithms, methods, and heuristics		
Unconstrained nonlinear: Methods calling ... [show]		
Constrained nonlinear [show]		
Convex optimization [show]		
Combinatorial [show]		
Metaheuristics [show]		
Categories (Algorithms and methods · Heuristics) · **Software**		

Categories: Optimization algorithms and methods