Newton-type Methods

Ayşen YILDIZ
Esra Nur ÇAKICI
H.Yavuz ERZURUMLU

What is Newton-type Method?

- One of the most famous numerical methods.
- An approach for solving nonlinear equations.
- An optimization method.

Application areas:

Relevance in machine learning, signal and image processing, computer networks, computer vision, and high dimensional statistics can be used.

What is Newton-type Method?

Google

newton type method

All Images Videos News Shopping More Settings Tools

About 3,040,000 results (0.27 seconds)

Scholarly articles for newton type method

A special **Newton-type** optimization **method** - Fischer - Cited by 767
A third-order **Newton-type method** to solve systems of ... - Darvishi - Cited by 138
Chaotic dynamics of a third-order **Newton-type method** - Amat - Cited by 96

[PDF] **Newton-type Methods - Stanford University**
https://web.stanford.edu/class/cme334/docs/newton-type-methods.pdf
by W Murray - 2010 - Cited by 13 - Related articles
Jul 5, 2010 - Keywords: nonlinear equations, optimization **methods**, modified **Newton**. 1 Introduction. As noted **Newton's method** is famous. **Type** it into the ...

Introduction to a Newton-type method for solving nonlinear equations ...
https://www.sciencedirect.com/science/article/pii/S0096300307005942
by R Thukral - 2008 - Cited by 22 - Related articles
Feb 1, 2008 - In this paper we shall introduce a new **Newton-type method**, namely rational Newton, for solving a nonlinear equation. This new method is ...

Newton's method - Wikipedia
https://en.wikipedia.org/wiki/Newton%27s_method
In numerical analysis, **Newton's method** (also known as the **Newton**–Raphson **method**), named after Isaac **Newton** and Joseph Raphson, is a **method** for finding successively better approximations to the roots (or zeroes) of a real-valued function. It is one example of a root-finding algorithm.
Newton's method in optimization · Secant method · Householder's method

Searches related to newton type method

newton method
newton's method **optimization**
newton **raphson** method **example**
how to find root in newton **raphson** method
newton type **optimization**
newton's method **multivariate**
newton's method **optimization algorithm**
alternatives to newton's method

Several Topics

- Estimating root of f(x) function: Newton Raphson Method
- One Dimensional Unconstrained Optimization
- Multi Dimensional Unconstrained Optimization

Newton Raphson Method - Algorithm

Algorithm:

- **Input:** initial x, $func(x), derivFunc(x)$
- **Output:** Root of Func()

1. Compute values of $func(x)$ and $derivFunc(x)$ for given initial x
2. Compute h: $h = \frac{func(x)}{derivFunc(x)}$
3. While h is greater than allowed error ε
   (a) $h = \frac{func(x)}{derivFunc(x)}$
   (b) $x = x - h$

An Example

Let us illustrate Newton's method with a concrete numerical example. The golden ratio ($\varphi \approx 1.618$) is the largest root of the polynomial $f(x) = x^2 - x - 1$; to calculate this root, we can use the Newton iteration

$$x_{k+1} = x_k - \frac{f(x)}{f'(x)} = x_k - \frac{x_k^2 - x_k - 1}{2x_k - 1} = \frac{x_k^2 + 1}{2x_k - 1}$$

An Example

$$x_0 = 1.0$$
$$x_1 = 2.0$$
$$x_2 = 1.6666666666666667$$
$$x_3 = 1.6190476190476191$$
$$x_4 = 1.6180344478216817$$
$$x_5 = 1.618033988749989$$
$$x_6 = 1.6180339887498947$$
$$x_7 = 1.6180339887498949$$
$$x_8 = 1.6180339887498949$$

Complexity:

$O((\log n)F(n))$

where $F(n)$ is the cost of calculating $f(x)/f'(x)$ with n-digit precision.

One Dimensional Unconstrained Optimization

Recall that the Newton-Raphson method is an open method that finds the root x of a function such that $f(x) = 0$

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

as a technique to find the minimum or maximum of $f(x)$. It should be noted that this equation can also be derived by writing a second-order Taylor series for $f(x)$ and setting the derivative of the series equal to zero. Newtons method is an open method similar to Newton-Raphson because it does not require initial guesses that bracket the optimum. In addition, it also shares the disadvantage that it may be divergent. Finally, it is usually a good idea to check that the second derivative has the correct sign to confirm that the technique is converging on the result you desire.

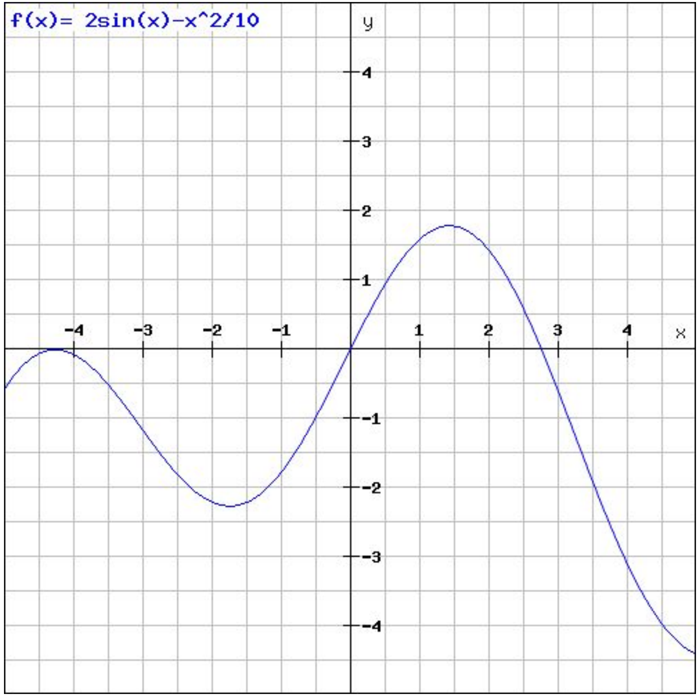
An Example

$$f(x) = 2\sin(x) - \frac{x^2}{10}$$

with an initial guess of $x_0 = 2.5$

$$f'(x) = 2\cos(x) - \frac{x}{5}$$

$$f''(x) = -2\sin(x) - \frac{1}{5}$$



An Example

Solution: The first and second derivatives of the function can be evaluated as

$$f'(x) = 2\cos(x) - \frac{x}{5}$$

$$f''(x) = -2\sin(x) - \frac{1}{5}$$

which can be subsituted into Eq. to give

$$x_{i+1} = x_i - \frac{2\cos(x_i) - x_i/5}{-2\sin(x_i) - 1/5}$$

Subsituting the initial guess yields

$$x_i = 2.5 - \frac{2\cos 2.5 - 2.5/5}{-2\sin 2.5 - 1/5} = 0.99508$$

which has a function value of 1.57859. The second iteration gives

$$x_i = 0.0995 - \frac{2\cos 0.0995 - 0.0995/5}{-2\sin 0.0995 - 1/5} = 1.46901$$

which has a function value of 1.77385.

i	x	$f(x)$	$f'(x)$	$f''(x)$
0	2.5	0.57194	-2.10229	-1.39694
1	0.99508	1.57859	0.88985	-1.87761
2	1.46901	1.77385	-0.09058	-2.18965
3	1.42764	1.77573	-0.00020	-2.17954
4	1.42755	1.77573	0.00000	-2.17952

f ’(x0)=0, f ’’(x0)>0 than x0 is minimum of function.
f ’(x0)=0, f ’’(x0)<0 than x0 is maximum of function.
f ’(x0)=0, f ’’(x0)<0 than x0 is try other methods.

i	x	$f(x)$	$f'(x)$	$f''(x)$
0	2.5	0.57194	-2.10229	-1.39694
1	0.99508	1.57859	0.88985	-1.87761
2	1.46901	1.77385	-0.09058	-2.18965
3	1.42764	1.77573	-0.00020	-2.17954
4	$\boxed{1.42755}$	1.77573	0.00000	-2.17952

Complexity :

Complexity is related with the initial value. It is $O(lg(n))$ which n is initial value.

Multidimensional Unconstrained Optimization

Extend the Newton's method for 1-D case to multidimensional case. Given f(X), approximate f(X) by a second order Taylor series at X = Xi : where (X is vector)

$$f(\vec{X}) \approx f(\vec{X}_i) + \nabla f'(\vec{X}_i)(\vec{X} - \vec{X}_i) + \frac{1}{2}(\vec{X} - \vec{X}_i)' H_i (\vec{X} - \vec{X}_i)$$

Multidimensional Unconstrained Optimization

$|H| > 0$ and $\partial^2 f/\partial x^2 > 0$, then $f(x, y)$ has a local minimum.
$|H| > 0$ and $\partial^2 f/\partial x^2 < 0$, then $f(x, y)$ has a local maximum.
$|H| < 0$, then $f(x, y)$ has a saddle point.

where H is the Hessian matrix:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ & \cdots & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Multidimensional Unconstrained Optimization

Stop-Criteria :

Difference between two vectors

$x^{k+1} - x^k <$ epsilon - very close to zero.

Algorithm:

- **Input:** $f : \mathbb{R}^n \to \mathbb{R}$ a twice-differentiable function $x^{(0)}$ an initial solution
- **Output:** x^*, a local minimum of the cost function f.

1. **begin**
2. $k \leftarrow 0;$
3. **while** STOP-CRIT **and** $(k < k_{max})$ **do**
4. $x^{k+1} \leftarrow x^k + \delta^k$;
5. with $\delta^k = -(H_f(x^k))^{-1} \bigtriangledown f(x^k)$;
6. $k \leftarrow k + 1$;
7. **return** x^k
8. **end**

Pitfalls and Improvings

We must calculate inverse of H for every iteration.

It requires both first and second partial derivatives, which may not be practical to obtain .

Using a step size of unity, the method may not converge.

We can use Quasi - Newton Method for to avoid calculating inverse of H.

Quasi - Newton Method

Quasi-Newton, or variable metric, methods seek to estimate the direct path to the optimum in a manner similar to Newton's method. However, notice that the Hessian matrix is composed of the second derivatives of f that vary from step to step. Quasi-Newton methods attempt to avoid these difficulties by approximating H with another matrix A using only first partial derivatives of f. The approach involves starting with an initial approximation of H−1 and updating and improving it with each iteration. The methods are called quasi-Newton because we do not use the true Hessian, rather an approximation. Further readings : DFP,BFGS

DFP -> Davidon-Fletcher-Powell
BFGS -> Broyden-Fletcher-Goldfarb-Shanno

Comparison Newton’s Method to Quasi- Newton

Newton’s Method	**Quasi-Newton Method**
Computationally expensive	Computationally cheap
Slow computation	Fast(er) computation
Need to iteratively calculate second derivative	No need for second derivative
Need to iteratively solve linear system of equations	No need to solve linear system of equations
Less convergence steps	More convergence steps
More precise convergence path	Less precise convergence path

Conclusions

We deal with newton type methods algorithms, its advantages and pitfalls.

We proposed to use this approach if we can calculate derivatives easily.

If we have second derivatives we can compute local minimum or maximum of given function. (1-D,2-D) The disadvantages of using this method are numerous. First of all, it is not guaranteed that Newton's method will converge if we select an initial root which that is too far from the exact root. Likewise, if our tangent line becomes parallel or almost parallel to the x-axis, we are not guaranteed convergence with the use of this method. Another disadvantage is that we must have a functional representation of the derivative of our function, which is not always possible if we working only from given data.

Questions?

Thanks :)