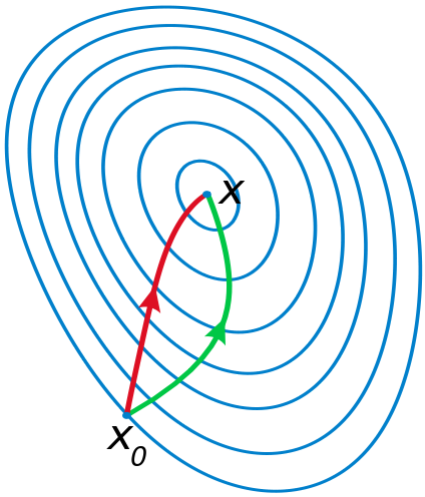
WIKIPEDIA

Newton's method in optimization

In calculus, **Newton's method** is an iterative method for finding the roots of a differentiable function F, which are 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. These solutions may be minima, maxima, or saddle points.[1]



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 (i.e. the second derivative) to take a more direct route.

Contents

Newton's Method

The central problem of optimization is minimization of functions. Let us first consider the case of univariate functions, i.e., functions of a single real variable. We will later consider the more general and more practically useful multivariate case.

Given a twice differentiable function $f : \mathbb{R} \to \mathbb{R}$, we seek to solve the optimization problem

$$\min_{x \in \mathbb{R}} f(x).$$

Newton's method attempts to solve this problem by constructing a sequence $\{x_k\}$ from an initial guess (starting point) $x_0 \in \mathbb{R}$ that converges towards a minimizer x_* of f by using a sequence of second-order Taylor approximations of f around the iterates. The second-order Taylor expansion of f around x_k is

$$f(x_k + t) \approx f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2.$$

The next iterate x_{k+1} is defined so as to minimize this quadratic approximation in t, and setting $x_{k+1} = x_k + t$. If the second derivative is positive, the quadratic approximation is a convex function of t, and its minimum can be found by setting the derivative to zero. Since

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}\left(f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2\right) = f'(x_k) + f''(x_k)t,$$

the minimum is achieved for

$$t = -\frac{f'(x_k)}{f''(x_k)}.$$

Putting everything together, Newton's method performs the iteration

$$x_{k+1} = x_k + t = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

Geometric interpretation

The geometric interpretation of Newton's method is that at each iteration, it amounts to the fitting of a paraboloid to the surface of $f(x)$ at the trial value x_k, having the same slopes and curvature as the surface at that point, and then proceeding to the maximum or minimum of that paraboloid (in higher dimensions, this may also be a saddle point).[2] Note that if f happens to *be* a quadratic function, then the exact extremum is found in one step.

Higher dimensions

The above iterative scheme can be generalized to d dimensions by replacing the derivative with the gradient (different authors use different notation for the gradient, including $f'(x) = \nabla f(x) = g_f(x) \in \mathbb{R}^d$), and the reciprocal of the second derivative with the inverse of the Hessian matrix (different authors use different notation for the Hessian, including $f''(x) = \nabla^2 f(x) = H_f(x) \in \mathbb{R}^{d\times d}$). One thus obtains the iterative scheme

$$x_{k+1} = x_k - [f''(x_k)]^{-1} f'(x_k), \qquad k \geq 0.$$

Often Newton's method is modified to include a small step size $0 < \gamma < 1$ instead of $\gamma = 1$:

$$x_{k+1} = x_k - \gamma[f''(x_k)]^{-1} f'(x_k).$$

This is often done to ensure that the Wolfe conditions are satisfied at each step of the method. For step sizes other than 1, the method is often referred to as the relaxed or damped Newton's method.

Convergence

If f is a strongly convex function with Lipschitz Hessian, then provided that x_0 is close enough to $x_* = \arg\min f(x)$, the sequence $x_0, x_1, x_2, \ldots$ generated by Newton's method will converge to the (necessarily unique) minimizer x_* of f quadratically fast. That is,

$$\|x_{k+1} - x_*\| \leq \frac{1}{2}\|x_k - x_*\|^2, \qquad \forall k \geq 0.$$

Computing the Newton direction

Finding the inverse of the Hessian in high dimensions to compute the Newton direction $h = -(f''(x_k))^{-1} f'(x_k)$ can be an expensive operation. In such cases, instead of directly inverting the Hessian, it's better to calculate the vector h as the solution to the system of linear equations

$$[f''(x_k)]h = -f'(x_k)$$

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 $f''(x_k)$ is a positive definite matrix. While this may seem like a limitation, it's often a useful indicator of something gone wrong; for example if a minimization problem is being approached and $f''(x_k)$ 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 x_{k+1} will need to be done with a method that will work for such, such as the $LDL^\top$ 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_k so as to make $f''(x_k) + B_k$ positive definite. One approach is to diagonalize the Hessian and choose B_k so that $f''(x_k) + B_k$ has the same eigenvectors as the Hessian, 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, μI, 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 $1/\mu$. This results in slower but more reliable convergence where the Hessian doesn't provide useful information.

Stochastic Newton's Method

Many practical optimization problems, and especially those arising in data science and machine learning, involve a function $f : \mathbb{R}^d \to \mathbb{R}$ which arises as an average of a very large number of simpler functions f_i:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x).$$

In supervised machine learning, $f_i(x)$ represents the loss of model parameterized by vector $x \in \mathbb{R}^d$ on data training point i, and $f(x)$ thus reflects the average loss of the model on the training data set. Problems of this type include linear least squares, logistic regression and deep neural network training.

In this situation, Newton's method for minimizing f takes the form

$$x_{k+1} = x_k - \left(\frac{1}{n}\sum_{i=1}^{n} f_i''(x_k)\right)^{-1} \left(\frac{1}{n}\sum_{i=1}^{n} f_i'(x_k)\right).$$

Recall that the key difficulty of standard Newton's method is the computation of Newton's step which is typically much more computationally demanding than the computation of the Hessian $f''(x_k)$ and the gradient $f'(x_k)$. However, in the setting considered here with f being the sum of a very large number of functions, the situation reverses and the *computation* of $f''(x_k)$ and $f'(x_k)$ by averaging the Hessians and gradients of the individual functions f_i becomes the bottleneck.

In this big n regime, the above issue can be resolved by considering the **stochastic Newton (SN) method** developed and analyzed by Kovalev, Mishchenko and Richtárik.[3] SN is a generalization of Newton's method which allows for a flexible choice of the set S_k of functions for which the computation of the Hessian and gradient is necessary. This set can be chosen to be $S_k = \{1, 2, \ldots, n\}$, in which case SN reduces to Newton's method. However, one can also choose $S_k = \{i\}$, where i is a random element of $\{1, 2, \ldots, n\}$.

The Method. In general, SN is a parametric family of methods with parameter $\tau \in \{1, 2, \ldots, n\}$ controlling the batch size. Given τ, in iteration k we let S_k be a random subset of $\{1, 2, \ldots, n\}$ chosen uniformly from all subsets of cardinality τ. That is, all subsets of cardinality τ are chosen with probability $1/\binom{d}{\tau}$. The two cases described above are special cases of this for $\tau = n$ and $\tau = 1$, respectively.

The Stochastic Newton method maintains a sequence of vectors $x_k^1, x_k^2, \cdots, x_k^n \in \mathbb{R}^d$ for $k \geq 0$. At the beginning, i.e., for $k = 0$, these vectors are initialized arbitrarily. A sensible choice is to set them to be equal. The method then performs the following steps:

$$\begin{aligned}
&\textit{Step } 1: \quad x_{k+1} = \left(\frac{1}{n}\sum_{i=1}^{n} f_i''(x_k^i)\right)^{-1} \left(\frac{1}{n}\sum_{i=1}^{n} f_i''(x_k^i)x_k^i - f_i'(x_k^i)\right)\\
&\textit{Step } 2: \quad \text{Sample } S_k \subseteq \{1, 2, \ldots, n\}\\
&\textit{Step } 3: \quad x_{k+1}^i = \begin{cases} x_{k+1} & i \in S_k \\ x_k^i & i \notin S_k \end{cases}.
\end{aligned}$$

Note that if $x_0^1 = x_0^2 = \cdots = x_0^n$ and $\tau = n$, SN reduces to Newton's method described above. However, in contrast with Newton's method, in iteration k, SN needs to compute the gradients and Hessians of functions f_i for $i \in S_k$ only. In particular, the batch size τ can be chosen to be a constant, in which case the cost of each iteration of SN is *independent* of n.

Convergence. For $\tau = n$, SN has local quadratic convergence rate identical to Newton's method. For $\tau < n$, SN has a local linear convergence rate independent of the condition number. In particular, it was shown by Kovalev, Mishchenko and Richtárik that if f is strongly convex and has Lipschitz Hessian, then as long as the initial iterates $x_0^1, x_0^2, \cdots, x_0^n$ are sufficiently close to the (necessarily) unique minimizer x_* of f, then

$$\mathbf{E}\left[\frac{1}{n}\sum_{i=1}^{n}\|x_k^i - x_*\|^2\right] \leq \left(1 - \frac{3\tau}{4n}\right)^k \left[\frac{1}{n}\sum_{i=1}^{n}\|x_0^i - x_*\|^2\right],$$

where $\mathbf{E}[\cdot]$ refers to mathematical expectation with respect to the randomness inherent in the algorithm.

This is a much better rate than what is obtainable by any stochastic first order method, such as stochastic gradient descent. Indeed, the convergence rate of all first order method depends on the condition number of f, which is typically defined as $\kappa = L/\mu$, where $0 < \mu \leq L$ are constants such that

$$\mu I \preceq f''(x) \preceq LI, \qquad \forall x \in \mathbb{R}^d.$$

There are various techniques which can to some extent *reduce* but which *can't completely eliminate* the effect of the conditioning κ on the convergence rate of first order methods. These techniques include adaptive stepsizes, minibatching, importance sampling, Polyak momentum, Nesterov's momentum and variance reduction. In contrast to all of these techniques, SN completely removes the effect of conditioning. However, as Newton's method, SN suffers from reliance on *local* convergence guarantees only.

See also

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

Notes

1. "Relative Extrema" (http://tutorial.math.lamar.edu/Classes/CalcIII/RelativeExtrema.aspx). *Lamar University*. Retrieved 28 August 2019.
2. Edwards, A. W. F. (1992). *Likelihood* (Expanded ed.). Baltimore: Johns Hopkins University Press. p. 129. ISBN 0-8018-4443-6.
3. Kovalev, Dmitry; Mishchenko, Konstantin; Richtárik, Peter (2019). "Stochastic Newton and cubic Newton methods with simple local linear-quadratic rates". arXiv:1912.01597 (https://arxiv.org/abs/