

Newton's Method

Hoàng Nam Dũng

Khoa Toán - Cơ - Tin học, Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội

Newton-Raphson method

<http://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec9.pdf>

<http://mathfaculty.fullerton.edu/mathews/n2003/Newton'sMethodProof.html>

<http://web.stanford.edu/class/cme304/docs/newton-type-methods.pdf>

Animation:

<http://mathfaculty.fullerton.edu/mathews/a2001/Animations/RootFinding/NewtonMethod/NewtonMethod.html>

Newton's method

Given unconstrained, smooth convex optimization

$$\min_x f(x),$$

where f is convex, twice differentiable, and $\text{dom}(f) = \mathbb{R}^n$. Recall that gradient descent chooses initial $x^{(0)} \in \mathbb{R}^n$, and repeats

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \dots$$

In comparison, **Newton's method** repeats

$$x^{(k)} = x^{(k-1)} - \left(\nabla^2 f(x^{(k-1)}) \right)^{-1} \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \dots$$

Here $\nabla^2 f(x^{(k-1)})$ is the Hessian matrix of f at $x^{(k-1)}$.

Newton's method interpretation

Recall the motivation for gradient descent step at x : we minimize the quadratic approximation

$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2t} \|y - x\|_2^2,$$

over y , and this yields the update $x^+ = x - t \nabla f(x)$.

Newton's method uses in a sense a **better quadratic approximation**

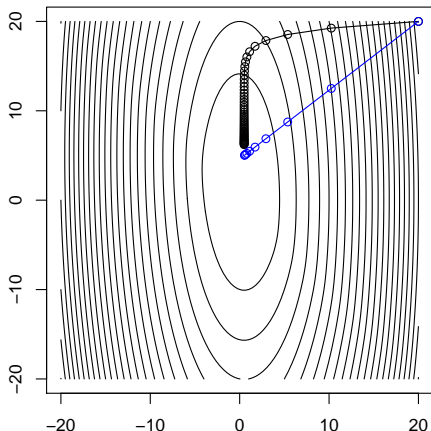
$$f(y) \approx f(x) + \nabla f(x)^T (y - x) + \frac{1}{2} (y - x)^T \nabla^2 f(x) (y - x),$$

and minimizes over y to yield $x^+ = x - (\nabla^2 f(x))^{-1} \nabla f(x)$.

Newton's method

Consider minimizing $f(x) = (10x_1^2 + x_2^2)/2 + 5 \log(1 + e^{-x_1 - x_2})$

We compare gradient descent (black) to Newton's method (blue), where both take steps of roughly same length

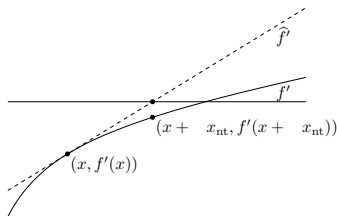


Linearized optimality condition

Alternative interpretation of Newton step at x : we seek a direction v so that $\nabla f(x + v) = 0$. Let $F(x) = \nabla f(x)$. Consider **linearizing** F around x , via approximation $F(y) \approx F(x) + DF(x)(y - x)$, i.e.,

$$0 = \nabla f(x + v) \approx \nabla f(x) + \nabla^2 f(x)v.$$

Solving for v yields $v = -(\nabla^2 f(x))^{-1} \nabla f(x)$.



From B & V page 486

History: work of Newton (1685) and Raphson (1690) originally focused on finding roots of polynomials. Simpson (1740) applied this idea to general nonlinear equations, and minimization by setting the gradient to zero.

Affine invariance of Newton's method

Important property Newton's method: **affine invariance**. Given f , nonsingular $A \in \mathbb{R}^{n \times n}$. Let $x = Ay$, and $g(y) = f(Ay)$. Newton steps on g are

$$\begin{aligned}y^+ &= y - (\nabla^2 g(y))^{-1} \nabla g(y) \\&= y - (A^T \nabla^2 f(Ay) A)^{-1} A^T \nabla f(Ay) \\&= y - A^{-1} (\nabla^2 f(Ay))^{-1} \nabla f(Ay)\end{aligned}$$

Hence

$$Ay^+ = Ay - (\nabla^2 f(Ay))^{-1} \nabla f(Ay),$$

i.e.,

$$x^+ = x - (\nabla^2 f(x))^{-1} \nabla f(x).$$

So progress is independent of problem scaling; recall that this is **not true** of gradient descent.

Newton decrement

At a point x , we define the Newton decrement as

$$\lambda(x) = \left(\nabla f(x)^T (\nabla^2 f(x))^{-1} \nabla f(x) \right)^{1/2}.$$

This relates to the difference between $f(x)$ and the minimum of its quadratic approximation:

$$\begin{aligned} f(x) - \min_y \left(f(x) + \nabla f(x)^T (y - x) + \frac{1}{2} (y - x)^T \nabla^2 f(x) (y - x) \right) \\ = f(x) - \left(f(x) - \frac{1}{2} \nabla f(x)^T (\nabla^2 f(x))^{-1} \nabla f(x) \right) \\ = \frac{1}{2} \lambda(x)^2. \end{aligned}$$

Therefore can think of $\lambda^2(x)/2$ as an approximate upper bound on the suboptimality gap $f(x) - f^*$.

Another interpretation of Newton decrement: if Newton direction is $v = -(\nabla^2 f(x))^{-1} \nabla f(x)$, then

$$\lambda(x) = (v^T \nabla^2 f(x) v)^{1/2} = \|v\|_{\nabla^2 f(x)},$$

i.e., $\lambda(x)$ is the **length of the Newton step** in the norm defined by the Hessian $\nabla^2 f(x)$.

Note that the Newton decrement, like the Newton steps, are affine invariant; i.e., if we defined $g(y) = f(Ay)$ for nonsingular A , then $\lambda_g(y)$ would match $\lambda_f(x)$ at $x = Ay$.

Backtracking line search

So far what we've seen is called **pure Newton's method**. This need not converge. In practice, we use **damped Newton's method** (i.e., Newton's method), which repeats

$$x^+ = x - t(\nabla^2 f(x))^{-1} \nabla f(x).$$

Note that the pure method uses $t = 1$.

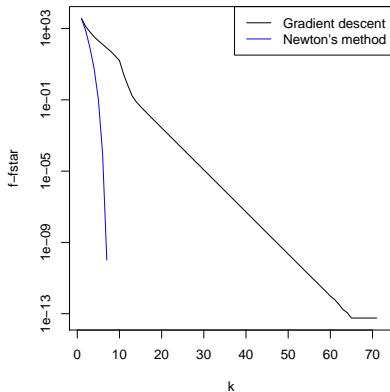
Step sizes here typically are chosen by backtracking search, with parameters $0 \leq \alpha \leq 1/2, 0 < \beta < 1$. At each iteration, we start with $t = 1$ and while

$$f(x + tv) > f(x) + \alpha t \nabla f(x)^T v,$$

we shrink $t = \beta t$, else we perform the Newton update. Note that here $v = -(\nabla^2 f(x))^{-1} \nabla f(x)$, so $\nabla f(x)^T v = -\lambda^2(x)$.

Example: logistic regression

Logistic regression example, with $n = 500$, $p = 100$: we compare gradient descent and Newton's method, both with backtracking



Newton's method: in a totally different regime of convergence...!

Convergence analysis

Assume that f convex, twice differentiable, having $\text{dom}(f) = \mathbb{R}^n$, and additionally

- ▶ ∇f is Lipschitz with parameter L .
- ▶ f is strongly convex with parameter m .
- ▶ $\nabla^2 f$ is Lipschitz with parameter M .

Theorem

Newton's method with backtracking line search satisfies the following two-stage convergence bounds

$$f(x^{(k)}) - f^* \leq \begin{cases} (f(x^{(0)}) - f^*) - \gamma k & \text{if } k \leq k_0 \\ \frac{2m^3}{M^2} \left(\frac{1}{2}\right)^{2^{k-k_0}+1} & \text{if } k > k_0. \end{cases}$$

Here $\gamma = \alpha\beta^2\eta^2m/L^2$, $\eta = \min\{1, 3(1 - 2\alpha)\}m^2/M$, and k_0 is the number of steps until $\|\nabla f(x^{(k_0+1)})\|_2 < \eta$.

Convergence analysis

In more detail, convergence analysis reveals $\gamma > 0, 0 < \eta \leq m^2/M$ such that convergence follows two stages

- Damped phase: $\|\nabla f(x^{(k)})\|_2 \geq \eta$, and

$$f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma.$$

- Pure phase: $\|\nabla f(x^{(k)})\| < \eta$, backtracking selects $t = 1$, and

$$\frac{M}{2m^2} \|\nabla f(x^{(k+1)})\|_2 \leq \left(\frac{M}{2m^2} \|\nabla f(x^{(k)})\|_2 \right)^2.$$

Note that once we enter pure phase, we won't leave, because

$$\frac{2m^2}{M} \left(\frac{M}{2m^2} \eta \right)^2 \leq \eta$$

when $\eta \leq m^2/M$.

Convergence analysis

Unraveling this result, what does it say? To get $f(x^{(k)}) - f^* \leq \varepsilon$, we need at most

$$\frac{f(x^{(0)}) - f^*}{\gamma} + \log \log(\varepsilon_0/\varepsilon,)$$

iterations, where $\varepsilon_0 = 2m^3/M^2$.

- ▶ This is called **quadratic convergence**. Compare this to linear convergence (which, recall, is what gradient descent achieves under strong convexity).
- ▶ The above result is a **local convergence rate**, i.e., we are only guaranteed quadratic convergence after some number of steps k_0 , where $k_0 \leq \frac{f(x^{(0)}) - f^*}{\gamma}$.
- ▶ Somewhat bothersome may be the fact that the above bound depends on L, m, M , and yet the **algorithm itself does not** ...

Self-concordance

A scale-free analysis is possible for **self-concordant functions**: on \mathbb{R} , a convex function f is called self-concordant if

$$|f'''(x)| \leq 2f''(x)^{3/2} \quad \text{for all } x,$$

and on \mathbb{R}^n is called self-concordant if its projection onto every line segment is so.

Theorem (Nesterov and Nemirovskii)

Newton's method with backtracking line search requires at most

$$C(\alpha, \beta)(f(x^{(0)}) - f^*) + \log \log(1/\varepsilon),$$

iterations to reach $f(x^{(k)}) - f^$, where $C(\alpha, \beta)$ is a constant that only depends on α, β .*

What kind of functions are self-concordant?

- ▶ $f(x) = -\sum_{i=1}^n \log(x_i)$ on \mathbb{R}_{++}^n .
- ▶ $f(X) = -\log(\det(X))$ on \mathbb{S}_{++}^n .
- ▶ If g is self-concordant, then so is $f(x) = g(Ax + b)$.
- ▶ In the definition of self-concordance, we can replace factor of 2 by a general $\kappa > 0$.
- ▶ If g is κ -self-concordant, then we can rescale: $f(x) = \frac{\kappa}{4}g(x)$ is self-concordant (2-self-concordant).

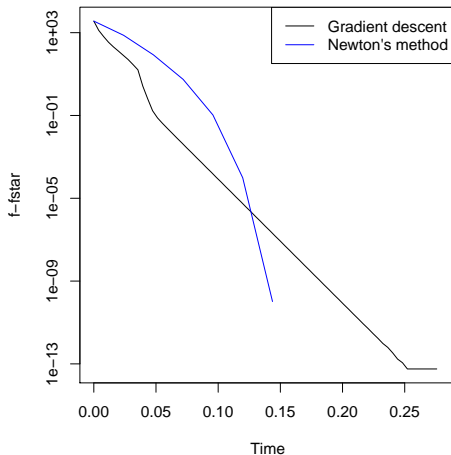
Comparison to first-order methods

At a high-level:

- ▶ **Memory:** each iteration of Newton's method requires $O(n^2)$ storage ($n \times n$ Hessian); each gradient iteration requires $O(n)$ storage (n -dimensional gradient).
- ▶ **Computation:** each Newton iteration requires $O(n^3)$ flops (solving a dense $n \times n$ linear system); each gradient iteration requires $O(n)$ flops (scaling/adding n -dimensional vectors).
- ▶ **Backtracking:** backtracking line search has roughly the same cost, both use $O(n)$ flops per inner backtracking step.
- ▶ **Conditioning:** Newton's method is not affected by a problem's conditioning, but gradient descent can seriously degrade.
- ▶ **Fragility:** Newton's method may be empirically more sensitive to bugs/numerical errors, gradient descent is more robust.

Newton method vs. gradient descent

Back to logistic regression example: now x -axis is parametrized in terms of time taken per iteration



Sparse, structured problems

When the inner linear systems (in Hessian) can be solved **efficiently and reliably**, Newton's method can thrive.

E.g., if $\nabla^2 f(x)$ is sparse and structured for all x , say banded, then both memory and computation are $O(n)$ with Newton iterations.

What functions admit a structured Hessian? Two examples:

- ▶ If $g(\beta) = f(X\beta)$, then $\nabla^2 g(\beta) = X^T \nabla^2 f(X\beta) X$. Hence if X is a structured predictor matrix and $\nabla^2 f$ is diagonal, then $\nabla^2 g$ is structured.
- ▶ If we seek to minimize $f(\beta) + g(D\beta)$, where $\nabla^2 f$ is diagonal, g is not smooth, and D is a structured penalty matrix, then the Lagrange dual function is $-f^*(-D^T u) - g^*(-u)$. Often $-D \nabla^2 f^*(-D^T u) D^T$ can be structured.

Quasi-Newton methods

If the Hessian is too expensive (or singular), then a quasi-Newton method can be used to approximate $\nabla^2 f(x)$ with $H \succ 0$, and we update according to

$$x^+ = x - tH^{-1}\nabla f(x).$$

- ▶ Approximate Hessian H is recomputed at each step. Goal is to make H^{-1} cheap to apply (possibly, cheap storage too).
- ▶ Convergence is fast: **superlinear**, but not the same as Newton. Roughly n steps of quasi-Newton make same progress as one Newton step.
- ▶ Very wide variety of quasi-Newton methods; common theme is to “propagate” computation of H across iterations.

Quasi-Newton methods






Davidon-Fletcher-Powell or DFP:

- ▶ Update H, H^{-1} via rank 2 updates from previous iterations; cost is $O(n^2)$ for these updates.
- ▶ Since it is being stored, applying H^{-1} is simply $O(n^2)$ flops.
- ▶ Can be motivated by Taylor series expansion.

Broyden-Fletcher-Goldfarb-Shanno or BFGS:

- ▶ Came after DFP, but BFGS is now much more widely used.
- ▶ Again, updates H, H^{-1} via rank 2 updates, but does so in a “dual” fashion to DFP; cost is still $O(n^2)$.
- ▶ Also has a limited-memory version, L-BFGS: instead of letting updates propagate over all iterations, only keeps updates from last m iterations; storage is now $O(mn)$ instead of $O(n^2)$.

References and further reading

-  S. Boyd and L. Vandenberghe (2004), *Convex optimization*, Chapters 9 and 10
-  Y. Nesterov (1998), *Introductory lectures on convex optimization: a basic course*, Chapter 2
-  Y. Nesterov and A. Nemirovskii (1994), *Interior-point polynomial methods in convex programming*, Chapter 2
-  J. Nocedal and S. Wright (2006), *Numerical optimization*, Chapters 6 and 7
-  L. Vandenberghe, *Lecture notes for EE 236C*, UCLA, Spring 2011-2012