10-425/625: Introduction to Convex Optimization **(Fall 2023)**

Lecture 17: Newton's Method Analysis

Instructor:[1] *Matt Gormley* *October 30, 2023*

17.1 Newton's Method

Recall from a previous lecture

17.1.1 The Algorithm

Newton's method Given unconstrained, smooth convex optimization

$$\min_x \; f(x)$$

where f is convex, twice differentable, and $\mathrm{dom}(f) = \mathbb{R}^n$.

Newton's method repeats

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

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

17.1.2 Newton's method interpretation

Newton decrement At a point x, we define the Newton decrement as

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

[1]These notes were originally written by Ryan Tibshirani for 10-725 Fall 2019 (original version: here) and were edited and adapted for 10-425/625.

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

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

Therefore can think of $\lambda^2(x)/2$ as an approximate upper bound on the sub-optimality gap $f(x) - f^\star$

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

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

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 $x = Ay$

Recall from a previous lecture

17.1.3 Damped Newton's method

Backtracking line search So far we've seen pure Newton's method. This need not converge. In practice, we use damped Newton's method (typically just called Newton's method), which repeats

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

Note that the pure method uses $t = 1$

Step sizes here are chosen by backtracking search, with parameters $0 < \alpha \leq 1/2$, $0 < \beta < 1$. At each iteration, start with $t = 1$, 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 $\nabla f(x)^T v = -\lambda^2(x)$

17.1.4 Analysis

Convergence analysis Recall that gradient descent converges at a rate of c^k for some constant c. We're going to see that Newton's method converges at a rate of $(1/2)^{2^k}$, a totally different regime of convergence! Note also, that we need backtracking line search for this work; Newton's method won't converge without it.

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

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

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

In short, there are two phases of the Newton's method progression: in the first phase ($k \le k_0$), it converges slowly. But then it reaches some point ($k > k_0$) after which it converges very fast—and, in this second phase, the backtracking line search will only take one step every time.

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

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

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

- Pure phase: $\|\nabla f(x^{(k)})\|_2 < \eta$, backtracking selects $t = 1$, and

  $$\frac{M}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \le \Big(\frac{M}{2m^2}\|\nabla f(x^{(k)})\|_2\Big)^2$$

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

  $$\frac{2m^2}{M}\Big(\frac{M}{2m^2}\eta\Big)^2 \le \eta$$

  when $\eta \le m^2/M$

Here we prove only the result for the pure phase, which is a bit simpler and more intuitive.

Proof: Assume we're in the pure phase, and backtracking line search gives us $t = 1$.

Fact 1: Since f is m-strongly convex, we know that:

$$f(x^{(k)}) - f(x^*) \le \frac{1}{2m}\|\nabla f(x^{(k)}\|_2^2$$

Proof of Fact 1:

$$f(y) \ge f(x) + \nabla f(x)^T(y-x) + \frac{m}{2}\|y-x\|_2^2$$

Now minimizing over both sides gives: $f(x^*) \ge \min_y f(x) + \nabla f(x)^T(y-x) + \frac{m}{2}\|y-x\|_2^2$

$$\text{take gradient:} 0 = \nabla f(x) + m(y-x) \qquad \Rightarrow y = -$$

$$\Rightarrow f(x^*) \ge f(x) - \frac{1}{2m}\|\nabla f(x)\|_2^2$$

$$\Rightarrow f(x) - f(x^*) \le \frac{1}{2m}\|\nabla f(x)\|_2^2$$

Fact 2: Once we are in the pure phase, letting $x^+ = x - -(\nabla^2 f(x))^{-1}\nabla f(x)$:

$$\frac{M}{2m^2}\|\nabla f(x^+\|_2^2 \le \left(\frac{M}{2m^2}\|\nabla f(x)\|_2^2\right)^2$$

Proof of Fact 2:

$$
\begin{aligned}
\|\nabla f(x^+)\|_2^2 &= \|\nabla f(x+v)\|_2^2 \text{ where } v = -(\nabla^2 f(x))^{-1}\nabla f(x) \\
&= \|\nabla f(x+v) - \nabla f(x) - \nabla^2 f(x)v\|_2^2 \text{ since } \nabla^2 f(x)v = \nabla f(x) \\
&= \|\int_0^1 \nabla^2 f(x+tv)dt - \nabla^2 f(x)v\|_2^2 \\
&\quad \text{by the fundamental theorem of calculus} \\
&= \int_0^1 \|\nabla^2 f(x+tv) - \nabla^2 f(x)v\|_2^2 dt \\
&\quad \text{by triangle inequality}
\end{aligned}
$$

The definition of the operator norm gives us that:

$$
\begin{aligned}
\|\nabla^2 f(x+tv) - \nabla^2 f(x)v\|_2^2 &\le \|\nabla^2 f(x+tv) - \nabla^2 f(x)\|_{op}\|v\|_2 \\
&\le Mt\|v\|_2\|v\|_2^2 = Mt\|v\|_2^2 \\
&\quad \text{By invoking the Lipschitz-ness of the Hessian}
\end{aligned}
$$

Returning to the broader inequality, we have:

$$
\begin{aligned}
\|\nabla f(x^+)\|_2^2 &\le M\|v\|_2^2 \int_0^1 tdt \\
&\le M\| - (\nabla^2 f(x))^{-1}\nabla f(x)\|_2^2 \\
&\le M\| - \|\nabla^2 f(x))^{-1}\|_{op}^2\|\nabla f(x)\|_2^2 \\
&\le -\frac{M}{2m^2}\|\nabla f(x)\|_2^2
\end{aligned}
$$

Where the last step is by strong convexity and since the inverse of the matrix and a matrix have reciprocal eigenvalues.

Multiplying both sides by $\frac{M}{2m^2}$ gives:

$$
\frac{M}{2m^2}\|\nabla f(x^+\|_2^2 \le \left(\frac{M}{2m^2}\|\nabla f(x)\|_2^2\right)^2
$$

Fact 3: Also in the pure phase:

$$
f(x^{(k)}) - f(x^*) \le \frac{2M^3}{m^2}\left(\frac{1}{2}\right)^{2^{k-k_0}}
$$

Proof of Fact 3: We've established that

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

Letting the LHS be a_k and the RHS be a_{k-1}, we have:

$$\begin{aligned} a_k &\leq a_{k-2}^4 \\ &\leq \dots \\ &\leq a_{k_0}^{2^{k-k_0}} \end{aligned}$$

Plugging back in yields:

$$\frac{M}{2m^2}\|\nabla f(x^{(k+1)}\|_2^2 \leq \left(\frac{M}{2m^2}\|\nabla f(x^{(k_0)})\|_2^2\right)^{2^{k-k_0}}$$

But at k_0 we know that $\|\nabla f(x^{k_0}\|_2^2 \leq \eta \leq \frac{m^2}{M}$. So:

$$\begin{aligned} \frac{M}{2m^2}\|\nabla f(x^{(k+1)}\|_2^2 &\leq \left(\frac{1}{2}\right)^{2^{k-k_0}} \\ f(x^k) - f(x^*) &\leq \frac{1}{2m}\|\nabla f(x^k)\|_2^2 \\ &\leq \frac{1}{2m}(\frac{2m^2}{M})^2(\frac{1}{2})^{2^{k-k_0+1}} \\ &\leq \frac{2m^3}{M^2}\left(\frac{1}{2}\right)^{2^{k-k_0+1}} \end{aligned}$$

∎

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

$$\frac{f(x^{(0)}) - f^\star}{\gamma} + \log\log(\epsilon_0/\epsilon)$$

iterations, where $\epsilon_0 = 2m^3/M^2$

- 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 $k_0 \leq \frac{f(x^{(0)})-f^\star}{\gamma}$
- 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

$$C(\alpha,\beta)\big(f(x^{(0)}) - f^\star\big) + \log\log(1/\epsilon)$$

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

What kind of functions are self-concordant?

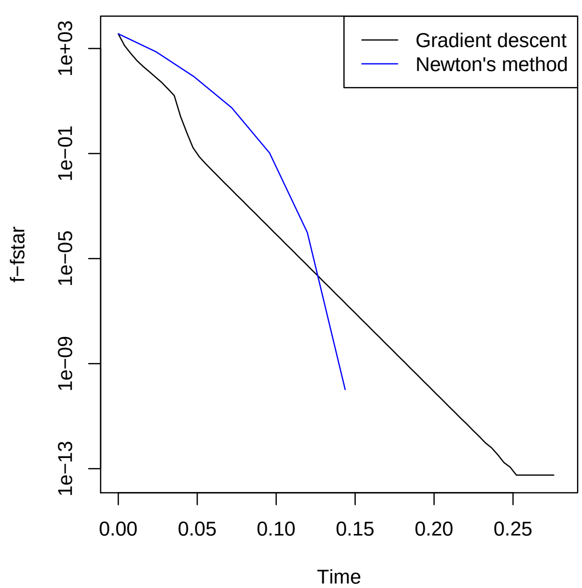
- Linear and quadratic functions
- $f(x) = -\sum_{i=1}^n \log(x_i)$ on $\mathbb{R}^n_{++}$
- $f(X) = -\log(\det(X))$ on $\mathbb{S}^n_{++}$
- If g is self-concordant, then so is $f(x) = g(Ax+b)$
- In the definition of self-concordance, we can replace factor of 2 by a general $\kappa > 0$
- If g is κ-self-concordant,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

17.1.5 Practicalities

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

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



Each gradient descent step is $O(p)$, but each Newton step is $O(p^3)$

Sparse, structured problems When the inner linear systems (in Hessian) can be solved efficiently and reliably, Newton's method can strive

For example, if $\nabla^2 f(x)$ is sparse/structured for all x, say banded, then both memory and computation are $O(n)$ per Newton iteration

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 $\nabla^2 f^*$ will be diagonal (e.g., when $f(\beta) = \sum_{i=1}^p f_i(\beta_i)$) so the Hessian in dual will be structured

17.1.6 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 "propogate" computation of H across iterations