Newton Method

Lecturer: Aarti Singh
Co-instructor: Pradeep Ravikumar

Convex Optimization 10-725/36-725

Materials courtesy: B. Poczos, R. Tibshirani, C. Carmanis & S. Sanghavi
Outline

Newton method

- Finding a root
- Unconstrained minimization
  - Motivation with quadratic approximation
  - Rate of Newton’s method
Newton method for finding a root
Newton method for finding a root

- Newton method: originally developed for finding a root of a function

- also known as the **Newton–Raphson method**

\[
\phi : \mathbb{R} \rightarrow \mathbb{R} \\
\phi(x^*) = 0 \\
x^* = ?
\]
Newton Method for Finding a Root

Goal: \( \phi : \mathbb{R} \rightarrow \mathbb{R} \)
\[ \phi(x^*) = 0 \]
\[ x^* = ? \]

Linear Approximation \((1^{st} \text{ order Taylor approx})\):
\[
\phi(x + \Delta x) = \phi(x) + \phi'(x) \Delta x + o(|\Delta x|)
\]

Therefore,
\[
0 \approx \phi(x) + \phi'(x) \Delta x
\]
\[
x^* - x = \Delta x = -\frac{\phi(x)}{\phi'(x)}
\]
\[
x_{k+1} = x_k - \frac{\phi(x)}{\phi'(x)}
\]
Goal: finding a root

\[ \hat{f}(x) = f(x_0) + f'(x_0)(x - x_0) \]

\[ x = x_0 + \Delta x_{NT} \]

In the next step we will linearize here in \( x \)
Example: Finding a Root

http://en.wikipedia.org/wiki/Newton%27s_method
Newton Method for Finding a Root

This can be generalized to multivariate functions

\[ F : \mathbb{R}^n \to \mathbb{R}^m \]

\[ 0_m = F(x^*) = F(x + \Delta x) = F(x) + \nabla F(x)\Delta x + o(|\Delta x|) \]

Therefore,

\[ 0_m = F(x) + \nabla F(x)\Delta x \]

\[ \Delta x = -[\nabla F(x)]^{-1}F(x) \]

\[ \Delta x = x_{k+1} - x_k, \text{ and thus} \]

\[ x_{k+1} = x_k - [\nabla F(x_k)]^{-1}F(x_k) \]

Newton method: Start from \( x_0 \) and iterate.
Newton method for minimization
Newton’s method for the optimization problem

\[
\min_x f(x)
\]

is the same as Newton’s method for finding a root of

\[
\nabla f(x) = 0.
\]

**History:** The 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.
Newton method for minimization

\[ f : \mathbb{R}^n \rightarrow \mathbb{R} \], \( f \) is twice differentiable

\[ \min_{x \in \mathbb{R}^n} f(x) \]

We need to find the roots of \( \nabla f(x) = 0_n \)

\[ \nabla f : \mathbb{R}^n \rightarrow \mathbb{R}^n \]

Newton system: \( \nabla f(x) + \nabla^2 f(x) \Delta x = 0_n \)

Newton step: \( \Delta x = x_{k+1} - x_k = -[\nabla^2 f(x)]^{-1} \nabla f(x) \)

Iterate until convergence, or max number of iterations exceeded

(divergence, loops, division by zero might happen…)

unconstrained
Motivation with Quadratic Approximation
Motivation with Quadratic Approximation

\[ f : \mathbb{R}^n \rightarrow \mathbb{R}, \ f \ \text{is twice differentiable} \]

\[ \min_{x \in \mathbb{R}^n} f(x) \]

unconstrained

Second order Taylor approximation:

Let \( \phi(x) = f(x_k) + \nabla^T f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) \)

Assume that

\[ \nabla^2 f(x_k) \succeq 0 \ [\text{i.e. } \phi \ \text{has strict global minimum}] \]

Now, if \( x_{k+1} \) is the global minimum of the quadratic function \( \phi \), then

\[ 0_n = \nabla \phi(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) \]

Newton step:

\[ \Delta x = x_{k+1} - x_k = -[\nabla^2 f(x)]^{-1} \nabla f(x) \]
Motivation with Quadratic Approximation

Quadratic approximation is good, when $x$ is close to $x^*$

$$\hat{f}(z) = f(x) + \nabla^T f(x)(z-x) + \frac{1}{2}(z-x)^T \nabla^2 f(x)(z-x)$$
Comparison with Gradient Descent
Comparison with Gradient Descent

Newton’s method: choose initial $x^{(0)} \in \mathbb{R}^n$, and

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

Compare to gradient descent: choose initial $x^{(0)} \in \mathbb{R}^n$, and

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

Newton method is obtained by minimizing over quadratic approximation:

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

Gradient descent uses a different quadratic approximation:

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

For $f(x) = (10x_1^2 + x_2^2)/2 + 5 \log(1 + e^{-x_1-x_2})$, compare gradient descent (black) to Newton’s method (blue), where both take steps of roughly same length
How good is the Newton method?
**Lemma [Descent direction]**

If $\nabla^2 f > 0$, then Newton step is a descent direction.

**Proof:**

We know that if a vector has negative inner product with the gradient vector, then that direction is a descent direction.

Newton step: $\Delta x = x_{k+1} - x_k = -[\nabla^2 f(x)]^{-1} \nabla f(x)$

$\Rightarrow \nabla f(x)^T \Delta x = - \nabla f(x)^T [\nabla^2 f(x)]^{-1} \nabla f(x) < 0$
Recall convergence rate for gradient descent:

$$f(x^{(k)}) - f^* \leq c^k \frac{L}{2} \|x^{(0)} - x^*\|^2_2$$

Constant $c$ depends adversely on condition number $L/m$ (higher condition number $\Rightarrow$ slower rate)

Can we convert it into well-conditioned problem by changing coordinates?

let $x = Ay$ and $g(y) = f(Ay)$.

$$\nabla g(y) = A^T \nabla f(Ay), \quad \nabla^2 g(y) = A^T \nabla^2 f(Ay) A$$

Can get $\nabla^2 g(y) = I$ if $A = [\nabla^2 f(x)]^{-1}$
Pre-Conditioning for Gradient descent

Can we convert it into well-conditioned problem by changing coordinates?

let $x = Ay$ and $g(y) = f(Ay)$.

Can get $\nabla^2 g(y) = I$ if $A = [\nabla^2 f(x)]^{-1/2}$

Running gradient descent for $g(y)$, gives best descent direction and convergence rate.

$$y_+ = y - \eta \nabla g(y)$$
$$= y - \eta A^T \nabla f(Ay),$$

$$Ay_+ = Ay - \eta AA^T \nabla f(Ay)$$

$$x_+ = x - \eta AA^T \nabla f(x).$$

Equivalent to Newton step on $f(x)$. 
Affine Invariance

Important property Newton’s method: affine invariance.

Assume $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable and $A \in \mathbb{R}^{n \times n}$ is nonsingular. Let $g(y) := f(Ay)$.

Newton step for $g$ starting from $y$ is

$$y^+ = y - (\nabla^2 g(y))^{-1}\nabla g(y).$$

It turns out that the Newton step for $f$ starting from $x = Ay$ is

$$x^+ = Ay^+.$$

Therefore progress is independent of problem scaling. By contrast, this is not true of gradient descent.

[Proof: HW3]
Affine Invariant stopping criterion

Stopping criterion for gradient descent:

$$\|\nabla f(x)\|_2 \leq \epsilon,$$

Not affine-invariant

Stopping criterion for Newton method:

$$\frac{\lambda^2(x)}{2} \leq \epsilon,$$

where

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

is the Newton decrement.

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 $x = Ay$. 
Affine Invariant stopping criterion

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

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

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

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

\[
\lambda(x) = (u^T \nabla^2 f(x) u)^{1/2} = \|u\| \|\nabla^2 f(x)\|
\]

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

- Quadratic convergence in the neighborhood of a strict local minimum [under some conditions].

- It can break down if $f''(x_k)$ is degenerate. [no inverse]

- It can diverge.

- It can be trapped in a loop.

- It can converge to a loop...
Damped Newton’s Method

We have seen pure Newton’s method, which need not converge. In practice, we instead 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 $t = 1$
Backtracking line search

\[ x^+ = x - t(\nabla^2 f(x))^{-1}\nabla f(x) \]

Step sizes here typically are chosen by backtracking search, with parameters \(0 < \alpha \leq 1/2\), \(0 < \beta < 1\). 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 \(\nabla f(x)^T v = -\lambda^2(x)\)
Convergence Rate
Theorem: Assume $F: \mathbb{R}^n \rightarrow \mathbb{R}^n$ is continuously differentiable and $x^* \in \mathbb{R}^n$ is a root of $F$, that is, $F(x^*) = 0$ such that $F'(x^*)$ is non-singular. Then

(a) There exists $\delta > 0$ such that if $\|x^{(0)} - x^*\| < \delta$ then Newton’s method is well defined and

$$\lim_{k \to \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0.$$ 

(b) If $F'$ is Lipschitz continuous in a neighborhood of $x^*$ then there exists $K > 0$ such that

$$\|x^{(k+1)} - x^*\| \leq K\|x^{(k)} - x^*\|^2.$$
Convergence analysis

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

- \( \nabla f \) 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

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

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 \( \|\nabla f(x^{(k_0+1)})\|_2 < \eta \).
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: $\|\nabla f(x^{(k)})\|_2 < \eta$, 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

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

when $\eta \leq m^2/M$
Convergence analysis

To reach $f(x^{(k)}) - f^* \leq \epsilon$, we need at most

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

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

- Somewhat bothersome may be the fact that the above bound depends on $L, m, M$, and yet the algorithm itself does not

Analysis can be improved e.g. for self-concordant functions
Comparison to first-order methods
Comparison to first-order methods

- **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
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 has a different regime of convergence.
Example: Logistic regression

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 $O(p^3)$