10-725/36-725: Convex Optimization **Fall 2019**

Lecture 14: Newton's Method *(October 14)*

Lecturer: Ryan Tibshirani *Scribes: Dennis Li, Divyansh Pareek, Shahriar Noroozizadeh*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

14.1 Last Time: dual correspondences

Given a function, $f : \mathbb{R}^n \to \mathbb{R}$, we define its **conjugate** $f^* : \mathbb{R}^n \to \mathbb{R}$,

$$f^*(y) = \max_x y^T x - f(x)$$

Key properties and examples of conjugates include:

- Conjugate f^* is always convex (regardless of convexity of f), because it is point wise maximum of function of y.
- When f is quadratic in $Q \succ 0$, f^* is a quadratic in Q^{-1}. In fact, when $f(x) = \frac{1}{2}x^T Q x$, where $Q \succ 0$, then $f^*(y) = \frac{1}{2}x^T Q^{-1} x$.
- When f is a norm, f^* is indicator of the dual norm unit ball.
- When f is closed and convex, $x \in \partial f^*(y) \iff y \in \partial f(x)$.

Conjugate's relationship with duality, also called Fenchel duality:

$$\text{Primal: } \min_x f(x) + g(x)$$
$$\text{Dual: } \max_u -f^*(u) - g^*(-u)$$

14.2 Newton's Method

Now, we're going to introduce a second-order method called Newton's method, and draw a comparison between the first order method gradient descent and 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

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

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

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

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

14.2.1 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 $x^+ = x - t\nabla f(x)$. $\nabla f(x)^T (y - x)$ is the first order term, and $\frac{1}{2t}\|y - x\|_2^2$ is the second order term that's the quadratic approximation of $f(y)$ around the point x.

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

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

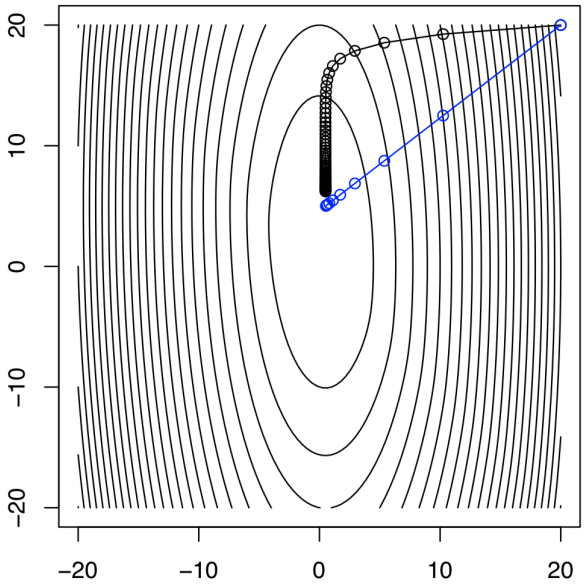
and minimizes over y to yield the update rule $x^+ = x - (\nabla^2 f(x))^{-1} \nabla f(x)$. Note that Newton's method does not use step size t for simplicity.

14.2.2 Example Newton's method vs gradient descent

Now, let's see an example of how Newton's method performs compared to gradient descent. We consider the problem of minimizing

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

Note that this is a nonquadratic function, because if it were, then Newton's method would just take one step and reaches the minimum.



We compare gradient descent (black curve) to Newton's method (blue curve), where both take steps of roughly same length, just moving towards different directions. Notice that Newton's method is able to go straight at the minimum, whereas gradient descent takes the direction always orthogonal to the tangent line at any one point of the contour.

14.3 Outline

The topics of today's lecture include:

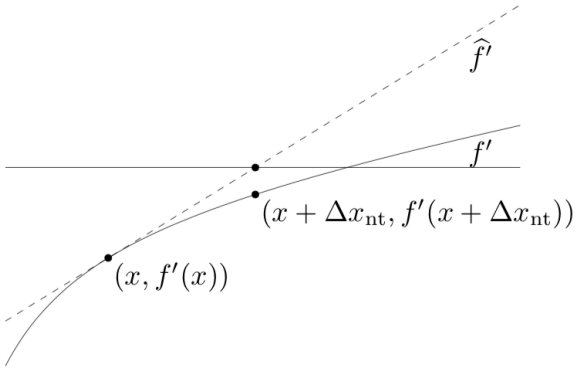
- Interpretations and properties
- Backtracking line search
- Convergence analysis
- Equality-constrained Newton
- Quasi-Newton preview

14.4 Linearized optimality condition

An alternative interpretation of Newton's step at x is that we seek a direction v so that $\nabla f(x+v) = 0$. Let $F(x) + \nabla f(x)$. Consider **linearizing** F around x, via first-order approximation.

$$0 = F(x+v) \approx F(x) + DF(x)v$$

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



In the above figure, $\Delta x_{nt} = -(\nabla^2 f(x))^{-1}\nabla f(x)$ is the Newton step, and we can think of the Newton step as the addition needed to satisfy linearized optimality condition, namely the v in $\nabla f(x+v) = 0$.

(Figure 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.

14.5 Affine invariance of Newton's method

An important property of Newton's method is **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.

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

As we just saw that for an arbitrary affine transformation A, applying Newton's method on y yields y^+, then applying A on y^+ to get $x^+ = Ay^+$, which is exactly the same as applying A on y first to get x and then applying Newton's method on x. Hence, we have shown that Newton's method is affine invariant (progress is independent of problem scaling), which is **not true** for gradient descent.

14.6 Newton Decrement

At a point x, we define the **newton decrement** as

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

For notational convenience, let's use $g_x = \nabla f(x)$ and $H_x = \nabla^2 f(x)$ to denote the gradient and hessian. *(If x were n-dimensional, then $g_x \in \mathbb{R}^n$ and $H_x \in \mathbb{R}^{n\times n}$).* This quantity naturally comes up when we think about the difference between $f(x)$ and the minimum of the quadratic approximation. That is

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

With this motivation, we can think of $\frac{1}{2}\lambda(x)^2 \leq \epsilon$ as a reasonable stopping condition.

14.6.1 Alternate interpretation of Newton decrement

The newton direction at a point x is given by

$$v_x = -(\nabla^2 f(x))^{-1}\nabla f(x) = -H_x^{-1}g_x$$

Then it's straightforward to see that

$$\lambda(x) = \sqrt{v_x^T H_x v_x} = \|v_x\|_{H_x}$$

This means that $\lambda(x)$ is the length of the newton step in the norm defined by the hessian at x.

14.6.2 Affine Invariance of Newton Decrement

Define a generic affine transformation as

$$g(y) = f(Ay + b)$$

Then at $x = Ay + b$ (and A non-singular), we can do the following algebra

$$\lambda_g(y)^2 = \nabla g(y)^T (\nabla^2 g(y))^{-1} \nabla g(y)$$

$$\implies \lambda_g(y)^2 = (A^T \nabla f(x))^T (A^T \nabla f(x) A)^{-1} A^T \nabla f(x)$$

$$\implies \lambda_g(y)^2 = \nabla f(x)^T A A^{-1} (\nabla f(x))^{-1} (A^T)^{-1} A^T \nabla f(x)$$

$$\implies \lambda_g(y)^2 = \lambda_f(x)^2$$

Hence the newton decrement, similar to the newton step, is **affine invariant**.

14.7 Backtracking Line Search

So far we've seen the **pure Newton's method**. This need not converge. This is because pure newton's method is *not* guaranteed to be a descent method. To adjust for this we use **damped Newton's method**. This also takes the step in the newton direction but a scaled version of that; guaranteeing that the criterion value decreases — ensuring descent.

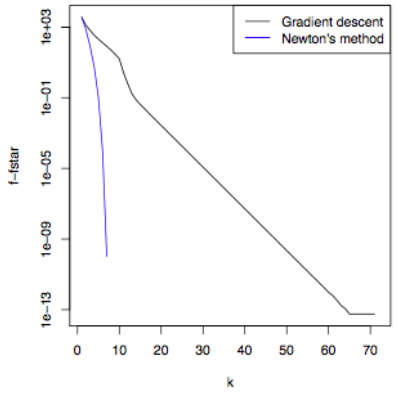
$$x^+ = x - t(\nabla^2 f(x))^{-1} \nabla f(x) \text{ where } t \leq 1$$

The step size t is chosen by backtracking line search. We use parameters $\alpha \in (0, \frac{1}{2}]$ and $\beta \in (0, 1)$. At each outer iteration, we start with $t = 1$ and run an inner loop :
while $f(x + tv) > f(x) + \alpha t \nabla f(x)^T v$
    $t = \beta t$
$x \leftarrow x + tv$
Here $v = -(\nabla^2 f(x))^{-1} \nabla f(x)$ which means $\nabla f(x)^T v = -\lambda(x)^2$.
So loop condition is $f(x + tv) > f(x) - \alpha t \lambda(x)^2$

Aside : Here starting with $t = 1$ at the beginning of each inner loop seems principled, since that is the pure Newton's method. So in spirit, we try to take the full newton step. But if that does not give us a decrease in the criterion (objective), we shrink the step size by a multiplicative factor.

14.8 Example : Logistic Regression

Number of data points $n = 500$, number of covariates/features $p = 100$. Here's a comparison of Gradient descent (with backtracking) and Newton's method (again, with backtracking).



This shows that newton's method is in a totally different regime of convergence. As we will shortly see, newton's method is a **(locally) quadratic convergence** method. In terms of order notation :

Method	Suboptimality after k iterations
Gradient Descent	$O(c_1^k)$
NM (in quadratic convergence regime)	$O(c_2^{2^k})$

Here $c_1 = (1 - \frac{m}{L})$ (since this is a strongly convex problem). We will see that $c_2 = \frac{1}{2}$ is a problem independent constant for Newton's method.

14.9 Convergence Analysis of Newton's method

Our assumptions are :

- f is convex, twice differentiable and has domain $dom(f) = \mathbb{R}^n$
- ∇f is Lipschitz with parameter L
- f is strongly convex with parameter m
- $\nabla^2 f$ is Lipschitz with parameter M

We first elaborate on the last point. It means that $\|\nabla^2 f(x) - \nabla^2 f(y)\|_{op} \leq M\|x - y\|_2$. We can use other norms also.

Theorem 14.1 *Newton's method (with backtracking line search) satisfies the two-stage convergence bounds* $\exists\ 0 < \eta \leq \frac{m^2}{M}$, $\gamma > 0$ *and* $k_0 \in \mathbb{Z}_+$ *such that*

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

For the proof, one can use $\gamma = \alpha\beta^2\eta^2\frac{m}{L^2}$. And k_0 is the number of iterations until $\|\nabla f(x^{(k_0+1)})\|_2 < \eta$ is satisfied. We can give a sketch of the proof for the two phases :
Damped phase : $\|\nabla^2 f(x^{(k)})\| \geq \eta$. The only thing we can say is that the criterion will reduce by at least

some fixed positive quantity in each iteration. What remains is to show that such a quantity $\gamma > 0$ exists. *(Side note : Most iterations will require backtracking steps in the damped phase.)* Also, note that the damped phase will continue for at most $k_0 \leq \frac{f(x^{(0)})-f^\star}{\gamma}$ iterations (since criterion can't decrease beyond $f^\star$).
Pure phase : $\|\nabla^2 f(x^{(k)})\| < \eta$. It can be shown that backtracking always selects $t = 1$. Further it can be shown that $\frac{M}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \leq (\frac{M}{2m^2}\|\nabla f(x^{(k)})\|_2)^2$. Meaning that $\|\nabla f(x^{(k)})\|_2$ converges quadratically to zero. Also, once we enter the pure phase we won't leave it because

$$\|\nabla f(x^{(k+1)})\|_2 \leq \frac{2m^2}{M^2}(\frac{M}{2m^2}\|\nabla f(x^{(k)})\|_2)^2 \leq \frac{2m^2}{M^2}(\frac{M}{2m^2}\eta)^2 \leq \frac{M}{2m^2}\eta^2 \leq \frac{M}{2m^2}\frac{m^2}{M}\eta = \frac{\eta}{2} \leq \eta \text{ (since } \eta \leq \frac{m^2}{M})$$

Now, we can derive a bound on the number of iterations needed to reach an ϵ suboptimal point. We need at most

$$\frac{f(x^{(0)}) - f^\star}{\gamma} + \log_2 \log_2(\frac{\epsilon_0}{\epsilon})$$

where $\epsilon_0 = \frac{2m^3}{M^2}$. A few notes :

- This is quadratic convergence, since the number of iterations for ϵ suboptimality is $\mathcal{O}(\log\log\frac{1}{\epsilon})$. Compare this to linear convergence $\mathcal{O}(\log\frac{1}{\epsilon})$ which is what gradient descent achieves under strong convexity.
- This quadratic convergence is local. After a fixed number of iterations $k_0 \leq \frac{f(x^{(0)})-f^\star}{\gamma}$ we're guaranteed this.
- The second term is practically a constant. $\log_2\log_2(\frac{1}{\epsilon}) \leq 6$ for $\epsilon = 10^{-10}$

The above convergence analysis is called the classical analysis. A good point of this analysis is that it explains the two different convergence regimes observed in practice with Newton's method (with backtracking). However it contains constants that are usually unknown in practice (like m, M). Further, the iteration is bound is riddled with all these constants (that are *not* affine invariant), and yet the underlying method is indeed affine invariant (scale-free).

14.10 Self-Concordance

A scale-free analysis is possible for self-concordant functions.
A convex function $f : \mathbb{R} \to \mathbb{R}$ is called self-concordant if

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

(Note that convexity ensures that $(f''(x))^{3/2}$ is well defined since $f''(x) \geq 0$ for all x)
A multivariate convex function $f : \mathbb{R}^n \to \mathbb{R}$ is called self-concordant if

$$g(t) = f(x + tv) \text{ is self-concordant } \forall\ x \in dom(f), \forall\ v \in \mathbb{R}^n$$

The above just means that the projection of f on any line is self-concordant.

14.10.1 Convergence analysis via Self-concordance

This bound does not depend on any unknown constants — gives an affine invariant bound.

Theorem 14.2 *Newton's method (with backtracking line search) requires at most*

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

iterations to reach an ϵ suboptimal point.

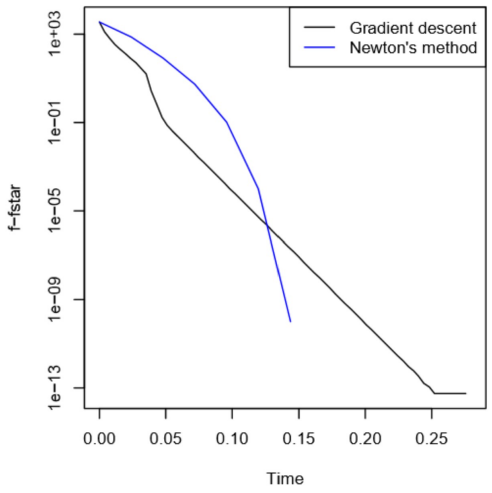
Note that here $C(\alpha,\beta)$ does not depend on any other constants like m, M as it did in the classical analysis.

14.10.2 Self Concordance Function Examples

- Linear functions (LP)
- Quadratic functions (QP)
- $f(X) = -\log(\det(X))$ on $\mathbb{S}^n_{++}$
- $f(x) = -\sum_{i=1}^n \log(x_i)$ on $\mathbb{R}^n_{++}$
- g : Self-concordant $\longrightarrow f(x) = g(Ax+b)$ also self-concordant.
- Instead of 2 in the definition a general κ can be replaced.
- g : κ-self-concordant $\longrightarrow f(x) = \frac{\kappa^2}{4}g(x)$: 2-self-concordant.

14.11 Comparison to 1st-order methods

Newton's method has its own pros and cons at high-level compared to the first order methods.
In terms of **Memory**, at each iteration we save an $n \times n$ Hessian which requires $O(n^2)$ storage, whereas for the gradient we only require $O(n)$ to store the n-dimensional gradient.
Regarding **Computation** also, we are solving a dense $n \times n$ system of linear equations for each iteration of Newton's method that take $O(n^3)$ as opposed to gradient iterations requiring scaling and adding n-dimensional vector that takes $O(n)$.
On the other hand, we have roughly the same cost regarding the **Backtracking** for both methods taking $O(n)$ time in the inner loop for contraction.
On the plus side for Newton's method compared to gradient descent, regarding **Conditioning**, we have that the gradient descent that can seriously degrade, Newton's method unlike will not be affected by problem's conditioning.
The following plot shows $f - f^*$ of logistic regression example for gradient descent and Newton's method with the horizontal axis being parameterized in terms of time taken per iteration. We have that each gradient descent step taking $O(p)$ as compared to $O(p^3)$ of Newton's step. We see a somewhat linear drop for gradient descent and a faster drop for Newton's method.



14.12 Sparse, structured problems

Newton's method strives when the inner linear systems in Hessian can be solved efficiently and reliably. This is the case when the Hessian is sparse, structured for all x that is called **banded**.

$H = vg \begin{cases} O(n^3) & \text{H: Dense} \\ O(n) & \text{H: Banded} \end{cases}$

Examples of functions with structured Hessian:

- $g(\beta) = f(X\beta) \longrightarrow \nabla^2 g(\beta) = X^T \nabla^2 f(X\beta) X \longrightarrow$ If $\nabla^2 f$ being diagonal and X being a structured predictor matrix $\longrightarrow \nabla^2 g$ is also structured. Correlation is local.
- Minimizing $f(\beta) + g(D\beta)$ with $\nabla^2 f$ diagonal, g non-smooth (prox hard to calculate), and D structured penalty matrix $\longrightarrow$ the Lagrange dual: $-f^*(-D^T u) - g^*(-u)$. For many cases including when $f(\beta) = \sum_{i=1}^p f_i(\beta_i)$, $\nabla^2 f^*$ is diagonal and thus the Hessian in the dual is structured as a result.

Example: $f(u) = -y^T u + \sum_{i=1}^n b(u_i)$ (for instance for least squares $b(u) = \frac{u^2}{2}$) and here the Hessian is diagonal. We could also use logistic regression ($log(1 + \exp(u)$.

14.13 Equality-constrained Newton's Method

For minimization problem with equality constrained of following we have several options.

$$\min_x f(x) \text{ subject to } Ax = b$$

The first option is to **eliminate the equality constraints** by considering x to be $Fy + x_0$ with F spanning the null space of A, $Ax_0 = b$ and then solving in terms of y. The second option is relying on deriving Lagrange **dual function** of $-f^*(-A^T v) - b^t v$, checking that the strong duality holds, and if we are lucky we can then express the optimal x^* in terms of v^*. Lastly, we can use the **Equality-constrained Newton** that is the most straightforward option in most applications. In this method:
Start with $x^{(0)}$ s.t. $Ax^{(0)} = b$.

Repeat updates:
$x^+ = x + tv$ where $v = \arg\min_{Az=0} \nabla f(x)^T z + \frac{1}{2} z^T \nabla^2 f(x) z$ (Direction in null space of A).
We have that x^+ is feasible because $Ax^+ = Ax + tAv = b + 0 = b$. We also have that v is solution for minimizing a quadratic program subject to equality constraints.
The KKT conditions satisfied by v (for some w) are:

$$\begin{pmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{pmatrix} \quad \begin{pmatrix} v \\ w \end{pmatrix} \quad = \quad \begin{pmatrix} -\nabla f(x) \\ 0 \end{pmatrix}$$

This results in v (Newton direction) being given by solving a bigger linear system but again in terms of the Hessian.

14.14 Quasi-Newton methods

When Hessian is too expensive or singular then we can use the **quasi-Newton** method that approximates $\nabla^2 f(x)$ with $H \succ 0$. The updates are such that: $x^+ = x - tH^{-1}\nabla f(x)$.
Here we approximate the Hessian (H) at each step with the goal of having the inverse to be cheap both in terms of application and storage. Quasi-Newton is converges fast (**superlinear**) but not as fast as Newton, as an estimate, every n steps of quasi-Newton method make the same progress as one Newton step. There are many different varities of quasi-Newton methods but almost all try to propagate the computation of hessian approximate among their iterations.

References

[1] R. Tibshirani, Lecture notes for 10-725, CMU, Fall 2019

[2] S. Boyd and L. Vandenberghe (2004), "Convex Optimization", Chapters 9 and 10

[3] Y. Nesterov(1998), "Introductory lectures on convex optimization: a basic course", Chapter 2

[4] Y. Nesterov and A. Nemirovskii (1994), "Interior-point polynomial methods in convex programming", Chapter 2

[5] J. Nocedal and S. Wright (2006), "Numerical optimization", Chapters 6 and 7

[6] L. Vandenberghe, Lecture notes for EE 236C, UCLA, Spring 2011-2012