10-725/36-725: Convex Optimization **Spring 2015**

Lecture 14: March 2: Newton's Method

Lecturer: Ryan Tibshirani *Scribes: Swabha Swayamdipta*

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

This lecture's notes illustrate some second order methods like Newton's method and sets the stage for other second order methods like the Barrier method and other interior point methods .

14.1 Recap on dual correspondences

The conjugate of a function, $f(x)$, is given by $f^* = \max_x y^T x - f(x)$. Conjugacy is intimately related to duality since evaluating the conjugate is very similar to the minimization of the Lagrangian. The conjugate of f is always affine, regardless of f, and hence the maximization is always a convex problem.

A key result to remember is that if the primal problem is of the form

$$\min_x f(x) + g(x)$$

then the dual problem can be written in terms of conjugates thus

$$\max_u \ -f^*(u) - g^*(-u)$$

14.2 Newton's method

Duality plays a very fundamental role in designing second-order methods for convex optimization. Newton's method is a second-order method in the simplest setting where we consider unconstrained smooth convex optimization (same as the setting for gradient descent).

$$\min_x f(x)$$

Recall that in gradient descent, the update in the kth iteration, $x^{(k)}$ moved in the direction of the negative gradient of the previous iteration

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

where t_k is the step-size. In contrast, in Newton's method we move in the direction of negative Hessian inverse of the gradient.

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

This is called the pure Newton's method, since there's no notion of a step size involved. As is evident from the update, Newton's method involves solving linear systems in the Hessian.

To motivate Newton's method, consider the following quadratic approximation

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

The Newton update is obtained by minimizing the above w.r.t. y. This quadratic approximation is better than the approximation used in gradient descent (given by 14.1), since it uses more information about the function via the Hessian.

$$f(y) \approx \ f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(x) \tag{14.1}$$

As an illustration, let us consider minimizing $f(x) = (10{x_1}^2 + {x_1}^2)/2 + 5\log(1+\exp -x_1 - x_2)$. The updates are illustrated in 14.2. We see that the gradient descent updates move in a direction orthogonal to the contours. However, the Newton's method spheres the contours in every step and moves in a direction orthogonal to the sphered contours. This results in a more direct path (not necessarily a straight line).

Fact: Newton's method minimizes a quadratic in a single step! This is true because the quadratic approximation is the same as the original function.

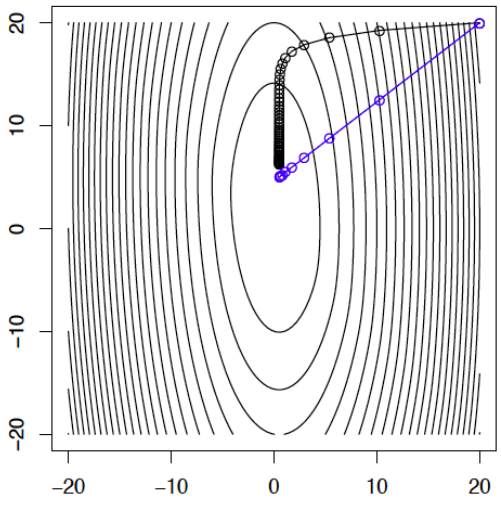


Figure 14.1: Newton's method(blue) vs. gradient descent(black) updates. For simplicity, approximately the same step length has been used for both methods.

14.3 Properties of Newton's method

Linearized optimality condition

An alternative motivation for Newton's method comes from the first-order(linearized) optimality condition. When we are at a point x, we seek a direction v such that $\nabla f(x+v) = 0$. Using a first order Taylor expansion of the gradient, we get

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

Solving this linearized system for v, we get $v = -(\nabla^2 f(x))^{-1}\nabla f(x)$, which is the Newton's update.

Historically, this method was devised first to find the roots of a polynomial equation. This was done independently by Newton and Ralphson. Later, Simpson applied the method to non - linear equations and also used it for minimization by setting the gradient equal to zero.

Affine Invariance of Newton's method

Consider an affine transformation $x = Ay$ of the variable x in the original minimization problem, $\min_x f(x)$ where $A \in \mathbb{R}^{n\times n}$ is an invertible matrix. Let $g(y) = f(Ay)$. The Newton step for g is

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

This shows that the Newton's method update on g is also the same as that on f. In other words, Newton's method is independent of linear scaling. This holds because in every step, we are using an approximation which involves sphering the contours. However, the same cannot be said for gradient descent (because it uses a different quadratic approximation), which is hence not affine invariant.

Newton Decrement

Given a smooth, convex function f, the Newton decrement is given by

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

The Newton decrement is proportional to the difference between $f(x)$ and minima of the quadratic approximation at x.

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

Hence, it provides an approximate bound on suboptimality gap $f(x) - f^*$. The bound is not exact because it takes into account the minimum of the quadratic approximation, not the actual minimum of $f(x)$. Regardless, it can be used as a stopping criterion for backtracking line search.

The Newton decrement is also the Hessian norm of the length of the Newton step, $v = -\nabla^2 f(x)^{-1}.\nabla f(x)$. We know that $||v||_A = \sqrt{(v^T A v)}$. For $A = \nabla^2 f(x)$, we get

$$||v||_{\nabla^2 f(x)} = \sqrt{\nabla f(x)^T . \nabla^2 f(x)^{-1} . \nabla f(x)}$$

The Newton decrement is also affine invariant.

14.4 Convergence Analysis

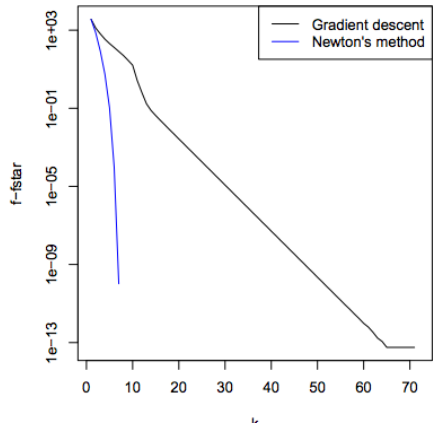
The pure Newton method doesn't necessarily converge. Depending on where we start, the Newton method can either converge or diverge "quadratically quickly". In practice, backtracking line search is used with Newton's method, with parameters $0 < \alpha \leq 1/2$, $0 < \beta < 1$ just like first-order methods. In Newton's method with backtracking, we 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. Here, $v = -\nabla^2 f(x)^{-1}.\nabla f(x)$. Note that $\nabla f(x)^T v = -\lambda^2(x)$.

For example, let us consider the convergence rates for Newton's method vs. gradient descent(which has linear convergence) for logistic regression in Fig. **??**. We see an entirely different regime of convergence(called quadratic convergence) for the Newton's method, it is extremely fast. In fact, this comparison is unfair since Newton's method involves solving the Hessian (in $O(n^3)$ flops), which might be expensive itself.

Figure 14.2: Convergence rates for Newton's method vs. gradient descent



Convergence Result

Let us assume that f is strongly convex with parameter m and twice differentiable and $\text{dom}(f) = \mathbb{R}^n$. Let us also assume that ∇f is Lipschitz with parameter L. These conditions guarantee that gradient descent on the same problem will have linear convergence. Recall that the gradient descent convergence rate depends adversely on the condition number, L/m.

Let us additionally assume that $\nabla^2 f(x)$ is Lipschitz with parameter H. Hence,

$$||\nabla^2 f(x) - \nabla^2 f(y)||_F \leq H||x - y||_2$$

where $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n \times n}$.

The convergence results hold for Newton's method with backtracking (with parameters α, β) and depend on two parameters, $\gamma > 0$ and $0 < \eta \leq m^2/H$. Let k_0 be the number of steps till $||\nabla f(x^{(k_0+1)}||_2 < \eta$. k_0 breaks the convergence theorem into two stages, where $\eta = \min\{1, 3(1-2\alpha)\}m^2/H$.

In the first stage, when $k \leq k_0$,

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

This might not be very fast, but we are guaranteed to decrease the criterion by γ in every step where $\gamma = \alpha\beta^2\eta^2 m/L^2$. If a function is poorly conditioned, then γ is fairly small and we cannot decrease the criterion by much in each step. The above bound is very conservative and this phase of convergence is called the damped Newton phase.

There exists a second regime of convergence when $k > k_0$,

$$f(x^{(k)}) - f^* \leq \frac{2m^3}{H^2}(\frac{1}{2})^{2^{k-k_0+1}}$$

This rate of convergence is extremely fast, and hence is named quadratic convergence. This phase is called the pure Newton phase. In the pure Newton phase, $t = 1$ is always satisfied for backtracking, there is no decay in the step size t. This means that once we enter the pure Newton phase, we won't leave it.

To reach an ϵ desired level of accuracy, where $f^{(k)} - f^* \leq \epsilon$, the number of iterations required to leave the first phase are

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

For the second phase, we can use strong convexity to prove that the number of iterations required is

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

where $\epsilon_0 = 2m^3/H^2$.

The $log(log(\epsilon_0/\epsilon))$ term in the convergence result makes the convergence quadratic. For instance, to get $\epsilon = 2^-32$ we need only 5 steps in Newton's method with backtracking!

But this quadratic convergence result is only local, it is guaranteed in the second or pure phase only. The global convergence rate for Newton's method is given by some latest results by Nesterov which suggest linear convergence.

Self-concordance

The convergence results for Newton's method might seem dissatisfying because we know that it is affine invariant but the results involve constants L, m and H. A scale-free analysis has been proposed by Nesterov and Nemirovskii for self-concordant functions, f. A function is self-concordant if it satisfies

$$|f'''(x)| \leq 2f''(x)^{3/2} \quad \forall x$$

.

For example, $f(x) = -\log x$ is self-concordant. If a function f is self concordant, then Newton's method with backtracking line search requires at most

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

steps to reach ϵ level of accuracy. This result is interesting because it does not involve the Lipschitz or strong convexity constants and is a better way of characterizing Newton's method.

14.5 Comparison with gradient descent

Let us assume that the $\text{dom}(f) \in \mathbb{R}^n$.

Memory : Each iteration of Newton's method requires $O(n^2)$ storage due to the $n \times n$ Hessian whereas each gradient iteration requires $O(n)$ storage for the n-dimensional gradient.

Computation : Each Newton iteration requires $O(n^3)$ flops as it solves a dense $n \times n$ linear system. Each gradient descent iteration requires $O(n)$ flops attributed to scaling/adding n-dimensional vectors.

Backtracking : Backtracking line search has roughly the same cost for both methods, which use $O(n)$ flops per inner backtracking step.

Conditioning : Newton's method is not affected by a problem's conditioning(due to affine invariance), but gradient descent can seriously degrade, since it depends adversely on the condition number.

Fragility : Newton's method may be empirically more sensitive to bugs/numerical errors, whereas gradient descent is more robust.

From the above, we can conclude that even though Newton's method has quadratic convergence as compared to linear convergence of gradient descent, in practice, computing the Hessian might make the method a lot slower. If the Hessian is sparse and structured(e.g. banded), then both memory and computation are $O(n)$.

14.6 Equality-constrained Newton's method

Let us now consider problems with equality constraints

$$\min_x \quad f(x) \quad \text{s.t.} \quad Ax = b$$

We can do a transformation of the variables (recall Optimization Basics, Lecture 3) by writing $x = My + x_0$ where M spans the null space of A, and choosing $Ax_0 = b$. Now the minimization problem is in terms of y. However, this problem can be hard because finding an M might not be easy and this transformation can result in a non-sparse Hessian.

Alternatively, we could compute the Lagrangian dual, $-f^*(-A^Tv) - b^Tv$ and proceed with an unconstrained problem. However, the dual might not be straightforward to derive, since conjugates are involved.

A third, most straightforward option is to use equality-constrained Newton's method. In this method, we take Newton steps which are confined to a region satisfied by the constraints. The Newton update is now $x^+ = x + tv$ where

$$v = \underset{Az=0}{\operatorname{argmin}} \nabla f(x)^T(z-x) + \frac{1}{2}(z-x)^T\nabla^2 f(x)(z-x)$$

This ensures that x^+ is feasible as $Ax^+ = Ax + tAv = b$. KKT conditions result in an an equality-constrained KKT matrix given below, which maintains the structure and sparseness of the Hessian. Note that we now have to solve a bigger linear system in the Hessian.

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

In case of general constraints, however, we cannot use projected methods like we did for gradient descent. Instead, we will have to rely on Barrier methods which will be introduced in the next lecture. Barrier methods are a special case of the more general interior point methods which will be visited in future lectures.

14.7 Quasi-Newton methods

Quasi-Newton methods are a class of methods that come into play when Hessians are either singular or too expensive to compute or solve linear systems in. These methods approximate the Hessian with some matrix,

$H \succ 0$, i.e. H needs to be strictly positive definite. H is recomputed in every step and hence must be cheap to store.

Quasi-Newton methods are superlinear, i.e. their convergence guarantees are somewhere between linear (like gradient descent) and quadratic (like Newton's method). Roughly n steps of quasi-Newton make same progress as one Newton step. The key idea used in quasi-Newton methods is to avoid a full computation of H across iterations. Two popular variants of quasi-Newton methods are DFP and BFGS, named after their authors.

14.7.1 Davidon-Fletcher-Powell or DFP

The Davidon-Fletcher-Powell method was the first quasi-Newton method to be introduced. This method makes an approximation(derived from a Taylor expansion) of the Hessian from the rank 2 updates from previous iterations. In each step, however, H^{-1} needs to be computed, hence resulting in $O(n^2)$ operations.

14.7.2 Broyden-Fletcher-Goldfarb-Shanno or BFGS

The Broyden-Fletcher-Goldfarb-Shanno method was introduced after the DFP, but is now much more widely used. This method also makes an approximation(derived from a Taylor expansion) of the Hessian from the rank 2 updates from previous iterations, but does so in a "dual" fashion to DFP. Hence, the cost is still $O(n^2)$.

A limited-memory version, L-BFGS is even more widely used, which instead of letting updates propogate over all iterations, only keeps updates from last m iterations. Hence, the storage required is $O(mn)$ instead of $O(n^2)$.

References

- S. Boyd and L. Vandenberghe(2004), "Convex Optimization," *Chapters 9 and 10.*
- R. Tibshirani(2015), "Newton's Method".