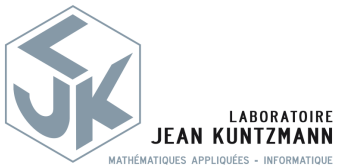

Lecture 7 : Second-order methods - Newton algorithm

This lecture is dedicated to the second-order methods, notably the Newton algorithms and its theoretical guarantee. In general, these methods can enjoy a much faster convergence rate than first-order counterparts. Nevertheless, the cost per iteration makes them very expensive, hence hindering their usage in large-scale optimization. Adapting second-order methods to real-world scale problems is an active research domain, and might enable us to handle challenging optimization problems.

1 Second-order methods - the Newton step

So far, all algorithms that we have seen in this course only exploit first-order information, i.e., their update rules only use gradients. It is, thus, very tempting to exploit higher-order derivatives to see whether we can break the fundamental limit of first-order methods. Newton method, possibly one of the most well-known second-order methods, will be our focus of today lecture.

Given a C^2 function, the Newton update is given by:

$$x_{k+1} = x_k - \alpha \nabla^2 f(x_k)^{-1} \nabla f(x_k). \tag{Newton}$$

When $\alpha = 1$, it is called the *pure Newton step*. If the step-size $\alpha \neq 1$, then this is a damped Newton step.

There are, in fact, many interpretations and fun facts about (Newton):

Original Newton method is to find zero of functions, and not an optimization algorithm The original Newton method was proposed to find the x such that $f(x) = 0$ of a differentiable function $f : \mathbb{R} \to \mathbb{R}$. Starting from a guess point, the algorithm updates its iterations as:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

This rule is proposed due to the first-order approximation: $f(x + \delta) = f(x) + f'(x)\delta$. The pure (Newton) is when we apply the original Newton method for $f : \mathbb{R}^n \to \mathbb{R}^n : x \mapsto \nabla f(x)$, or more precisely:

$$\nabla^2 f(x_k)(x_{k+1} - x_k) - \nabla f(x_k) = 0.$$

Pure Newton step is the steepest descent method of a special metric In fact, the descent direction of gradient descent can be view as the minimizer of the following constrained problem:

$$\begin{aligned} \underset{p\in\mathbb{R}^d}{\text{Minimze}} \quad & \nabla f(x_k)^\top p \\ \text{s.t.} \quad & \|p\|_2 \leq 1. \end{aligned}$$

The Newton step has the same interpertation, but with $\|p\|_f = p^\top \nabla^2 f(x_k) p$, i.e.,

$$\begin{aligned} \underset{p\in\mathbb{R}^d}{\text{Minimze}} \quad & \nabla f(x_k)^\top p \\ \text{s.t.} \quad & \|p\|_f \leq 1. \end{aligned}$$

We can get the optimal direction by considering $\nabla^2 f^{1/2} p$.

Pure Newton step is the minimization of the surrogate of the function f In fact, the gradient descent step can be viewed as the minimizer of the following surrogate function:

$$f_{\text{GD}}(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2\alpha}\|x - x_k\|^2$$

where the minimizer is given by:

$$x^\star = x_k - \alpha \nabla f(x_k) = \text{gradient descent step.}$$

For Newton step, the surrogate function is the second order Taylor approximation of f, i.e.:

$$f_{\text{Newton}}(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2}(x - x_k)^\top \nabla^2 f(x_k)(x - x_k),$$

given that $\nabla^2 f(x_k) \succ 0$.

Affine invariant of Newton step An important of (Newton) is that it is independent of linear or affine changes of coordinates. Suppose that we have a non-singular matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, and define $g = f \circ \mathbf{A}$, we have:

$$\nabla g(x) = \mathbf{A}^\top \nabla f(\mathbf{A}x) \qquad , \qquad \nabla^2 g(x) = \mathbf{A}^\top \nabla^2 f(\mathbf{A}x)\mathbf{A}.$$

Therefore, the Newton update of the function g at the point $y = \mathbf{A}^{-1}x$ is:

$$\begin{aligned} \mathbf{A}^{-1}x - \left(\nabla^2 g(\mathbf{A}^{-1}x)\right)^{-1} \nabla g(\mathbf{A}^{-1}x) &= x - (\mathbf{A}^\top \nabla^2 f(x)\mathbf{A})^{-1}\mathbf{A}^\top \nabla f(x) \\ &= \mathbf{A}^{-1}x - \mathbf{A}^{-1}\nabla^2 f(x)^{-1}\nabla f(x) \\ &= \mathbf{A}^{-1}(x - \nabla^2 f(x)^{-1}\nabla f(x)). \end{aligned}$$

Thus, the Newton update of the function g is that of the function f, up to the inverse of non-singular linear transformation $\mathbf{A}$. This is the unique property of Newton methods, unlike other first-order methods.

2 Newton algorithm and theoretical guarantees

The Newton algorithm is given as below: in each iteration:

1. Compute the Newton step p_k as in (Newton) and decrement $\lambda^2 := \nabla f(x)^\top \nabla^2 f(x)^{-1} \nabla f(x)$.
2. Line search: find step-size α so that $f(x - \alpha p_k) \leq f(x) - c\alpha\lambda^2$ (Armijo rule) by backtracking.
3. Update: $x_{k+1} = x_k - \alpha p_k$.

Theoretical guarantees We employ two assumptions

Assumption 2.1 (Strongly convex and smooth)**.** *The function f is m-strongly convex and M-smooth, i.e.,*

$$m\mathbf{I} \leq \nabla^2 f(x) \leq M\mathbf{I}, \forall x.$$

Assumption 2.2 (L-Lipschitz hessian)**.** *The Hessian of f is Lipschitz continuous with constant L, i.e.,*

$$\|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \leq L\|x - y\|_2, \forall x, y.$$

In the analysis, the Newton algorithm will go through two stage:

1. *Damped Newton phase*: In this stage, the value function will decrease steadily by a constant $\gamma > 0$ and the norm of $\|\nabla f(x)\| \leq \delta > 0$.
2. *Pure Newton phase*: In the second stage, the norm of the gradient becomes small and the line search will always pick $\alpha_k = 1$ and:

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

This is also known as quadratic convergence.

Before entering the technicalities, remind us of two identities that will be used repeatedly throughout the proof:

$$\begin{aligned} f(y) &= f(x) + \int_0^1 \nabla f(x + t(y - x))(y - x)dt, \\ f(y) &= f(x) + \nabla f(x)^\top (y - x) + \int_0^1 (1 - t)(y - x)^\top \nabla^2 f(x + t(y - x))(y - x)dt. \end{aligned} \tag{1}$$

Damped Newton phase By M-smoothness of the function f, we have:

$$\begin{aligned} f(x_k - \alpha p_k) &\leq f(x_k) - \alpha \nabla f(x_k)^\top p_k + \alpha^2 \frac{M\|p_k\|^2}{2} \\ &\leq f(x_k) - \alpha\lambda_k^2 + \frac{M}{2m}\alpha^2\lambda_k^2. \end{aligned}$$

Therefore, if we choose $\alpha \leq \frac{m}{M}$, it will satisfy the exit condition of the line search for $c < \frac{1}{2}$ because:

$$f(x_k - \alpha p_k) \leq f(x_k) - \alpha\left(1 - \frac{M}{2m}\alpha\right)\lambda^2. \tag{2}$$

Therefore, our step-size α found by line-search will satisfy:

$$\alpha_k \geq \beta\frac{m}{M},$$

where $\beta > 0$ is a constant dependent on our line search algorithm. As such:

$$f(x_{k+1}) - f(x) \leq -\frac{1}{2}\alpha\lambda^2 \leq -\beta\frac{m}{2M}\lambda^2 \leq -\alpha\beta\eta^2\frac{m}{M^2}.$$

because $\|\nabla f(x)\|_2 < \eta$ and $\lambda^2 \geq (1/M)\|\nabla f(x)\|^2$.

Quadratically convergent phase We prove next that if:

$$\eta \leq \frac{m^2}{2L},$$

then the line search will choose unit step, i.e., $\alpha_k = 1$. Indeed, since f is L-smooth, we have:

$$\begin{aligned}
f(x_{k+1}) &= f(x_k - \underbrace{\nabla^2 f(x_k)^{-1}\nabla f(x_k))}_{p_k} \\
&= f(x_k) - \nabla f(x_k)^\top p_k + \int_0^1 (1-t) p_k^\top \nabla^2 f(x_k - t p_k) p_k dt \\
&= f(x_k) - \frac{1}{2}\lambda_k^2 + \int_0^1 (1-t) p_k^\top (\nabla^2 f(x_k - t p_k) - \nabla^2 f(x_k)) p_k dt \\
&\leq f(x_k) - \frac{1}{2}\lambda_k^2 + \left\| \int_0^1 (1-t) p_k^\top (\nabla^2 f(x_k - t p_k) - \nabla^2 f(x_k)) p_k dt \right\|^2 \\
&\leq f(x_k) - \frac{1}{2}\lambda_k^2 + \frac{L}{6} \cdot \|p_k\|^3.
\end{aligned}$$

Because

$$\begin{aligned}
\lambda_k^2 = \|\nabla^2 f(x_k)^{-\frac{1}{2}} \nabla f(x_k)\|_2^2 = \|\nabla^2 f(x_k)^{\frac{1}{2}} p_k\|_2^2 \geq m\|p_k\|^2, \\
\|\nabla f(x_k)\| = \|\nabla^2 f(x_k) p_k\|_2^2 \geq m\|p_k\|,
\end{aligned}$$

we have:

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2}\lambda_k^2 + \frac{L}{6m^2}\lambda_k^2 \|\nabla f(x_k)\| = f(x_k) - \frac{1}{2}\lambda_k^2 \left(1 - \frac{L\|\nabla f(x_k)\|}{3m^2}\right).$$

Therefore, if $\eta \leq \frac{m^2}{2L}$, we have $f(x_k - d_k) \leq f(x_k) - \frac{1}{4}\lambda_k^2$ and line search will choose $\alpha_k = 1$.

Next, we will argue that when $\|\nabla f(x_k)\| \leq \frac{m^2}{2L}$, $\|\nabla f(x_\ell)\|, \ell \geq k$ will be smaller than this amount forever. Indeed,

$$\begin{aligned}
\|\nabla f(x_{k+1})\|_2 &= \|\nabla f(x_k - \nabla^2 f(x_k)^{-1}\nabla f(x_k))\|_2 \\
&= \left\| \nabla f(x_k) - \int_0^1 \nabla^2 f(x - t\nabla^2 f(x_k)^{-1}\nabla f(x_k)) \nabla^2 f(x_k)^{-1}\nabla f(x_k) dt \right\|_2 \\
&= \left\| \int_0^1 \left(\nabla^2 f(x - t\nabla^2 f(x_k)^{-1}\nabla f(x_k)) - \nabla^2 f(x_k)\right) \nabla^2 f(x_k)^{-1}\nabla f(x_k) \right\|_2 \\
&\leq \int_0^1 \left\| \left(\nabla^2 f(x - t\nabla^2 f(x_k)^{-1}\nabla f(x_k)) - \nabla^2 f(x_k)\right) \nabla^2 f(x_k)^{-1}\nabla f(x_k) \right\|_2 \\
&\leq L \int_0^1 t \left\| \nabla^2 f(x_k)^{-1}\nabla f(x_k) \right\| \|\nabla^2 f(x_k)^{-1}\| \|\nabla f(x_k)\|_2 \\
&\leq \frac{L}{2m^2}\|\nabla f(x_k)\|_2^2 \leq \|\nabla f(x_k)\|_2.
\end{aligned}$$

Moreover, the above inequality also shows that:

$$\|\nabla f(x_k)\|_2 = O\left(c^{2^k}\right), 0 \leq c < 1,$$

which is known as quadratic convergence speed of the norm of the gradient. This is much faster than linear convergence rate, which is read as: $\|\nabla f(x_k)\|_2 = O(c^k), c \in [0, 1)$. Finally, to get the quadratic convergence for the distance $\|x_k - x^\star\|$, we have:

$$m\|x_k - x^\star\|_2^2 \leq \langle \nabla f(x_k) - \nabla f(x^\star), x_k - x^\star \rangle \leq \|\nabla f(x_k)\| \|x_k - x^\star\|.$$

Hence,

$$\|\nabla f(x_k)\| \geq m\|x_k - x^\star\|_2.$$

Note that when one enters the quadratically convergent phase, to reach an error ϵ, one needs roughly $\log_2 \log_2 \frac{1}{\epsilon}$. For machine precision error (approximately 10^{-16}), one practically needs at most 6 iterations. Therefore, the number of iterations of Newton method is upper-bounded by:

$$\frac{f(x_0) - f^\star}{\eta} + 6.$$

The Newton method is suitable for small to medium sized problems since it suffers from a couple of problems:

1. Inversing the Hessian is expensive for large-scaled optimization problems since it might cost up to $O(d^3)$ basic operations where d is the problem dimension. Without special structure of the Hessian, this complexity per iteration is prohibitive.

2. If the function f is not strongly convex, the Hessian is not invertible and renders the Newton method useless. There are a couple of workarounds such as:

   (a) We choose the direction p_k as:

   $$p_k \in \operatorname*{argmin}_{p\in\mathbb{R}^d} \|\nabla^2 f(x_k)p - \nabla f(x_k)\|_2.$$

   In case $\nabla^2 f(x_k)$ is invertible, then we recover the Newton method. Otherwise, one needs a use a least square solver to find d_k.

   (b) We add a pertubation to the Hessian matrix and compute the direction as:

   $$p_k = (\nabla^2 f(x_k) + \epsilon \mathbf{I})^{-1} \nabla f(x_k).$$

   where $\epsilon > 0$ is a hyper-parameter of the algorithm.

3. If the function is not (strongly) convex, then the Newton direction is not necessarily a descent direction, i.e.,

   $$\lambda^2(x_k) := \nabla f(x_k)^\top \nabla^2 f(x_k)^{-1} \nabla f(x_k) \geq 0,$$

   if $\nabla f(x_k)$ is an eigenvector of $\nabla^2 f(x_k)$ with a negative eigenvalue. Using a similar model to the Newton method, many authors replaced the exact hessian matrix by a semi-definite approximation $\mathbf{B}_k$. These methods are known as quasi-Newton methods and they update the iteration along the direction d defined as:

   $$p_k \in \operatorname*{argmin}_{p\in\mathbb{R}^d} f(x_k) + \nabla f(x_k)^\top p + \frac{1}{2} p^\top \mathbf{B}_k p.$$

   If one chooses $\mathbf{B}_k = \nabla^2 f(x_k)$, we recover the Newton method.