Newton's and Quasi-Newton Methods

Axel Böhm

December 6, 2021

1. Introduction
2. Newton's method
3. Convergence analysis
4. Quasi-Newton methods

1-dimensional case: Newton-Raphson method

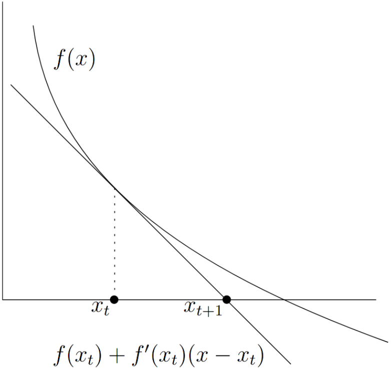
Objective: Find zero of differentiable $f : \mathbb{R} \to \mathbb{R}$.

Strategy: Solve

$$f(x_k) + f'(x_k)(x - x_k) = 0.$$

Method: Gives

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



The Babylonian method

- compute square root of $R \in \mathbb{R}_+$
- find zero of $f(x) = x^2 - R$
- use Newton-Raphson:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - R}{2x_k} = \frac{1}{2}\left(x_k + \frac{R}{x_k}\right)$$

- Starting from $x_0 > 0$ we have

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{R}{x_k}\right) \geq \frac{x_k}{2}.$$

- Starting from $x_0 = R \geq 1$, it takes $\mathcal{O}(\log R)$ steps to get to $x_k - \sqrt{R} < \frac{1}{2}$.

The Babylonian method - Takeoff

Note that

$$x_{k+1} - \sqrt{R} = \frac{1}{2}\left(x_k + \frac{R}{x_k}\right) - \sqrt{R} = \frac{x_k}{2} + \frac{R}{2x_k} - \sqrt{R} = \frac{1}{2x_k}\left(x_k - \sqrt{R}\right)^2$$

For simplicity $R \geq 1/4$, then $x_k \geq \sqrt{R} \geq 1/2$. Hence

$$x_{k+1} - \sqrt{R} = \frac{1}{2x_k}\left(x_k - \sqrt{R}\right)^2 \leq \left(x_k - \sqrt{R}\right)^2$$

If $x_0 - \sqrt{R} < \frac{1}{2}$ (ensured after $\mathcal{O}(\log R)$ steps).

$$x_k - \sqrt{R} \leq \left(x_0 - \sqrt{R}\right)^{2^k} \leq \left(\frac{1}{2}\right)^{2^k}$$

To achieve $x_k - \sqrt{R} < \epsilon$ we only need $k = \log\log(\epsilon^{-1})$ steps!

The Babylonian method - Takeoff

Note that

$$x_{k+1} - \sqrt{R} = \frac{1}{2}\left(x_k + \frac{R}{x_k}\right) - \sqrt{R} = \frac{x_k}{2} + \frac{R}{2x_k} - \sqrt{R} = \frac{1}{2x_k}\left(x_k - \sqrt{R}\right)^2$$

For simplicity $R \geq 1/4$, then $x_k \geq \sqrt{R} \geq 1/2$. Hence

$$x_{k+1} - \sqrt{R} = \frac{1}{2x_k}\left(x_k - \sqrt{R}\right)^2 \leq \left(x_k - \sqrt{R}\right)^2$$

If $x_0 - \sqrt{R} < \frac{1}{2}$ (ensured after $\mathcal{O}(\log R)$ steps).

$$x_k - \sqrt{R} \leq \left(x_0 - \sqrt{R}\right)^{2^k} \leq \left(\frac{1}{2}\right)^{2^k}$$

To achieve $x_k - \sqrt{R} < \epsilon$ we only need $k = \log\log(\epsilon^{-1})$ steps!

The Babylonian method - Example

$R = 1000$, in double arithmetic

- 7 steps to get to $x_7 - \sqrt{1000} < 1/2$
- 3 steps to get to $\sqrt{1000}$ up to *machine precision*
- First phase: $\approx$ one more correct digit per iteration
- Second phase: $\approx$ double the number of correct digits per iteration

In practice: $\log \log x \leq 5$.

Newton's method for optimization

- Goal: Find global minimum x^* of convex, differentiable function f.
- Strategy: Search for zero of derivative.
- **1-dimensional case:** Apply Newton-Raphson method to f':

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

  (requires twice differentiable and $f'' > 0$)
- ***d*-dimensional case:** Newtons methods for minimizing convex $f : \mathbb{R}^d \to \mathbb{R}$:

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

Newton's method as adaptive gradient descent

General update scheme:

$$x_{k+1} = x_k - H(x_k)\nabla f(x_k)$$

for some matrix $H(x) \in \mathbb{R}^{d\times d}$.

- Newton's method: $H = \nabla^2 f(x_k)^{-1}$.
- Gradient descent: $H = \alpha\,\mathrm{Id}$

Newton's methods **adapts** to the local geometry of f at x_k

$\to$ *no need to choose a stepsize.*

Convergence in one step on quadratic functions

A quadratic function

$$f(x) = \frac{1}{2}x^T Mx + q^T x + c$$

is called *nondegenerate* if M is invertible.

- $x^* := M^{-1}q$ is the unique solution of $\nabla f(x) = 0$.
- x^* is the unique global minimum if f is convex.

Lemma (arbitrary x_0)

On nondegenerate quadratic functions, Newtons method yields $x_1 = x^*$.

Proof.

We have $\nabla f(x) = Mx - q$ and $\nabla^2 f(x) = M$. Therefore

$$x_1 = x_0 - \nabla^2 f(x_0)^{-1} \nabla f(x_0) = x_0 - M^{-1}(Mx_0 - q) = M^{-1}q = x^*. \qquad \square$$

Minimizing the second-order Taylor approximation

Alternative interpretation of Newton's method:

Minimize (local) quadratic model of f.

Lemma

*Let f be convex, twice differentiable and $\nabla^2 f(x) \succ 0$. Then x_{k+1} resulting from **Newton's step** satisfies*

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle x - x_k, \nabla^2 f(x_k)(x - x_k) \rangle \right\}$$

Local Convergence

We will prove:

Under suitable conditions on f and close to the minimum Newton's method approximates solution up to an error ϵ in $\log\log(1/\epsilon)$ iterations.

- much faster than anything so far..
- only locally

We call this a local convergence result.

Global convergence statements are more difficult to obtain.

Theorem

Let f be convex with unique global minimum x^*, and X a ball around x^* s.t.

(i) *Bounded inverse Hessians: There exists $\mu > 0$*

$$\|\nabla^2 f(x)^{-1}\| \leq \frac{1}{\mu}, \quad \forall x \in X$$

(ii) *Lipschitz continuous Hessians: There exists $B > 0$*

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq B\|x - y\|, \quad \forall x, y \in X$$

Then, for $x_{k+1} = N_f(x_k)$ we have

$$\|x_{k+1} - x^*\| \leq \frac{B}{2\mu}\|x_k - x^*\|^2.$$

Super-exponential speed

Corollary

In the setting of previous theorem, if

$$\|x_k - x^*\| \leq \frac{\mu}{B},$$

then

$$\|x_k - x^*\| \leq \frac{\mu}{B}\left(\frac{1}{2}\right)^{2^k-1}$$

Close to the global minimum, we will reach distance to the minimum less than ϵ in at most $\log\log(1/\epsilon)$ steps.

As for the last phase of Babylonian method.

Super-exponential speed - intuition

- Almost constant Hessians close to optimality...
- so f behaves almost like a quadratic
- on which Newton's converge in one step

Lemma

If

$$\|x_0 - x^*\| \leq \frac{\mu}{B}$$

the Hessians in Newton's method satisfy the relative error bound

$$\frac{\|\nabla^2 f(x_k) - \nabla^2 f(x^*)\|}{\|\nabla^2 f(x^*)\|} \leq \left(\frac{1}{2}\right)^{2^k - 1}.$$

Proof of convergence theorem

We abbreviate $H = \nabla^2 f(x_k)$, $x = x_k$, $x^+ = x_{k+1}$

$$\begin{aligned} x^+ - x^* &= x - x^* - H^{-1}\nabla f(x) \\ &= x - x^* + H^{-1}(\nabla f(x^*) - \nabla f(x)) \\ &= x - x^* + H^{-1}\int_0^1 H(x + t(x^* - x))(x^* - x)\,\mathrm{d}t, \end{aligned}$$

where we used the fundamental theorem of calculus

$$\int_a^b h'(t)\,\mathrm{d}t$$

with

$$\begin{aligned} h(t) &= \nabla f(x + t(x^* - x)) \\ h'(t) &= \nabla^2 f(x + t(x^* - x))(x^* - x). \end{aligned}$$

Proof of convergence theorem

We abbreviate $H = \nabla^2 f(x_k)$, $x = x_k$, $x^+ = x_{k+1}$

$$
\begin{aligned}
x^+ - x^* &= x - x^* - H^{-1}\nabla f(x) \\
&= x - x^* + H^{-1}(\nabla f(x^*) - \nabla f(x)) \\
&= x - x^* + H^{-1}\int_0^1 H(x + t(x^* - x))(x^* - x)\,\mathrm{d}t,
\end{aligned}
$$

where we used the fundamental theorem of calculus

$$
\int_a^b h'(t)\,\mathrm{d}t
$$

with

$$
\begin{aligned}
h(t) &= \nabla f(x + t(x^* - x)) \\
h'(t) &= \nabla^2 f(x + t(x^* - x))(x^* - x).
\end{aligned}
$$

Proof of convergence theorem II

So far

$$x^+ - x^* = x - x^* + H^{-1} \int_0^1 H(x + t(x^* - x))(x^* - x)\,\mathrm{d}t$$

With

$$x - x^* = H(x)^{-1} \int_0^1 -H(x)(x^* - x)$$

we get

$$x^+ - x^* = H^{-1} \int_0^1 (H(x + t(x^* - x)) - H(x))(x^* - x)\,\mathrm{d}t.$$

Using norms

$$\|x^+ - x^*\| \leq \|H^{-1}\| \left\| \int_0^1 H(x + t(x^* - x)) - H(x)(x^* - x)\,\mathrm{d}t \right\|$$

Proof of convergence theorem III

$$
\begin{aligned}
\|x^+ - x^*\| &= \|H^{-1}\| \left\| \int_0^1 (H(x + t(x^* - x)) - H(x))(x^* - x)\, \mathrm{d}t \right\| \\
&\leq \|H^{-1}\| \|x^* - x\| \int_0^1 \|(H(x + t(x^* - x)) - H(x))\| \, \mathrm{d}t
\end{aligned}
$$

Use **bounded inverse Hessians** and **Lipschitz continuity** of the Hessian to conclude

$$
\begin{aligned}
\|x^+ - x^*\| &\leq \frac{1}{\mu} \|x^* - x\| \int_0^1 B \|t(x^* - x)\| \, \mathrm{d}t \\
&= \frac{B}{\mu} \|x^* - x\|^2 \int_0^1 t \, \mathrm{d}t = \frac{B}{2\mu} \|x - x^*\|^2. \qquad \square
\end{aligned}
$$

Strong convexity $\Rightarrow$ Bounded inverse Hessians

- How to ensure bounded inverse Hessians?

Lemma

Let $f : \mathbb{R}^d \to \mathbb{R}$ be C^2 and ***strongly convex*** *with parameter μ, i.e.*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2, \quad \forall x, y.$$

Then, $\nabla^2 f(x)$ is invertible and $\|\nabla^2 f(x)\|^{-1} \leq 1/\mu$ for all x.

Downside of Newton's method

Computational bottleneck in every step:

- compute Hessian
- invert Hessian or solve $\nabla^2 f(x_k)\Delta x = -\nabla f(x_k)$

Matrix has size $d \times d$, taking $\mathcal{O}(d^3)$ to invert.
In many applications the dimension d is large (**too large** to even store Hessian).

When training a ML model d is the *number or parameters* of our ML model (number of features for linear model).

The secant method

Another iterative method for finding zeros in 1-d. Recall Newton-Raphson:

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

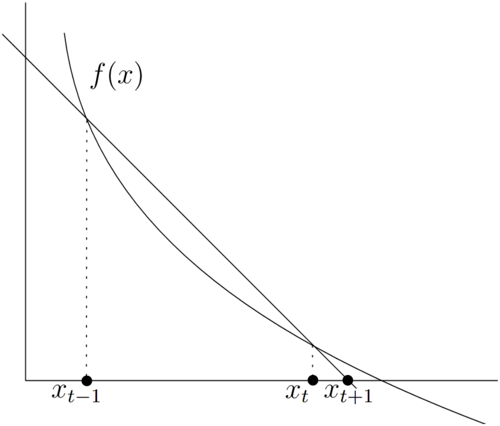
Use finite difference approximation of $f'(x_k)$:

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

We obtain the **secant method**:

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

The secant method II



Constructs the line through $(x_{k-1}, f(x_{k-1}))$ and $(x_k, f(x_k))$.

The secant method III

- is a *derivative-free* version of the Newton-Raphson method.
- For optimization: Apply secant method to f' to optimize f:

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

- yields a second-derivative free version of Newton's method.

What about higher dimensions? Can't divide vectors..

The secant method III

- is a *derivative-free* version of the Newton-Raphson method.
- For optimization: Apply secant method to f' to optimize f:

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

- yields a second-derivative free version of Newton's method.

What about higher dimensions? Can't divide vectors..

The secant condition

In 1-d:

$$H_k := \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}} \approx f''(x_k)$$
$$\Leftrightarrow f'(x_k) - f'(x_{k-1}) = H_k(x_k - x_{k-1}),$$

the secant condition.

- Newton's method: $x_{k+1} = x_k - f''(x_k)^{-1} f'(x_k)$
- Secant method: $x_{k+1} = x_k - {H_k}^{-1} f'(x_k)$

Quasi-Newton methods

$$\nabla f(x_k) - \nabla f(x_{k-1}) = H_k(x_k - x_{k-1}) \approx \nabla^2 f(x_k)(x_k - x_{k-1})$$

We therefore hope that $H_k \approx \nabla^2 f(x_k)$.

$$\text{Secant:} \quad x_{k+1} = x_k = H_k^{-1} \nabla f(x_k)$$

- $d = 1$: unique number H_k satisfying the secant condition
- $d > 0$: secant condition $\nabla f(x_k) - \nabla f(x_{k-1}) = H_k(x_k - x_{k-1})$ has **infinitely** many *symmetric* solutions

> *Any scheme of choosing in each step of the secant method a symmetric H_k that satisfies the secant condition defines a Quasi-Newton method.*

Quasi-Newton methods II

- Newton's method is a Quasi-Newton method $\Leftrightarrow$ f is a nondegenerate **quadratic** function.
- $\Rightarrow$ Quasi-Newton methods *do not generalize* Newton's method but form a family of *related* algorithms.
- First Quasi-Newton method by William Davidon in 1956
- But the paper got rejected for lacking a convergence analysis,
- was finally officially published in 1991
- methods of choice in a number of relevant machine learning applications

Developing a Quasi-Newton method

- want to avoid matrix inversion $\Rightarrow$ directly deal with the inverse H_k^{-1}
- Given: iterates x_{k-1}, x_k and matrix H_{k-1}^{-1}
- Seeking: next matrix H_k^{-1} needed in next Quasi-Newton step

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

- How to choose H_k^{-1}?
- Newton's method: $\nabla^2 f(x_k)$ fluctuates only very little in the region of very fast convergence.
- Makes sense to have $H_k \approx H_{k-1}$ or $H_k^{-1} \approx H_{k-1}^{-1}$

Greenstadt's family of Quasi-Newton methods

Greenstadt [Gre70]: Update

$$H_k^{-1} = H_{k-1}^{-1} + E_k,$$

with E_k an error matrix.

- Try to minimize error subject to H_k satisfying the **secant condition**!

Simple error measure: Frobenius norm

$$\|E\|_F^2 := \sum_{i=1}^{d} \sum_{j=1}^{d} E_{ij}^2$$

BFGS method

- special version of Greenstadt
- is named after Broyden, Fletcher, Goldfarb and Shanno who all came up with it independently around 1970. Greenstadt mostly forgotten.
- Newton's method needs to compute and invert Hessians
  $\Rightarrow$ cost of $\mathcal{O}(d^3)$ per iteration
- Any method in Greenstadt's family avoids computation of Hessian. **Only gradients are needed**.
- In the BFGS method, the cost per iteration drops to $\mathcal{O}(d^2)$.
- even this can be prohibitive $\rightarrow$ **limited memory BFGS**
- uses observation that we do not need H_k^{-1}, only $H_k^{-1}\nabla f(x_k)$