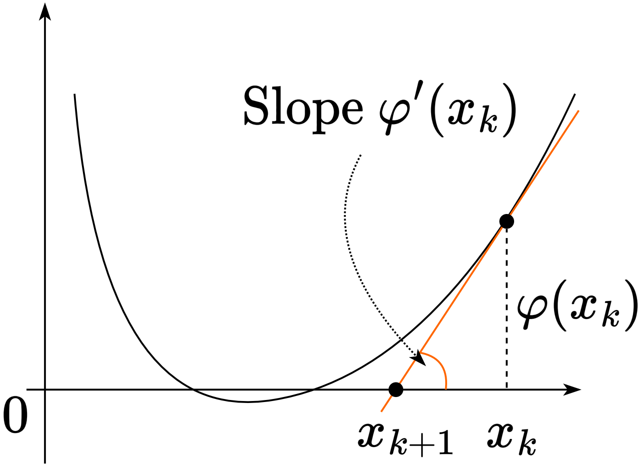
Newton method. Quasi-Newton methods

Seminar

Optimization for ML. Faculty of Computer Science. HSE University

First interpretation of Newton method (solution of linearized equations)



Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$. We want to find the root of $\varphi(x) = 0$.
The whole idea came from building a linear approximation at the point x_k and find its root, which will be the new iteration point:

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

We get an iterative scheme:

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

Now, if we consider $\varphi(x) \equiv \nabla f(x)$, this will become a Newton optimization method:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

Example of Newton linearization method

Question

Apply Newton method to find the root of $\varphi(t) = 0$ and determine the convergence area:

$$\varphi(t) = \frac{t}{\sqrt{1+t^2}}$$

Example of Newton linearization method

Question

Apply Newton method to find the root of $\varphi(t) = 0$ and determine the convergence area:

$$\varphi(t) = \frac{t}{\sqrt{1+t^2}}$$

1. Let's find the derivative:

$$\varphi'(t) = -\frac{t^2}{(1+t^2)^{\frac{3}{2}}} + \frac{1}{\sqrt{1+t^2}}$$

Example of Newton linearization method

Question

Apply Newton method to find the root of $\varphi(t) = 0$ and determine the convergence area:

$$\varphi(t) = \frac{t}{\sqrt{1+t^2}}$$

1. Let's find the derivative:

$$\varphi'(t) = -\frac{t^2}{(1+t^2)^{\frac{3}{2}}} + \frac{1}{\sqrt{1+t^2}}$$

2. Then the iteration of the method takes the form:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)} = x_k - x_k(x_k^2+1) = -x_k^3$$

Example of Newton linearization method

Question

Apply Newton method to find the root of $\varphi(t) = 0$ and determine the convergence area:

$$\varphi(t) = \frac{t}{\sqrt{1+t^2}}$$

1. Let's find the derivative:

$$\varphi'(t) = -\frac{t^2}{(1+t^2)^{\frac{3}{2}}} + \frac{1}{\sqrt{1+t^2}}$$

2. Then the iteration of the method takes the form:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)} = x_k - x_k(x_k^2+1) = -x_k^3$$

It is easy to see that the method converges only if $|x_0| < 1$, emphasizing the **local** nature of the Newton method.

Second interpretation of Newton method (local quadratic Taylor approximation minimizer)

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k\rangle.$$

Second interpretation of Newton method (local quadratic Taylor approximation minimizer)

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}(x)$, i.e. $\nabla f^{II}(x_{k+1}) = 0$.

$$\begin{aligned}
x_{k+1} &= \arg\min_x \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \left\langle \nabla^2 f(x_k)(x-x_k), x-x_k \right\rangle \right\} \\
\nabla f^{II}_{x_k}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k) \\
\left[\nabla^2 f(x_k)\right]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \\
x_{k+1} &= x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k).
\end{aligned}$$

Second interpretation of Newton method (local quadratic Taylor approximation minimizer)

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

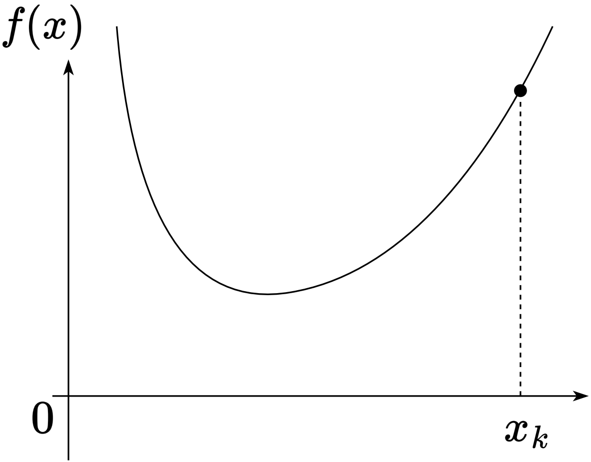
$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}(x)$, i.e. $\nabla f^{II}(x_{k+1}) = 0$.

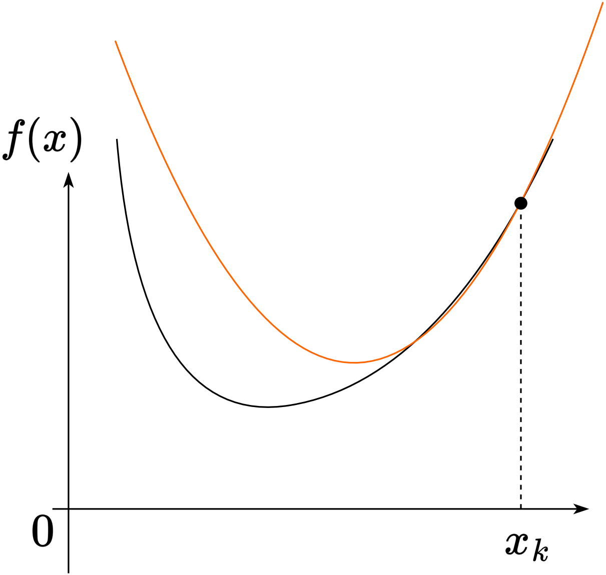
$$\begin{aligned}
x_{k+1} &= \arg\min_x \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \left\langle \nabla^2 f(x_k)(x-x_k), x-x_k \right\rangle \right\} \\
\nabla f^{II}_{x_k}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k) \\
\left[\nabla^2 f(x_k)\right]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \\
x_{k+1} &= x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k).
\end{aligned}$$

Pay attention to the restrictions related to the need for the Hessian to be non-degenerate (for the method to work), as well as for it to be positive definite (for convergence guarantee).

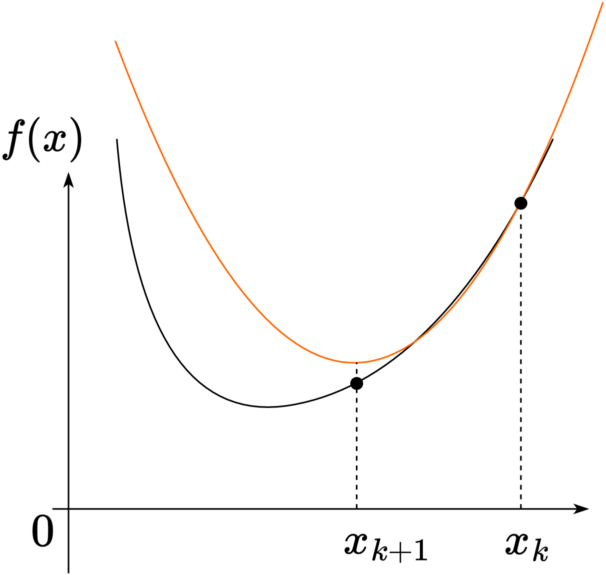
Newton method as a local quadratic Taylor approximation minimizer



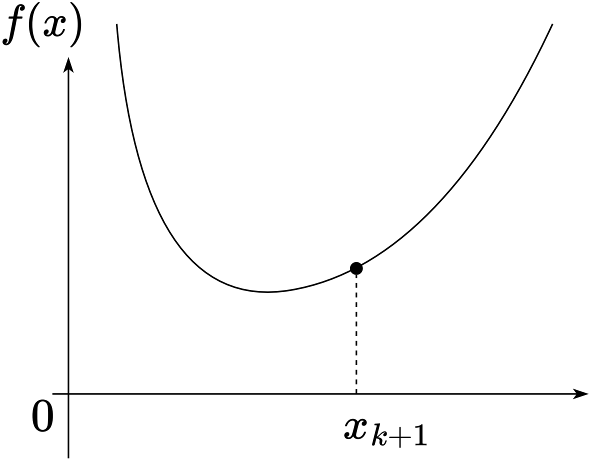
Newton method as a local quadratic Taylor approximation minimizer



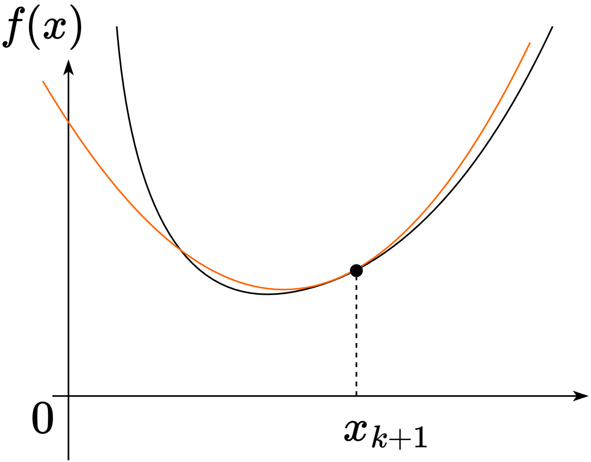
Newton method as a local quadratic Taylor approximation minimizer



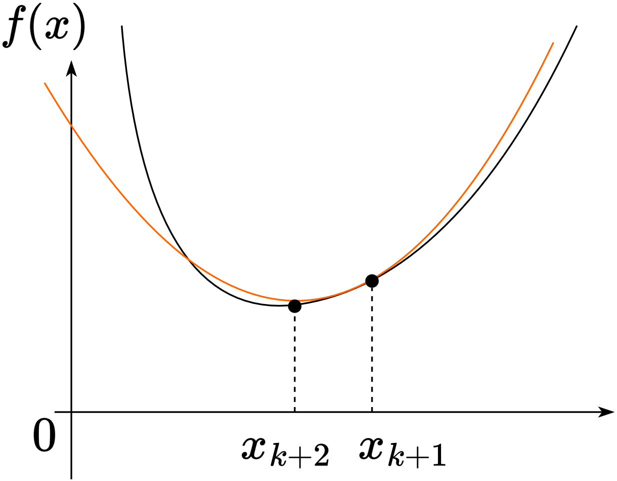
Newton method as a local quadratic Taylor approximation minimizer



Newton method as a local quadratic Taylor approximation minimizer



Newton method as a local quadratic Taylor approximation minimizer



Newton method vs gradient descent

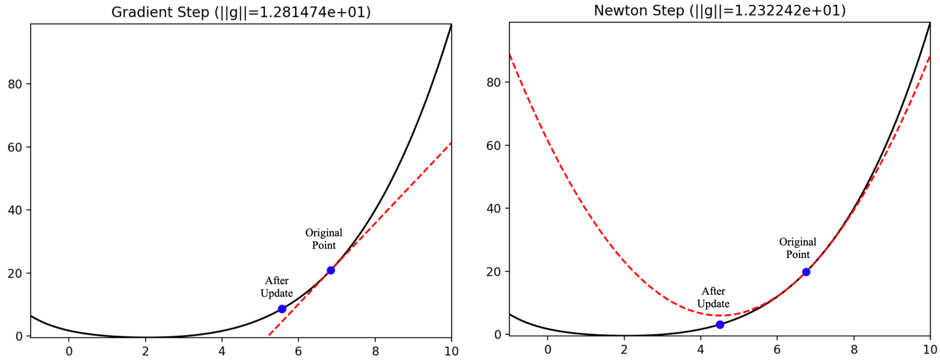


Figure 7: The loss function is depicted in black, the approximation as a dotted red line

The gradient descent $\equiv$ linear approximation

The Newton method $\equiv$ quadratic approximation

Convergence

Theorem

Let $f(x)$ be a strongly convex twice continuously differentiable function at $\mathbb{R}^n$, for the second derivative of which inequalities are executed: $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$. Then Newton method with a constant step

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

locally converges to solving the problem with superlinear speed. If, in addition, Hessian is M-Lipschitz continuous, then this method converges locally to x^* at a quadratic rate:

$$\|x_{k+1} - x^*\|_2 \leq \frac{M \|x_k - x^*\|_2^2}{2 \left(\mu - M \|x_k - x^*\|_2\right)}$$

Convergence

Theorem

Let $f(x)$ be a strongly convex twice continuously differentiable function at $\mathbb{R}^n$, for the second derivative of which inequalities are executed: $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$. Then Newton method with a constant step

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

locally converges to solving the problem with superlinear speed. If, in addition, Hessian is M-Lipschitz continuous, then this method converges locally to x^* at a quadratic rate:

$$\|x_{k+1} - x^*\|_2 \leq \frac{M \|x_k - x^*\|_2^2}{2\left(\mu - M \|x_k - x^*\|_2\right)}$$

"**Converge locally**" means that the convergence rate described above is guaranteed to occur only if the starting point is quite close to the minimum point, in particular $\|x_0 - x^*\| < \frac{2\mu}{3M}$

Affine invariance

Question

Consider a function $f(x)$ and a transformation with an invertible matrix A. Let's figure out how the iteration step of Newton method will change after applying the transformation.

Affine invariance

Question

Consider a function $f(x)$ and a transformation with an invertible matrix A. Let's figure out how the iteration step of Newton method will change after applying the transformation.

1. Let's $x = Ay$ and $g(y) = f(Ay)$.

Affine invariance

Question

Consider a function $f(x)$ and a transformation with an invertible matrix A. Let's figure out how the iteration step of Newton method will change after applying the transformation.

1. Let's $x = Ay$ and $g(y) = f(Ay)$.
2. Consider a quadratic approximation:

$$g(y+u) \approx g(y) + \langle g'(y), u \rangle + \frac{1}{2} u^\top g''(y) u \to \min_u$$

$$u^* = -\left(g''(y)\right)^{-1} g'(y) \quad y_{k+1} = y_k - \left(g''\left(y_k\right)\right)^{-1} g'\left(y_k\right)$$

Affine invariance

> **Question**
>
> Consider a function $f(x)$ and a transformation with an invertible matrix A. Let's figure out how the iteration step of Newton method will change after applying the transformation.

1. Let's $x = Ay$ and $g(y) = f(Ay)$.
2. Consider a quadratic approximation:

$$g(y+u) \approx g(y) + \langle g'(y), u \rangle + \frac{1}{2} u^\top g''(y) u \rightarrow \min_u$$

$$u^* = -\left(g''(y)\right)^{-1} g'(y) \quad y_{k+1} = y_k - \left(g''\left(y_k\right)\right)^{-1} g'\left(y_k\right)$$

3. Substitute explicit expressions for $g''\left(y_k\right), g'\left(y_k\right)$:

$$y_{k+1} = y_k - \left(A^\top f''\left(Ay_k\right) A\right)^{-1} A^\top f'\left(Ay_k\right) = y_k - A^{-1}\left(f''\left(Ay_k\right)\right)^{-1} f'\left(Ay_k\right)$$

Affine invariance

> **Question**
>
> Consider a function $f(x)$ and a transformation with an invertible matrix A. Let's figure out how the iteration step of Newton method will change after applying the transformation.

1. Let's $x = Ay$ and $g(y) = f(Ay)$.
2. Consider a quadratic approximation:

$$g(y+u) \approx g(y) + \left\langle g'(y), u \right\rangle + \frac{1}{2} u^\top g''(y) u \to \min_u$$

$$u^* = -\left(g''(y)\right)^{-1} g'(y) \quad y_{k+1} = y_k - \left(g''\left(y_k\right)\right)^{-1} g'\left(y_k\right)$$

3. Substitute explicit expressions for $g''\left(y_k\right), g'\left(y_k\right)$:

$$y_{k+1} = y_k - \left(A^\top f''\left(Ay_k\right) A\right)^{-1} A^\top f'\left(Ay_k\right) = y_k - A^{-1} \left(f''\left(Ay_k\right)\right)^{-1} f'\left(Ay_k\right)$$

4. Thus, the method's step is transformed by linear transformation in **the same way** as the coordinates:

$$Ay_{k+1} = Ay_k - \left(f''\left(Ay_k\right)\right)^{-1} f'\left(Ay_k\right) \quad x_{k+1} = x_k - \left(f''\left(x_k\right)\right)^{-1} f'\left(x_k\right)$$

Summary of Newton method

Pros

- quadratic convergence near the solution
- high accuracy of the obtained solution
- affine invariance

Summary of Newton method

Pros

- quadratic convergence near the solution
- high accuracy of the obtained solution
- affine invariance

Cons

- no global convergence
- it is necessary to store the hessian on each iteration: $\mathcal{O}(n^2)$ memory
- it is necessary to solve linear systems: $\mathcal{O}(n^3)$ operations
- the Hessian can be degenerate
- the Hessian may not be positively determined $\to$ direction $-(f''(x))^{-1}f'(x)$ may not be a descending direction

Summary of Newton method

Pros

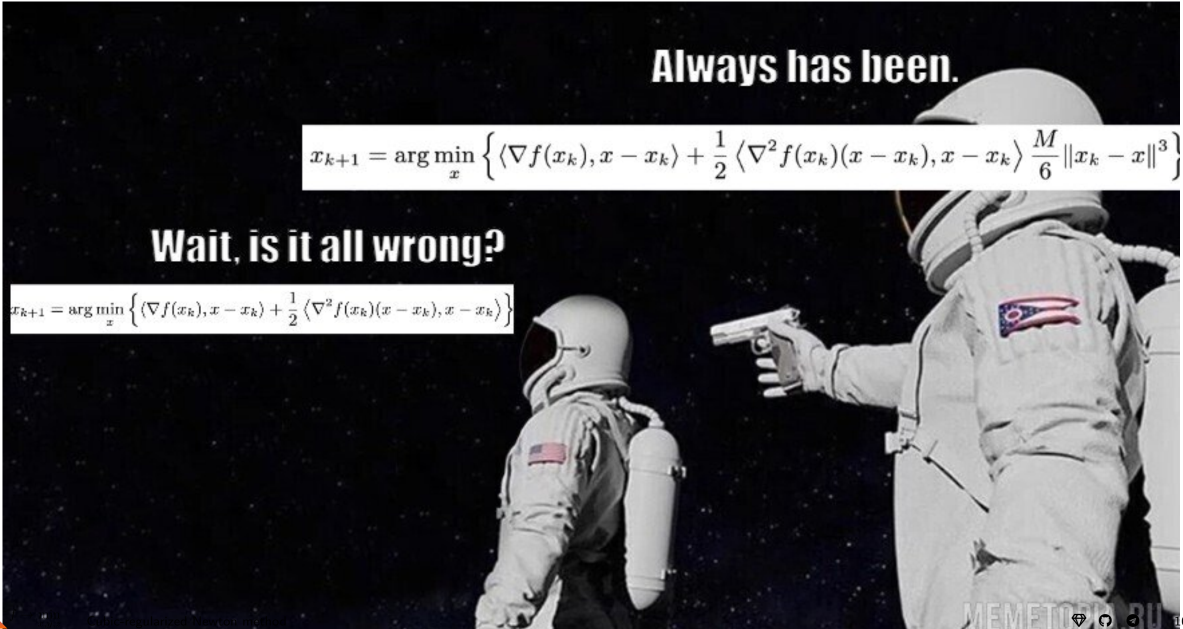
- quadratic convergence near the solution
- high accuracy of the obtained solution
- affine invariance

Cons

- no global convergence
- it is necessary to store the hessian on each iteration: $\mathcal{O}(n^2)$ memory
- it is necessary to solve linear systems: $\mathcal{O}(n^3)$ operations
- the Hessian can be degenerate
- the Hessian may not be positively determined $\to$ direction $-(f''(x))^{-1}f'(x)$ may not be a descending direction

Cubic-regularized Newton method and Quasi Newton methods partially solve these problems!

A bit of base.



Wait, is it all wrong?
$x_{k+1} = \arg\min_x \left\{ \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \left\langle \nabla^2 f(x_k)(x - x_k), x - x_k \right\rangle \right\}$
Always has been.
$x_{k+1} = \arg\min_x \left\{ \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \left\langle \nabla^2 f(x_k)(x - x_k), x - x_k \right\rangle \frac{M}{6} \|x_k - x\|^3 \right\}$

Intuition on how to improve Newton method

Gradient Descent recap

If f has L-Lipschitz gradient, then

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2 .$$

So, each step of gradient descent for function f with L-Lipschitz gradient is a minimization of majorizing paraboloid:

$$\begin{aligned} x_{k+1} &= \arg\min_x \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L}{2} \|x - x_k\|^2 \right\} \\ &= x_k - \frac{1}{L} \nabla f(x_k). \end{aligned}$$

Intuition on how to improve Newton method

Gradient Descent recap

If f has L-Lipschitz gradient, then

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|^2 .$$

So, each step of gradient descent for function f with L-Lipschitz gradient is a minimization of majorizing paraboloid:

$$\begin{aligned} x_{k+1} &= \arg\min_x \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L}{2} \|x - x_k\|^2 \right\} \\ &= x_k - \frac{1}{L} \nabla f(x_k). \end{aligned}$$

But if function f has M-Lipschitz Hessian, it is easy to show that

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \left\langle \nabla^2 f(x)(y - x), y - x \right\rangle + \frac{M}{6} \|y - x\|^3 .$$

What if we use the same logic as in gradient descent for function with M-Lipschitz Hessian?

Cubic-regularized Newton method

If f has M-Lipschitz Hessian, then

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \left\langle \nabla^2 f(x)(y - x), y - x \right\rangle + \frac{M}{6} \|y - x\|^3 .$$

Minimizing the right-hand side of this inequality, we come to Cubic-regularized Newton method

$$x_{k+1} = \arg\min_x \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \left\langle \nabla^2 f(x_k)(x - x_k), x - x_k \right\rangle + \frac{M}{6} \|x - x_k\|^3 \right\} . \tag{1}$$

Question

What problems do you see in (1)?

Cubic-regularized Newton method

If f has M-Lipschitz Hessian, then

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \left\langle \nabla^2 f(x)(y-x), y - x \right\rangle + \frac{M}{6} \|y - x\|^3 .$$

Minimizing the right-hand side of this inequality, we come to Cubic-regularized Newton method

$$x_{k+1} = \arg\min_x \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \left\langle \nabla^2 f(x_k)(x - x_k), x - x_k \right\rangle + \frac{M}{6} \|x - x_k\|^3 \right\} . \tag{1}$$

❗ Challenges

1. We can't get explicit expression for x_{k+1} (without argmin) from (1) as we could in gradient descent.
2. The subproblem inside (1) can be non-convex.

Cubic-regularized Newton method

If f has M-Lipschitz Hessian, then

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \left\langle \nabla^2 f(x)(y-x), y-x \right\rangle + \frac{M}{6} \|y - x\|^3.$$

Minimizing the right-hand side of this inequality, we come to Cubic-regularized Newton method

$$x_{k+1} = \arg\min_x \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \left\langle \nabla^2 f(x_k)(x - x_k), x - x_k \right\rangle + \frac{M}{6} \|x - x_k\|^3 \right\}. \tag{1}$$

! Challenges

1. We can't get explicit expression for x_{k+1} (without argmin) from (1) as we could in gradient descent.
2. The subproblem inside (1) can be non-convex.

Solutions

1. We can use numerical methods with fast convergence
2. The subproblem is equivalent to a convex one-dimensional optimization problem. [a]
3. The subproblem can be made convex with proper regularization coefficient. [b]

[a] Nesterov, Y. (2018). Lectures on convex optimization. Springer.

[b] Nesterov, Y. (2021). Implementable tensor methods in unconstrained convex optimization. Mathematical Programming.

Convergence [1]

Theorem

Let $f(x)$ be μ-strongly convex function with M-Lipschitz Hessian. Then, Cubic-regularized Newton Method (1) converges globally superlinearly as

$$f(x_{k+1}) - f^* \leq \gamma_k (f(x_k) - f^*), \;\; \gamma_k \to 0.$$

[1] Kamzolov, D., et al. (2024). Optami: Global superlinear convergence of high-order methods. Accepted to ICLR 2025.

Quasi-Newton methods intuition

For the classic task of unconditional optimization $f(x) \to \min\limits_{x \in \mathbb{R}^n}$ the general scheme of iteration method is written as:

$$
x_{k+1} = x_k + \alpha_k d_k
$$

Quasi-Newton methods intuition

For the classic task of unconditional optimization $f(x) \to \min\limits_{x \in \mathbb{R}^n}$ the general scheme of iteration method is written as:

$$x_{k+1} = x_k + \alpha_k d_k$$

In the Newton method, the d_k direction (Newton direction) is set by the linear system solution at each step:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

Quasi-Newton methods intuition

For the classic task of unconditional optimization $f(x) \to \min_{x \in \mathbb{R}^n}$ the general scheme of iteration method is written as:

$$x_{k+1} = x_k + \alpha_k d_k$$

In the Newton method, the d_k direction (Newton direction) is set by the linear system solution at each step:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

i.e. at each iteration it is necessary to **compute** hessian and gradient and **solve** linear system.

Quasi-Newton methods intuition

For the classic task of unconditional optimization $f(x) \to \min\limits_{x \in \mathbb{R}^n}$ the general scheme of iteration method is written as:

$$
x_{k+1} = x_k + \alpha_k d_k
$$

In the Newton method, the d_k direction (Newton direction) is set by the linear system solution at each step:

$$
B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)
$$

i.e. at each iteration it is necessary to **compute** hessian and gradient and **solve** linear system.

Note here that if we take a single matrix of $B_k = I_n$ as B_k at each step, we will exactly get the gradient descent method.

The general scheme of quasi-Newton methods is based on the selection of the B_k matrix so that it tends in some sense at $k \to \infty$ to the truth value of the Hessian $\nabla^2 f(x_k)$.

Quasi-Newton method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \ldots$, repeat:

1. Find $d_k : B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Quasi-Newton method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \dots$, repeat:

1. Find $d_k : B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Quasi-Newton method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \dots$, repeat:

1. Find $d_k : B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Basic Idea: As B_k already contains information about the Hessian, use a suitable matrix update to form B_{k+1}.

Quasi-Newton method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \dots$, repeat:

1. Find $d_k : B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Basic Idea: As B_k already contains information about the Hessian, use a suitable matrix update to form B_{k+1}.

Reasonable Requirement for B_{k+1} (motivated by the secant method):

$$\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1}(x_{k+1} - x_k) = B_{k+1} d_k$$
$$\Delta y_k = B_{k+1} d_k$$

Quasi-Newton method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \dots$, repeat:

1. Find $d_k : B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Basic Idea: As B_k already contains information about the Hessian, use a suitable matrix update to form B_{k+1}.

Reasonable Requirement for B_{k+1} (motivated by the secant method):

$$\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1}(x_{k+1} - x_k) = B_{k+1} d_k$$
$$\Delta y_k = B_{k+1} d_k$$

In addition to the secant equation, we want:

- B_{k+1} to be symmetric
- B_{k+1} to be "close" to B_k
- $B_k \succ 0 \Rightarrow B_{k+1} \succ 0$

Problem 1: Symmetric Rank-One (SR1) update

Let's try an update with rank-one matrix:

$$B_{k+1} = B_k + auu^T$$

Question

What a and u can we choose? How the update of the B_{k+1} would look like?

Problem 1: Symmetric Rank-One (SR1) update

Let's try an update with rank-one matrix:

$$B_{k+1} = B_k + auu^T$$

> **Question**
>
> What a and u can we choose? How the update of the B_{k+1} would look like?

SR1 convergence

$$
B_{k+1} = B_k + \frac{(\Delta y_k - B_k d_k)(\Delta y_k - B_k d_k)^T}{(\Delta y_k - B_k d_k)^T d_k}
$$

called the symmetric rank-one (SR1) update or Broyden method.

Theorem

Let

- f be twice continuously differentiable, has unique stationary point x^*,
- $0 \succ \nabla^2 f(x^2)$, $\nabla^2 f(x)$ is Lipschitz continuous in a neighborhood x^*,
- the sequence of matrices $\{B_k\}$ is bounded in norm,
- $|(\Delta y_k - B_k d_k)^T d_k| \geq r \|d_k\| \|\Delta y_k - B_k d_k\|$, $0 < r \ll 1$.

Then in SR1 $x_k \to x^*$ superlinearly.

SR1 with inverse update

How can we solve

$$B_{k+1} d_{k+1} = -\nabla f(x_{k+1}),$$

in order to take the next step? In addition to propagating B_k to B_{k+1}, let's propagate inverses, i.e., $C_k = B_k^{-1}$ to $C_{k+1} = (B_{k+1})^{-1}$.

Sherman-Morrison Formula:

The Sherman-Morrison formula states:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}$$

SR1 with inverse update

How can we solve

$$B_{k+1} d_{k+1} = -\nabla f(x_{k+1}),$$

in order to take the next step? In addition to propagating B_k to B_{k+1}, let's propagate inverses, i.e., $C_k = B_k^{-1}$ to $C_{k+1} = (B_{k+1})^{-1}$.

Sherman-Morrison Formula:

The Sherman-Morrison formula states:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}$$

Thus, for the SR1 update, the inverse is also easily updated:

$$C_{k+1} = C_k + \frac{(d_k - C_k \Delta y_k)(d_k - C_k \Delta y_k)^T}{(d_k - C_k \Delta y_k)^T \Delta y_k}$$

In general, SR1 is simple and cheap, but it has a key drawback: it does not preserve positive definiteness.

Problem 2: Broyden-Fletcher-Goldfarb-Shanno (BFGS) update

Let's now try a rank-two update:

$$B_{k+1} = B_k + auu^T + bvv^T.$$

Question

What a, u, b and v can we choose? How the update of the B_{k+1} would look like?

BFGS convergence

$$B_{k+1} = B_k - \frac{B_k d_k d_k^T B_k}{d_k^T B_k d_k} + \frac{\Delta y_k \Delta y_k^T}{d_k^T \Delta y_k}$$

called the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update.

Theorem

Let $f(x)$ be twice continuously differentiable, have Lipschitz Hessian at x^* and additionally $\sum_{k=1}^{\infty} \|x_k - x^*\| \leq \infty$. Then in BFGS $x_k \to x^*$ superlinearly.

BFGS update with inverse

Woodbury Formula

The Woodbury formula, a generalization of the Sherman-Morrison formula, is given by:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

BFGS update with inverse

Woodbury Formula

The Woodbury formula, a generalization of the Sherman-Morrison formula, is given by:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Applied to our case, we get a rank-two update on the inverse C:

$$C_{k+1} = C_k + \frac{(d_k - C_k\Delta y_k)d_k^T}{\Delta y_k^T d_k} + \frac{d_k(d_k - C_k\Delta y_k)^T}{\Delta y_k^T d_k} - \frac{(d_k - C_k\Delta y_k)^T\Delta y_k}{(\Delta y_k^T d_k)^2} d_k d_k^T$$

$$C_{k+1} = \left(I - \frac{d_k \Delta y_k^T}{\Delta y_k^T d_k}\right) C_k \left(I - \frac{\Delta y_k d_k^T}{\Delta y_k^T d_k}\right) + \frac{d_k d_k^T}{\Delta y_k^T d_k}$$

This formulation ensures that the BFGS update, while comprehensive, remains computationally efficient, requiring $O(n^2)$ operations. Importantly, BFGS update preserves positive definiteness. Recall this means $B_k \succ 0 \Rightarrow B_{k+1} \succ 0$. Equivalently, $C_k \succ 0 \Rightarrow C_{k+1} \succ 0$

L-BFGS main idea

- L-BFGS does not store full matrix B_k (C_k), instead it stores two sequences of vectors of length $m : m < n$
- memory reduces from $O(n^2)$ to $O(mn)$, making it more sutable for high-dimensional problems

Computational experiments

- Computation experiments for Quasi-Newtom, CG and GD
- Computational experiments for Newton and Quasi Newton methods .