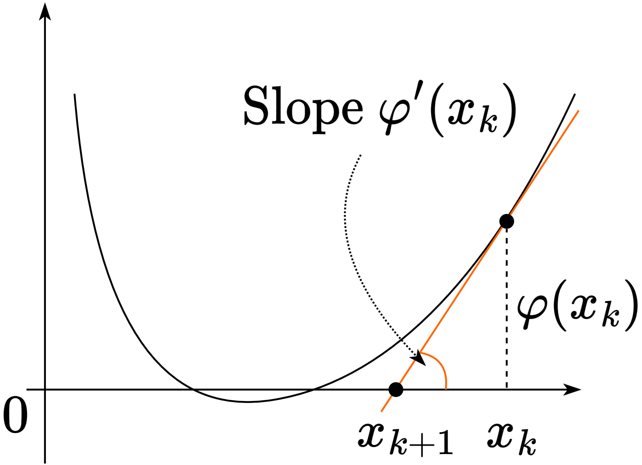
Newton method. Quasi-Newton methods. K-FAC

Daniil Merkulov

Optimization methods. MIPT

Idea of Newton method of root finding



Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$.
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:

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

Which will become a Newton optimization method in case $f'(x) = \varphi(x)$[a]:

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

[a] Literally we aim to solve the problem of finding stationary points $\nabla f(x) = 0$

Newton method as a 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:

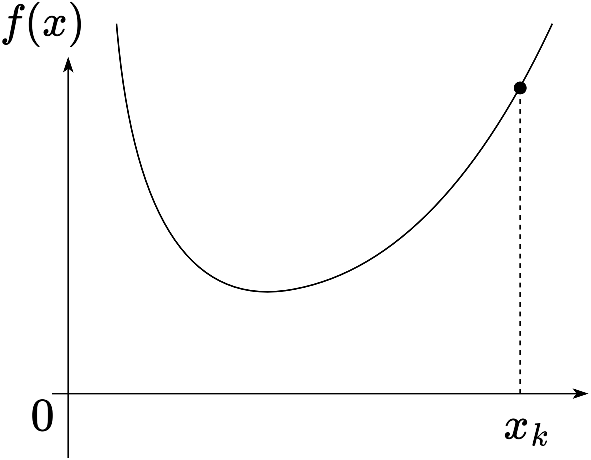
$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

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

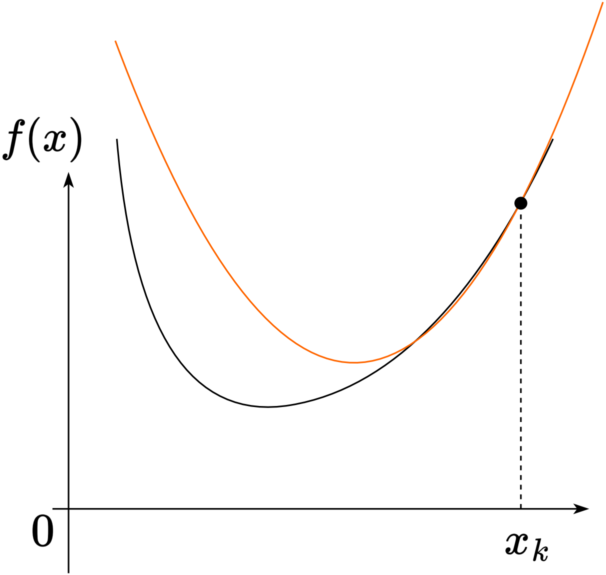
$$\begin{aligned}
\nabla f^{II}_{x_k}(x_{k+1}) &= \nabla f(x_{k}) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_{k}) \\
\left[\nabla^2 f(x_k)\right]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_{k}) \\
x_{k+1} &= x_k -\left[\nabla^2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

Let us immediately note the limitations related to the necessity of the Hessian's non-degeneracy (for the method to exist), as well as its positive definiteness (for the convergence guarantee).

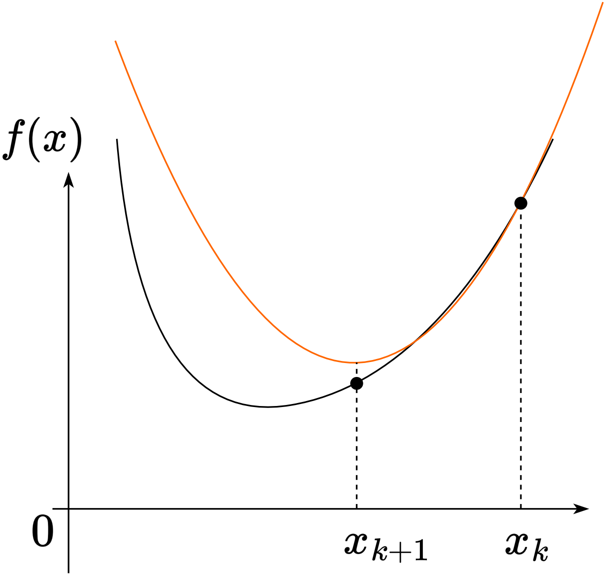
Newton method as a local quadratic Taylor approximation minimizer



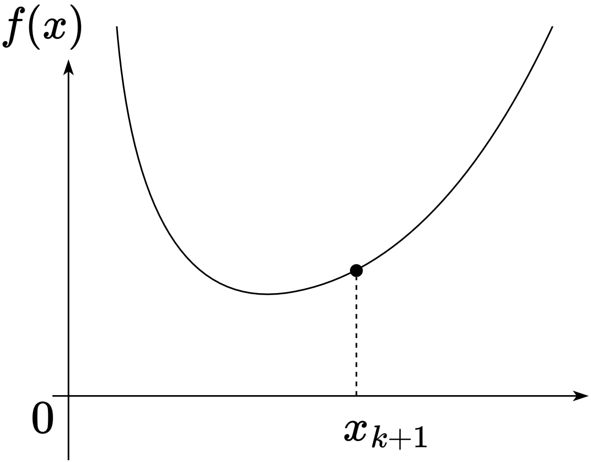
Newton method as a local quadratic Taylor approximation minimizer



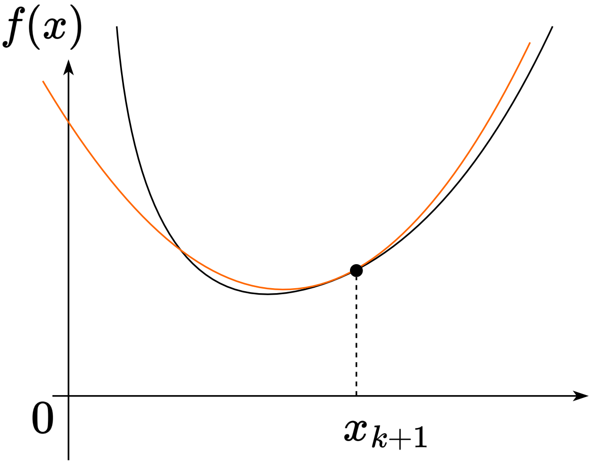
Newton method as a local quadratic Taylor approximation minimizer



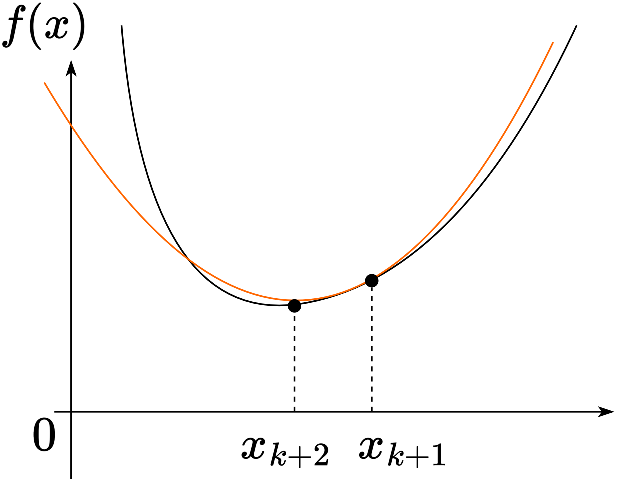
Newton method as a local quadratic Taylor approximation minimizer



Newton method as a local quadratic Taylor approximation minimizer



Newton method as a local quadratic Taylor approximation minimizer



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's method with a constant step 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.

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's method with a constant step 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.

Proof

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's method with a constant step 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.

Proof

1. We will use Newton-Leibniz formula

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau$$

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's method with a constant step 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.

Proof

1. We will use Newton-Leibniz formula

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau$$

2. Then we track the distance to the solution

$$x_{k+1} - x^* = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) - x^* = x_k - x^* - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) =$$
$$= x_k - x^* - \left[\nabla^2 f(x_k)\right]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau$$

Convergence

3.

$$
\begin{aligned}
&= \left(I - \left[\nabla^2 f(x_k)\right]^{-1} \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))d\tau\right)(x_k - x^*) = \\
&= \left[\nabla^2 f(x_k)\right]^{-1} \left(\nabla^2 f(x_k) - \int_0^1 \nabla^2 f(x^* + \tau(x_k - x^*))d\tau\right)(x_k - x^*) = \\
&= \left[\nabla^2 f(x_k)\right]^{-1} \left(\int_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*))d\tau\right)\right)(x_k - x^*) = \\
&= \left[\nabla^2 f(x_k)\right]^{-1} G_k(x_k - x^*)
\end{aligned}
$$

Convergence

3.

$$
\begin{aligned}
&= \left(I - \left[\nabla^2 f(x_k) \right]^{-1} \int_0^1 \nabla^2 f(x^* + \tau (x_k - x^*)) d\tau \right) (x_k - x^*) = \\
&= \left[\nabla^2 f(x_k) \right]^{-1} \left(\nabla^2 f(x_k) - \int_0^1 \nabla^2 f(x^* + \tau (x_k - x^*)) d\tau \right) (x_k - x^*) = \\
&= \left[\nabla^2 f(x_k) \right]^{-1} \left(\int_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau (x_k - x^*)) d\tau \right) \right) (x_k - x^*) = \\
&= \left[\nabla^2 f(x_k) \right]^{-1} G_k (x_k - x^*)
\end{aligned}
$$

4. We have introduced:

$$
G_k = \int_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau (x_k - x^*)) d\tau \right).
$$

Convergence

5. Let's try to estimate the size of G_k:

$$\|G_k\| = \left\| \int_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) \right\| \leq$$

$$\leq \int_0^1 \left\| \nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) \right\| d\tau \leq \qquad \text{(Hessian's Lipschitz continuity)}$$

$$\leq \int_0^1 M\|x_k - x^* - \tau(x_k - x^*)\| d\tau = \int_0^1 M\|x_k - x^*\|(1 - \tau) d\tau = \frac{r_k}{2} M,$$

where $r_k = \|x_k - x^*\|$.

Convergence

5. Let's try to estimate the size of G_k:

$$\|G_k\| = \left\| \int_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) d\tau \right) \right\| \leq$$

$$\leq \int_0^1 \left\| \nabla^2 f(x_k) - \nabla^2 f(x^* + \tau(x_k - x^*)) \right\| d\tau \leq \qquad \text{(Hessian's Lipschitz continuity)}$$

$$\leq \int_0^1 M \|x_k - x^* - \tau(x_k - x^*)\| d\tau = \int_0^1 M \|x_k - x^*\| (1 - \tau) d\tau = \frac{r_k}{2} M,$$

where $r_k = \|x_k - x^*\|$.

6. So, we have:

$$r_{k+1} \leq \left\| \left[\nabla^2 f(x_k) \right]^{-1} \right\| \cdot \frac{r_k}{2} M \cdot r_k$$

and we need to bound the norm of the inverse hessian

Convergence

7. Because of Hessian's Lipschitz continuity and symmetry:

$$
\begin{aligned}
\nabla^2 f(x_k) - \nabla^2 f(x^*) &\succeq -Mr_k I_n \\
\nabla^2 f(x_k) &\succeq \nabla^2 f(x^*) - Mr_k I_n \\
\nabla^2 f(x_k) &\succeq \mu I_n - Mr_k I_n \\
\nabla^2 f(x_k) &\succeq (\mu - Mr_k) I_n
\end{aligned}
$$

Convexity implies $\nabla^2 f(x_k) \succ 0$, i.e. $r_k < \frac{\mu}{M}$.

$$
\begin{aligned}
\left\| \left[\nabla^2 f(x_k) \right]^{-1} \right\| &\leq (\mu - Mr_k)^{-1} \\
r_{k+1} &\leq \frac{r_k^2 M}{2(\mu - Mr_k)}
\end{aligned}
$$

Convergence

7. Because of Hessian's Lipschitz continuity and symmetry:

$$
\begin{aligned}
\nabla^2 f(x_k) - \nabla^2 f(x^*) &\succeq -Mr_k I_n \\
\nabla^2 f(x_k) &\succeq \nabla^2 f(x^*) - Mr_k I_n \\
\nabla^2 f(x_k) &\succeq \mu I_n - Mr_k I_n \\
\nabla^2 f(x_k) &\succeq (\mu - Mr_k) I_n
\end{aligned}
$$

   Convexity implies $\nabla^2 f(x_k) \succ 0$, i.e. $r_k < \frac{\mu}{M}$.

$$
\begin{aligned}
\left\| \left[\nabla^2 f(x_k) \right]^{-1} \right\| &\leq (\mu - Mr_k)^{-1} \\
r_{k+1} &\leq \frac{r_k^2 M}{2(\mu - Mr_k)}
\end{aligned}
$$

8. The convergence condition $r_{k+1} < r_k$ imposes additional conditions on r_k : $\quad r_k < \frac{2\mu}{3M}$

   Thus, we have an important result: Newton's method for the function with Lipschitz positive-definite Hessian converges **quadratically** near $(\|x_0 - x^*\| < \frac{2\mu}{3M})$ to the solution.

Summary

What's nice:

- quadratic convergence near the solution x^*

What's not nice:

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance

What's not nice:

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration: $\mathcal{O}(n^2)$ memory

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration: $\mathcal{O}(n^2)$ memory
- it is necessary to solve linear systems: $\mathcal{O}(n^3)$ operations

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) 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 at x^*

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) 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 at x^*
- the hessian may not be positively determined $\to$ direction $-(f''(x))^{-1}f'(x)$ may not be a descending direction

Newton method problems

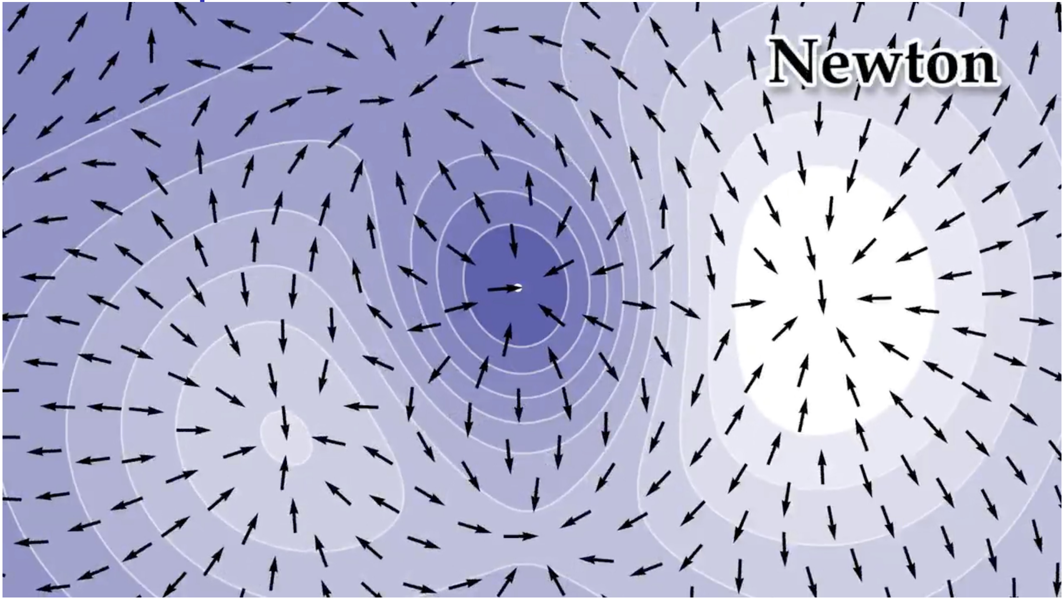


Figure 7: Illustration

Newton method problems

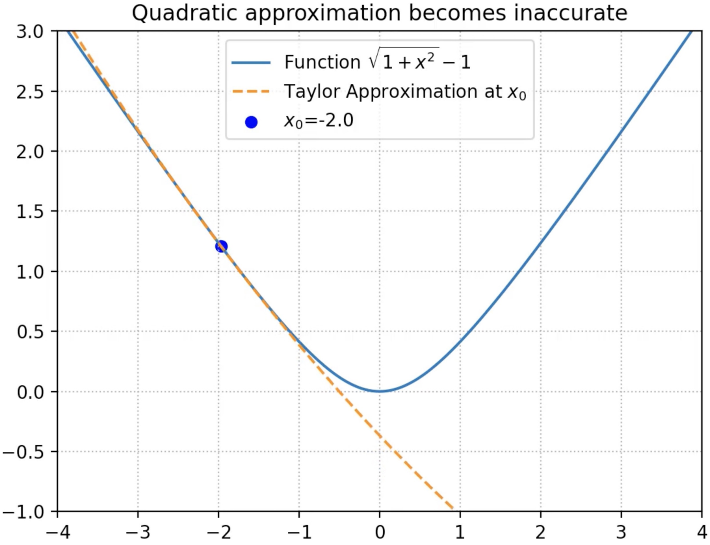


Figure 8: Illustration

The idea of adapive metrics

Given $f(x)$ and a point x_0. Define $B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : d(x, x_0) = \varepsilon^2\}$ as the set of points with distance ε to x_0. Here we presume the existence of a distance function $d(x, x_0)$.

$$
x^* = \text{arg} \min_{x \in B_\varepsilon(x_0)} f(x)
$$

Then, we can define another *steepest descent* direction in terms of minimizer of function on a sphere:

$$
s = \lim_{\varepsilon \to 0} \frac{x^* - x_0}{\varepsilon}
$$

Let us assume that the distance is defined locally by some metric A:

$$
d(x, x_0) = (x - x_0)^\top A (x - x_0)
$$

Let us also consider first order Taylor approximation of a function $f(x)$ near the point x_0:

$$
f(x_0 + \delta x) \approx f(x_0) + \nabla f(x_0)^\top \delta x \tag{1}
$$

Now we can explicitly pose a problem of finding s, as it was stated above.

$$
\begin{aligned}
&\min_{\delta x \in \mathbb{R}^{\ltimes}} f(x_0 + \delta x) \\
&\text{s.t. } \delta x^\top A \delta x = \varepsilon^2
\end{aligned}
$$

Using equation (1 it can be written as:

$$
\begin{aligned}
&\min_{\delta x \in \mathbb{R}^{\ltimes}} \nabla f(x_0)^\top \delta x \\
&\text{s.t. } \delta x^\top A \delta x = \varepsilon^2
\end{aligned}
$$

Using Lagrange multipliers method, we can easily conclude, that the answer is:

$$
\delta x = - \frac{2 \varepsilon^2}{\nabla f(x_0)^\top A^{-1} \nabla f(x_0)} A^{-1} \nabla f
$$

Which means, that new direction of steepest descent is nothing else, but $A^{-1} \nabla f(x_0)$.
Indeed, if the space is isotropic and $A = I$, we immediately have gradient descent formula, while Newton method uses local Hessian as a metric matrix.

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:

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

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

$$s_k = -B_k \nabla f(x_k), \quad B_k = f_{xx}^{-1}(x_k)$$

i.e. at each iteration it is necessary to **compensate** hessian and gradient and **resolve** 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 true value of inverted Hessian in the local optimum $f_{xx}^{-1}(x_*)$. Let's consider several schemes using iterative updating of B_k matrix in the following way:

$$B_{k+1} = B_k + \Delta B_k$$

Then if we use Taylor's approximation for the first order gradient, we get it:

$$\nabla f(x_k) - \nabla f(x_{k+1}) \approx f_{xx}(x_{k+1})(x_k - x_{k+1}).$$

Quasi-Newton method

Now let's formulate our method as:

$$\Delta x_k = B_{k+1} \Delta y_k, \text{ where } \quad \Delta y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

in case you set the task of finding an update ΔB_k:

$$\Delta B_k \Delta y_k = \Delta x_k - B_k \Delta y_k$$

Broyden method

The simplest option is when the amendment ΔB_k has a rank equal to one. Then you can look for an amendment in the form

$$\Delta B_k = \mu_k q_k q_k^\top.$$

where μ_k is a scalar and q_k is a non-zero vector. Then mark the right side of the equation to find ΔB_k for Δz_k:

$$\Delta z_k = \Delta x_k - B_k \Delta y_k$$

We get it:

$$\mu_k q_k q_k^\top \Delta y_k = \Delta z_k$$
$$\left(\mu_k \cdot q_k^\top \Delta y_k\right) q_k = \Delta z_k$$

A possible solution is: $q_k = \Delta z_k$, $\mu_k = \left(q_k^\top \Delta y_k\right)^{-1}$.

Then an iterative amendment to Hessian's evaluation at each iteration:

$$\Delta B_k = \frac{(\Delta x_k - B_k \Delta y_k)(\Delta x_k - B_k \Delta y_k)^\top}{\langle \Delta x_k - B_k \Delta y_k, \Delta y_k \rangle}.$$

Davidon–Fletcher–Powell method

$$\Delta B_k = \mu_1 \Delta x_k (\Delta x_k)^\top + \mu_2 B_k \Delta y_k (B_k \Delta y_k)^\top.$$

$$\Delta B_k = \frac{(\Delta x_k)(\Delta x_k)^\top}{\langle \Delta x_k, \Delta y_k \rangle} - \frac{(B_k \Delta y_k)(B_k \Delta y_k)^\top}{\langle B_k \Delta y_k, \Delta y_k \rangle}.$$

Broyden–Fletcher–Goldfarb–Shanno method

$$\Delta B_k = QUQ^\top, \quad Q = [q_1, q_2], \quad q_1, q_2 \in \mathbb{R}^n, \quad U = \begin{pmatrix} a & c \\ c & b \end{pmatrix}.$$

$$\Delta B_k = \frac{(\Delta x_k)(\Delta x_k)^\top}{\langle \Delta x_k, \Delta y_k \rangle} - \frac{(B_k \Delta y_k)(B_k \Delta y_k)^\top}{\langle B_k \Delta y_k, \Delta y_k \rangle} + p_k p_k^\top.$$

Code

- Open In Colab

Code

- Open In Colab
- Comparison of quasi Newton methods

Natural Gradient Descent

K-FAC