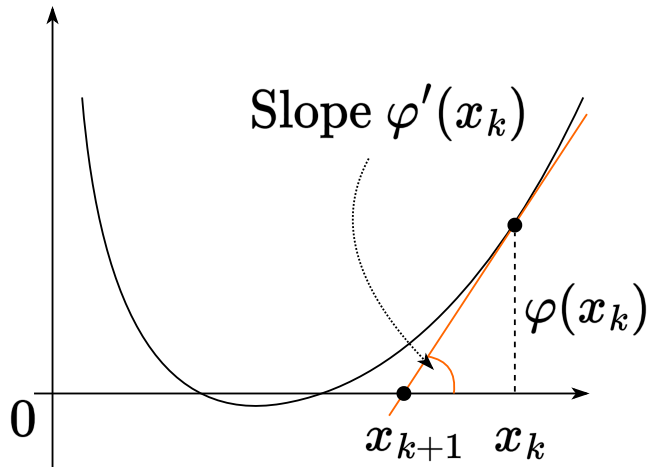


# Newton method. Quasi-Newton methods

## Seminar

Optimization for ML. Faculty of Computer Science. HSE University

## Idea of Newton method of root finding



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

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  $f'(x) = \varphi(x)^a$ :

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

---

<sup>a</sup>Literally we aim to solve the problem of finding stationary points  $\nabla f(x) = 0$

# Idea of Newton method of root findin

## i Question

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

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

# Idea of Newton method of root findin

## i Question

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

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

1. Let's find the derivative:

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

# Idea of Newton method of root findin

## i Question

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

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

1. Let's find the derivative:

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

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

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

# Idea of Newton method of root finding

## i Question

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

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

1. Let's find the derivative:

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

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.

## 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  $x_k$ :

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

# 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  $x_k$ :

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

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

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

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

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

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



# 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  $x_k$ :

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

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

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

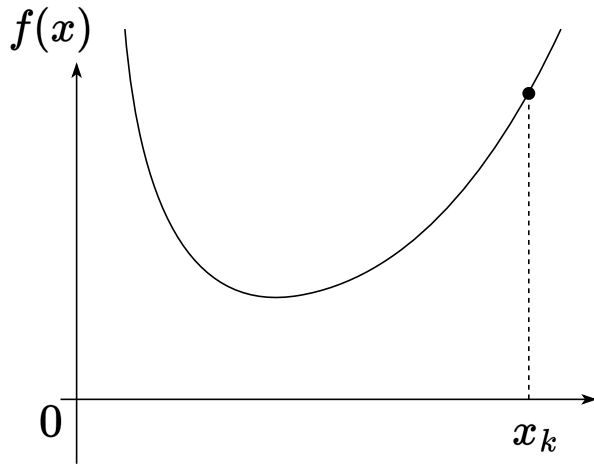
$$\nabla^2 f(x_k)(x_{k+1} - x_k) = -\nabla f(x_k)$$

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

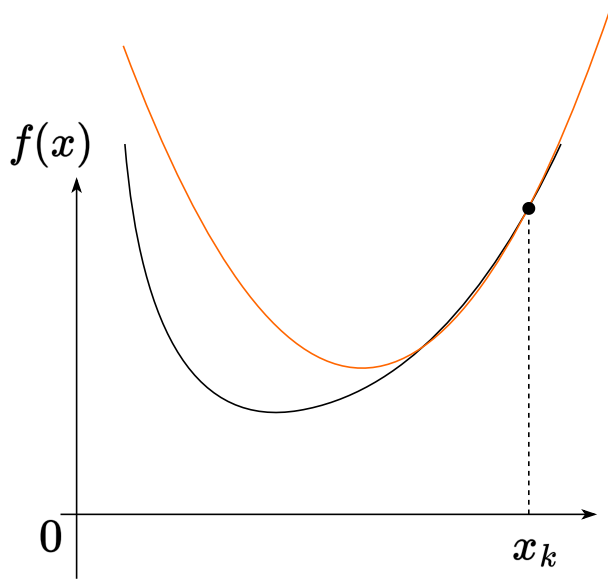
$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k).$$

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).

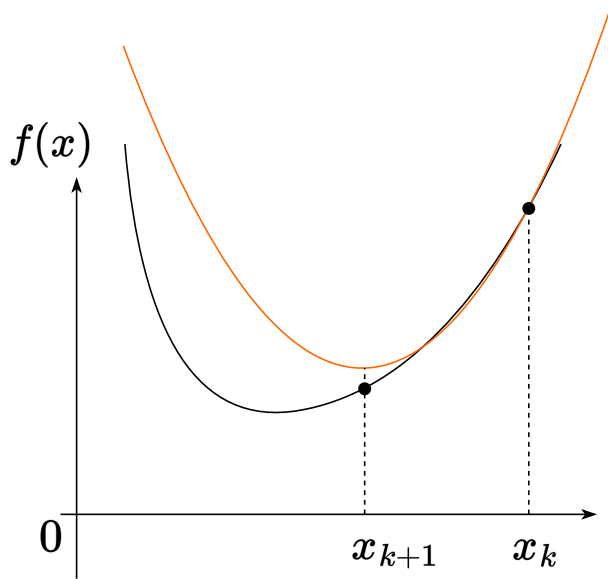
## Newton method as a local quadratic Taylor approximation minimizer



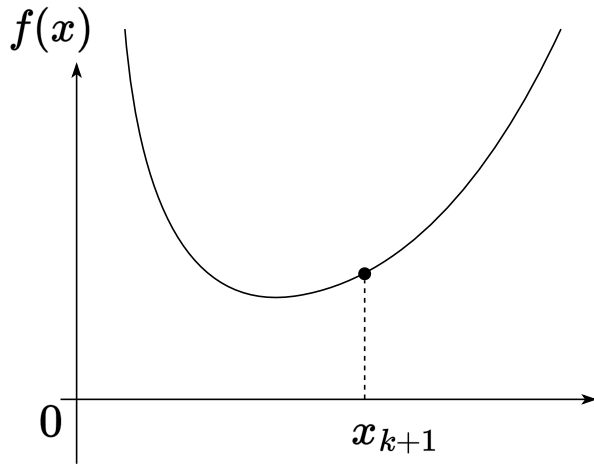
## Newton method as a local quadratic Taylor approximation minimizer



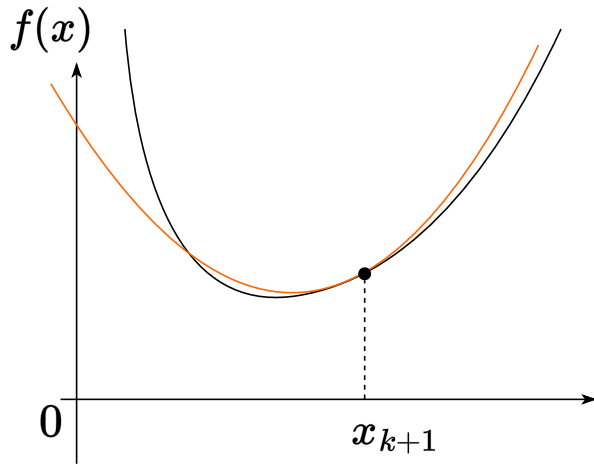
## Newton method as a local quadratic Taylor approximation minimizer



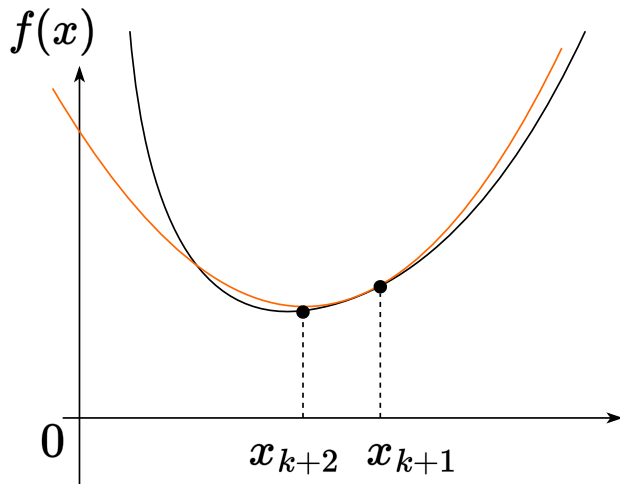
## Newton method as a local quadratic Taylor approximation minimizer



## Newton method as a local quadratic Taylor approximation minimizer



## Newton method as a local quadratic Taylor approximation minimizer



# Newton method vs gradient descent

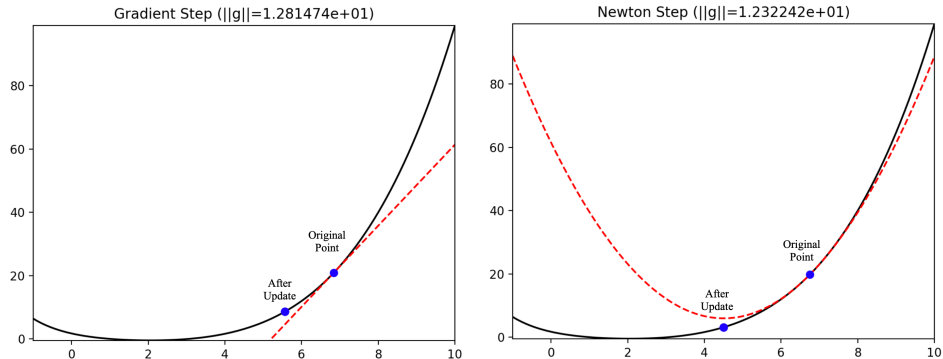


Figure 7: The loss function is depicted in black, the approximation as a dotted red line

The gradient descent  $\equiv$  linear approximation

The Newton method  $\equiv$  quadratic approximation



# Convergence

## i 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

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

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:

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

# Convergence

## i 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

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

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:

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

**“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

## i Question

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

# Affine invariance

## i Question

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

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

# Affine invariance

## i Question

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

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

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

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

# Affine invariance

## i Question

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

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

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

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

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

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

# Affine invariance

## i Question

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

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

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

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

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

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

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

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

# Summary of Newton's method

## Pros

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




# Summary of Newton's method

## Pros

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

## Cons


- 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  $\rightarrow$  direction  $-(f''(x))^{-1}f'(x)$  may not be a descending direction 

# Summary of Newton's method

## Pros

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

## Cons

- 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  $\rightarrow$  direction  $-(f''(x))^{-1}f'(x)$  may not be a descending direction 

Quasi Newton methods partially solve these problems!

# Quasi Newton methods

For the classic task of unconditional optimization  $f(x) \rightarrow \min_{x \in \mathbb{R}^n}$  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:

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

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.

# Quasi Newton methods

For the classic task of unconditional optimization  $f(x) \rightarrow \min_{x \in \mathbb{R}^n}$  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:

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

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  $k \rightarrow \infty$  to the true value of inverted Hessian in the local optimum  $f_{xx}^{-1}(x_*)$ .

# Quasi Newton methods

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:

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

Now let's formulate our method as:

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

in case you set the task of finding an update  $\Delta B_k$ :

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

## Broyden method

The simplest option is when the amendment  $\Delta B_k$  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  $\mu_k$  is a scalar and  $q_k$  is a non-zero vector. Then mark the right side of the equation to find  $\Delta B_k$  for  $\Delta z_k$ :

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

We get it:

$$\begin{aligned}\mu_k q_k q_k^\top \Delta y_k &= \Delta z_k \\ (\mu_k \cdot q_k^\top \Delta y_k) q_k &= \Delta z_k\end{aligned}$$

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

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

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



# Computational experiments

Let's look at computational experiments for Newton and Quasi Newton methods 🔄.