Intuition

Newton's method to find the equation' roots

Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$. Let there be equation $\varphi(x^*) = 0$. Consider a linear approximation of the function $\varphi(x)$ near the solution ($x^* - x = \Delta x$):

$$\varphi(x^*) = \varphi(x + \Delta x) \approx \varphi(x) + \varphi'(x)\Delta x.$$

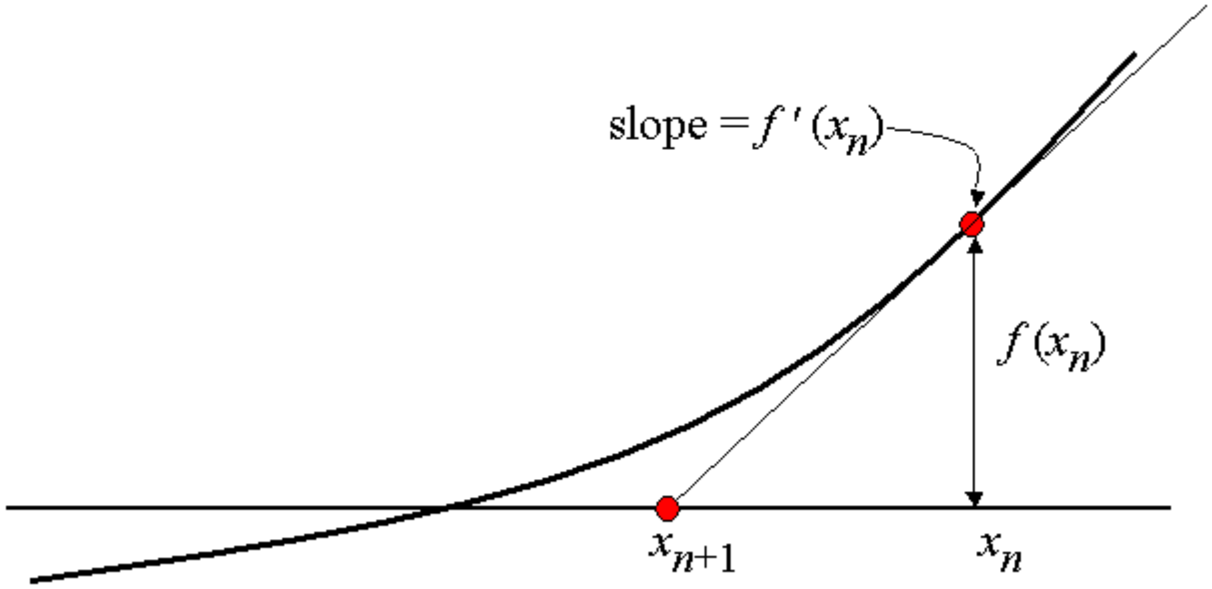
We get an approximate equation:

$$\varphi(x) + \varphi'(x)\Delta x = 0$$

We can assume that the solution to equation $\Delta x = -\dfrac{\varphi(x)}{\varphi'(x)}$ will be close to the optimal $\Delta x^* = x^* - x.$

We get an iterative scheme:

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



This reasoning can be applied to the unconditional minimization task of the $f(x)$ function by writing down the necessary extremum condition:

$$f'(x^*) = 0$$

Here $\varphi(x) = f'(x), \; \varphi'(x) = f''(x)$. Thus, we get the Newton optimization method in its classic form:

$$x_{k+1} = x_k - \left[f''(x_k) \right]^{-1} f'(x_k). \tag{Newton}$$

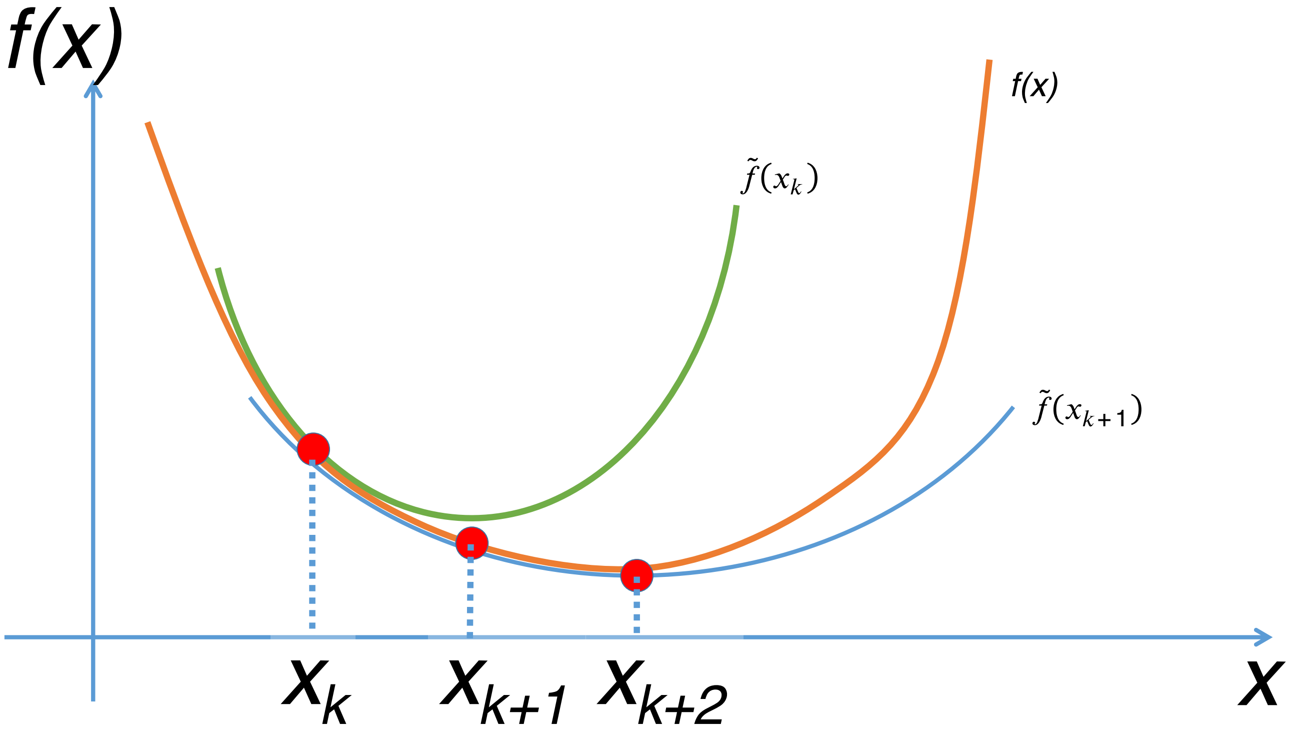
With the only clarification that in the multidimensional case:
$x \in \mathbb{R}^n, \; f'(x) = \nabla f(x) \in \mathbb{R}^n, \; f''(x) = \nabla^2 f(x) \in \mathbb{R}^{n \times n}$.

Second order Taylor approximation of the function

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

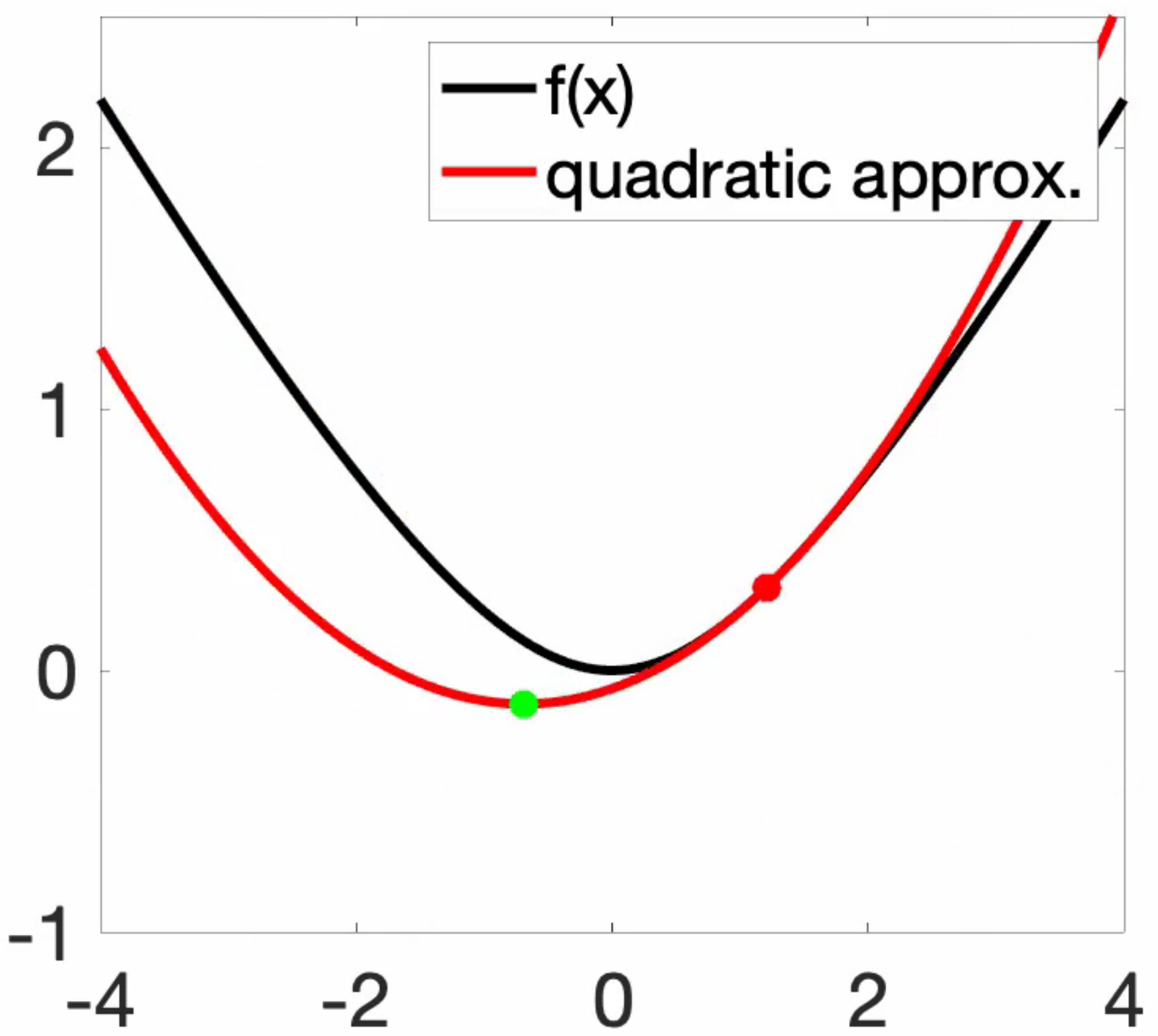
$$\tilde{f}(x) = f(x_k) + \langle f'(x_k), x - x_k \rangle + \frac{1}{2} \langle f''(x_k)(x - x_k), x - x_k \rangle.$$

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



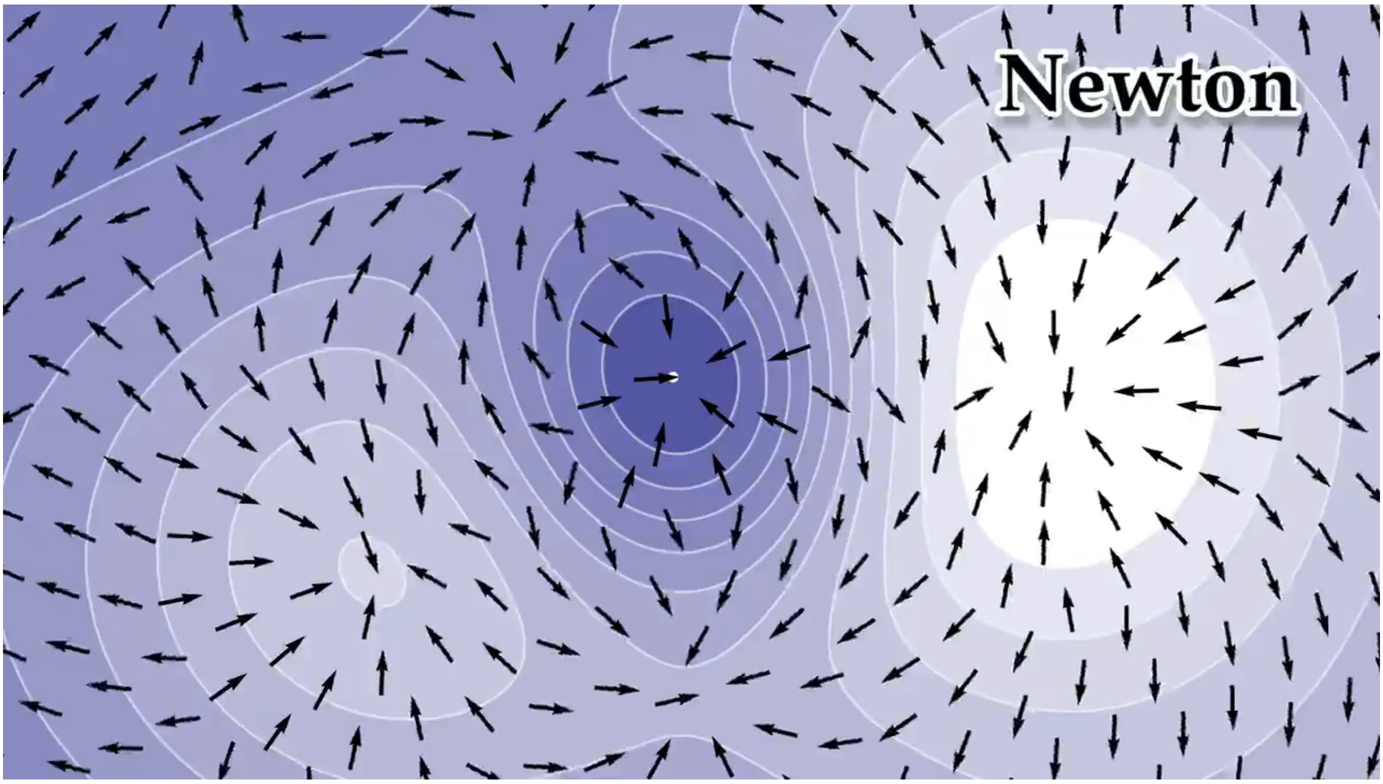
$$\begin{aligned} \nabla \tilde{f}(x_{k+1}) &= f'(x_k) + f''(x_k)(x_{k+1} - x_k) = 0 \\ f''(x_k)(x_{k+1} - x_k) &= -f'(x_k) \\ \left[f''(x_k) \right]^{-1} f''(x_k)(x_{k+1} - x_k) &= -\left[f''(x_k) \right]^{-1} f'(x_k) \\ x_{k+1} &= x_k - \left[f''(x_k) \right]^{-1} f'(x_k). \end{aligned}$$

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).



Quadratic approximation and Newton step (in green) for varying starting points (in red). Note that when the starting point is far from the global minimizer (in 0), the Newton step totally overshoots the global minimizer. Picture was taken from the post.

Convergence



Let's try to get an estimate of how quickly the classical Newton method converges. We will try to enter the necessary data and constants as needed in the conclusion (to illustrate the methodology of obtaining such estimates).

$$
\begin{aligned}
x_{k+1} - x^* = x_k - \left[f''(x_k)\right]^{-1} f'(x_k) - x^* = x_k - x^* - \left[f''(x_k)\right]^{-1} f'(x_k) = \\
= x_k - x^* - \left[f''(x_k)\right]^{-1} \int_0^1 f''(x^* + \tau(x_k - x^*))(x_k - x^*) d\tau = \\
= \left(1 - \left[f''(x_k)\right]^{-1} \int_0^1 f''(x^* + \tau(x_k - x^*)) d\tau\right)(x_k - x^*) = \\
= \left[f''(x_k)\right]^{-1} \left(f''(x_k) - \int_0^1 f''(x^* + \tau(x_k - x^*)) d\tau\right)(x_k - x^*) = \\
= \left[f''(x_k)\right]^{-1} \left(\int_0^1 \left(f''(x_k) - f''(x^* + \tau(x_k - x^*)) d\tau\right)\right)(x_k - x^*) = \\
= \left[f''(x_k)\right]^{-1} G_k (x_k - x^*)
\end{aligned}
$$

Used here is: $G_k = \int_0^1 \left(f''(x_k) - f''(x^* + \tau(x_k - x^*)) d\tau\right)$. Let's try to estimate the size of G_k:

$$
\begin{aligned}
\|G_k\| = \left\| \int_0^1 \left(f''(x_k) - f''(x^* + \tau(x_k - x^*)) d\tau\right) \right\| \leq \\
\leq \int_0^1 \left\| f''(x_k) - f''(x^* + \tau(x_k - x^*)) \right\| d\tau \leq \qquad \text{(Hessian's Lipschitz continuity)} \\
\leq \int_0^1 M \|x_k - x^* - \tau(x_k - x^*)\| d\tau = \int_0^1 M \|x_k - x^*\|(1 - \tau) d\tau = \frac{r_k}{2} M,
\end{aligned}
$$

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

So, we have:

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

Already smells like quadratic convergence. All that remains is to estimate the value of Hessian's reverse.

Because of Hessian's Lipschitz continuity and symmetry:

$$\begin{aligned} f''(x_k) - f''(x^*) &\succeq -Mr_k I_n \\ f''(x_k) &\succeq f''(x^*) - Mr_k I_n \\ f''(x_k) &\succeq lI_n - Mr_k I_n \\ f''(x_k) &\succeq (l - Mr_k) I_n \end{aligned}$$

So, (here we should already limit the necessity of being $f''(x_k) \succ 0$ for such estimations, i.e. $r_k < \frac{l}{M}$).

$$\left\| \left[f''(x_k) \right]^{-1} \right\| \leq (l - Mr_k)^{-1}$$

$$r_{k+1} \leq \frac{r_k^2 M}{2(l - Mr_k)}$$

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

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

Theorem

Let $f(x)$ be a strongly convex twice continuously differentiated function at $\mathbb{R}^n$, for the second derivative of which inequalities are executed: $lI_n \preceq f''(x) \preceq LI_n$. Then Newton's method with a constant step locally converges to solving the problem with superlinear speed. If, in addition, Hessian is Lipschitz continuous, then this method converges locally to x^* at a quadratic rate.

Summary

It's nice:

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

- the parameters have little effect on the convergence rate

It’s not nice:

- 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 at x^*
- the hessian may not be positively determined → direction $-(f''(x))^{-1}f'(x)$ may not be a descending direction

Possible directions

- Newton’s damped method (adaptive stepsize)
- Quasi-Newton methods (we don’t calculate the Hessian, we build its estimate - BFGS)
- Quadratic evaluation of the function by the first order oracle (superlinear convergence)
- The combination of the Newton method and the gradient descent (interesting direction)
- Higher order methods (most likely useless)

Materials

- Going beyond least-squares – I : self-concordant analysis of Newton method
- Going beyond least-squares – II : Self-concordant analysis for logistic regression
- Picture with gradient and Newton field was taken from this tweet by Keenan Crane.
- About global damped Newton convergence issue. Open in Colab

Code

Open in Colab

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:

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

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
- Comparison of quasi Newton methods