

Deep learning and continuous optimization

Spring semester 2025/26

Kristóf Bérczi

Eötvös Loránd University
Institute of Mathematics
Department of Operations Research



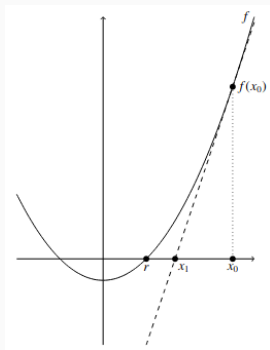
Lecture 4: Newton's method

Finding a root of a univariate function I

Input: A sufficiently differentiable function $g : \mathbb{R} \rightarrow \mathbb{R}$.

Goal: Find one of its roots, that is, a point r s.t. $g(r) = 0$.

Setup: Zeroth- and first-order access to g and point x_0 that is sufficiently close to some root of g .



Idea: Given a point x , consider the tangent through $(x, g(x))$, and let x' be the intersection with the x -axis.

$$x' = x - \frac{g(x)}{g'(x)}$$

⇒ We hope to make progress in reaching a zero of g .

Finding a root of a univariate function II

Algorithm

- 1 Start with $x_0 \in \mathbb{R}$.
- 2 For $t = 0, 1, \dots$, let

$$x_{t+1} := x_t - \frac{g(x_t)}{g'(x_t)}$$

Remarks:

- The method requires the differentiability of g .
- The convergence heavily depends on the starting point.

Example: Minimize $f(x) := ax - \log x$ over all positive $x > 0$.

Solution: Take $g(x) := f'(x)$, and find a root of g . As f is convex, the root of g is an optimizer for f . We have

$$g(x) := f'(x) = a - \frac{1}{x}.$$

Example 1

While the solution is obviously $\frac{1}{a}$, let us apply Newton's method.

Reason 1: To illustrate the method.

Reason 2: Early computers used Newton's method to compute the reciprocal as it only involved addition, subtraction, and multiplication.

We have

$$x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)} = 2x_t - ax_t^2.$$

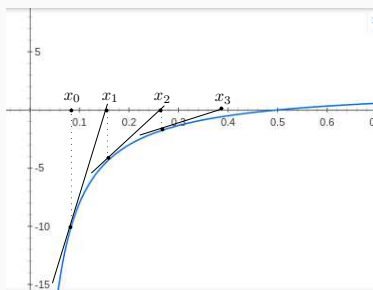
Define $e_t := 1 - ax_t$. Then

$$e_{t+1} = e_t^2.$$

- If $|e_0| < 1$, then $e_t \rightarrow 0$.
- If $|e_0| = 1$, then $e_t = 1$ for $t \geq 1$. \Rightarrow
- If $|e_0| > 1$, then $e_t \rightarrow \infty$.
- If $0 < x_0 < \frac{2}{a}$, then $x_t \rightarrow \frac{1}{a}$.
- If $x_0 = \frac{2}{a}$, then $x_t = 0$ for $t \geq 1$.
- If $x_0 > \frac{2}{a}$, then $x_t \rightarrow -\infty$.

Example II

- If $0 < x_0 < \frac{2}{a}$, then $x_t \rightarrow \frac{1}{a}$.
- If $x_0 = \frac{2}{a}$, then $x_t = 0$ for $t \geq 1$.
- If $x_0 > \frac{2}{a}$, then $x_t \rightarrow -\infty$.



⇒ The right starting point has a crucial impact on whether the algorithm succeeds or fails.

Note: By modifying the function g , e.g. $g(x) := x - \frac{1}{a}$, we get a different algorithm to compute $\frac{1}{a}$. However, some of them might not make sense ($x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)} = \frac{1}{a}$), or might not be efficient.

Convergence I

Recall: The distance $e_t = 1 - ax_t = a\left(\frac{1}{a} - x_t\right)$ was squared at every iteration in the example.

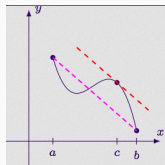
Question: Do we get quadratic convergence in general? **YES!**

Thm.

Suppose $g : \mathbb{R} \rightarrow \mathbb{R}$ is twice differentiable and its second derivative is continuous, $r \in \mathbb{R}$ is a root of g , $x_0 \in \mathbb{R}$ is a starting point, and $x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}$. Then $|x_1 - r| \leq M|x_0 - r|^2$, where $M = \sup_{\xi \in (r, x_0)} \left| \frac{g''(\xi)}{2g'(\xi)} \right|$.

The proof relies on the **Mean value theorem**, stating that if $h : \mathbb{R} \rightarrow \mathbb{R}$ is a continuous function on the closed interval $[a, b]$ and differentiable on the open interval (a, b) , then there exists $c \in (a, b)$ s.t.

$$h'(c) = \frac{h(b) - h(a)}{b - a}.$$



Convergence II

Proof of the theorem.

By considering the second-order Taylor approximation of g around x_0 , we have

$$g(r) = g(x_0) + (r - x_0)g'(x_0) + \frac{1}{2}(r - x_0)^2 g''(\xi)$$

for some $\xi \in (r, x_0)$.

From the definition of x_1 , we know that $g(x_0) = g'(x_0)(x_0 - x_1)$. Furthermore, $g(r) = 0$ as r is a root, hence

$$0 = g'(x_0)(x_0 - x_1) + (r - x_0)g'(x_0) + \frac{1}{2}(r - x_0)^2 g''(\xi).$$

This implies

$$g'(x_0)(x_1 - r) = \frac{1}{2}(r - x_0)^2 g''(\xi).$$

Therefore

$$|x_1 - r| = \left| \frac{g''(\xi)}{2g'(x_0)} \right| |x_0 - r|^2 \leq M |x_0 - r|^2,$$

where M is as stated in the theorem. □

Convergence III

Assuming M is a small constant (say $M \leq 1$, and this holds throughout the procedure) and $|x_0 - r| < \frac{1}{2}$, the theorem implies quadratically fast convergence of x_t to r .

Indeed, after t steps we have

$$|x_t - r| \leq |x_0 - r|^{2^t} \leq 2^{-2^t}.$$

⇒ If $t \approx \log \log \frac{1}{\varepsilon}$, then $|x_t - r| \leq \varepsilon$.

Summary: Newton's method is very efficient!

See the Newton-Raphson method on GeoGebra!

Multivariate functions

Input: A sufficiently differentiable function $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$.

Goal: Find one of its roots, that is, a point r s.t. $g(r) = 0$.

[**Be careful:** here 0 denotes the all-zero vector.]

Setup: Zeroth- and first-order access to g and point x_0 that is sufficiently close to some root of g .

Original idea: Given a point x , define

$$x' = x - \frac{g(x)}{g'(x)}$$

\Rightarrow Now $g(x)$ is a vector while $g'(x)$ is the **Jacobian** matrix of g at x , i.e. $J_g(x)$ is the matrix of partial derivatives

$$\left[\frac{\partial g_i}{\partial x_j}(x) \right]_{1 \leq i, j \leq n}$$

Hence the update rule becomes

$$x_{t+1} := x_t - J_g(x_t)^{-1} g(x_t).$$

Newton's method for unconstrained optimization

What is the connection between convex programs and Newton's method?

Key observation: Minimizing a differentiable convex function in the unconstrained setting is equivalent to finding a root of its derivative.

Input: Sufficiently differentiable convex function f .

Goal: Find $x^* := \arg \min_{x \in \mathbb{R}^n} f(x)$.

Recall:

- ∇f is a function from \mathbb{R}^n to \mathbb{R}^n .
- The Jacobian $J_{\nabla f}$ is the Hessian $\nabla^2 f$.

⇒ The update rule is

$$x_{t+1} := x_t - (\nabla^2 f(x_t))^{-1} \nabla f(x_t).$$

For ease of notation, we define the **Newton step** at x to be

$$n(x) := -(\nabla^2 f(x_t))^{-1} \nabla f(x_t).$$

Hence $x_{t+1} := x_t + n(x_t)$.

Newton's method as a second-order method

Suppose we would like to find a global minimum of f and x_0 is our current approximate solution. Let

$$\tilde{f}(x) := f(x_0) + \langle x - x_0, \nabla f(x_0) \rangle + \frac{1}{2}(x - x_0)^T \nabla^2 f(x_0)(x - x_0).$$

Idea: Set the next point to be the minimizer of \tilde{f} .

[Roughly, we hope that \tilde{f} approximates f locally, and so the new point should be an even better approximation to x^ .]*

We have to find $x_1 := \arg \min_{x \in \mathbb{R}^n} \tilde{f}(x)$. Assuming that f is strictly convex (and so $\nabla^2 f(x_0)$ is invertible), this is equivalent to solving $\nabla \tilde{f}(x) = 0$, that is,

$$\nabla f(x_0) + \nabla^2 f(x_0)(x - x_0) = 0,$$

leading to $x_1 = x_0 - (\nabla^2 f(x_0))^{-1} \nabla f(x_0) = x_0 + n(x_0)$.

\Rightarrow We recovered Newton's method!

Consequence: When applied to strictly convex quadratic functions, i.e. of the form $h(x) = x^T Mx + b^T x$ for $M \succ 0$, then after one iteration we land in the unique minimizer.

Newton's method vs. gradient descent

Is Newton's method a "better" algorithm?

Pros: It uses the Hessian to perform the iterations, hence it is more powerful.

Cons: One iteration is now more costly, as a second-order oracle is needed.

More precisely, to compute x_{t+1} , we need to solve the following system:

$$(\nabla^2 f(x_t)) x = \nabla f(x_t).$$

- In general, this takes $O(n^3)$ time using Gaussian elimination, or $O(n^\omega)$ using fast matrix multiplication.
- If the Hessian has a special form, e.g. it is Laplacian, then there are nearly-linear time Laplacian solvers.

Newton-Euclidean condition

NE condition

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a function, x^* be one of its minimizers, x_0 be arbitrary. We say that the **NE**(M) condition is satisfied for $M > 0$ if there exists an Euclidean ball $B(x^*, R)$ of radius R containing x_0 and constants $h, L > 0$ such that $M \geq \frac{L}{2h}$ and

- for every $x \in B(x^*, R)$, we have $\|\nabla^2 f(x)^{-1}\| \leq \frac{1}{h}$,
- for every $x, y \in B(x^*, R)$, we have $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|_2$.

Here the norm of a matrix is the so-called spectral norm, defined as

$$\|A\| := \sup_{x \in \mathbb{R}^n} \frac{\|Ax\|_2}{\|x\|_2}.$$

Thm.

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and x^* be one of its minimizers. Let x_0 be arbitrary and define $x_1 := x_0 + n(x_0)$. If the **NE**(M) condition is satisfied, then

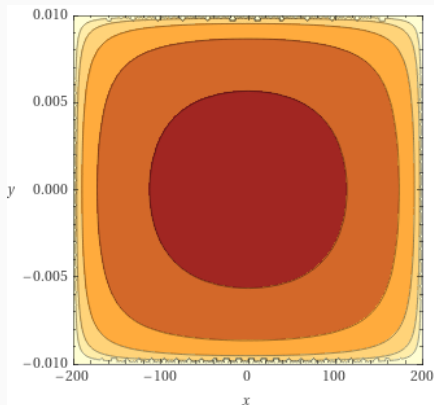
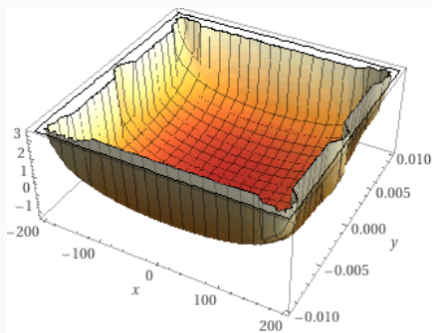
$$\|x_1 - x^*\|_2 \leq M\|x_0 - x^*\|_2^2.$$

Problem with the convergence I

Fact: The theorem is stated with respect to quantities based on Euclidean norm $\| \cdot \|_2$, which makes it hard to apply in many cases.

Example: For $K_1, K_2 > 0$ (large constants), consider

$$f(x_1, x_2) := -\log(K_1 - x_1) - \log(K_1 + x_1) - \log\left(\frac{1}{K_2} - x_2\right) - \log\left(\frac{1}{K_2} + x_2\right).$$



Problem with the convergence II

Now the Hessian of f is

$$\nabla^2 f(x) = \begin{pmatrix} \frac{1}{(K_1 - x_1)^2} + \frac{1}{(K_1 + x_1)^2} & 0 \\ 0 & \frac{1}{(\frac{1}{K_2} - x_2)^2} + \frac{1}{(\frac{1}{K_2} + x_2)^2} \end{pmatrix}$$

\Rightarrow It can be verified that M , which determines the quadratic convergence of Newton's method, is at least $\Omega(K_1^2 K_2^2)$. Therefore, even when the initial point is close to the optimal solution x^* , the guarantee in the theorem is too weak to imply that in one step the distance drops.

However, Newton's method does in fact converge rapidly to x^* !

Local norm I

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a strictly convex function, i.e., the Hessian $\nabla^2 f(x)$ is positive definite for every $x \in \mathbb{R}^n$. We define the **local inner product** at every point x as

$$\langle u, v \rangle_x := u^T \nabla^2 f(x) v \quad \text{for } u, v \in \mathbb{R}^n.$$

The corresponding **local norm** is

$$\|u\|_x := \sqrt{u^T \nabla^2 f(x) u} \quad \text{for } u \in \mathbb{R}^n.$$

Recall: When deriving the gradient descent algorithm, we picked the direction of steepest descent which is a solution to the following problem:

$$\arg \max_{\|u\|=1} (-\langle \nabla f(x), u \rangle).$$

The optimal direction w.r.t. the Euclidean norm $\|\cdot\| = \|\cdot\|_2$ is in the direction $-\nabla f(x)$.

Idea: What if instead maximize over all u of local norm 1? That is,

$$\arg \max_{\|u\|_x=1} (-\langle \nabla f(x), u \rangle) = \arg \max_{u^T \nabla^2 f(x) u = 1} (-\langle \nabla f(x), u \rangle).$$

[We would like to capture the “shape” of f around x with our choice of the norm, and our best guess is the quadratic term given by the Hessian.]

Using Cauchy–Schwarz, the optimal solution is in the direction

$$-\nabla^2 f(x)^{-1} \nabla f(x),$$

which is exactly the **Newton step!**

Indeed, set $v := \nabla^2 f(x)^{-1} \nabla f(x)$, and observe that

$$\begin{aligned} -\langle \nabla f(x), u \rangle &= -\left\langle \nabla^2 f(x)^{\frac{1}{2}} v, \nabla^2 f(x)^{\frac{1}{2}} u \right\rangle \\ &\leq \sqrt{v^T \nabla^2 f(x) v} \sqrt{u^T \nabla^2 f(x) u} \\ &= \|v\|_x \|u\|_x, \end{aligned}$$

and equality holds if and only if $\nabla^2 f(x)^{\frac{1}{2}} u = -\nabla^2 f(x)^{\frac{1}{2}} v$. This is the same as

$$u = -v = -\nabla^2 f(x)^{-1} \nabla f(x).$$

Conclusion

Newton's method can be interpreted as a **steepest descent** algorithm, where the Newton step is the direction of steepest descent with respect to the local norm.

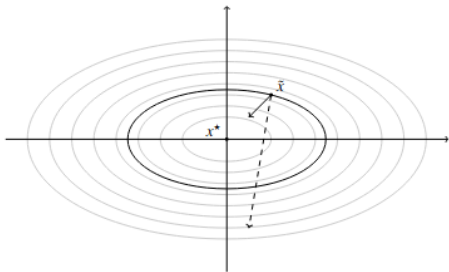


Figure 9.2 Illustration of the difference in directions suggested by gradient descent and Newton's method for the function $x_1^2 + 4x_2^2$. $\bar{x} = (1, 1)$ and $x^* = (0, 0)$. The solid arrow is a step of length $1/2$ in the direction $(-1, -1)$ (Newton step) and the dashed arrow is a step of length $1/2$ in the direction $(-1, -4)$ (gradient step).