MSc Data Science Ramp-Up Course in Mathematics
Part: Continuous Optimization

Prof. Dr. Christian Kirches

Summer 2024

The following is a summary of some essential topics that I would have treated in a one-semester BSc class in continuous optimization at TU Braunschweig. It is intentionally written in a brief and informal style. The intention is to transfer general concepts while, at certain times, glancing over the finer details to maintain simplicity and accessibility of the exposition. Almost every topic addressed here deserves a broader and much more detailed discussion. Textbooks on calculus and continuous optimization should be consulted in parallel to reading this summary.

I highly recommend the textbook *Numerical Optimization* by J. Nocedal and S.J. Wright, 2nd edition, Springer, 2006 (ISBN 978-0-387-40065-5). The university libraries has copies. A Springer ebook can be obtained from `https://link.springer.com/book/10.1007/978-0-387-40065-5`

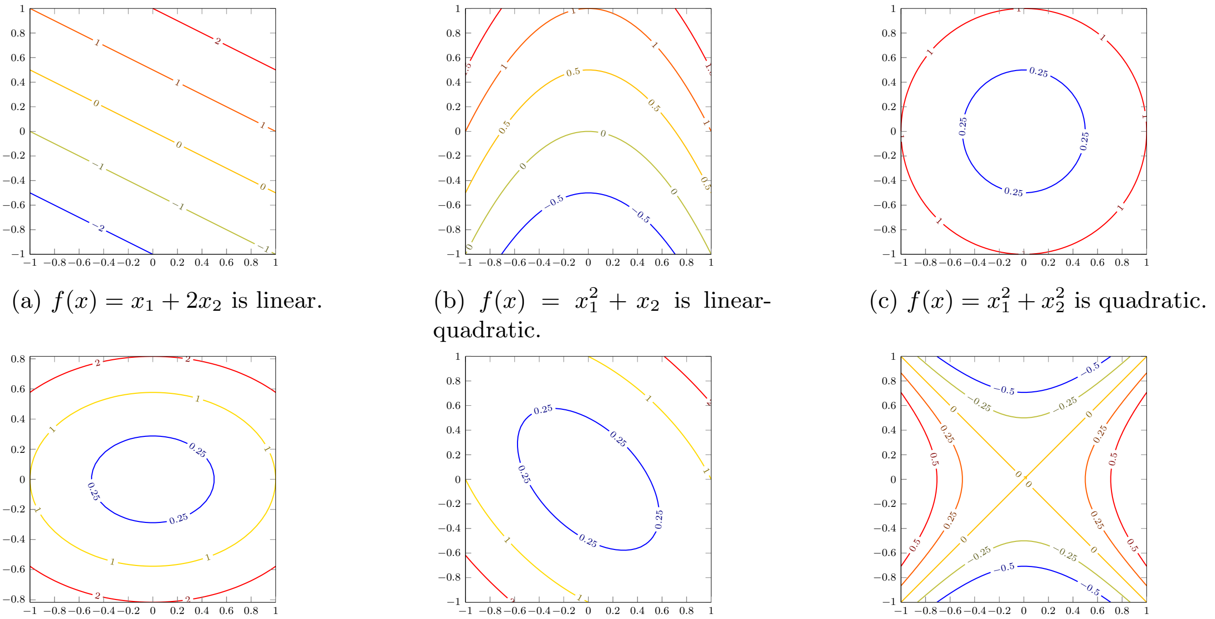
1 Calculus Prerequisites

Level Sets of Functions

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, a level set $N_c(f)$ of f contains all points in $\mathbb{R}^n$ at which the function has the same value c,

$$N_c(f) := \{x \in \mathbb{R}^n \mid f(x) = c\}.$$

There is one level set for every function value $c \in \mathbb{R}$. Level sets are empty for values c that f never assumes.



(a) $f(x) = x_1 + 2x_2$ is linear.

(b) $f(x) = x_1^2 + x_2$ is linear-quadratic.

(c) $f(x) = x_1^2 + x_2^2$ is quadratic.

(d) $f(x) = x_1^2 + 3x_2^2$ is quadratic.

(e) $f(x) = x_1^2 + x_2^2 + x_1x_2$ is quadratic.

(f) $f(x) = x_1^2 - x_2^2$ is quadratic but nonconvex.

Gradient, Jacobian, Directional Derivatives

The **gradient** of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ in a point $x \in \mathbb{R}^n$ is a column vector in $\mathbb{R}^n$ denoted by $\nabla f(x)$, or alternatively a row vector denoted by $f'(x)$. Its elements are the differential quotients

$$(f'(x))_i = (\nabla f(x))_i = \lim_{h \to 0} \frac{1}{h}(f(x + he_i) - f(x)),$$

which are the slopes of f along the n different coordinate axes e_i, $1 \leq i \leq n$.

In practice, the gradient is computed by applying the known rules of differential calculus (product rule, chain rule, etc.), which we assume you know. When visualized by an arrow, a gradient is perpendicular to the level set tangent and points into the direction of steepest ascent.

For a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ we have a gradient for every one of the m component functions $f_j : \mathbb{R}^n \to \mathbb{R}$, $1 \leq j \leq m$. The gradient of f becomes an $n \times m$ matrix composed of columns,

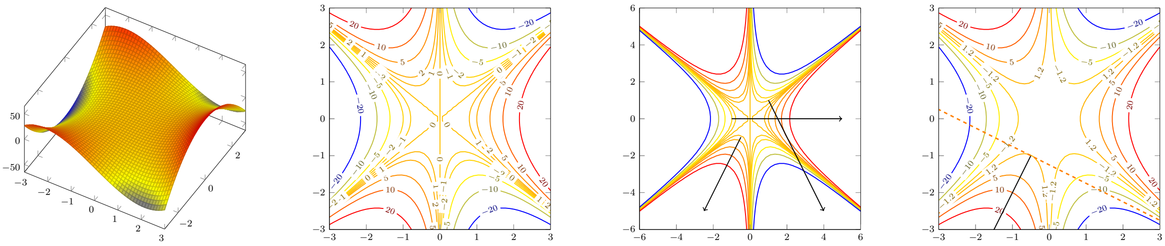
$$\nabla f(x) = \left(\begin{array}{c|c|c} \nabla f_1(x) & \cdots & \nabla f_m(x) \end{array} \right) \in \mathbb{R}^{n \times m},$$

and its transpose is called the **Jacobian**[1] $f'(x)$ of f at x, composed of rows

$$f'(x) = \begin{pmatrix} f_1'(x) \\ \hline \vdots \\ \hline f_m'(x) \end{pmatrix} \in \mathbb{R}^{m \times n}.$$

A **directional derivative** represents the slope of f at $x \in \mathbb{R}^n$ into a direction $d \in \mathbb{R}^n$ that is *not* necessarily one of the coordinate axes e_i. It is given by the dot product ($m = 1$) or matrix-vector product ($m > 1$)

$$\nabla f(x)^T d = \langle \nabla f(x), d \rangle = f'(x) d \in \mathbb{R}.$$



(a) The graph of $f(x) = 2x_1^3 - 3x_1x_2^2$.

(b) Some level sets of f.

(c) Gradients $\nabla f(x)$ in several locations x.

(d) The gradient is a direction of steepest ascent, and is perpendicular to the level set tangent.

Hessian

The **Hessian**[2] of a function $f : \mathbb{R}^n \to \mathbb{R}$ in a point $x \in \mathbb{R}^n$ is the $n \times n$ matrix $\nabla^2 f(x)$ of mixed second order partial differentials

$$(\nabla^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}.$$

If the second derivative of f is continuous, the matrix $(\nabla^2 f(x))$ is symmetric (Schwarz[3] theorem). The **curvature** of f along the coordinate axes e_i is found on the diagonal,

$$e_i^T \; \nabla^2 f(x) \; e_i = (\nabla^2 f(x))_{ii}.$$

The curvature into a direction $d \in \mathbb{R}^n$ that is *not* necessarily one of the coordinate axes e_i is found by computing the bilinear form

$$d^T \; \nabla^2 f(x) \; d \in \mathbb{R}.$$

This requires computing the matrix-vector product $v = \nabla^2 f(x) \; d$ and then the dot product $d^T v$.

[1] Carl Gustav Jacob Jacobi, German mathematician

[2] Otto Hesse, German mathematician

[3] Hermann Amandus Schwarz, German mathematician

Taylor Expansion, Tangents, Quadratic Models

A twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ admits a Taylor[4] expansion

$$f(x+d) = f(x) + \nabla f(x)^T d + \tfrac{1}{2} d^T \nabla^2 f(x) d + o(||d||^2).$$

This involves the function's value, directional derivative, and curvature into a direction $d \in \mathbb{R}^n$. As $d \to 0$, the error term $o(||d||^2)$ approches zero faster than $||d||^2$ does.

The function

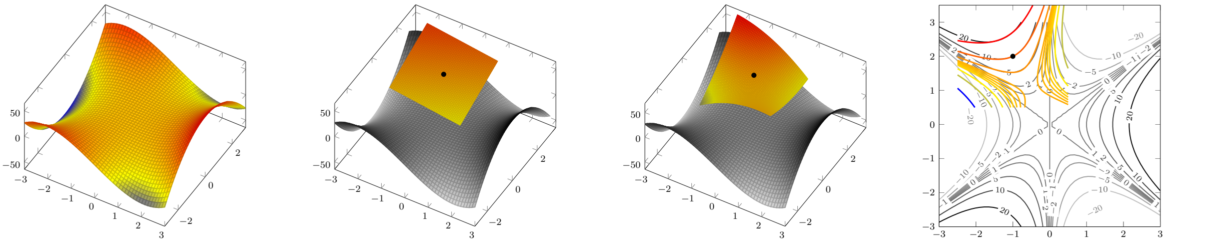
$$\ell(d) = f(x) + \nabla f(x)^T d$$

is a **linear model** of the behavior of the function f, taken in a fixed point $x \in \mathbb{R}^n$, and evaluated when leaving x into direction $d \in \mathbb{R}^n$. It is frequently called a **tangent** to f in x.

The function

$$q(d) = f(x) + \nabla f(x)^T d + \tfrac{1}{2} d^T \nabla^2 f(x) d$$

is a **quadratic model** of the behavior of the function f, taken in a fixed point $x \in \mathbb{R}^n$, and evaluated when leaving x into direction $d \in \mathbb{R}^n$.



(a) The graph of $f(x) = 2x_1^3 - 3x_1x_2^2$.

(b) A linear model of f in a location x.

(c) A quadratic model of f in a location x.

(d) Overlay of level sets of f and the same quadratic model.

A twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$ is vector-valued and its Taylor expansion

$$f(x+d) = f(x) + \nabla f(x)^T d + \tfrac{1}{2} d^T \nabla^2 f(x) d + o(||d||^2)$$

works with the gradient matrix $\nabla f(x) \in \mathbb{R}^{n \times m}$ and the Hessian is a third-order tensor $\nabla^2 f(x) \in \mathbb{R}^{n \times n \times m}$. The bilinear form $d^T \nabla^2 f(x) d \in \mathbb{R}^m$ is vector-valued. It is sometimes easier to work with Taylor expansions of the component functions $f_j : \mathbb{R}^n \to \mathbb{R}$, $1 \le j \le m$.

Convex Functions

A function $f : D \to \mathbb{R}$ is convex on a subset $D \subset \mathbb{R}^n$ if one of the following holds:

1. f has nonnegative curvature everywhere on D,

$$\nabla^2 f(x) \text{ positive semidefinite } \forall x \in D.$$

2. *Any* tangent to f never become greater than f on D,

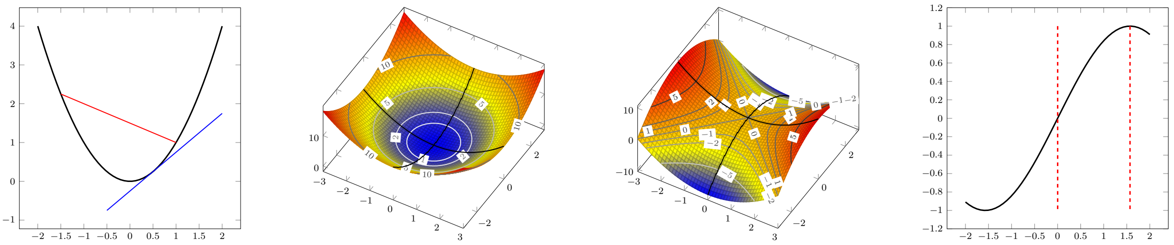
$$f(x) + \nabla f(x)^T (y-x) \le f(y) \ \forall x, y \in D.$$

3. Secants between *any* two points on the graph of f never become smaller than f on D,

$$f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y) \ \forall x, y \in D, \lambda \in (0,1).$$

If the inequalities are strict (equiv. if $\nabla^2 f(x)$ is positive definite) on D, the function f is called *strictly convex* on D.

[4] Brook Taylor, British mathematician



(a) A convex function (black) with a tangent (blue) and a secant between two points (red).

(b) $f(x) = x_1^2 + x_2^2$ is convex. The Hessian $\nabla^2 f$ has a double eigenvalue 2.

(c) $f(x) = x_1^2 - x_2^2$ is convex along x_1 but not along x_2, so not a convex function. Eigenvalues of $\nabla^2 f$ are 2 and -2.

(d) $f(x) = \sin(x)$ is convex on $D = [0, \pi]$ but not on $D = \mathbb{R}$.

Subsets of $\mathbb{R}^n$

A point $x \in \mathbb{R}^n$ is called **feasible** with respect to a set $\mathcal{F} \subset \mathbb{R}^n$ if $x \in \mathcal{F}$. It is called **infeasible** with respect to $\mathcal{F}$ otherwise.
Testing feasibility is not a computational concept unless an **outer description** of $\mathcal{F}$ is accessible. We frequently assume that functions $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^k$ exist such that

$$\mathcal{F} := \{x \in \mathbb{R}^n \mid g(x) = 0 \in \mathbb{R}^n,\ h(x) \geq 0 \in \mathbb{R}^m\},$$

i.e., membership of x in $\mathcal{F}$ can be tested by evaluating g and h and checking the signs of their vector return values.

The **boundary** of a set $\mathcal{F}$ is denoted by $\partial\mathcal{F}$ and consists of all point $x \in \mathcal{F}$ with neighborhoods that contain points $y \notin \mathcal{F}$. If $\partial\mathcal{F} \in \mathcal{F}$, then $\mathcal{F}$ is called **closed**. Subsets of $\mathbb{R}^n$ that are bounded and closed are called **compact**. The **interior** int $\mathcal{F}$ is the set $\mathcal{F} \setminus \partial\mathcal{F}$. If $x \in \partial\mathcal{F}$ implies $x \notin \mathcal{F}$, then $\mathcal{F}$ is called **open**.

A set $\mathcal{C}$ satisfying $\lambda x \in \mathcal{C}$ for all $x \in \mathcal{C}$ and all $\lambda > 0$ is called a **cone**. It is **pointed** if $0 \in \mathcal{C}$. Finite unions and intersections of cones are cones. Finite intersections of convex cones are convex, but unions need not be.

The cone $\mathcal{C}^* = \{y \in \mathbb{R}^n \mid \langle x, y\rangle \geq 0\ \forall x \in \mathcal{F}\}$ is called the **dual cone** of a set $\mathcal{F}$. The cone $\mathcal{C}^\circ = \{y \in \mathbb{R}^n \mid \langle x, y\rangle \leq 0\ \forall x \in \mathcal{F}\}$ is called the **polar cone** of a set $\mathcal{F}$. Both are closed and convex.

2 Unconstrained Optimization Theory

Definition 1. (Unconstrained Nonlinear Program)
The unconstrained minimization problem

$$\boxed{\min_{x \in \mathbb{R}^n} \ f(x)} \tag{MIN}$$

for an *objective function* $f : \mathbb{R}^n \to \mathbb{R}$ is called an *unconstrained nonlinear program.*

Definition 2. (Minimizers)
Let $x^* \in \mathbb{R}^n$. If there exists a number $\varepsilon > 0$ such that

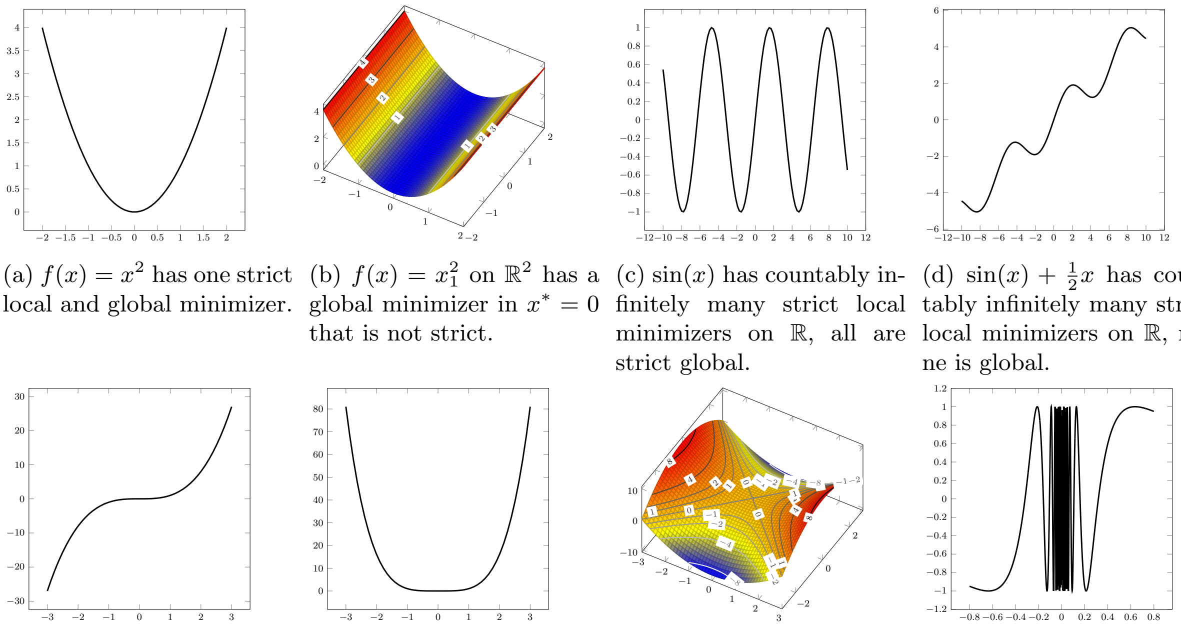
$$f(x^*) \leq f(y) \text{ for all } y \in U_\varepsilon(x) := \{y \in \mathbb{R}^n \mid ||y - x^*|| < \varepsilon\}, \tag{1}$$

then x^* is called a **local minimizer** of (MIN). If the inequality holds strictly on $U_\varepsilon(x) \setminus \{x^*\}$, then x^* is called a **strict local minimizer**. If ε can be chosen arbitrarily large, then x^* is called a **global minimizer**. The value $f(x^*)$ is called a (strict) local/global **minimum**.

Identifying *global* minimizers is an $\mathcal{NP}$-hard task in general. Assuming that the $\mathcal{P} \neq \mathcal{NP}$ hypothesis is true, this roughly means that on deterministic computers there is no algorithm that finds a global minimizer within a runtime that is "fast", i.e. a polynomial in the number n of unknowns. The problem isn't "unsolvable", though, as there are "slow" algorithms that solve it within a runtime that is exponential in n.

n	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$	$O(1.001^n)$	$O(2^{n/100})$
10^2	5 ns	100 ns	461 ns	10 μs	1 ms	1 ns	2 ns
10^3	7 ns	1 μs	7 μs	1 ms	1 s	2.7 ns	1 μs
10^4	9 ns	10 μs	92 μs	100 ms	14 ms	22 μs	–
10^5	12 ns	100 μs	1 ms	10 s	11.5 d	–	–
10^6	14 ns	1 ms	14 ms	17 min	31.7 y	–	–
10^9	21 ns	1 s	21 s	31.7 y	–	–	–

Fast versus slow algorithms. The table shows runtimes depending on n for algorithms of different runtime complexity. It is assumed that one operation takes one nanosecond (10^{-9} seconds, i.e. a machine at 1 GHz clock speed) and that the constant hidden by O is 1. Other choices would introduce a constant factor but would not change the general picture. "–" indicates runtimes certainly in excess of your lifetime.



(a) $f(x) = x^2$ has one strict local and global minimizer.

(b) $f(x) = x_1^2$ on $\mathbb{R}^2$ has a global minimizer in $x^* = 0$ that is not strict.

(c) $\sin(x)$ has countably infinitely many strict local minimizers on $\mathbb{R}$, all are strict global.

(d) $\sin(x) + \frac{1}{2}x$ has countably infinitely many strict local minimizers on $\mathbb{R}$, none is global.

(e) $f(x) = x^3$ has a saddle point in $x^* = 0$, but no minimizers. Still $\nabla f(0) = 0$.

(f) $f(x) = x^4$ has a local and global minimizer in $x^* = 0$ but $\nabla^2 f(0) = 0$.

(g) $f(x) = x_1^2 - x_2^2$ has a saddle point in $x^* = 0$, but no minimizers.

(h) $\sin(1/x)$ has countably infinitely many strict local and global minimizers in the finite interval $[-1, 1]$.

Hence we only discuss local minimizers. Even the definition of a *local* minimizer is not suitable for computation as it requires verifying the condition $f(x^*) \le f(y)$ *for uncountably infinitely many* choices of y in a certain set. In its place, the following necessary condition is used to characterize local minimizers.

Proposition 3. (Necessary Optimality Condition of First Order)
Let x^* be a local minimizer of (MIN). Then the following holds:

$$\nabla f(x^*) = 0.$$

The condition is not sufficient, which motivates the following definition.

Definition 4. (Stationary Point)
If a point $x \in \mathbb{R}^n$ satisfies $\nabla f(x) = 0$, then x is called a **stationary point** of (MIN).

As a consequence, all local minimizers are stationary points. Vice versa, stationary points are the *candidates* for local minimizers, but some of them may not actually be minimizers (cf. examples (a) and (e)). In general, a gap remains between stationarity and local minimizers. The situation is much better for convex functions f.

Proposition 5. (Necessary and Sufficient Optimality Condition of First Order for Convex Functions)
Let f be convex. If and only if $\nabla f(x^*) = 0$ then x^* is a local and global minimizer of (MIN).

Compare examples (a) and (f). Moreover, the second order necessary optimality condition is always satisfied for convex functions. By convexity, a local minimizer also is a global one. If f is even *strictly convex*, the sufficient optimality condition of second order is always satisfied and the local minimizer is both unique and global. None of this however means that convex optimization problems can be solved exactly in polynomial time.

To try and tell local minimizer apart from stationary points, curvature information from the Hessian of f may be used to state second order conditions.

Proposition 6. (Necessary Optimality Condition of Second Order)
Let x^* be a local minimizer of (MIN). Then the following holds:

$$\nabla f(x^*) = 0 \text{ and } \nabla^2 f(x^*) \text{ is positive semidefinite.}$$

Again, all local minimizer satisfy these conditions. However, there may be additional points that satisfy them without being local minimizers (compare examples (a), (e) and (f)). Curvature information also allows to formulate a sufficient condition.

Proposition 7. (Sufficient Optimality Condition of Second Order)
If a point $x^* \in \mathbb{R}^n$ satisfies

$$\nabla f(x^*) = 0 \text{ and } \nabla^2 f(x^*) \text{ is positive definite,}$$

then x^* is a strict local minimum of (MIN).

Sufficient conditions, if satisfied, certify the presence of a local minimizer. If not satisfied, no statement is made, i.e. we *must not conclude* that the point in question is *not* a local minimizer. There are local minimizers that don't satisfy sufficient conditions, cf. example (f).

Unconstrained Optimization Algorithms

We cannot generally expect to describe local minimizers of (MIN) in terms of an algebraic expression (this is possible for fourth-order polynomials only). Instead, we can only hope to approximate them using iterative methods. The most basic one is gradient descent.

Definition 8. (Steepest Descent Direction)
The unit direction $d \in \mathbb{R}^n$ that minimizes the directional derivative

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & \nabla f(x)^T d \\ \text{s.t.} \quad & \|d\| = 1 \end{aligned} \tag{2}$$

is called the direction of *steepest descent* of f in the point x w.r.t. the norm $||\cdot||$. For the Euclidean norm $||\cdot||_2$, it is given by

$$d = -\frac{\nabla f(x)}{||\nabla f(x)||_2}. \tag{3}$$

This approach minimizes the first order Taylor term. The result shows that the gradient of a function is a direction of steepest ascent, and the antigradient is a direction of steepest descent. As we're looking for minimizers, this motivates gradient descent.

Definition 9. (Gradient Descent)
An algorithm producing a sequence of iterates $\{x^{(k)}\} \subset \mathbb{R}^n$ by starting in $x^{(0)} \in \mathbb{R}^n$ and letting

$$x^{(k+1)} \leftarrow x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}) \tag{4}$$

with step sizes $\alpha^{(k)} \in (0, 1]$ is called a *gradient descent* algorithm.

Gradient descent can be understood as replacing f (which we can't minimize exactly in one step) by its linear model $\ell(d)$ in $x^{(k)}$. Step sizes (called learning rates in ML) may be determined using line search.

Definition 10. (Armijo Line Search)
For a point $x \in \mathbb{R}^n$ and a descent direction $d \in \mathbb{R}^n$ with $\nabla f(x)^T d < 0$, a step size $\alpha \in (0, 1]$ satisfying

$$f(x + \alpha d) < f(x) \tag{5}$$

gives montonically decresing functions values $\{f(x^{(k)})\}$. A step size satisfying

$$f(x + \alpha d) < f(x) + \alpha\gamma \nabla f(x)^T d \tag{6}$$

for a fixed tuning factor $\gamma \in (0, 1]$ is called an Armijo *step size.* The procedure of finding one is called Armijo *Line Search*, and may, for example, proceed by trying a monotonically descending sequence of step sizes $\alpha \in \{\beta^\ell\}$ for $\ell \geq 0$ and $\beta \in (0, 1)$, e.g. $\beta = 0.5$.

We are interested in the speed of continuous optimization algorithms, which is measured in both the number of iterations required and the CPU time per iteration required. The number of iterations depends on the rate of convergence of an algorithm.

Definition 11. (Rates of Convergence)
For a convergent sequence $\{x^{(k)}\} \subset \mathbb{R}^n$ with limit $x^* \in \mathbb{R}^n$, we say that it is *linearly convergent* if there is an iteration index $\ell \in \mathbb{N}$ such that for all later iteration indices $k \geq \ell$ the following holds:

$$\frac{||x^{(k+1)} - x^*||}{||x^{(k)} - x^*||} \leq \kappa \text{ for some constant } \kappa \in [0,1). \tag{7}$$

We say that is *superlinearly convergent* if

$$\frac{||x^{(k+1)} - x^*||}{||x^{(k)} - x^*||} \leq \kappa^{(k)} \text{ for a sequence } \{\kappa^{(k)}\} \subset [0,1). \tag{8}$$

and $\kappa^{(k)} \to 0$ as $k \to \infty$. We say that is *quadratically convergent* if

$$\frac{||x^{(k+1)} - x^*||}{||x^{(k)} - x^*||^2} \leq \omega \text{ for some constant } \omega \in [0,\infty). \tag{9}$$

Proposition 12. (Global Convergence of Line Search Gradient Descent)
The gradient descent algorithm with Armijo line search is linearly convergent from any starting point $x^{(0)} \in \mathbb{R}^n$.

This means the method is comparably slow and will typically take many thousands of iterations, the rate κ depending on both the nonlinearity and the anisotropy (eigenvalues of the Hessian) of the problem. Faster methods make use of second order information.

By working with the quadratic model $q(d)$ obtained from a second order TAYLOR expansion we obtain

$$f(x^{(k)} + d) - f(x^{(k)}) \approx \nabla f(x^{(k)})d + \tfrac{1}{2}d^T\nabla^2 f(x^{(k)})d$$

This motivates minimizing the right-hand side to find a direction d that will, in general, be different from the steepest descent direction:

$$\min_{d\in\mathbb{R}^n} \; \nabla f(x^{(k)})d + \tfrac{1}{2}d^T\nabla^2 f(x^{(k)})d$$

If $\nabla^2 f(x^{(k)})$ is positive definite (c.f. the sufficient condition of second order), the result is

$$d^{(k)} = -\left(\nabla^2 f(x^{(k)})\right)^{-1} \nabla f(x^{(k)}).$$

The difference to gradient descent is obvious: The inverse of the Hessian applys a basis transformation to the antigradient. Only if the curvature information happens to be identity will the two approaches yield identical steps.

Definition 13. (NEWTON[5] Type Directions)
Given (an approximation of) the Hessian of the objective $B \approx \nabla^2 f(x)$ and assuming that B is invertible, the direction $d \in \mathbb{R}^n$ that minimizes the directional derivative for a direction of unit length in the B-norm

$$\begin{aligned} &\min_{d\in\mathbb{R}^n} \; \nabla f(x)^T d \\ &\text{s.t. } ||d||_{B,2} = 1 \end{aligned} \tag{10}$$

is called the **Newton-type direction** of f in the point x for (the approximation) B (with respect to the Euclidean norm). It is given by

$$d = -B^{-1}\frac{\nabla f(x)}{||\nabla f(x)||_2}. \tag{11}$$

[5] ISAAC NEWTON, British mathematician

Definition 14. (Newton-Type Method)
An algorithm producing a sequence of iterates $\{x^{(k)}\}$ by letting

$$x^{(k+1)} \leftarrow x^{(k)} - \alpha^{(k)} {B^{(k)}}^{-1} \nabla f(x^{(k)}) \tag{12}$$

with step sizes $\alpha^{(k)} \in (0,1]$ is called a Newton-*Type* algorithm.

If $B = \nabla^2 f(x)$, we call this the *exact or classical* Newton *method*. Depending on the source and type of an approximation $B \approx \nabla^2 f(x)$, one speaks of *Quasi*-Newton or Newton-Type methods. Popular approximations include the BFGS update, the SR-1 update, or the Gauss-Newton approximation in case of least-squares objectives f.

Proposition 15. (Global Convergence of Newton-Type Methods)
A Newton-Type algorithm with Armijo line search is linearly convergent from any starting point $x^{(0)} \in \mathbb{R}^n$ if $B = Id$ is choosen whenever the Newton-type direction $d_N = -{B^{(k)}}^{-1} \nabla f(x^{(k)})$ is not a descent direction. In a suitable (possibly small) neighborhood of a minimizer x^*, it is superlinearly convergent if the $\{B^{(k)}\}$ are certain approximations of the exact Hessian, and it is quadratically convergent if the $\{B^{(k)}\}$ are the exact Hessians of the objective f.