Part III

Optimization

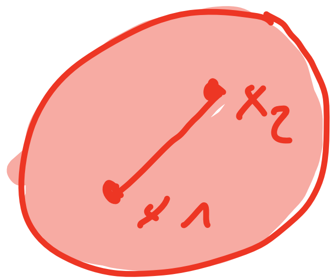
Convex sets and functions

Convex set

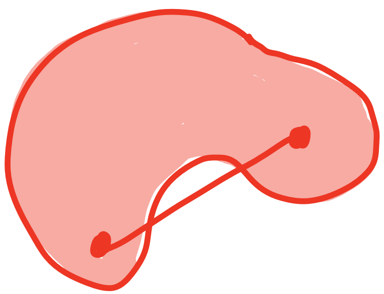
Consider a set $\Omega \subset V$ in a vector space V. The set Ω is called convex if for all $t \in [0,1]$ and for all $x_1, x_2 \in \Omega$,

$$t x_1 + (1-t) x_2 \in \Omega$$

Intuition: the line connecting x_1, x_2 is completely inside Ω.



convex

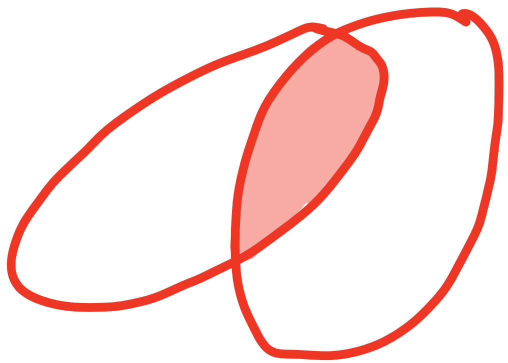


not convex

Intersections of convex sets are convex

Easy to see from the definition:

If A, B are two convex sets, then also $A \cap B$ is convex.



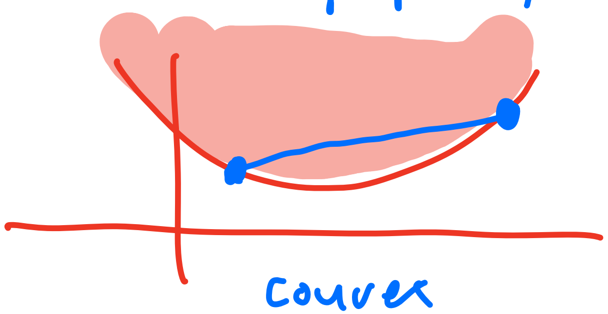
Convex function

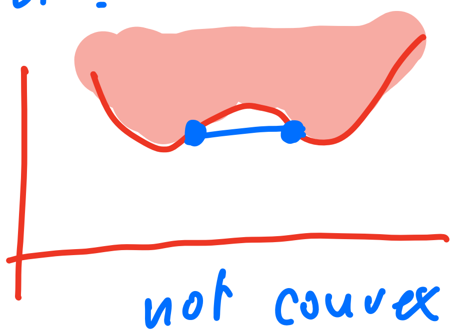
- Consider a vector space V, a set $\Omega \subset V$, and a function $f: \Omega \to \mathbb{R}$. The function f is called convex if

  strictly convex

  - Ω is a convex set
  - $\forall t \in [0,1]: \quad f(tx + (1-t)y) \leq t f(x) + (1-t) f(y)$ <

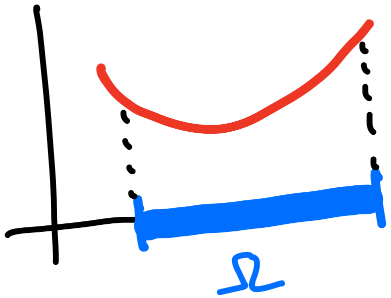
Intuition: the line connecting $(x, f(x))$ and $(y, f(y))$ is "above" the graph of the function.

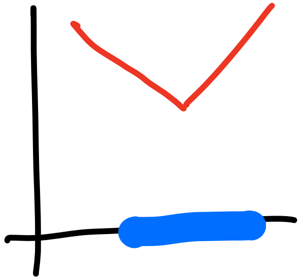


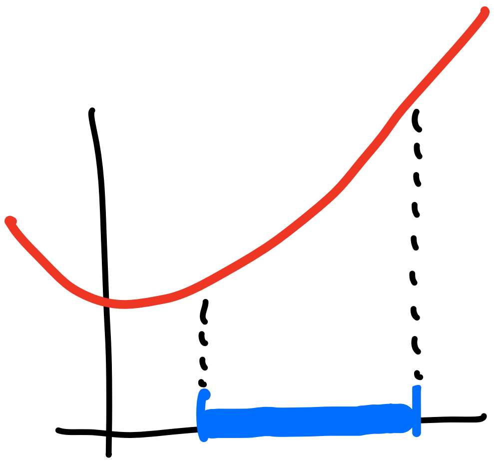


Examples

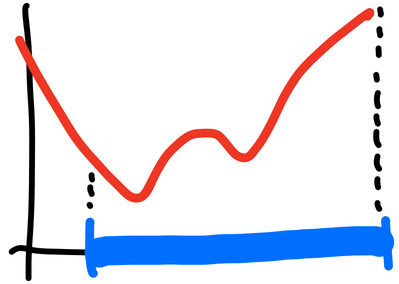
convex:

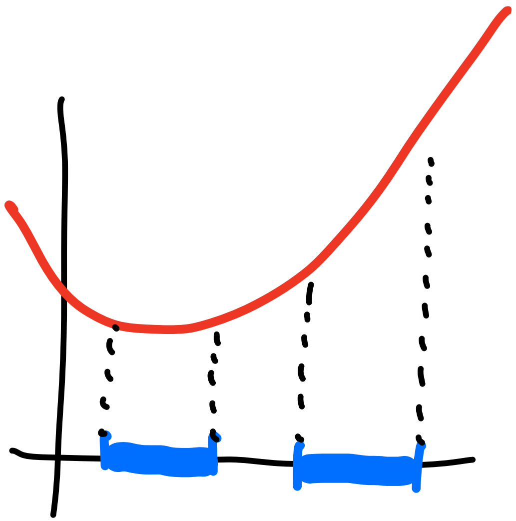






not convex

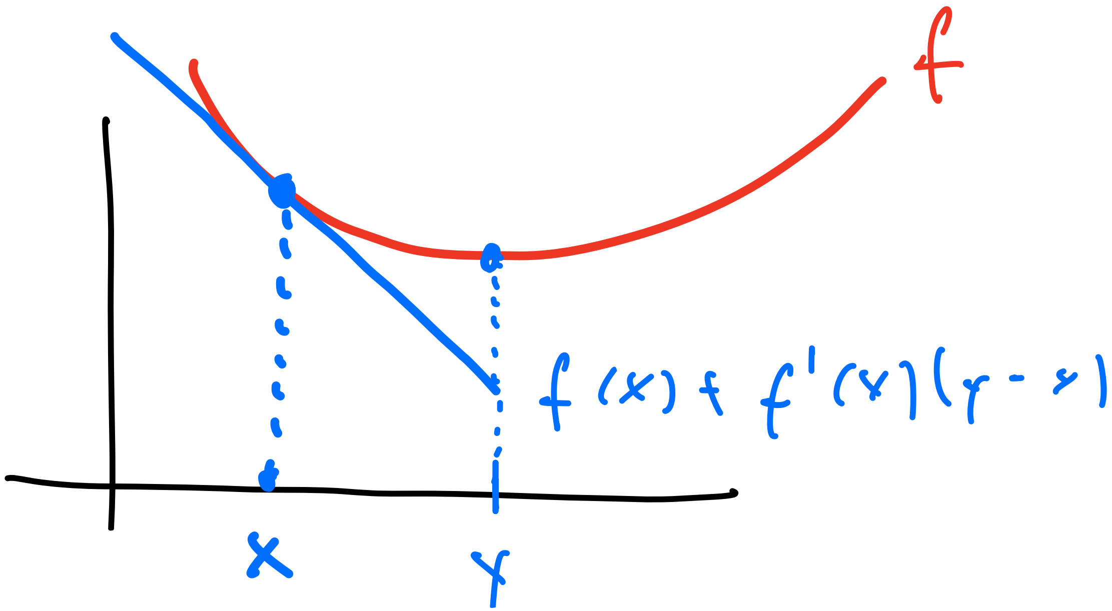




Differentiable, convex fcts

Prop A differentiable function $f: \mathbb{R}^d \to \mathbb{R}$ is convex iff for all $x, y \in \mathbb{R}^d$,

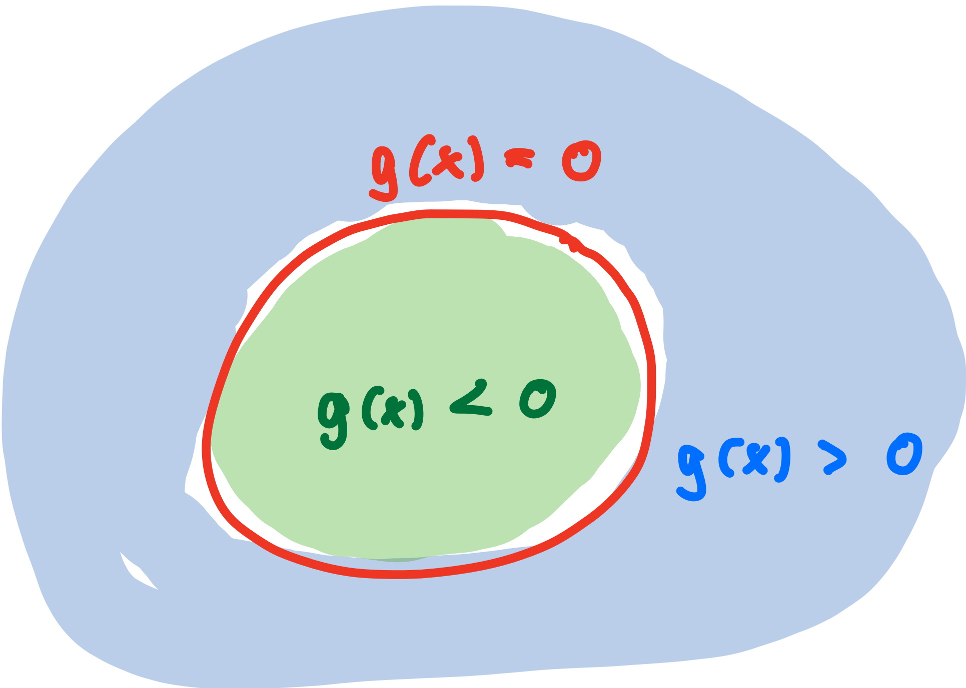
$$f(y) \geq f(x) + \nabla f(x)(y-x)$$

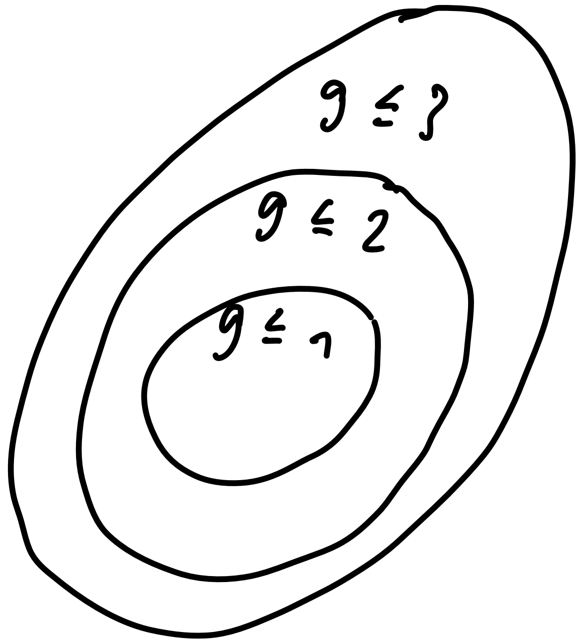


f convex $\Rightarrow$ sublevel sets convex

Observation: for convex function g, the sublevel sets

$S_a := \{ x \mid g(x) \leq a \}$ are convex.





(Funnily it is not true the other way round: a function can have all sublevel sets convex, while the function itself is not convex).

Strongly convex fct

Consider a fct defined on a convex domain. For simplicity let us assume that it is differentiable. We say that

f is μ-strongly convex iff for all $x, y \in \mathbb{R}^d$

$$f(y) \geq f(x) + \nabla f(x)(y-x) + \frac{\mu}{2}\|y-x\|_2^2$$

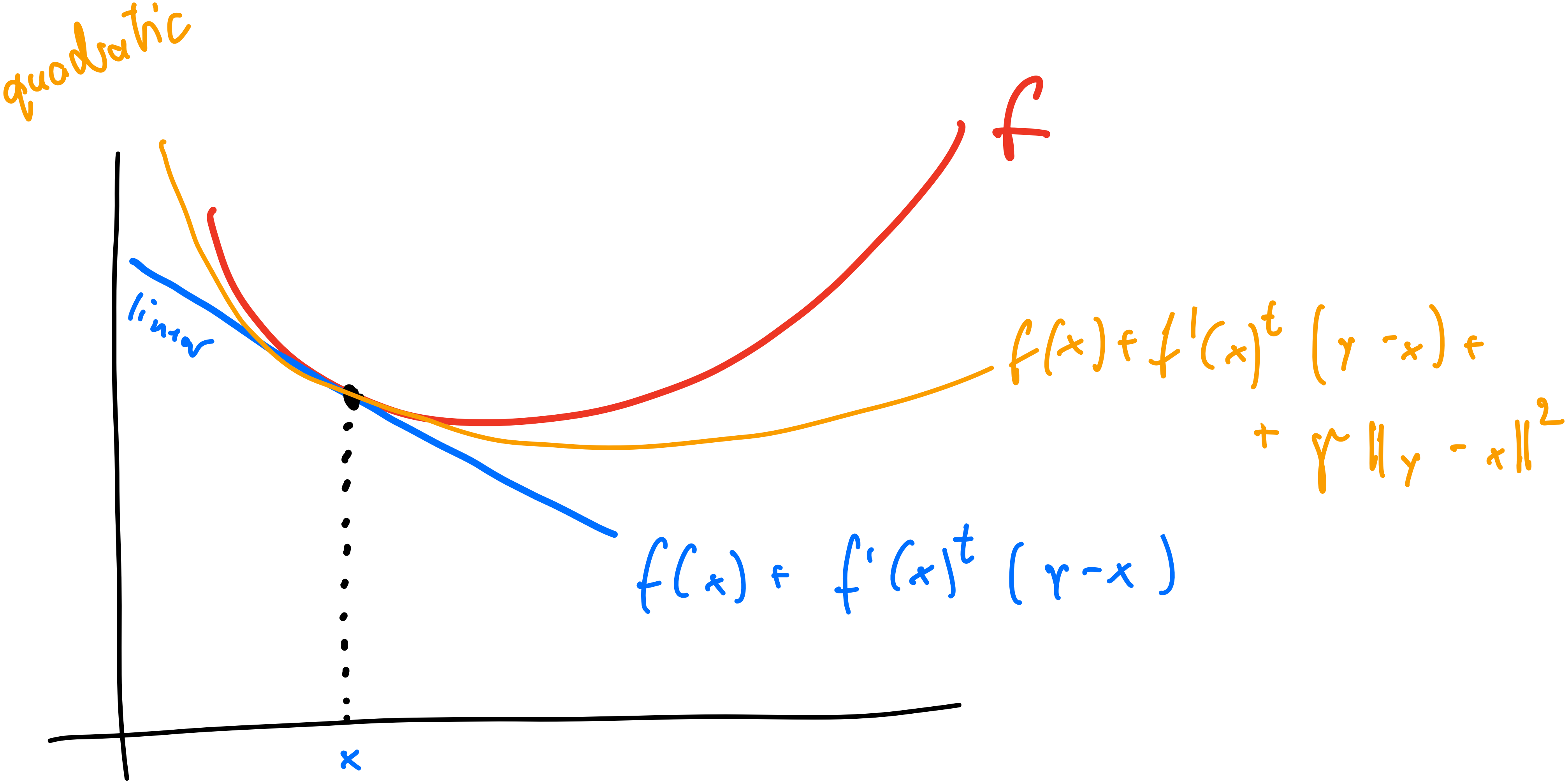
Intuition: "lower bound μ on the curvature"

Intuition on $\mathbb{R}$:

- a linear function is convex but not strongly convex
- if f is twice differentiable, then f is convex iff $f'' \geq 0$.

  f is μ-strongly convex if $f'' > 0$ with $f'' \geq \mu$.

Strongly convex function

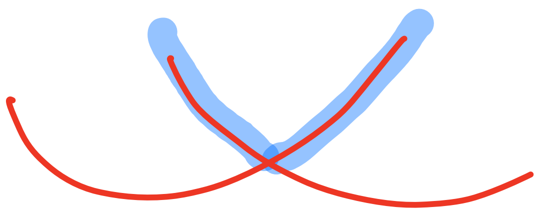


function f is "above" the quadratic approximation of f

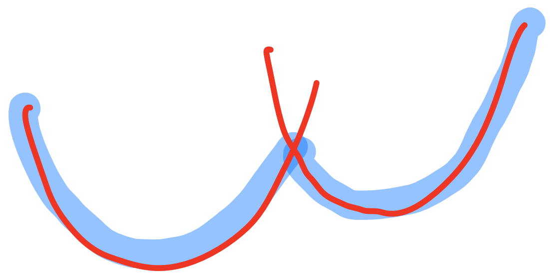
Operations that preserve convexity of functions

- **Weighted sums:** $f_1, \dots, f_n$ convex, $w_1, \dots, w_n > 0$. Then $f := \sum w_i f_i$ is convex.

- **Pointwise max:** f_1, f_2 convex. Then $g(x) := \max\{f_1(x), f_2(x)\}$ is convex



pointwise maximum is convex



⚠ pointwise minimum is <u>not</u> convex

L-Lipschitz and L-smooth

- A function f is called L-Lipschitz if there exists a constant $L > 0$ such that

$$|f(u) - f(v)| \leq L \cdot \|u - v\| \quad \text{for all } u, v \in \Omega$$

If f is differentiable this is equivalent to $\|\nabla f\| \leq L$.

Intuition: "upper bound on steepness"

- A differentiable function is called L-smooth if its gradient (!) is L-Lipschitz.

"upper bound L on the curvature"

Convexity and second derivatives

$\Rightarrow$ Hessian symmetric

<u>Proposition</u>: $f : \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^d$ open, convex. Assume that f is twice continuously differentiable. Then:

- f is convex iff the Hessian of f is positive semi-definite $\forall x$.
- f is strictly convex iff the Hessian positive definite.
- f is strongly convex with param μ if the the smallest eigenvalue λ_{min} of the Hessian satisfies

$$\lambda_{min}(H(x)) \geq \mu \quad \text{for all } x \in \Omega.$$

<u>Proof</u>: exercise

Convexity and first derivatives

Proposition

Let $f : \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^d$ be continuously differentiable and Ω an open convex set, then:

- f is convex iff $\forall x, y \in \Omega$:
  $$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$$

- f is strictly convex iff we have strict inequality.
- f is strongly convex with parameter μ iff $\forall x, y \in \Omega$,
  $$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|x - y\|_2^2$$

Proof exercise

Condition number

lower bound on curvature

upper bound on curvature

If f is μ-strongly convex and β-smooth, and is twice differentiable, then all eigenvalues of the Hessian are in the interval $[\mu, \beta]$.

We then denote by $\kappa := \beta/\mu \geq$ the "condition number"

(often it is also defined with the Hessian directly, or ratio of the largest to smallest eigenvalue of the Hessian)

Will see: important for gradient descent.

Jensen's inequality

Intuition: Let f be convex.

- Convex: if $\lambda_1 + \lambda_2 = 1$, $\lambda_1, \lambda_2 \geq 0$ we have

$$f\left(\sum_{i=1}^{2} \lambda_i x_i\right) \leq \sum_{i=1}^{2} \lambda_i f(x_i)$$

- Can extend this to finite sums: if $\sum_{i=1}^{m} \lambda_i = 1$, $\lambda_i \geq 0$, then

$$f\left(\sum_{i=1}^{m} \lambda_i x_i\right) \leq \sum_{i=1}^{m} \lambda_i f(x_i)$$

- Can extend this to integrals:

(arbitrary measure)

$$f\left(\int_S x \, dP(x)\right) \leq \int_S f(x) \, dP(x)$$

[or $f(E(x)) \leq E(f(x))$ where E is the expectation]

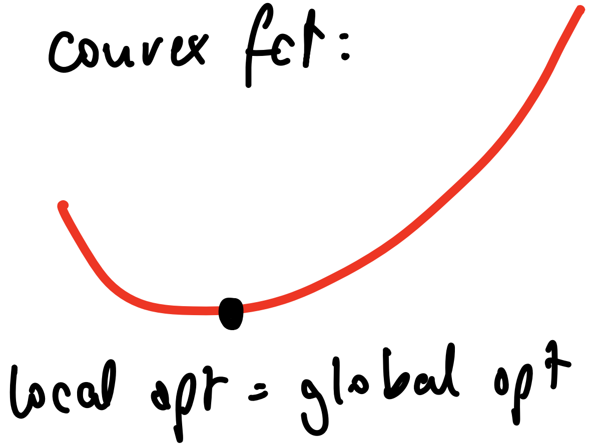
This is called Jensen's inequality for convex functions.

Convex functions and global optima

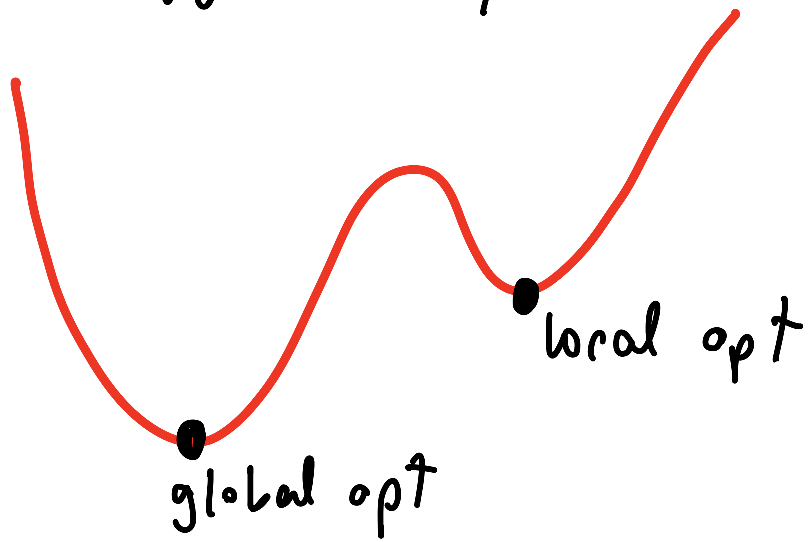
Theorem:

Any local minimum of a convex function is also a global minimum!

convex fct:



non-convex fcts



Optimization problems

General optimization problem

objective function

$$\min_{x \in D} \; f(x)$$

domain of f

$$\text{subject to} \quad g_i(x) \le 0 \quad (i = 1, \dots, r)$$

inequality constraint

$$h_j(x) = 0 \quad (j = 1, \dots, s)$$

equality constraint

A point $x \in D$ that satisfies all constraints is called feasible.

A minimizer is typically denoted by x^*, the optimal value as f^*.

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

Convex optimization problem

An optimization problem is called convex if

- the objective fct f is convex (and defined on a convex domain)
- the inequality constraint fcts g_i are convex
- the equality constraint fcts h_j are linear:

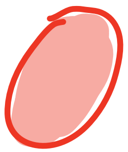
$$\langle a_j, x \rangle - b_j = 0$$

Feasible set is convex

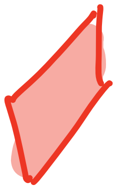
The set of feasible points of a convex optimization problem is convex:

- domain D is convex

$g(x) \le 0$



- For each inequality constraint $g_i(x) \le 0$, the set $\{x \mid g_i(x) \le 0\}$ is convex.



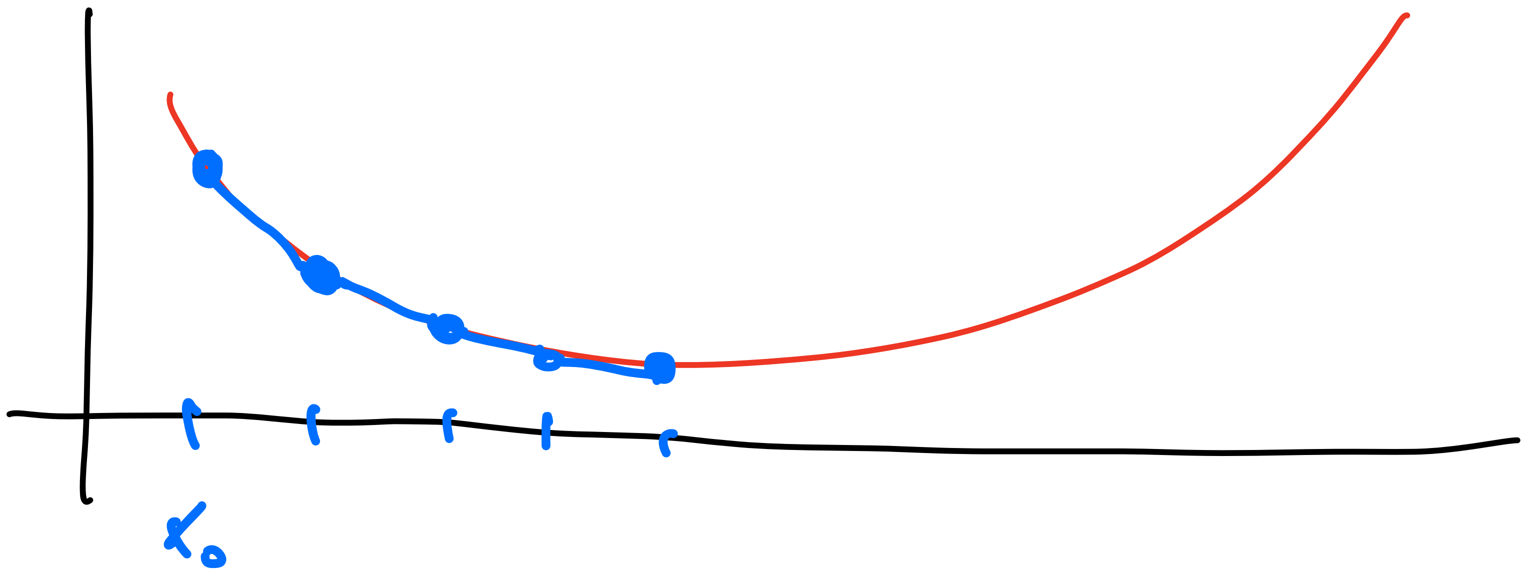
- For each equality constraint $h_j(x) = 0$, the set $\{x \mid h_j(x) = 0\}$ is convex

- The intersection of convex sets is convex.

Standard algorithms to solve convex problems

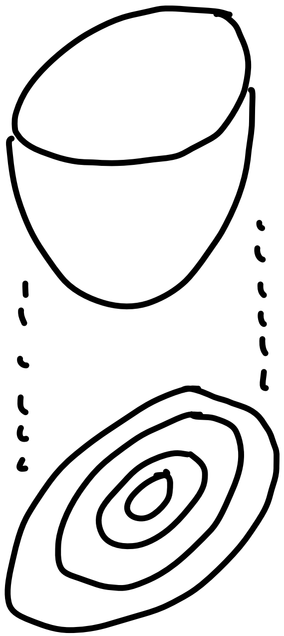
... some variant of gradient descent,

see next lecture ...



What if the set of feasible points is not convex?

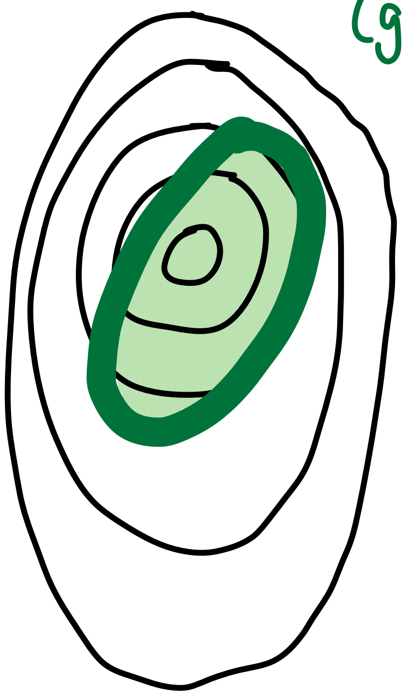
objective function



level sets of objective fct

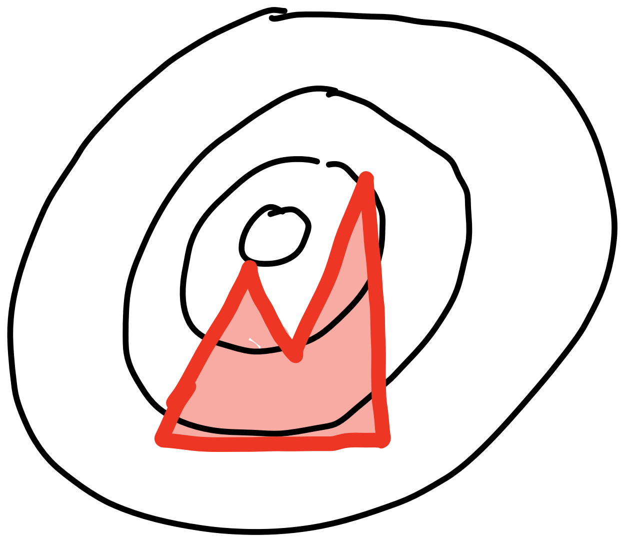
feasible points form a convex set

(good!)



gradient descent works!

feasible points form a non convex set (bad)



gradient descent might fail to find the global optimum

Local, global, unique solutions

Theorem Consider a convex optimization problem. Then:

- Any locally optimal point is also globally optimal.
- If the objective fct f is strictly convex and there exists a global optimum x^*, then it is unique.

Linear optimization problem

Given $c \in \mathbb{R}^d$, $A \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^d$, a linear program in standard form looks as follows:

$$\min_{x \in \mathbb{R}^d} \quad c^t x$$

linear objective fct

$$\text{subject to} \quad Ax - b \leq 0$$

linear constraints (read the inequality component-wise)

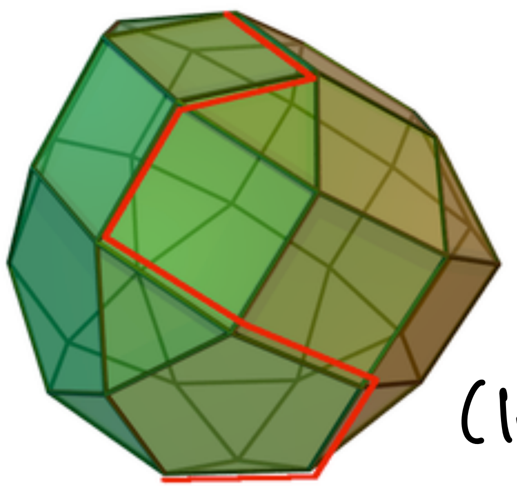
A linear program is a particular case of a convex optimization problem.

Standard alg. to solve linear programs

The **simplex algorithm:** it exploits the geometric structure of the domain

- domain is a simplex (because of linear constraints)
- optimum sits in one of the corners

Simplex alg jumps from corner to corner in a clever way



(Image: wikipedia)

Quadratic optimization problem

Given $c \in \mathbb{R}^d$, $Q \in \mathbb{R}^{d \times d}$ symmetric, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$,
$B \in \mathbb{R}^{k \times d}$, $\gamma \in \mathbb{R}^k$

a quadratic program in standard form is given as

$$\min_{x \in \mathbb{R}^d} \; \frac{1}{2} x^t Q x + c^t x \qquad \text{quadratic objective}$$

$$\text{s.t.} \quad A x \leq b \qquad \text{linear constraints}$$

$$B x = \gamma$$

If Q is positive definite, then the problem is convex.

Quadratic optimization with quadratic constraints

⚠ If the constraints are quadratic, the problem is not always convex

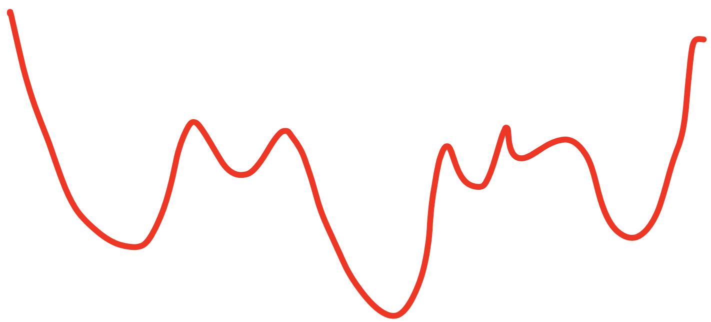
QCQP
quadratically
constraint
quadratic
program

Different kinds of optimization problems

We often distinguish different kinds of optimization problems:

(• discrete or not)

- Differentiable (do first derivatives exist)? gradients, first order methods
- Twice differentiable (do second derivatives exist)? Hessians, second-order methods
- convex / non-convex
- Does the function just have one (global) minimum or do local minima exist?



Equivalent optimization problems

Consider the optimization problem

$$\min_{x \in \mathbb{R}} f(x) \quad \text{subject to} \quad x \in C$$

Let $h : \mathbb{R} \to \mathbb{R}$ be a monotone function and consider

$$\min_{x \in \mathbb{R}} h(f(x)) \quad \text{s.t.} \quad x \in C$$

The two are equivalent optimization problems: they have the same solutions.

Example: $\sqrt{x}$ is not convex, but

$$(\sqrt{x})^4 = x^2 \quad \text{is convex}$$

Change of variables

Consider a bijective mapping $\Phi : \mathbb{R} \to \mathbb{R}$, and assume that its image covers the set $C \subset \mathbb{R}$. Then the two problems

$$\min_x f(x) \quad \text{s.t.} \quad x \in C$$

$$\min_y f(\Phi(y)) \quad \text{s.t.} \quad \Phi(y) \in C$$

are equivalent (have the same solutions)

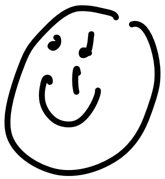
Example: $\min x + y$ s.t. $x^2 + y^2 = 1$

use polar coordinates: $x = \cos\theta$, $y = \sin\theta$

Then the problem is $\min \cos\theta + \sin\theta$ without any constraint!

(constraint is automatically satisfied, we got rid of it)

Equivalent optimization problems

 There are many other ways to transform optimization problems.

Eliminating equality constraints, introducing slack variables, ...

In practice, it can make a big difference!

→ see machine learning lectures for many examples...

First vs. second order methods

First order methods exploit gradient information to find a direction of descent:

- gradient descent (GD)
- stochastic gradient descent (SGD)

Second order methods additionally exploit the second derivatives (Hessian) to determine the step size:

- Newton
- BFGS

Constraint convex optimization:

primal, dual, Lagrangian

Literature: Boyd: Convex optimization.

Lagrangian: intuitive point of view

Lagrange multiplier for equality constraints

Consider the following convex optimization problem:

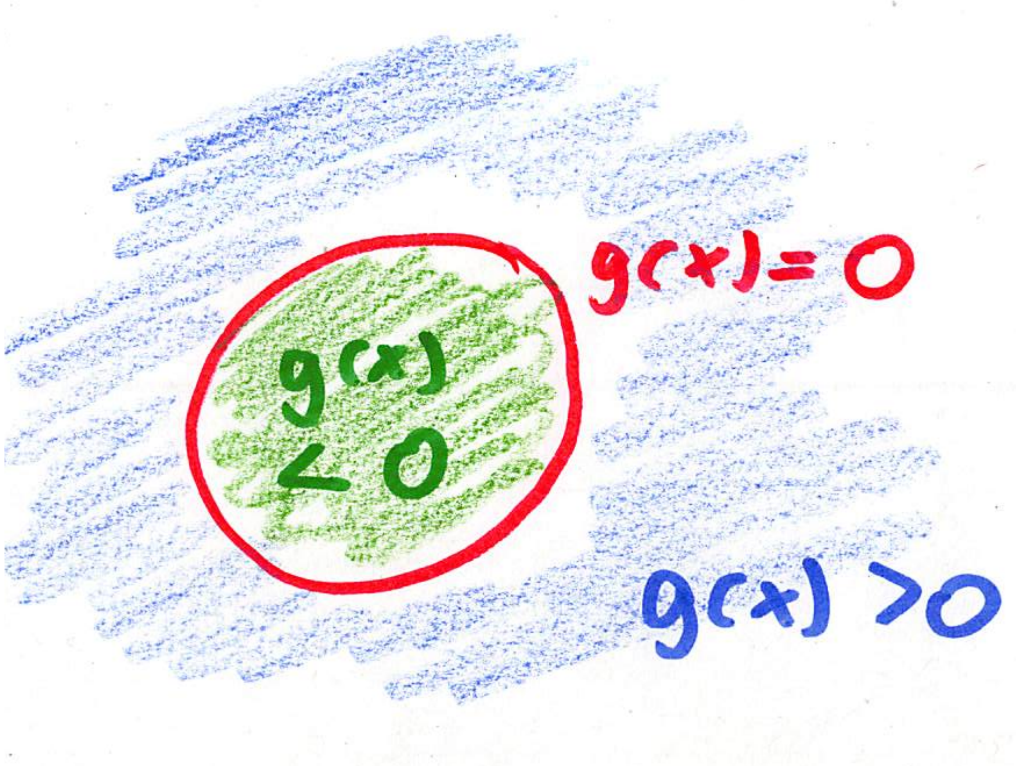
$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g(x) = 0 \end{aligned}$$

where f and g are convex.

We make several simplifying assumptions for now
(e.g., convexity), in order to get an intuition
for the Lagrange approach.
Later we then prove everything formally, without many of the assumptions.

Lagrange multiplier for equality constraints (2)

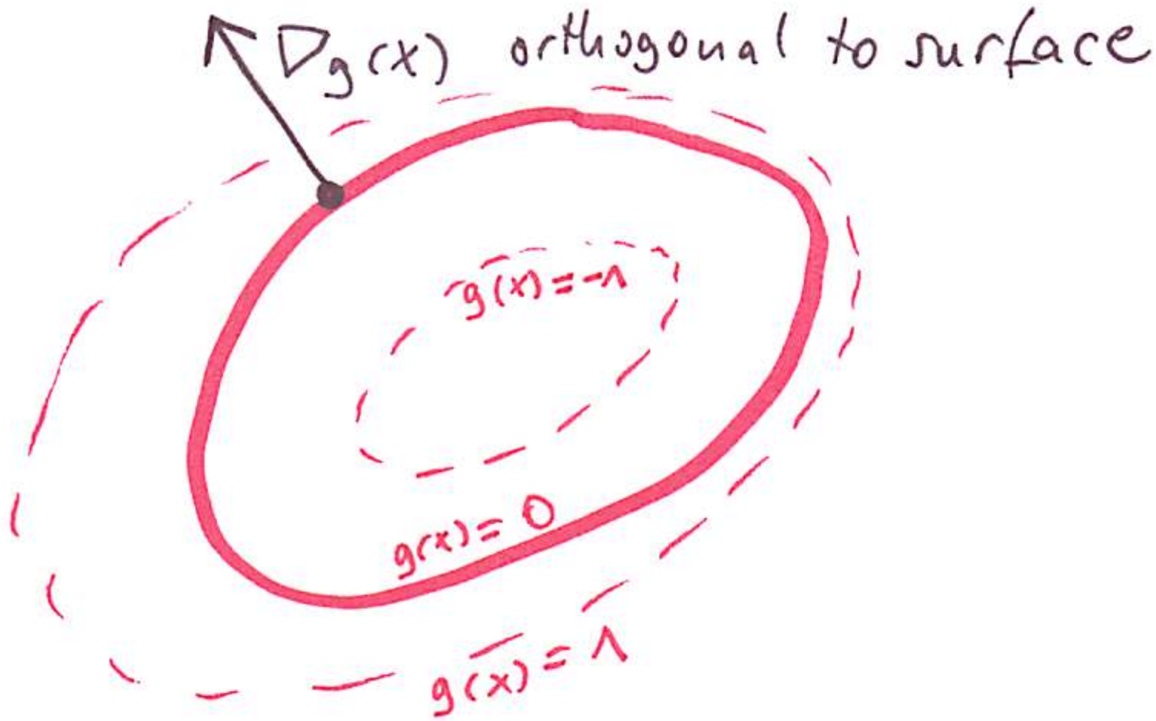
Recall: if g is convex, then it sublevel-sets are convex:



Sublevel set: $\{x|g(x) \leq c\}$ (the green set in the figure)

Lagrange multiplier for equality constraints (3)

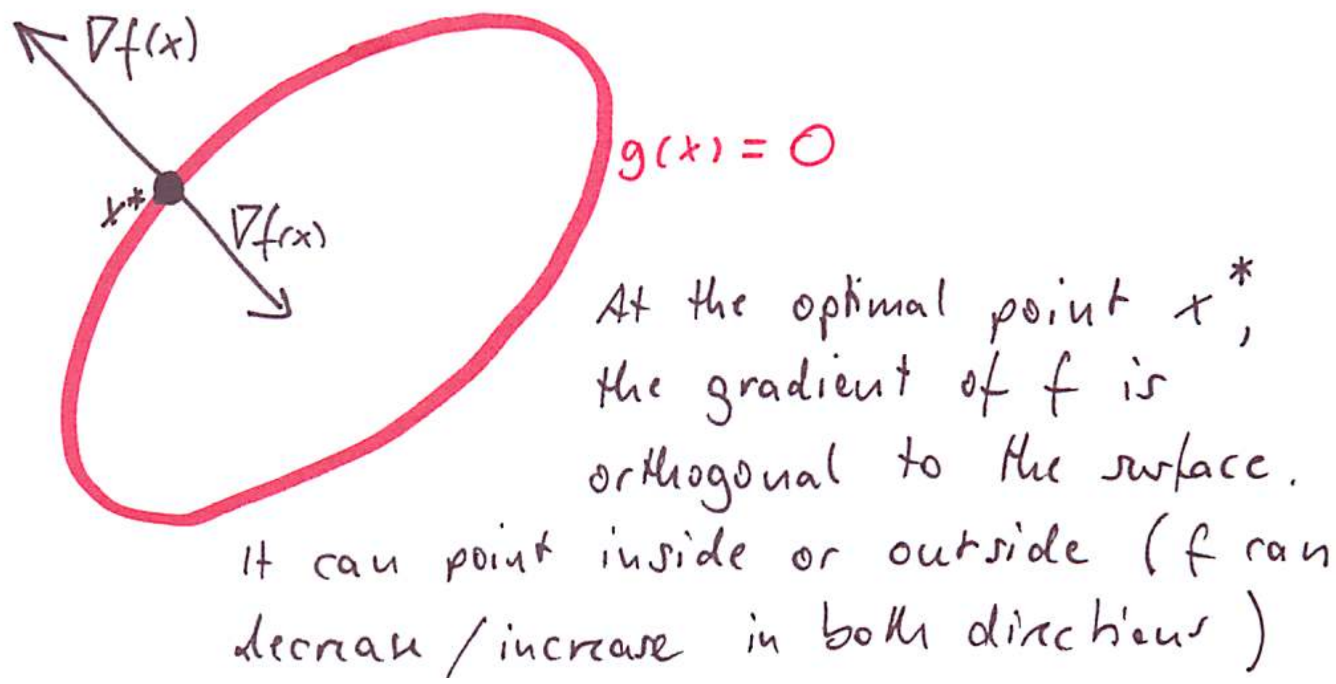
Gradient (equality constraint): For any point x on the "surface" $\{g(x) = 0\}$ the gradient $\nabla g(x)$ is orthogonal to the surface itself.



Intuition: to increase / decrease $g(x)$, you need to move away from the surface, not walk along the surface.

Lagrange multiplier for equality constraints (4)

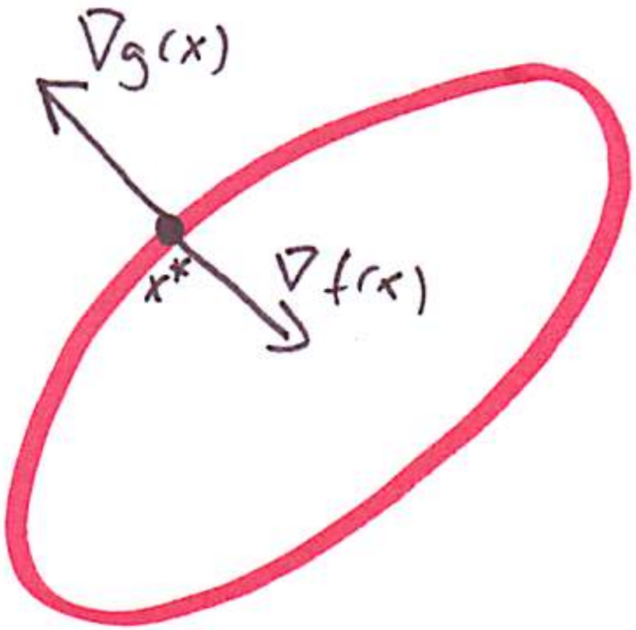
Gradient (objective function): Consider the point x^* on the surface $\{g(x) = 0\}$ for which $f(x)$ is minimized. This point must have the property that $\nabla f(x)$ is orthogonal to the surface.



Intuition: otherwise we could move a little along the surface to decrease $f(x)$.

Lagrange multiplier for equality constraints (5)

Conequence: at the optimal point, $\nabla g(x)$ and $\nabla f(x)$ are parallel, that is there exists some $\nu \in \mathbb{R}$ such that $\nabla f(x) + \nu \nabla g(x) = 0$.



At the optimal point x^*, ∇g and ∇f are parallel to each other (and can point either in the same or the opposite direction).

Lagrange multiplier for equality constraints (6)

We now define the Lagrangian function

$$L(x, \nu) = f(x) + \nu g(x)$$

where $\nu \in \mathbb{R}$ is a new variable called Lagrance multiplier. Now observe:

- The condition $\nabla f(x) + \nu \nabla g(x) = 0$ is equivalent to $\nabla_x L(x, \nu) = 0$
- The condition $g(x) = 0$ is equivalent to $\nabla_\nu L(x, \nu) = 0$.

To find an optimal point x^* we need to find a saddle point of $L(x, \nu)$, that is a point such that both $\nabla_x L(x, \nu)$ and $\nabla_\nu L(x, \nu)$ vanish.

Simple example

Consider the problem to minimize $f(x)$ subject to $g(x) = 0$, where $f, g : \mathbb{R}^2 \to \mathbb{R}$ are defined as

$$f(x_1, x_2) = x_1^2 + x_2^2 - 1$$
$$g(x_1, x_2) = x_1 + x_2 - 1$$

Observe: it is hard to solve this problem by naive methods because it is unclear how to take care of the constraints!

Solution by the Lagrange approach:

Write it in the standard form:

$$\text{minimize } x_1^2 + x_2^2 - 1$$
$$\text{subject to } x_1 + x_2 - 1 = 0$$

Simple example (2)

The Lagrangian is

$$L(x, \nu) = \underbrace{x_1^2 + x_2^2 - 1}_{f(x_1, x_2)} + \nu(\underbrace{x_1 + x_2 - 1}_{g(x_1, x_2)})$$

Now compute the derivatives and set them to 0:

$$\nabla_{x_1} L = 2x_1 + \nu \overset{!}{=} 0$$

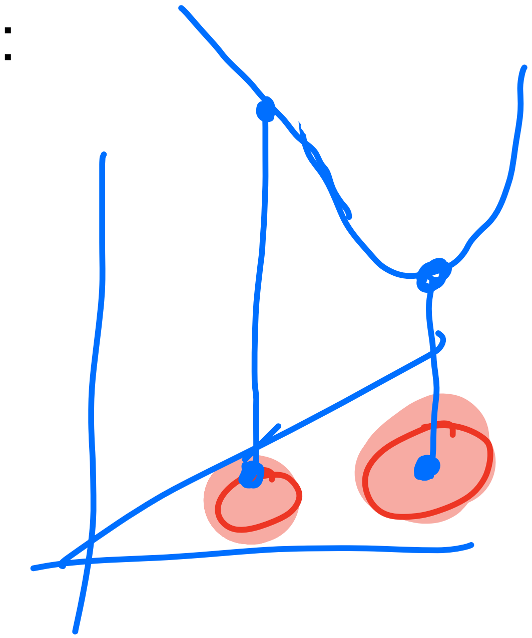
$$\nabla_{x_2} L = 2x_2 + \nu \overset{!}{=} 0$$

$$\nabla_{\nu} L = x_1 + x_2 - 1 \overset{!}{=} 0$$

If we solve this linear system of equations we obtain $(x_1^*, x_2^*) = (0.5, 0.5)$.

Lagrange multiplier for inequality constraints

Consider the following convex optimization problem:

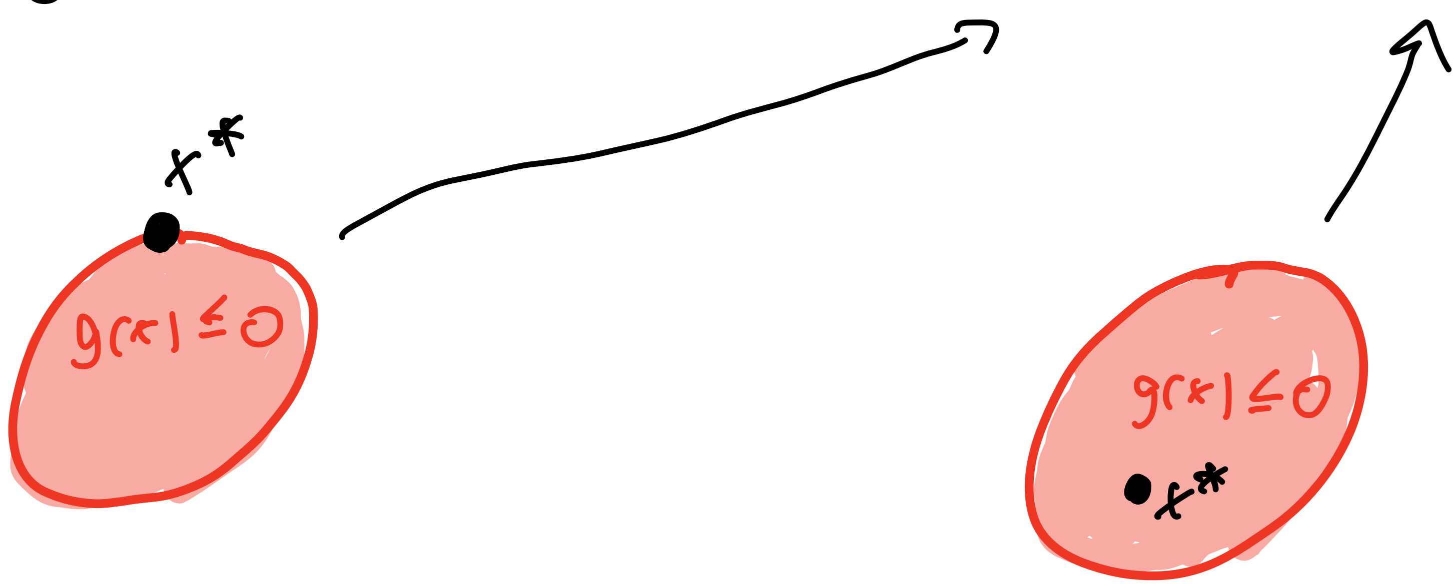


$$\text{minimize } f(x)$$

$$\text{subject to } g(x) \leq 0$$

where f and g are convex.

We now distinguish two cases: constraint is "active" or "inactive":



Lagrange multiplier for inequality constraints (2)

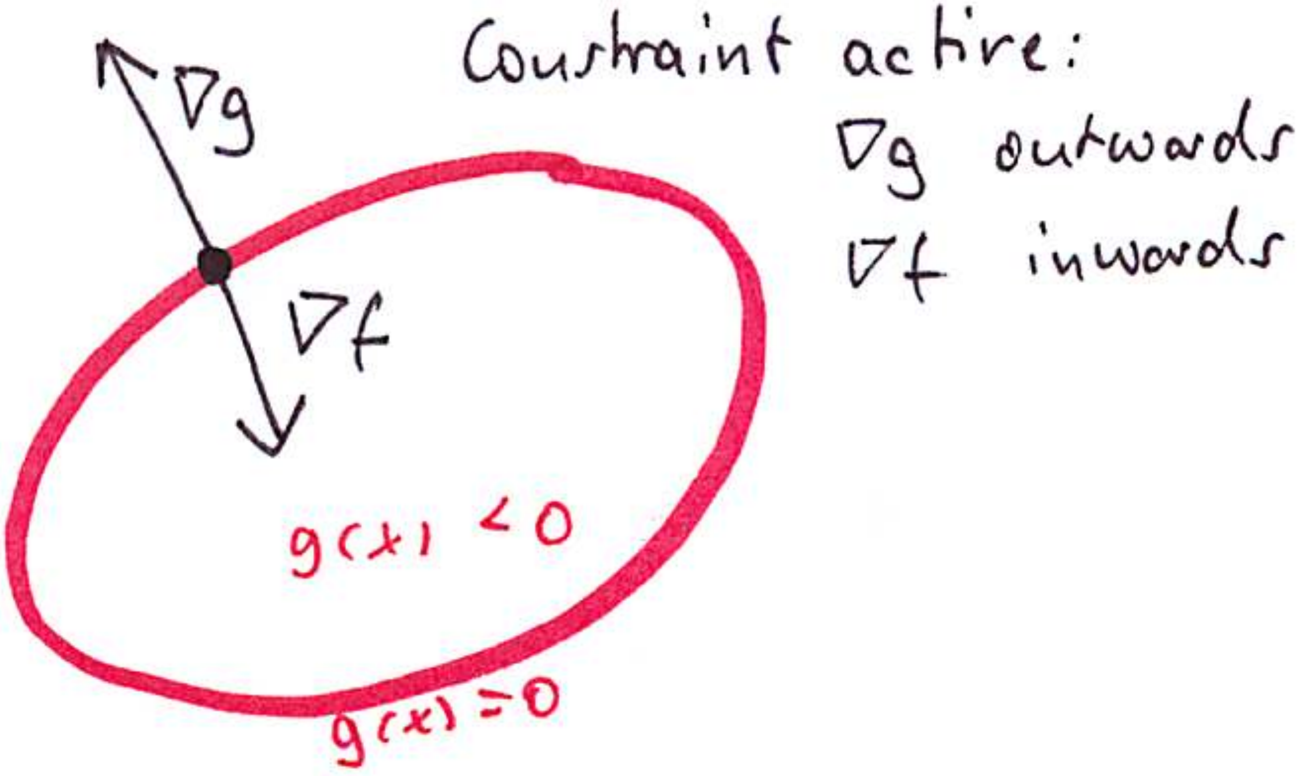
Case 1: Constraint is "active", that is the optimal point is *on* the surface $g(x) = 0$.

Again ∇f and ∇g are parallel in the optimal point.

But furthermore, the direction of derivatives matters:

- The derivative of g points outwards (at any point on the surface $g = 0$). This is always the case if g is convex.
- Then the derivative of f is directed inwards (otherwise we could decrease the objective by walking inside).

Lagrange multiplier for inequality constraints (3)



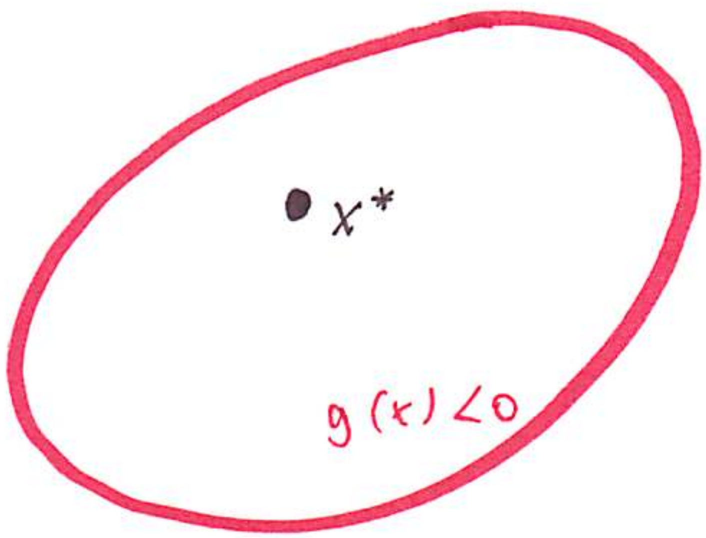
So we have $\nabla f(x) = -\lambda \nabla g(x)$ for some value $\lambda > 0$.

Lagrange multiplier for inequality constraints (4)

Case 2: Constraint is "inactive", that is the optimal point is not on the surface $g(x) = 0$ but somewhere in the interior.

- Then we have $\nabla f = 0$ at the solution (otherwise we could decrease the objective value).
- We do not have any condition on ∇g (it is as if we would not have this constraint).

Constraint inactive. No condition on ∇g



Lagrange multiplier for inequality constraints (5)

We can summarize both cases using the Lagrangian again. We now define the Lagrangian

$$L(x, \lambda) = f(x) + \lambda g(x)$$

where the Lagrange multiplier has to be positive: $\lambda \geq 0$.

- Case 1: constraint active, $\lambda > 0$.
  - Need to find a saddle point: $\nabla_x L(x, \lambda) = \nabla_\lambda L(x, \lambda) = 0$.
- Case 2: constraint inactive, $\lambda = 0$.
  - Then $L(x, \lambda) = f(x)$. Hence $\nabla_x L(x, \lambda) = \nabla_x f(x) \overset{!}{=} 0$, $\nabla_\lambda L(x, \lambda) \equiv 0$.
- So in both cases we have again a saddle point of the Lagrangian.

Lagrange multiplier for inequality constraints (6)

Also in both cases we have $\lambda g(x^*) = 0$.

- Constraint active: $\lambda > 0$, $g(x^*) = 0$.
- Constraint inactive: $\lambda = 0$, $g(x^*) \neq 0$.

This is called the Karush-Kuhn-Tucker (KKT) condition.

Simple example

What are the side lengths of a rectangle that maximize its area, under the assumption that its perimeter is at most 1?

We need to solve the following optimization problem:

$$\text{maximize}\, x \cdot y \text{ subject to } 2x + 2y \leq 1$$

Bring the problem in standard form:

$$\text{minimize}(-x \cdot y) \text{ subject to } 2x + 2y - 1 \leq 0$$

Form the Lagrangian:

$$L(x, y, \lambda) = -xy + \lambda(2x + 2y - 1)$$

Simple example (2)

Saddle point conditions / derivatives:

$$\partial L/\partial x = -y + 2\lambda \stackrel{!}{=} 0$$
$$\partial L/\partial y = -x + 2\lambda \stackrel{!}{=} 0$$
$$\partial L/\partial \lambda = 2x + 2y - 1 \stackrel{!}{=} 0$$

Solving this system of three equations gives $x = y = 0.25$.

Simple example (3)

Now need to see: when does this approach work, when does it not work, what can we prove about it?

Lagrangian: formal point of view

Lagranigan and dual: formal definition

Consider the primal optimization problem

$$\begin{aligned} &\text{minimize } f_0(x) \\ &\text{subject to } f_i(x) \leq 0 \;\; (i = 1, ..., m) \\ &\qquad\qquad\quad h_j(x) = 0 \;\; (j = 1, ..., k) \end{aligned}$$

Denote by x^* a solution of the problem and by $p^* := f_0(x^*)$ the objective value at the solution.

Lagranigan and dual: formal definition (2)

Define the corresponding Lagrangian as follows:

- For each equality constraint j introduce a new variable $\nu_j \in \mathbb{R}$, and for each inequality constraint i introduce a new variable $\lambda_i \geq 0$. These variables are called Lagrange multipliers.

- Then define

$\lambda = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{pmatrix}$ $\nu = \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_k \end{pmatrix}$

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{k} \nu_j h_j(x)$$

Define the dual function $g : \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$ by

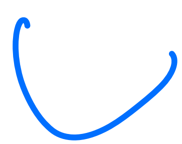
$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu)$$

Dual function as lower bound on primal

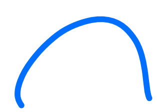
Proposition 1 (Dual function is concave)

No matter whether the primal problem is convex or not, the dual function is always concave in (λ, ν).

Proof. For fixed x, $L(x, \lambda, \nu)$ is linear in λ and ν and thus concave. The dual function as a pointwise infimum over concave functions is concave as well. ☺



convex



concave

f concave $\Leftrightarrow$ $-f$ convex

Note that concave is good, because we are going to maximize this function later on.

Dual function as lower bound on primal (2)

Proposition 2 (Dual function as lower bound on primal)

For all $\lambda_i \geq 0$ and $\nu_j \in \mathbb{R}$ we have $g(\lambda, \nu) \leq p^*$.

solution of primal

Proof.

- Let x_0 be a feasible point of the primal problem (that is, a point that satisfies all constraints).
- For such a point we have

$$\sum_{i=1}^{m} \underbrace{\lambda_i}_{\geq 0} \underbrace{f_i(x_0)}_{\leq 0} + \sum_{i=1}^{k} \nu_j \underbrace{h_j(x_0)}_{=0} \leq 0$$

Dual function as lower bound on primal (3)

- This implies

$$L(x_0, \lambda, \nu) = f_0(x_0) + \overbrace{\sum_{i=1}^{m} \lambda_i f_i(x_0) + \sum_{j=1}^{k} \nu_j h_j(x_0)}^{\leq 0} \leq f_0(x_0)$$

  Note that this property holds in particular when x_0 is x^*.

- Moreover, for any x_0 (and in particular for $x_0 := x^*$) we have

$$\underbrace{\inf_x L(x, \lambda, \nu)}_{g(\lambda,\nu)} \leq L(x_0, \lambda, \nu)$$

- Combining the last two properties gives

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) \leq L(x^*, \lambda, \nu) \leq f_0(x^*)$$

☺

Dual optimization problem

Have seen: the dual function provides a lower bound on the primal value. Finding the highest such lower bound is the task of the dual problem:

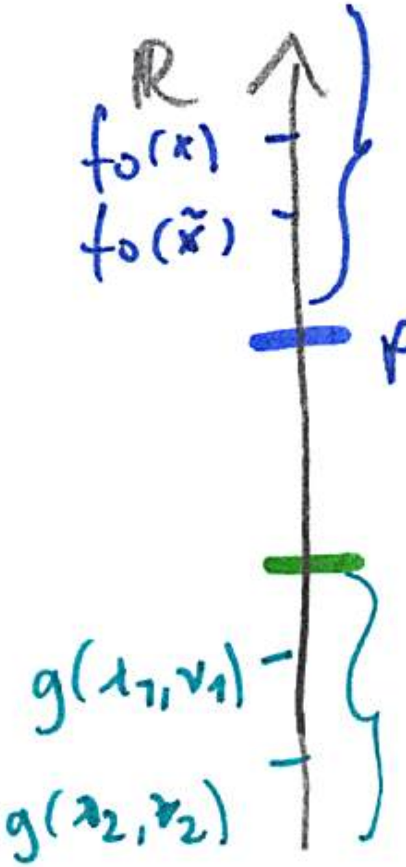
We define the dual optimization problem as

$$\max_{\lambda,\nu} g(\lambda, \nu) \text{ subject to } \lambda_i \geq 0, \nu_j \in \mathbb{R}$$

Denote the solution of this problem by λ^*, ν^* and the corresponding objective value $d^* := g(\lambda^*, \nu^*)$.

Dual optimization problem (2)

Dual vs Primal, some intuition:

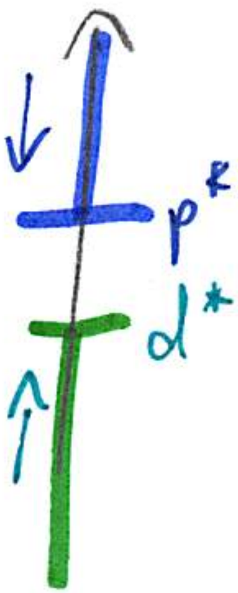


Can find upper bounds on p^* by evaluating $f_0(x)$ for diffnt values of x.

Want to minimize $f_0(x)$.

Can find lower bounds on p^* by evaluating $g(\lambda, \nu)$ for diffnt values λ, ν.

Dual optimization problem (3)



Primal problem: find best upper bound on p^*
(= find p^*)

Dual problem: find best lower bound on p^*
(= find d^*)

Weak duality

Proposition 3 (Weak duality)

The solution d^* of the dual problem is always a lower bound for the solution of the primal problem, that is $d^* \leq p^*$.

Proof. Follows directly from Proposition 2 above. ☺

We call the difference $p^* - d^*$ the duality gap.

Strong duality

- We say that strong duality holds if $p^* = d^*$.
- This is not always the case, just under particular conditions. Such conditions are called constraint qualifications in the optimization literature.
- Convex optimization problems often satisfy strong duality, but not always.

Strong duality (2)

Examples:

- Linear problems have strong duality
- Quadratic problems have strong duality
- There exist many convex problems that do not satisfy strong duality. Here is an example:

$$\begin{aligned} \text{minimize}_{x,y} \, & \exp(-x) \\ \text{subject to } & x/y \leq 0 \\ & y \geq 0 \end{aligned}$$

One can check that this is a convex problem, yet $p^* = 1$ and $d^* = 0$.

Strong duality: how to convert the solution of the dual to the one of the primal

By strong duality: $p^* = d^*$, that is we get the same objective values. But how can we recover the primal variables x^* that lead to this solution, if we just know the dual variables λ^*, ν^* of the optimal dual solution?

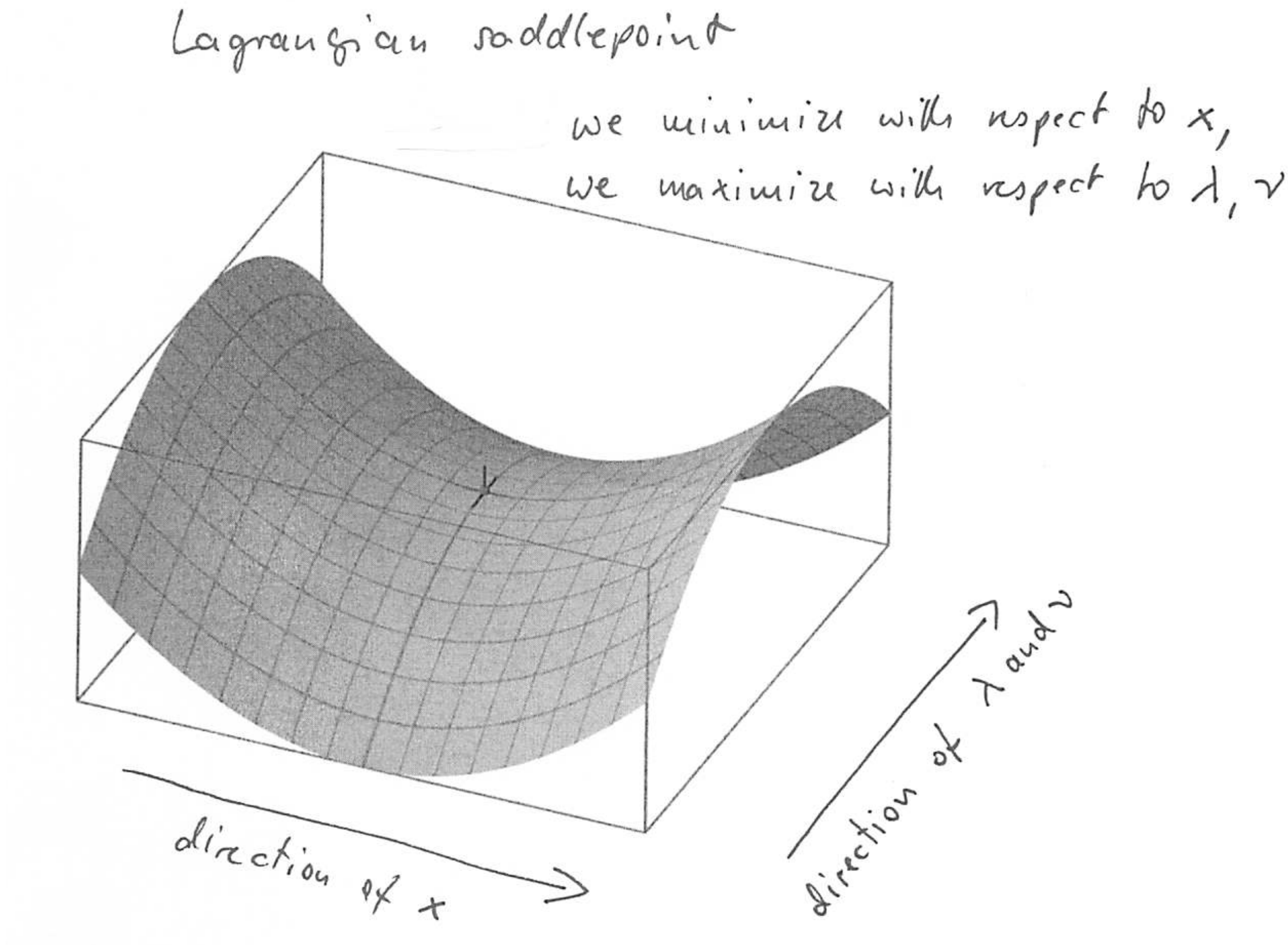
EXERCISE!

Strong duality implies saddle point

Proposition 4 (Strong duality implies saddle point)

Assume strong duality holds, let x^* be the solution of the primal and (λ^*, ν^*) the solution of the dual optimization problem. Then (x^*, λ^*, ν^*) is a saddle point of the Lagrangian.

Strong duality implies saddle point (2)



Strong duality implies saddle point (3)

Proof.

- We first show that x^* is a minimizer of $L(x, \lambda^*, \nu^*)$:
  - By the strong duality assumption we have $f_0(x^*) = g(\lambda^*, \nu^*)$.
  - With this we get

$$f_0(x^*) = g(\lambda^*, \nu^*) \overset{\text{def}}{=} \inf_x L(x, \lambda^*, \nu^*) \leq L(x^*, \lambda^*, \nu^*) \leq f_0(x^*)$$

    (last inequality follows from Proposition 3).
  - Because we have the same term on the left and side, we have equality everywhere.
  - So in particular, $\inf_x L(x, \lambda^*, \nu^*) = L(x^*, \lambda^*, \nu^*)$.
- Then we show that (λ^*, ν^*) are maximizers of $L(x^*, \lambda, \nu)$.
  - This follows from the definition of (λ^*, ν^*) as solutions of $\max_{\lambda,\nu} \underbrace{\min_x L(x, \lambda, \nu)}_{g(\lambda,\nu)}$.

Strong duality implies saddle point (4)

- Taken together we get

$$L(x^*, \lambda, \nu) \leq L(x^*, \lambda^*, \nu^*) \leq L(x, \lambda^*, \nu^*)$$

That is, (x^*, λ^*, ν^*) is a saddle point of the Lagrangian:

  - It is a minimum for x (with fixed λ^*, ν^*).
  - It is a maximum for (λ, ν) (with fixed x^*).

☺

Saddle point always implies primal solution

Proposition 5 (Saddlepoint implies primal solution)

If (x^*, λ^*, ν^*) is a saddle point of the Lagrangian, then x^* is always a solution of the primal problem.

Proof. Not very difficult, but we skip it. ☺

Remarks:

- This proposition always holds (not only under strong duality).
- This proposition gives sufficient conditions for optimality. Under additional assumptions (constraint qualifications) it is also a necessary condition.

Why is this whole approach useful?

- Whenever we have a saddle point of the Lagrangian, we have a solution of our constraint optimization problem. This is great, because otherwise we would not know how to solve it.
- If strong duality holds, we even know that any solution must be a saddle point. So if we don't find a saddle point, then we know that no solution exists.
- If your original minimization problem is not convex, at least its dual is a concave maximization problem (or, by changing the sign, a convex minimization problem). If the duality gap is small, then it might make sense to solve the dual instead of the primal (you will not find the optimal solution, but maybe a solution that is close).

Gradient descent

Literature: • Francis Bach: Learning theory from first principles. Forthcoming book, pdf online.

• Bottou, Curtis, Nocedal: Optimization methods for large scale machine learning. 2018

Gradient descent (vanilla version)

Assume we want to solve an optimization problem

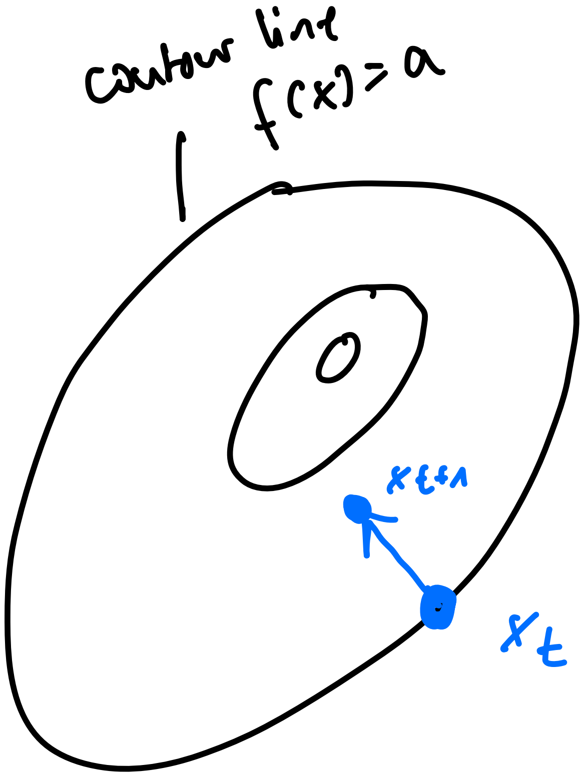
$$\min_{w \in \Omega} f(w)$$

where $f: \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^d$, f differentiable.

Gradient descent starts with a randomly chosen start point w_0 and then makes a small step in the direction opposite of the gradient $\nabla f(w_0)$. of step size γ_t:

$$\boxed{w_{t+1} = w_t - \gamma_t \nabla f(w_t)}$$

Intuition: Gradient descent and contour lines



Observe:

- gradient of a function is orthogonal to its contour lines.
- Gradient descent steps are orthogonal to contour lines.

Why does it help if f is differentiable?

Well, we want to compute gradients...

If we don't have gradients, our life really becomes hard.

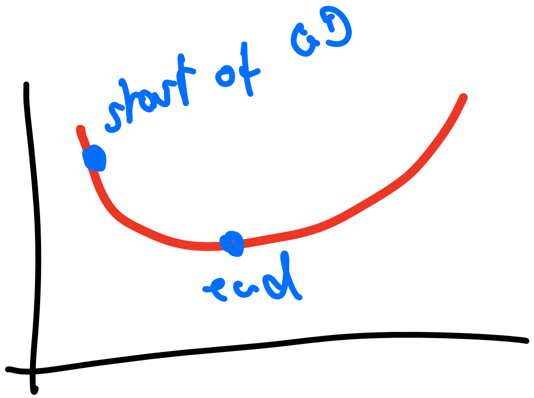
In ML, we always construct our problems in such a way that the loss fct is differentiable in the end. We might sacrifice many things when modeling a problem, but not differentiability.

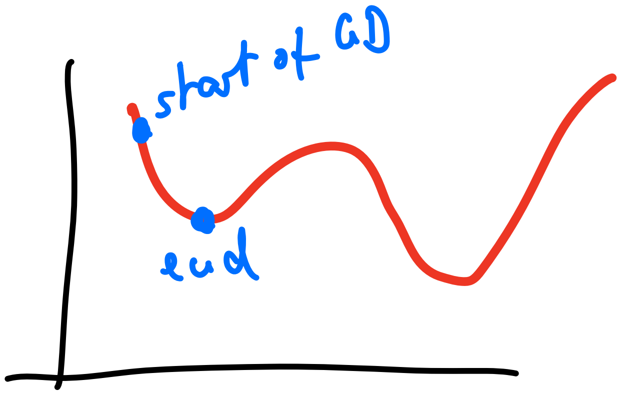
($\rightsquigarrow$ surrogate loss functions)

Why does convexity help?

For non-convex fct, GD typically finds local optima, but not the global one.

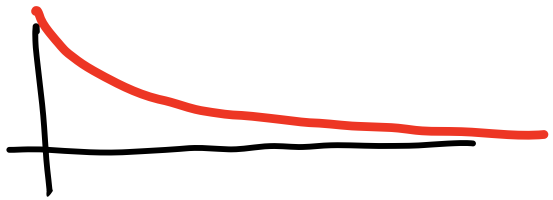
If f is convex, we know we found the global optimum.



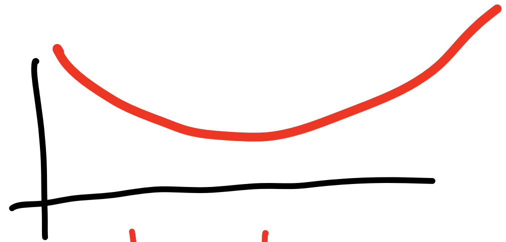


Why can it help if we have strong convexity?

Functions that are strongly convex cannot have long, "arbitrarily flat" parts. On such parts, GD would take forever. But if f is strongly convex, this does not exist.



convex but not strongly convex



strongly convex.

Mathematically, for strongly convex f we can estimate how far we are at most from the optimal pt.

In ML, we can sometimes achieve strong convexity through regularization.

Why does smoothness help?

- GD makes steps in direction of gradient.
- but if gradient itself changes wildly, then already after a small step we can be in trouble.
- If f is L-smooth, gradient only changes slowly, so gradient step makes sense.
- The smoother f, the larger steps we can dare to take.

  for example in the theorem below we choose a constant step size $\gamma_t = \frac{1}{L}$

  (in practice, one often decreases the step size over time, see below)

Recap: Condition number

If f is μ-strongly convex and β-smooth, and is twice continuously differentiable, then all eigenvalues of the Hessian are in the interval $[\mu, \beta]$.

We then denote by $K := \beta/\mu \geq$ the "condition number"

(often it is also defined with the Hessian directly, or ratio of the largest to smallest eigenvalue of the Hessian)

It always holds that $\mu \leq \beta$. Thus $K \geq 1$

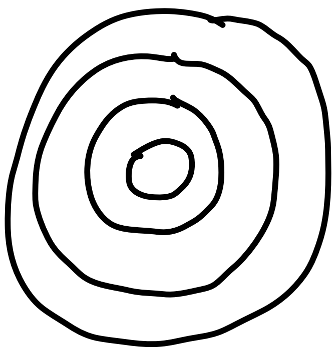
Intuition for condition number

Condition number small $\Rightarrow \mu \simeq \beta$

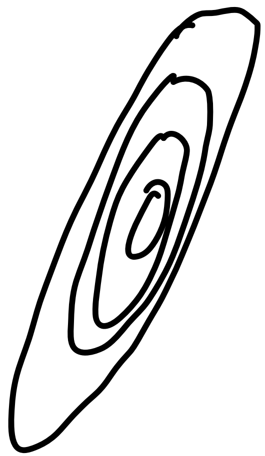
$\Rightarrow$ contour lines of the function are close to a circle

Condition number large

$\Rightarrow$ contour lines very "elongated"

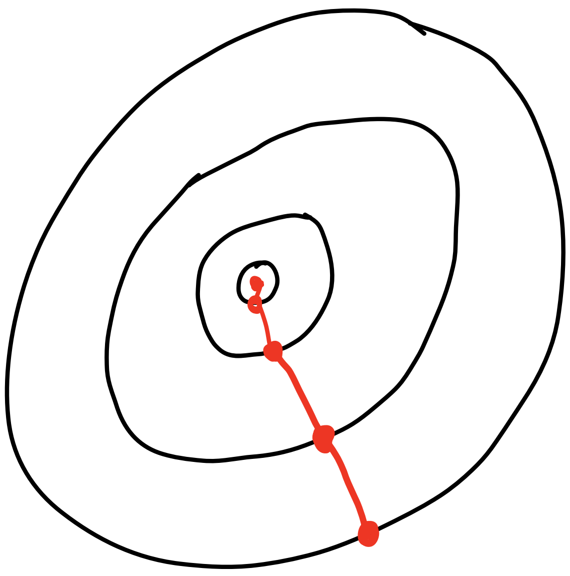


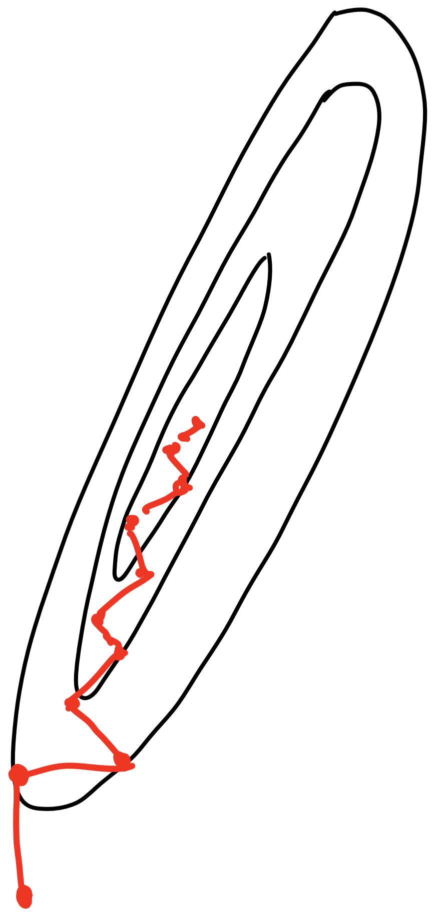
K small



K large

Intuition: speed of convergence





κ small, GD converges fast

Steps orthogonal to the contour lines take us pretty straight to the center.

κ large, GD converges slowly

Steps orthogonal to the contour lines jump wildly

Choosing a starting point

- Typically, a random point. Parameters of the initialized with Gaussian noise of "the correct size"

- Sometimes one uses a "warm start": use a first heuristic to guess a good starting point.
  (note that this is not what fine tuning is, see next slide)

Fine tuning

In complex models: Somebody trains, say, an image classifier or a language model on large amounts of data. We take the trained model and would like to run GD directly on the model — but typically we don't have access to the original architecture and parameter space. Then we just use the representation that has been learned by the original model and train a second classifier on top.

Stopping conditions

- Once you observe that the objective doesn't change a lot ... a bit unclear in practice.

- In deep learning, people often continue training even though the training error is pretty much 0 ... representation still might change.

⚠ As opposed to "traditional" numerics, we are not interested to minimize the objective fct. We want a small test error resp. a "good" representation of the data...

Convergence of GD (smooth, convex)

upper bd on curvature

no lower bound on curvature

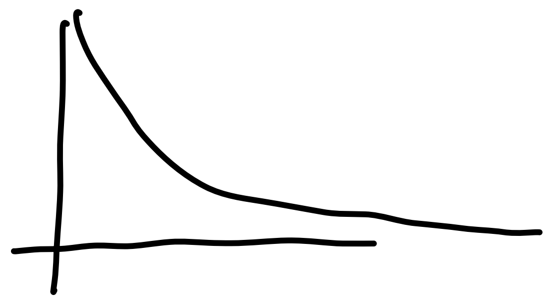
<u>Theorem</u>: Assume that F is L-smooth and convex with a global minimizer η_*. Choosing step size $\gamma_t = 1/L$, the iterates $(\theta_t)_t$ of GD satisfy

$$F(\theta_t) - F(\eta_*) \leq \frac{1}{t} \cdot \frac{L}{2} \cdot \|\theta_0 - \eta_*\|_2^2$$

"linear convergence"

Some more intuition for this theorem

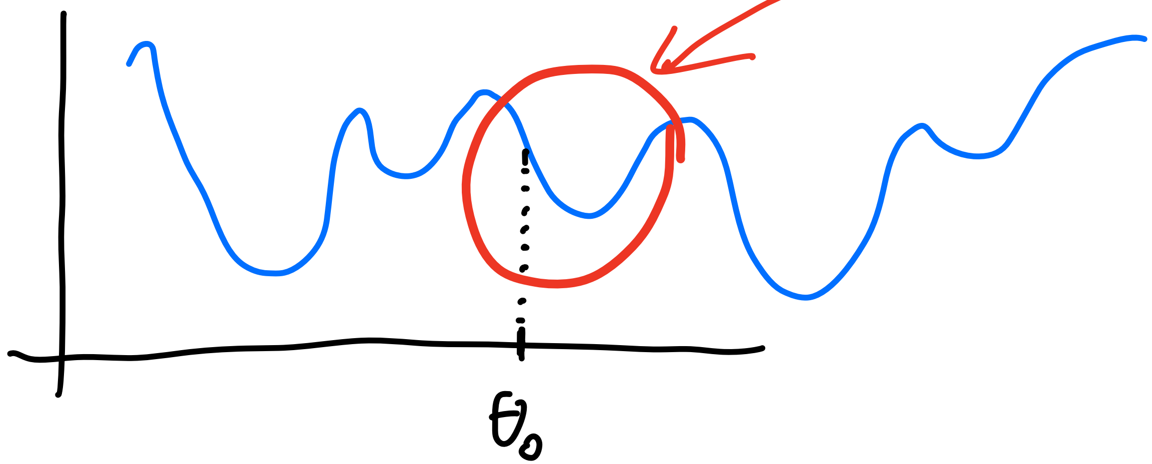
- Convex but not strongly convex. Could have a situation like this, need to make sure that a minimum does exist (assumption on η^*):



- If we would walk from θ_0 to η^* on a direct line with steps of size $\frac{1}{L}$, we would need $\|\theta_0 - \eta^*\| \cdot L$ many steps ($\rightsquigarrow$ constant in the bound)

Some more intuition for this theorem

- Note that in ML we often do not have globally convex problems. So this bound might tell us something about how fast we converge to the local optimum in the basin of attraction of our starting point.



No guarantee about convergence to global optimum!!!

Convergence of GD (smooth, strongly convex)

Theorem: Assume that f is L-smooth and μ-strongly convex.

Denote by $K = L/\mu$ the condition number, let η_* be global minim.

Choosing step size $\mu_t = \frac{1}{L}$, the iterates $(\theta_t)_t$ of GD on f satisfy

$$f(\theta_t) - f(\eta_*) \leq \left(1 - \frac{1}{K}\right)^t (f(\theta_0) - f(\eta_*))$$

$$\leq \underbrace{\exp(-t/K)}_{\text{"exponential convergence!"}} (f(\theta_0) - f(\eta_*)) \approx \exp(-t)$$

Remarks about this theorem

- K is always ≥ 1, so $\left(1 - \frac{1}{K}\right) \in \,]0,1[$, so $\left(1 - \frac{1}{K}\right)^t \longrightarrow 0$ as $t \to \infty$.

- Constant $(F(\theta_*) - F(\eta_*))$ on the rhs now measures the "distance" between start and end in the obj. fct, not the orig space. Can do this because of strong convexity.

- If we don't have restrictions on the domain of f, strong convexity implies the existence of global minimizer η^*.

- Convergence speed is surprisingly fast!

Convex vs strongly convex

- Convergence for strongly convex case is much faster!

Issues with the step size

- If the step size is too small, convergence of GD can take forever.
- If the step size is too large, GD might never converge because we always miss the optimum.
  (the algorithm could even diverge).
- In ML, step size is called the learning rate.

Learning rate decay

Unless one does something more clever, one typically uses a learning rate decay, for example

$$\alpha_t = \alpha_0 \exp(-k \cdot t) \approx \exp(-t) \quad \text{(exponential decay)}$$

$$\alpha_t = \alpha_0 / (1 + k \cdot t) \approx \frac{1}{1+t} \quad \text{(inverse decay)}$$

The parameter k is the decay rate.

Line search to determine step size

Want to perform a gradient step $x_{t+1} = x_t - \alpha_t \nabla f(x_t)$

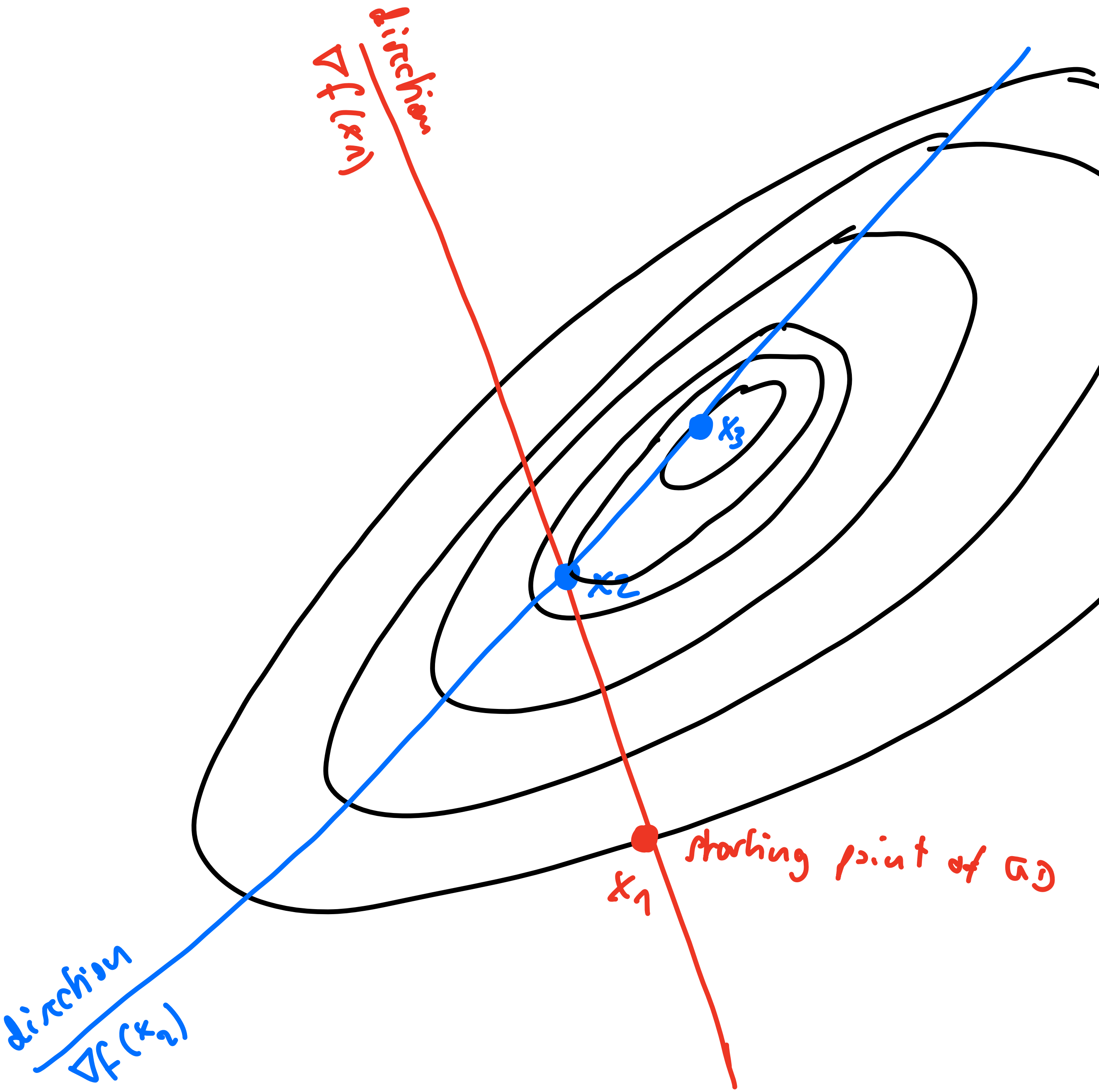
where we choose the step size α_t such that we minimize $f(x)$ in this direction:

$$\alpha_t = \underset{\alpha}{\operatorname{argmin}}\, f\left(x_t - \alpha \nabla f(x_t)\right)$$

Can for example use binary search to approximate α_t.

Computationally expensive, not so often used in ML.

Line search intuition



search along the whole line in direction of $\nabla f(x_1)$ for the point on which obj. fct f is minimized

then again search along line in direction of $\nabla f(x_2)$ for point on which obj. fct f is minimized

etc

Using momentum: idea

Consider a situation where we zig-zag slowly towards our destination:

Idea: let the descent direction "inherit" some part from the previous one, such that we get more of an "average" feeling:

Traditional: $w_{t+1} = w_t - \alpha_t \nabla f(w_t)$

Momentum: $w_{t+1} = w_t - \left(\beta \nabla f(w_{t-1}) + \alpha \nabla f(w_t) \right)$

momentum; friction

previous gradient

current gradient

Using momentum : idea

- It avoids the zig-zag
- It gains speed in steep areas that might help to travel faster over flat parts
- Might "overshoot" at the solution

"Like a marble that runs on the loss surface"

Vanishing gradient problem

... occurs if the gradients corresponding to some parameters get so small that they barely change.

Particularly the case in deep networks, where the gradients of the network parameters get multiplied during backpropagation. Partial derivatives of parameters in the first layers can get very small.

Particularly for loss functions with
i.e. tanh,
(not for ReLU)

Lots of suggestions in DNN literature, eg batch normalization

Exploding gradient problem

"Opposite" of vanishing gradients: in early layers, gradients get too large $\rightsquigarrow$ uncontrollable behavior

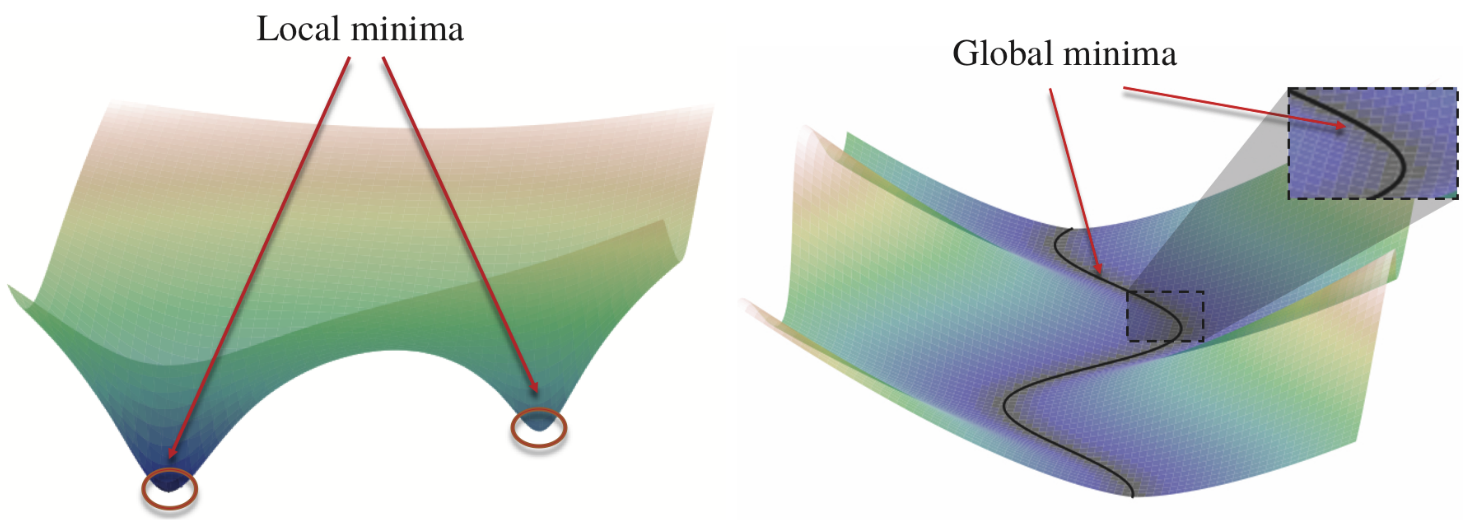
Reason either if weights are too large ($\rightsquigarrow$ initialization!)

or gradients too large $|\nabla w_i| > 1$ ($\rightsquigarrow$ gradient clipping)

Lots of solutions discussed in the DNN literature (eg skip connections)

Just one little teaser: loss landscape in deep learning

- The "loss landscape" in deep learning (when we try to optimize parameters of a neural network) is highly non-convex.
- However, it seems that "many" of the local optima one finds are already global optima!



(a) under-parametrized models (b) over-parametrized models

⇝ Belkin: Fit without fear: remarkable properties of deep learning, 2021.

Stochastic gradient descent

Literature: • Francis Bach: Learning theory from first principles. Forthcoming book, pdf online.

• Bottou, Curtis, Nocedal: Optimization methods for large scale machine learning. 2018

Motivation

In ML we typically minimize the training loss of the form

$$g(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_w(x_i), y_i)$$

where $(x_i, y_i)_{i=1,\dots,n}$ are our training points and ℓ is a loss fct, for example the squared loss, the logistic loss, and w are the parameters we optimize.

The gradient $\nabla g(w)$ is

$$\nabla g(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(f_w(x_i), y_i)$$

n terms.

Idea: "sample" the gradient

Consider the gradient $\nabla g(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(f_w(x_i), y_i)$

If n is large,

- It is really costly to compute it
- The data set might have redundancies, so we see similar info often.
- From a statistical point of view, whenever you see a large sum of random quantities, you guess that
  - the sum is close to its expected value
  - this might still be true if you subsample terms

Also, because we are interested in the test error in the end, a bit of stochasticity might, perhaps, prevent severe overfitting.

Stochastic gradient descent (vanilla)

Given training pts $(x_i, y_i)_{i=1\dots n}$ and training loss

$$g(w) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell(f_w(x_i), y_i)}_{=: \ell_i(w)}$$ with gradient

$$\nabla g(w) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{\nabla \ell(f_w(x_i), y_i)}_{= \nabla \ell_i(w)} .$$

In each step of the algorithm,

- sample one training pt (x_{i_0}, y_{i_0}) and compute the simplified gradient $\nabla \ell(f_{w_t}(x_{i_0}), y_{i_0}) =: \nabla \ell_i(w_t)$
- make gradient step:

$$w_{t+1} = w_t - \alpha_t \nabla \ell_i(w_t)$$

until convergence

Stochastic gradient descent (mini-batch)

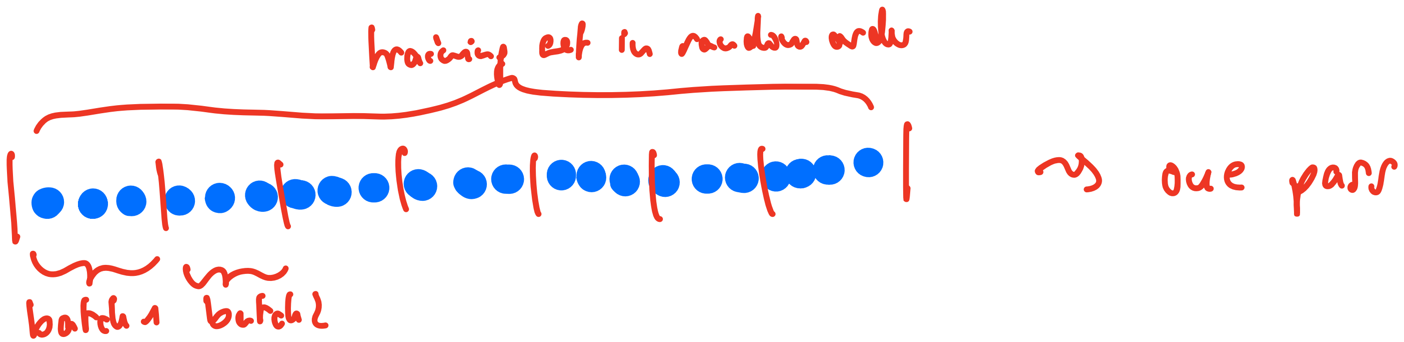
Instead of subsampling one point at each time, one typically subsamples a "mini-batch" consisting of a number k of data points.

Then apply the same principle:

- sample $(x_{i_1}, y_{i_1}), \dots, (x_{i_k}, y_{i_k})$
- Simplified gradient $\nabla \sum_{s=1}^{k} \ell_{i_s}(w_t) =: \nabla g_k(w_t)$
- gradient step $w_{t+1} = w_t - \alpha_t \, g_k(w_t)$

To implement this, choose a random permutation of the data set. Then pick one "batch" after the other, until one has made one full pass through the data.

Typically, one uses several passes through the data set.



Note that both variants (vanilla and mini-batch) use the same principle: they replace the full gradient by a random estimate of the gradient.

- introduces noise
- larger batches ⇒ less noise, smaller variance of the estimate,

Stochastic gradient descent (general view)

More generally, one can consider any estimate of the gradient $\hat{g}$ and use it in gradient descent:

$$w_{t+1} = w_t - \alpha_t \, \hat{g}(w_t)$$

Typically one would require that this estimate is unbiased (see later in the statistics part of the lecture).

"on average correct"

First remarks

- As the name suggests, SGD is not a deterministic algorithm: noisy updates!
- Some of the steps might even make a "wrong" step such that f is increasing. We hope though that "on average" it will be fine.
- We save compute and storage, compared to the standard GD.
- All guarantees for SGD would need to be statements "in expectation" or "with high probability".
- Subsampling is an unbiased estimate of the gradient, but the random errors do not decrease as we go along!

  Instead we will use a decreasing step size, which will make the errors smaller and smaller.

Convergence of SGD (convex)

Theorem: Assume that F is convex, B-Lipschitz and has a minimizer θ_* that satisfies $\|\theta_* - \theta_0\|_2^2 \leq D$. Assume that the gradients of the SGD iterates are all bounded by a constant B, that is $\|g_t(\theta_{t-1})\|_2^2 \leq B^2$ for all $t \geq 1$, and that the estimate g_t of the gradient is unbiased. Choose the step size $\gamma_t = \frac{D}{B} \cdot \frac{1}{\sqrt{t}}$.

Then the iterates of SGD on F satisfy

$$\underbrace{E\big(F(\bar{\theta}_t)}_{\text{"average iterate"}} - F(\theta_*)\big) \leq DB^2 \, \frac{2 + \log(t)}{2 \cdot \sqrt{t}}$$

slower than for GD

where

$$\bar{\theta}_t = \frac{\sum_{s=1}^{t} \gamma_s \theta_{s-1}}{\sum_{s=1}^{t} \gamma_s}$$

Digesting this theorem

- No assumption about strong convexity (could have really flat parts), no ass. on smoothness. Instead only that f is Lipschitz.
- Step size is decreasing (otherwise SGD would not converge because variance does not decrease)
- This bound is expressed in terms of the "average iterate", which is a form of stabilizing the result.

Convergence of SGD (strongly convex)

Consider a regularized problem of the form $G(\theta) = F(\theta) + \frac{\mu}{2}\|\theta\|_2^2$. ⊛

<u>Theorem</u> : Assume F is convex, B-Lipschitz, μ-strongly convex. Consider the regularized problem ⊛ and assume that it admits a unique minimizer θ_*. Then, under the same assumptions as before (unbiased, bounded $\|g_t(\theta)_t\|_2^2$) and choosing the step size $\gamma_t = \frac{1}{\mu t}$,

$$E\left(G(\bar{\theta}_t) - G(\theta_*)\right) \leq \frac{2B^2(1+\log t)}{\mu \cdot t}.$$

steps decrease faster

faster convergence

Remarks

- Smoothness does not help to improve a lot in the SGD case.
- SGD converges slower than GD in terms of accuracy, but needs much less computation (in terms of n, the number of training pts). So if the computational budget is limited, it is the method of choice.
- More refined bounds are needed to understand the behavior of different SGD versions (eg sampling one gradient vs using minibatches).

Comparing the bounds

Resulting error $|F(\theta) - F(\theta_*)|$ as a function of the number t of steps performed:

	Convex	Strongly Convex
Nonsmooth	Deterministic: $1/\sqrt{t}$ Stochastic: $1/\sqrt{t}$	Deterministic: $B^2/(t\mu)$ Stochastic: $B^2/(t\mu)$
Smooth	Deterministic: $1/t^2$ Stochastic: $1/\sqrt{t}$ Finite sum: n/t	Deterministic: $\exp(-t\sqrt{\mu/L})$ Stochastic: $L/(t\mu)$ Finite sum: $\exp(-\min\{1/n, \mu/L\}t)$

(tables from the Francis Bach book)

Final Remarks on SGD

To run SGD in practice, people use millions of tricks, and the choice of parameters and their scaling as learning proceeds makes a lot of difference!

Studying the behavior of SGD for all these scaling laws is challenging in practice and in theory!

... beyond this lecture ...

Why is SGD so popular in ML?

- Large scale problems ⇝ computational issues
- Redundant data
- "Stochasticity" introduced by SGD acts as regularizer: in the end we don't care about the training error (so it does not matter if we don't optimize it perfectly), but we want to have a small test error. The SGD noise might help to prevent overfitting.

Second order methods

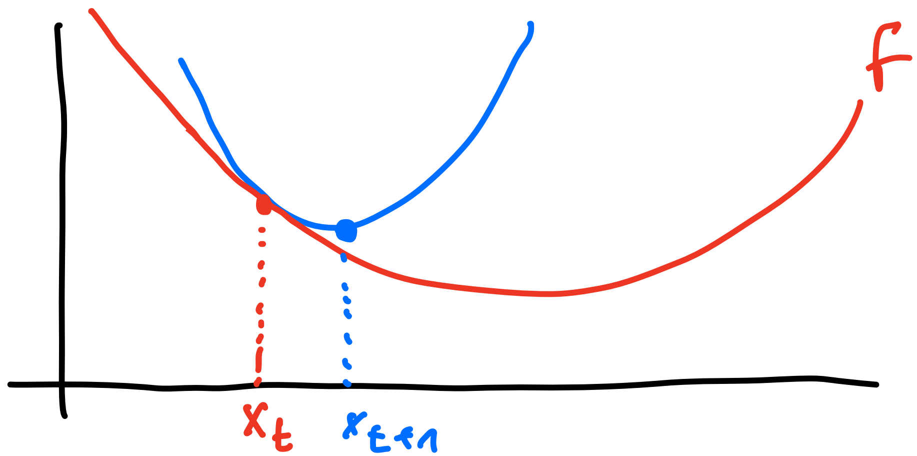
Newton

Intuition (1-dim)

Gradient descent only considers the first derivatives of the function.

Newton's method also looks at the second derivative and exploits it to choose a good step size:

Intuitively: fit not just a line but a parabola to the function at the current data point (given slope and curvature). Then proceed to the point where this parabola is minimal:



Newton method (1-dim)

Consider the Taylor expansion of the objective fct to the second order:

$$f(w_t + \varepsilon) \approx f(w_t) + \varepsilon \cdot f'(w_t) + \frac{1}{2} \varepsilon^2 f''(w_t).$$

Now search for ε that minimizes $f(w_t + \varepsilon)$:

$$\frac{d}{d\varepsilon}\left(f(w_t + \varepsilon)\right) = \dots = f'(w_t) + \varepsilon f''(w_t) \overset{!}{=} 0$$

$$\Rightarrow \quad \varepsilon = -\frac{f'(w_t)}{f''(w_t)}$$

Set update rule: $w_{t+1} = w_t - \dfrac{f'(w_t)}{f''(w_t)}$.

Newton method (d-dim)

Can derive a similar argument in the d-dim case, resulting in the update

$$w_{t+1} = w_t - \underbrace{H^{-1}}_{\text{Hessian}} \big(\underbrace{\nabla f(w_t)}_{\text{gradient}} \big)$$

- Particularly useful on convex fcts with pd Hessians...
- Convergence on smooth functions might be faster than for standard GD
- Trouble if Hessians are indefinite (saddle pts) or not even invertible

Computational costs

- Computationally costly (Hessian and its inverse!)
- Approximation algorithms to the inverse of the Hessian exist:
    - Conjugate gradient
    - Quasi Newton methods
    - BFGS algorithm