Optimization

The way machine learning applies models to data can roughly be categorized into two ways:

Optimization:

- The solution is a specific value.
- We define mathematically what a "good" solution is and search for the best one.

Simulation (such as MCMC):

- The solution is given by distribution (e.g. the distribution itself, an expectation, etc)
- We approximate the solution by samples drawn from the distribution.

Extremal Principles

	Purpose	Objective function
Maximum likehood estimation	Fit a statistical model	Likelihood
Empirical risk minimization	Fit a classifier/predictor	Error rate (more generally: empirical risk)
Variational inference	Approximate a posterior	Distance between approximate and true posterior

The list is far from exhaustive.

Min and argmin

$$\min_{\mathbf{x}} f(\mathbf{x}) = \text{ smallest value of } f(\mathbf{x}) \text{ for any } \mathbf{x}$$

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) = \text{ value of } \mathbf{x} \text{ for which } f(\mathbf{x}) \text{ is minimal}$$

Minimum with respect to subset of arguments

$$\min_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}) = \text{ smallest value of } f(\mathbf{x}, \mathbf{y}) \text{ for any } \mathbf{x} \text{ if } \mathbf{y} \text{ is kept fixed}$$

Optimization problem

For a given function $f : \mathbb{R}^d \to \mathbb{R}$, a problem of the form

$$\text{find} \quad \mathbf{x}^* := \arg\min_{\mathbf{x}} f(\mathbf{x})$$

is called a **minimization problem**. If $\arg\min$ is replaced by $\arg\max$, it is a **maximization problem**. Minimization and maximization problems are collectively referred to as **optimization problems**.

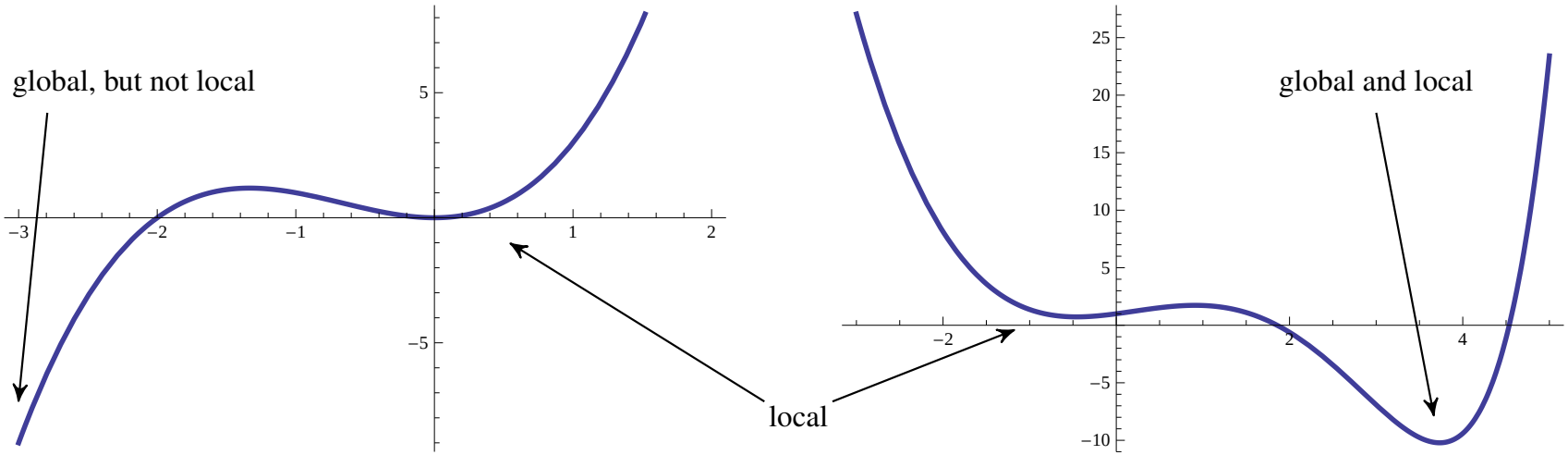
Minimization vs Maximization

For any function f, we have

$$\min f(\mathbf{x}) = -\max(-f(\mathbf{x})) \qquad \text{and} \qquad \arg\min f(\mathbf{x}) = \arg\max(-f(\mathbf{x}))$$

That means:

- If we know how to minimize, we also know how to maximize, and vice versa.
- We do not have to solve both problems separately; we can just generically discuss minimization.



Local and global minima

A minimum of f at x is called:

- **Global** if f assumes no smaller value on its domain.
- **Local** if there is some open interval (a, b) containing x (or an open neighbourhood if we work in $\mathbb{R}^d$) such that $f(x)$ is a global minimum of f restricted to that set.

Analytic criteria for local minima

Recall that x is a local minimum of f if

$$f'(x) = 0 \qquad \text{and} \qquad f''(x) > 0 \ .$$

In $\mathbb{R}^d$,

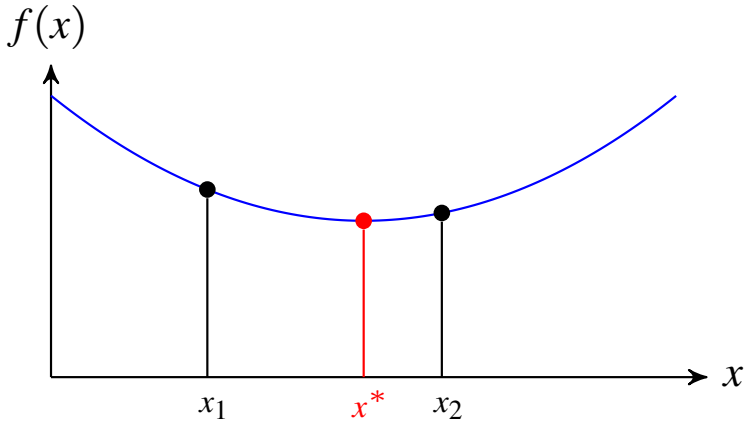
$$\nabla f(\mathbf{x}) = 0 \qquad \text{and} \qquad H_f(\mathbf{x}) = \Big(\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}) \Big)_{i,\,j=1,\ldots,d} \text{ positive definite.}$$

The $d \times d$-matrix $H_f(\mathbf{x})$ is called the **Hessian matrix** of f at $\mathbf{x}$.

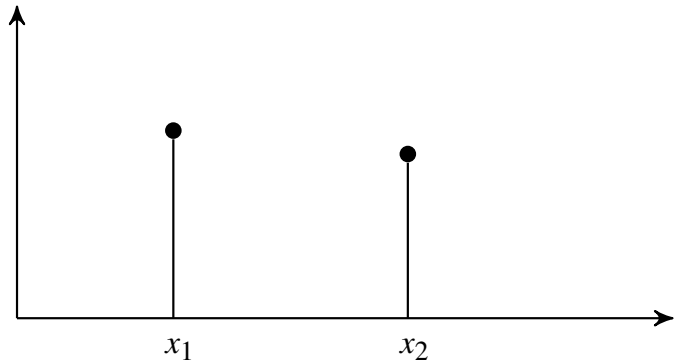
Typical situation

- Given is a function $f : \mathbb{R}^d \to \mathbb{R}$.
- The dimension d is usually very large.
  (In neural network training problems: Often in the millions.)
- We cannot plot or "look at" the function.
- We can only evaluate its value $f(\mathbf{x})$ point by point.

One-dimensional Illustration



- Our goal is to find x^*.
- We can evaluate the function at points of our choice, say x_1 and x_2.

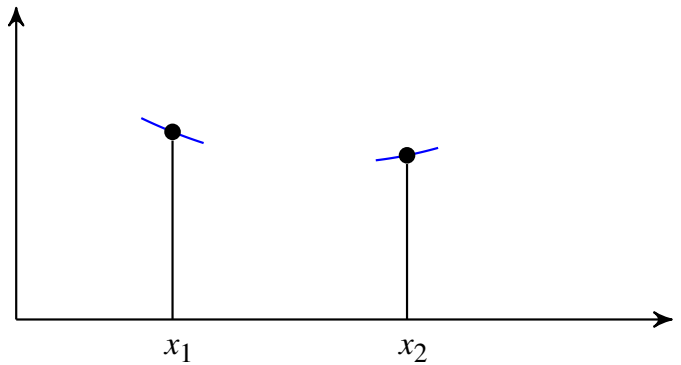


- However, we cannot "see" the function.
- All we know are values at a few points.

Task

Based on the values we know, we have to:

- Either make a decision what x^* is.
- Or gather more information, by evaluating f at additional points. In that case, we have to decide which point to evaluate next.



- If we can compute the derivatives $f'(x_1)$ and $f'(x_2)$, we have (the slope of) linear approximations to f at both points that are locally exact.
- That is: We can substitute the derivatives for the two short blue lines in the figure.
- We can tell from the sign of the derivative in which direction the function decreases.
- We also know that $f'(x) = 0$ if x is a minimum.

Basic idea

Start with some point x_0. Compute the derivative $f'(x_0)$ at x. Then:

- "Move downhill": Choose some $c > 0$, and set $x_1 = x_0 + c$ if $f'(x_0) < 0$ and $x_1 = x_0 - c$ if $f'(x_0) > 0$.
- Compute $f'(x_1)$. If it is 0 (possibly a minimum), stop.
- Otherwise, move downhill from x_1, etc.

Observations

- Since the sign of f' is determined by whether f increases or decreases, we can summarize the case distinction above by setting

$$x_1 = x_0 - \text{sign}(f'(x_0)) \cdot c$$

- If f changes rapidly, it may be a good strategy to make a large step (choose a large c), since we presumably are still far from the minimum. If f changes slowly, c should be small.
- One way of doing so is to choose c as the magnitude of f', since $|f'|$ has exactly this property. In that case:

$$x_1 = x_0 - \text{sign}(f'(x_0)) \cdot |f'(x_0)| = x_0 - f'(x_0)$$

The algorithm obtained by applying this step repeatedly is called **gradient descent**.

Gradient Descent

Gradient descent searches for a minimum of a differentiable function f.

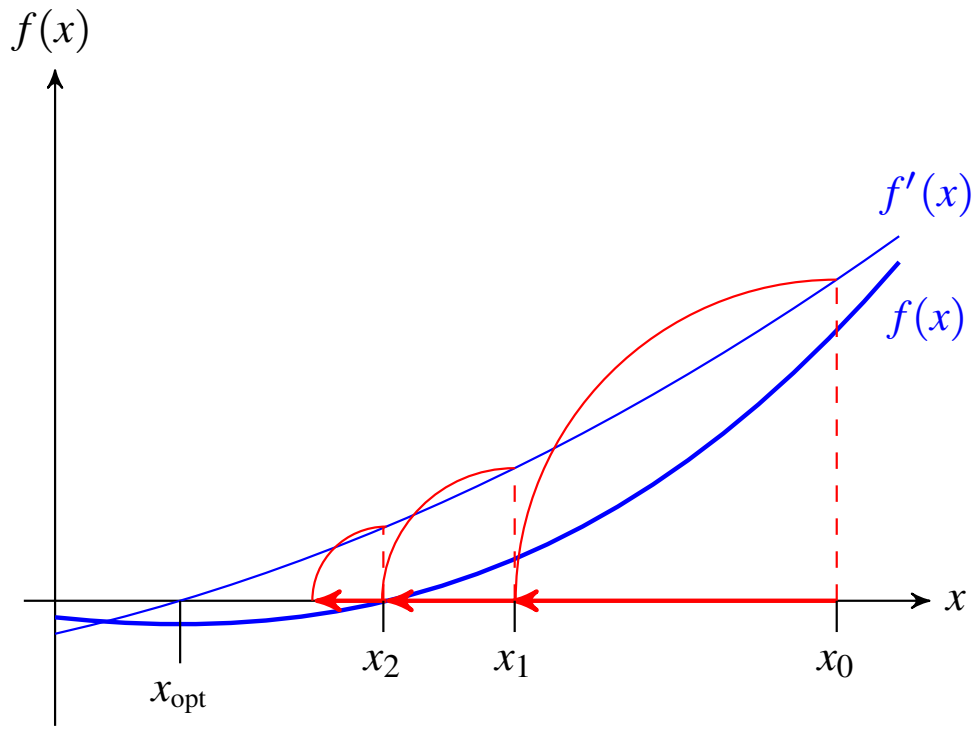
Algorithm

Start with some point $x_0 \in \mathbb{R}$ and fix a precision $\varepsilon > 0$.

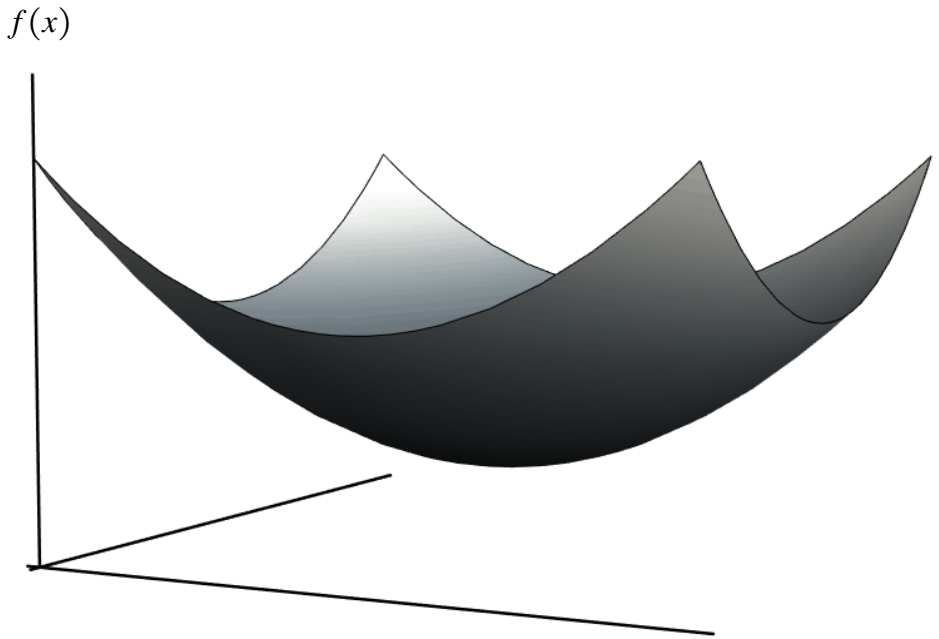
Repeat for $t = 0, 1, 2, \ldots$:

1. Check whether $|f'(x_t)| < \varepsilon$. If so, report the solution $x^* := x_t$ and terminate.
2. Otherwise, set

$$x_{t+1} := x_t - f'(x_t)$$

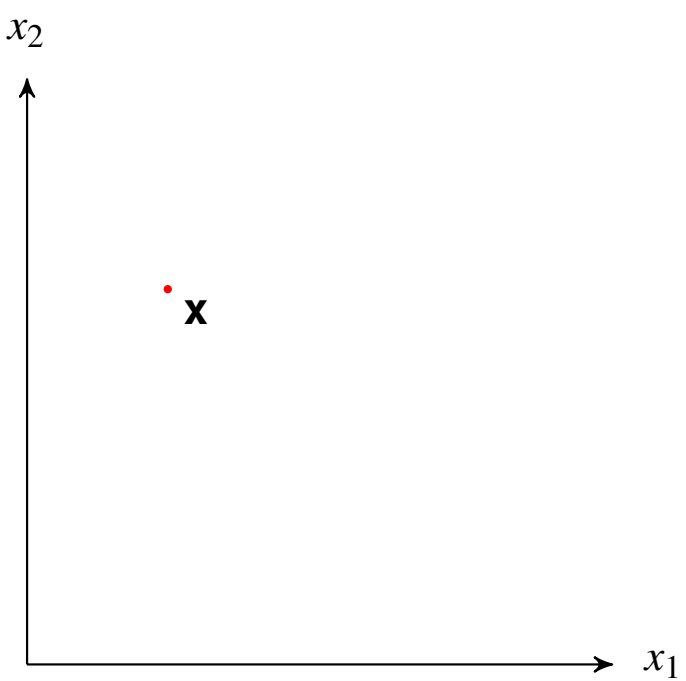


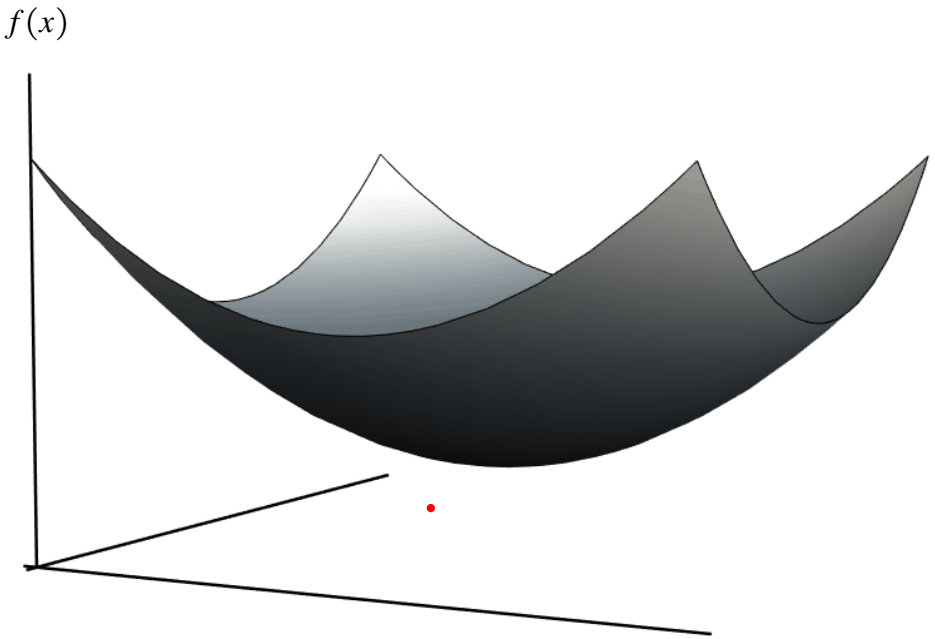
Derivatives in Multiple Dimensions



- We now ask how to define a derivative in multiple dimensions.
- Consider a function $f : \mathbb{R}^d \to \mathbb{R}$. What is the derivative of f at a point $\mathbf{x}$?
- For simplicity, we assume $d = 2$ (so that we can plot the function).

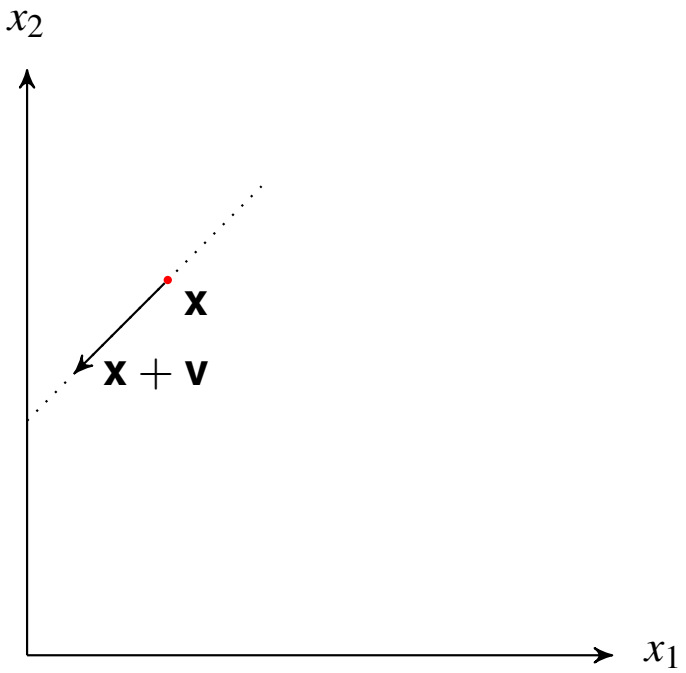
Derivatives in Multiple Dimensions

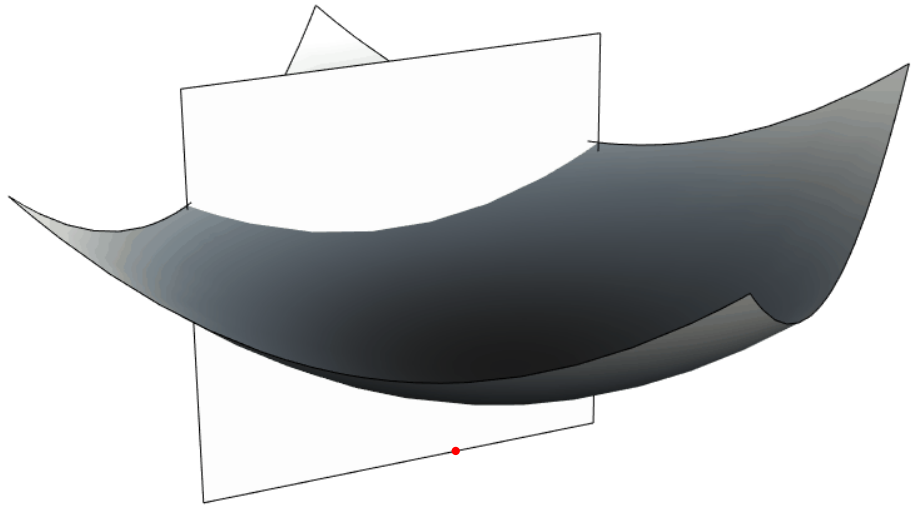




- We fix a point $\mathbf{x} = (x_1, x_2)$ in $\mathbb{R}^2$, marked red above.
- We will try to turn this into a 1-dimensional problem, so that we can use the definition of a derivative we already know.

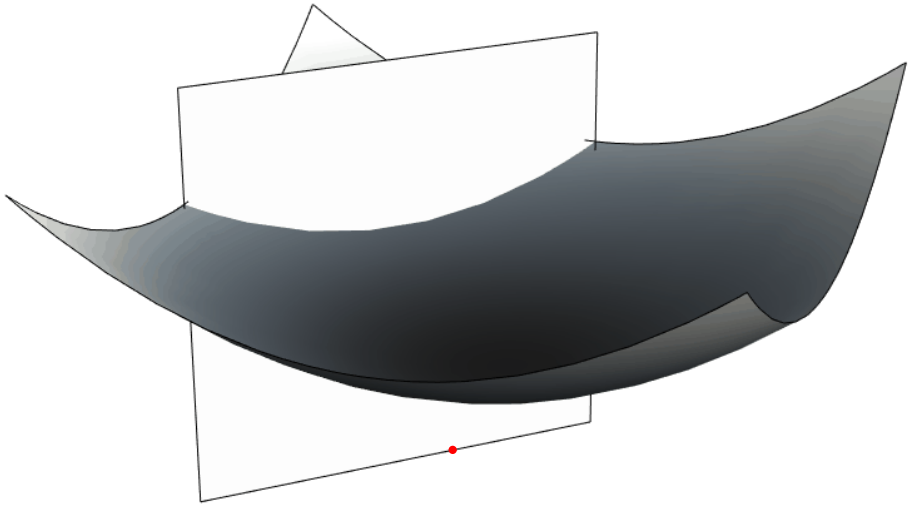
Reducing to one Dimension

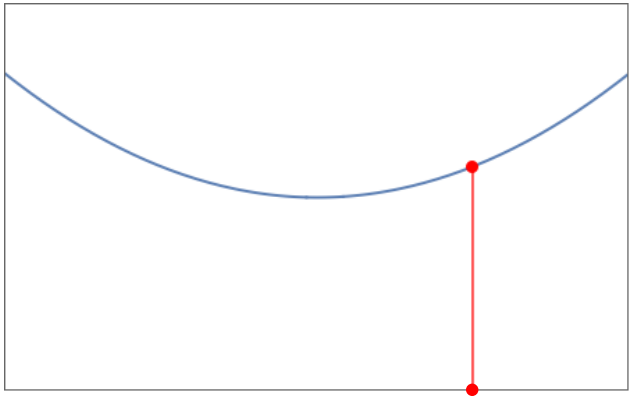




- To make the problem 1-dimensional, fix some vector $\mathbf{v} \in \mathbb{R}_2$, and draw a line through $\mathbf{x}$ in direction of $\mathbf{v}$.
- Then intersect f with a plane given by this line: In the coordinate system of f, choose the plane that contains the line and is orthogonal to $\mathbb{R}^2$.
- The plane contains the point $\mathbf{x}$.
- Note we can do that even if $d > 2$. We still obtain a plane.

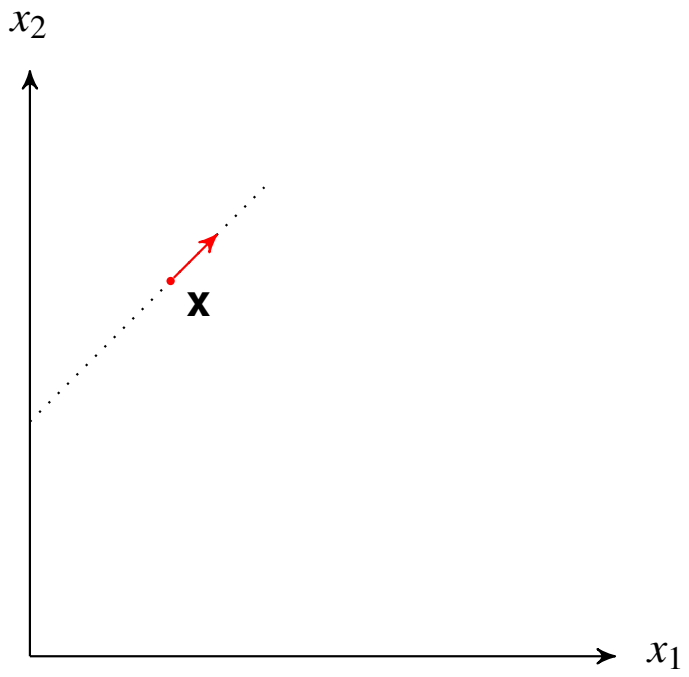
Reducing to one Dimension





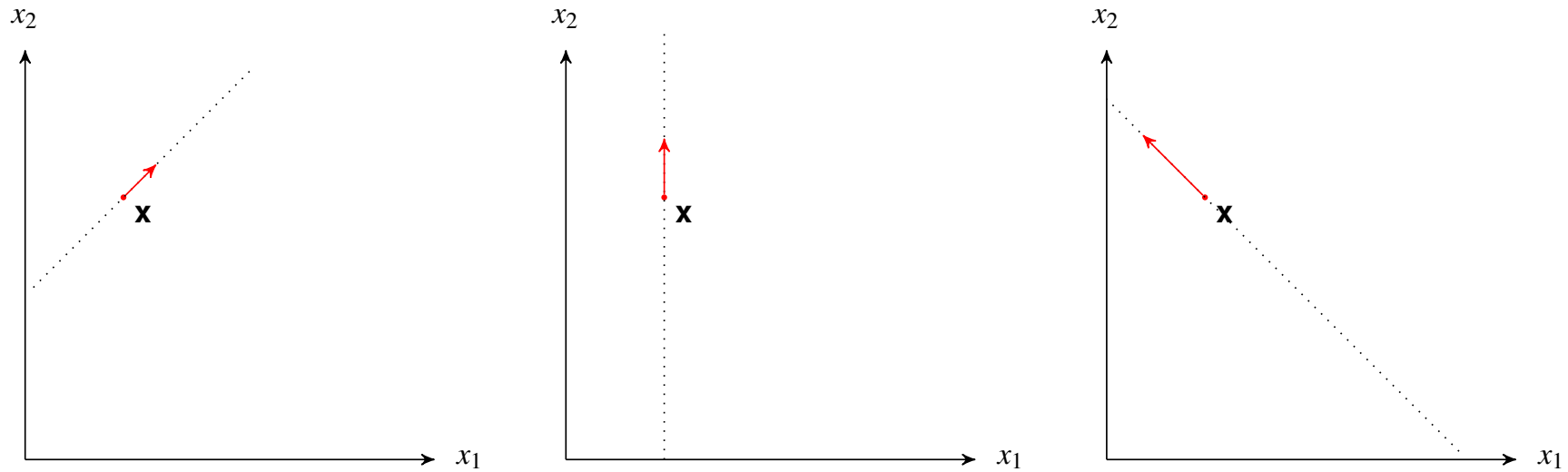
- The intersection of f with the plane is a 1-dimensional function f_H, and $\mathbf{x}$ corresponds to a point x_H in its domain.
- We can now compute the derivative f'_H of f_H at x_H. The idea is to use this as the derivative of f at $\mathbf{x}$.

Back to Multiple Dimensions



- In the domain of f, we draw a vector from $\mathbf{x}$ *in direction of* H such that:
  1. The vector is oriented to point in the direction in which f_H increases.
  2. Its length is the value of the derivative $f'_H(x_H)$.
- That completely determines the vector (shown in red above).
- There is one problem still to be solved: f_H depends on H, that is, on the direction of the vector $\mathbf{v}$. Which direction should we use?

THE GRADIENT



- We now rotate the plane H around $\mathbf{x}$. For each position of the plane, we get a new derivative $f'_H(x_H)$, and a new red vector.
- We choose the plane for which f'_H is largest:

$$H^* := \arg \max_{\text{all rotations of } H} f'_H(x_H)$$

Provided that f_H is differentiable for all H, one can show that this is always unique (or $f'_H(x_H)$ is zero for all H).

- We then define the vector

$$\nabla f(\mathbf{x}) := \text{ vector given by } H^* \text{ as above}$$

The vector $\nabla f(\mathbf{x})$ is called the **gradient** of f at $\mathbf{x}$.

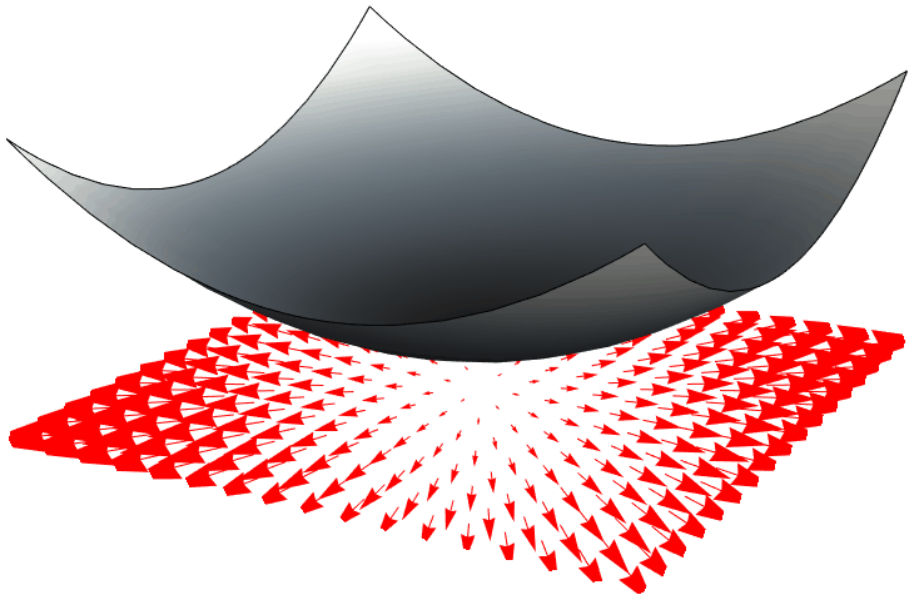
Properties of the Gradient

The gradient $\nabla f(\mathbf{x})$ of $f : \mathbb{R}^d \to \mathbb{R}$ at a point $\mathbf{x} \in \mathbb{R}^d$ is a vector in the domain $\mathbb{R}^d$ in the direction in which f most rapidly increases at $\mathbf{x}$.

- Recall that a contour line (or contour set) of f is a set of points along which f remains constant,
$$C[f, c] := \{\mathbf{x} \in \mathbb{R}^d \,|\, f(\mathbf{x}) = c\} \qquad \text{for some } c \in \mathbb{R}.$$
- One can show that if $C[f, c]$ contains $\mathbf{x}$, the gradient at $\mathbf{x}$ is orthogonal to the contour:
$$\nabla f(\mathbf{x}) \perp C[f, c] \qquad \text{if } \mathbf{x} \in C[f, c]\ .$$
- Intuition: The gradient points in the direction of maximal change, whereas $C[f, c]$ is a direction in which there is no change. Locally, these two are orthogonal.

Gradients are orthogonal to contour lines.

Gradients and Contour Lines



- For this parabolic function, all contour lines are concentric circles around the minimum.
- The picture above shows the gradients plotted at various points in the plane.

$$f : \mathbb{R}^d \to \mathbb{R}$$

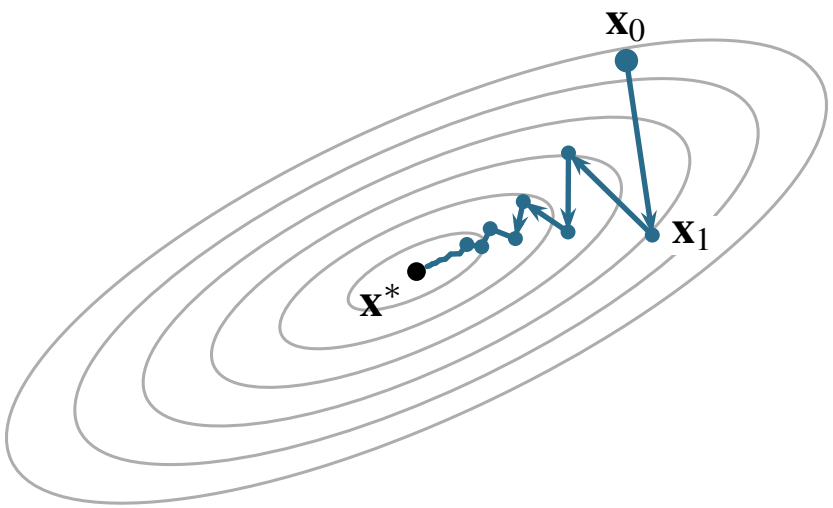
Algorithm

Start with some point $\mathbf{x}_0 \in \mathbb{R}^d$ and fix a precision $\varepsilon > 0$.

Repeat for $t = 0, 1, 2, \ldots$:

1. Check whether $\|\nabla f(\mathbf{x}_t)\| < \varepsilon$. If so, report the solution $\mathbf{x}^* := \mathbf{x}_t$ and terminate.
2. Otherwise, set

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \nabla f(\mathbf{x}_t)$$



$$f : \mathbb{R}^d \to \mathbb{R}$$

Algorithm

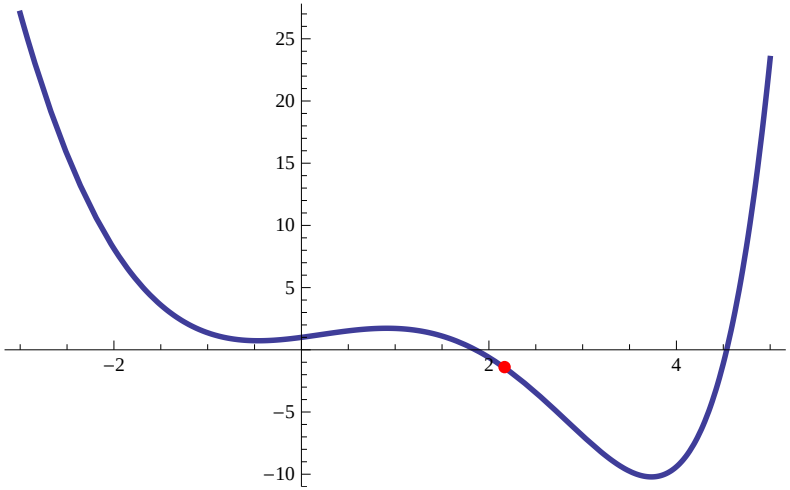
Start with some point $\mathbf{x}_0 \in \mathbb{R}^d$ and fix a precision $\varepsilon > 0$.

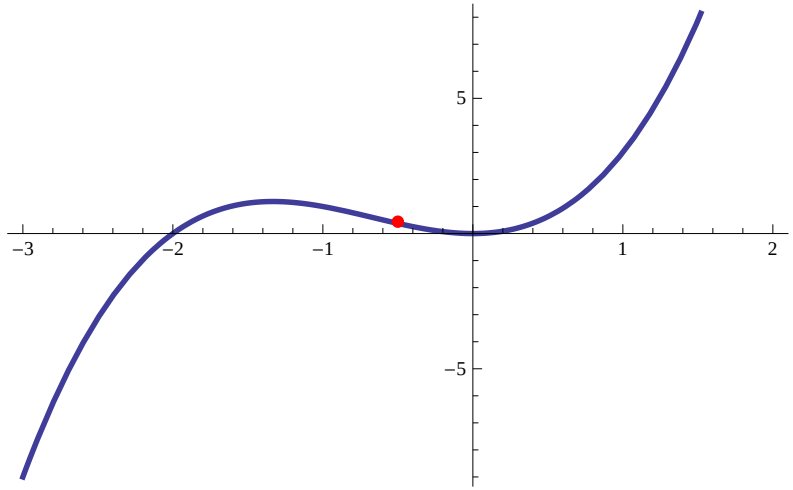
Repeat for $t = 0, 1, 2, \ldots$:

1. Check whether $\|\nabla f(\mathbf{x}_t)\| < \varepsilon$. If so, report the solution $\mathbf{x}^* := \mathbf{x}_t$ and terminate.
2. Otherwise, set
$$\mathbf{x}_{t+1} := \mathbf{x}_t - \alpha(t)\nabla f(\mathbf{x}_t)$$

Here, $\alpha(t) > 0$ is a coefficient that may depend on t. It is called the **step size** in optimization, or the **learning rate** in machine learning.

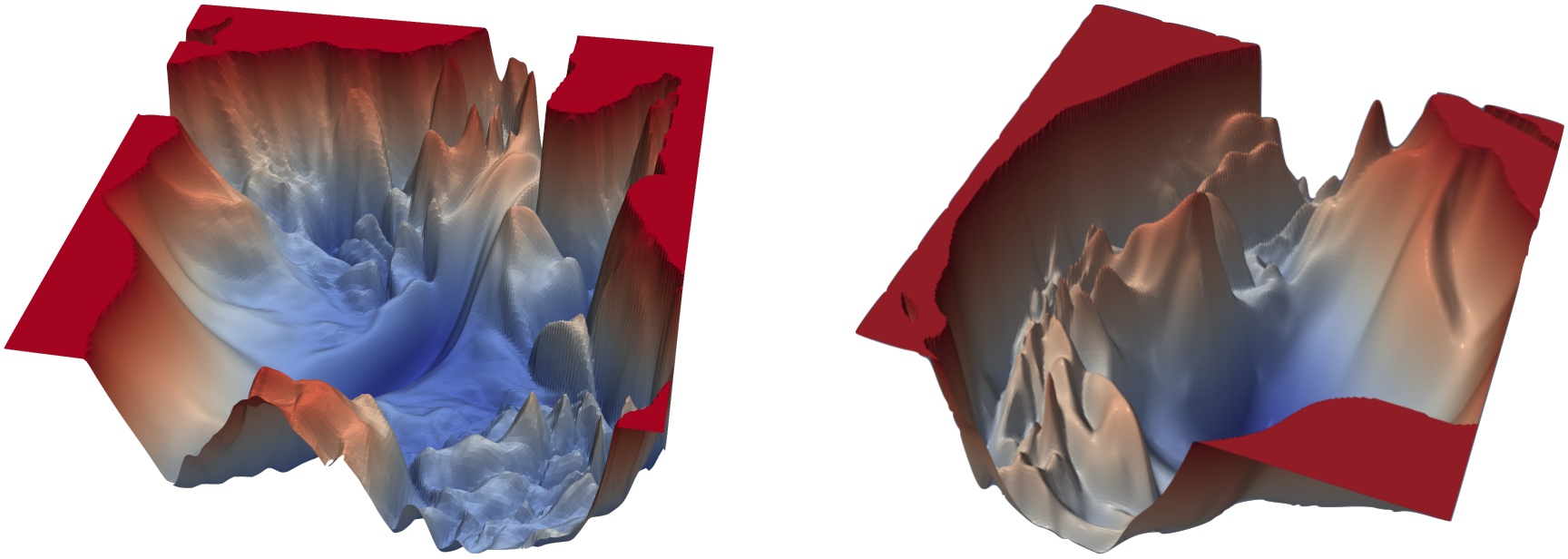
Gradient Descent and Local Minima





- Suppose for both functions above, gradient descent is started at the point marked red.
- It will "walk downhill" as far as possible, then terminate.
- For the function on the left, the minimum it finds is global. For the function on the right, it is only a local minimum.
- Since the derivative at both minima is 0, gradient descent cannot detect whether they are global or local.

For smooth functions, gradient descent finds *local* minima. If the function is complicated, there may be no way to tell whether the solution is also a global minimum.



Source: Li et al., arxiv 1712.09913

Mention saddle points.

Summary so far

- The derivative/gradient provides local information about how a function changes around a point $\mathbf{x}$.
- Optimization algorithms: If we know the gradient at our current location $\mathbf{x}$, we can use this information to make a step in "downhill" direction, and move closer to a (local) minimum.

What we do not know yet

That assumes that we can compute the gradient. There are two possibilities:

- For some functions, we are able to derive $\nabla f(\mathbf{x})$ as a function of $\mathbf{x}$. Gradient descent can evaluate the gradient by evaluating that function.
- Otherwise, we have to estimate $\nabla f(\mathbf{x})$ by evaluating the function f at points close to $\mathbf{x}$.

For now, we will assume that we can compute the gradient as a function.

Next: Newton's method

- Gradient descent is a **first-order method**. It uses only the first derivative.
- A similar method that uses two derivatives is called Newton's method (or the Newton–Raphson algorithm).
- Roughly speaking, higher-order methods converge in fewer steps, at the price of more computation per step.

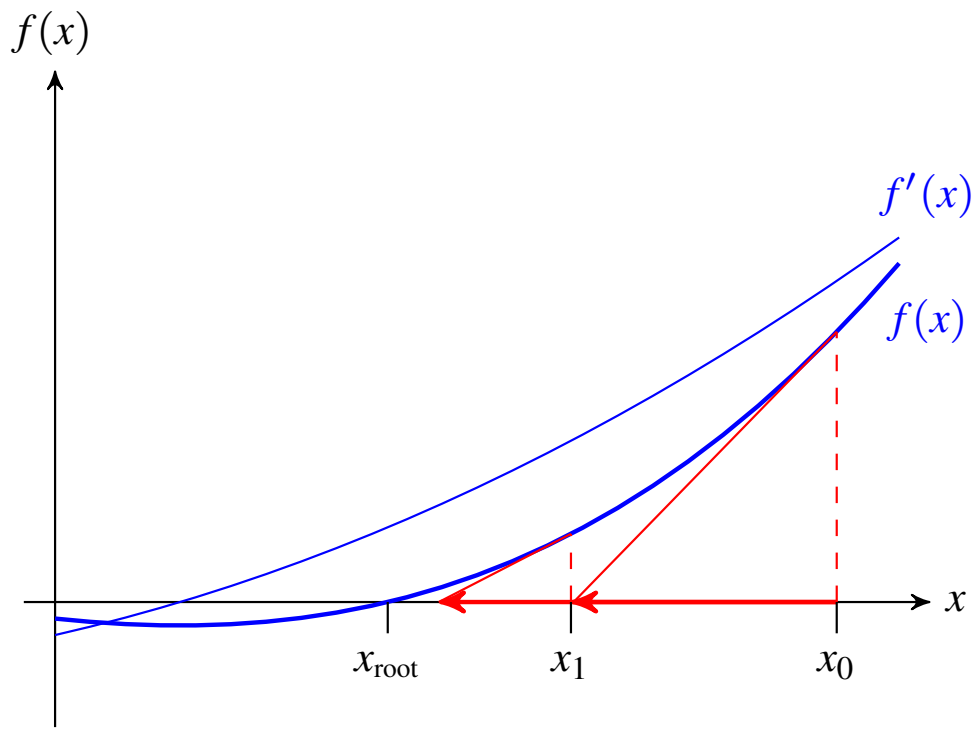
Algorithm

Newton's method searches for a **root** of f, i.e. it solves the equation $f(x) = 0$.

1. Start with some point $x_0 \in \mathbb{R}$ and fix a precision $\varepsilon > 0$.
2. Repeat for $t = 0, 1, 2, \ldots$

$$x_{t+1} := x_t - f(x_t)/f'(x_t)$$

3. Terminate when $|f(x_t)| < \varepsilon$.



Function evaluation

Some numerical evaluations of functions ($\sqrt{a}$, $1/a$, etc) are implemented using Newton's method. To evaluate g at a, we have to transform $x = g(a)$ into an equivalent equation of the form

$$f(x, a) = 0 \ .$$

We then fix a and solve for x using Newton's method for roots.

Example: Square root

To eveluate $g(a) = \sqrt{a}$, we can solve

$$f(x, a) = x^2 - a = 0 \ .$$

- This is how `sqrt()` is implemented in many software implementations.
- On most modern platforms, it compiles to a hardware instruction (eg `FSQRT` on x86 or `VSQRT` on ARM).
- A variant of Newton's algorithm is then run by the hardware (often with a workaround to avoid the division step, which is expensive).

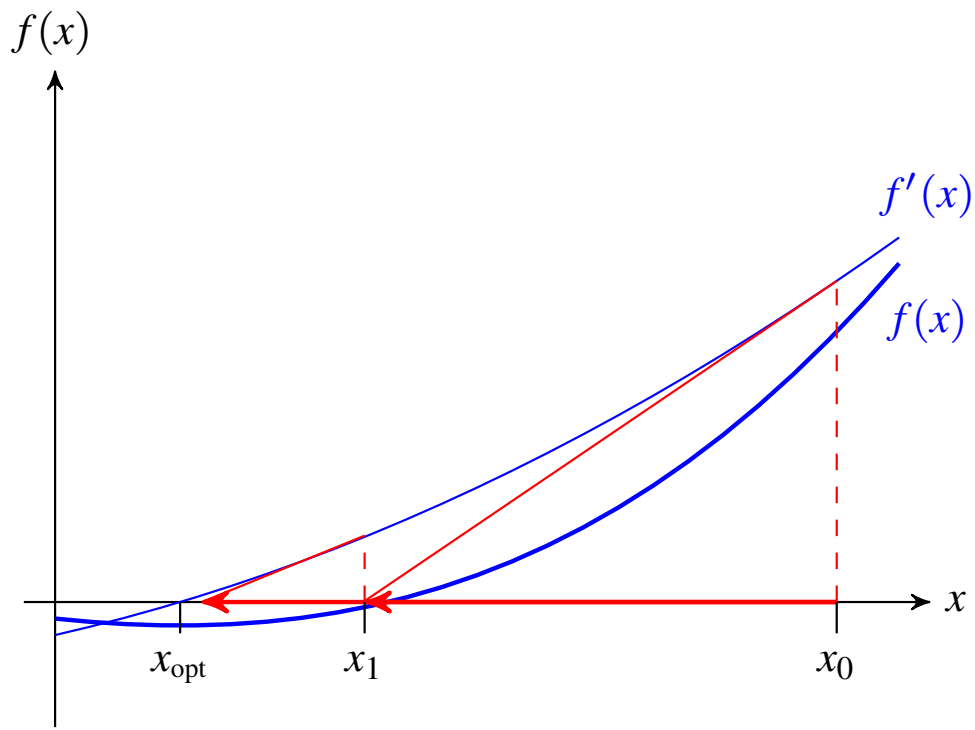
Algorithm

We can use Newton's method for minimization by applying it to solve $f'(x) = 0$.

1. Start with some point $x \in \mathbb{R}$ and fix a precision $\varepsilon > 0$.
2. Repeat for $t = 1, 2, \ldots$

$$x_{t+1} := x_t - f'(x_t)/f''(x_t)$$

3. Terminate when $|f'(x_t)| < \varepsilon$.



Recall: Gradient descent

$$\mathbf{x}_{t+1} := \mathbf{x}_t - \nabla f(\mathbf{x}_t)$$

Newton's method for minima

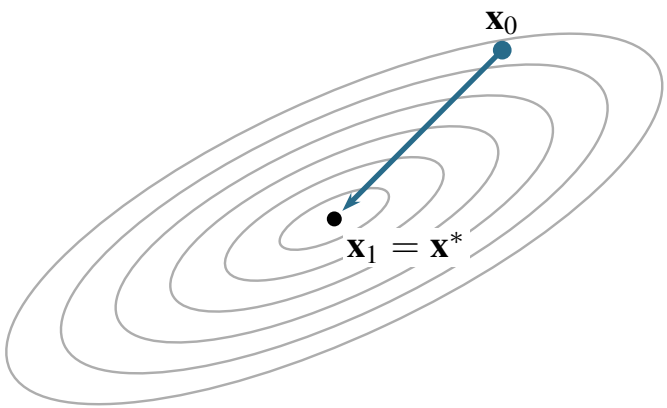
$$\mathbf{x}_{t+1} := \mathbf{x}_t - H_f^{-1}(\mathbf{x}_t) \cdot \nabla f(\mathbf{x}_t)$$

That requires that the matrix $H_f(\mathbf{x})$ is invertible.

The Hessian measures the curvature of f.

Effect of the Hessian

Multiplication by H_f^{-1} in general changes the direction of $\nabla f(\mathbf{x}_t)$. The correction takes into account how $\nabla f(\mathbf{x})$ changes away from $\mathbf{x}_t$, as estimated using the Hessian at $\mathbf{x}_t$.



Convergence

- The algorithm always converges if $f'' > 0$ (or H_f positive definite), if the step size is chosen properly. This can be implemented via line search.
- The speed of convergence separates into two phases:
  - In a (possibly small) region around the minimum, f can always be approximated by a quadratic function.
  - Once the algorithm reaches that region, the error decreases at quadratic rate. Roughly speaking, the number of correct digits in the solution doubles in each step.
  - Before it reaches that region, the convergence rate is linear.

High dimensions

- The required number of steps hardly depends on the dimension of $\mathbb{R}^d$. Even in $\mathbb{R}^{10000}$, you can usually expect the algorithm to reach high precision in half a dozen steps.
- Caveat I: The individual steps can become very expensive, since we have to invert H_f in each step, which is of size $d \times d$.
- Caveat II: High-dimensional functions tend to have many more local minima then low-dimensional ones. Even if Newton still converges quickly, we have to ask even more carefully what it is converging to.

Next: Constrained Optimization

So far

- If f is differentiable, we can search for local minima using gradient descent.
- If f is sufficiently nice (twice continuously differentiable), we know how to speed up the search process using Newton's method.

Constrained problems

- The numerical minimizers use the criterion $\nabla f(\mathbf{x}) = 0$ for the minimum.
- In a constrained problem, the minimum is *not* identified by this criterion.

Next steps

We will figure out how the constrained minimum can be identified. We have to distinguish two cases:

- Problems involving only equalities as constraints (easy).
- Problems also involving inequalities (a bit more complex).

- An **optimization problem** for a given function $f : \mathbb{R}^d \to \mathbb{R}$ is a problem of the form

$$\min_{\mathbf{x}} f(\mathbf{x})$$

  which we read as "find $\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$".

- A **constrained optimization problem** adds additional requirements on $\mathbf{x}$,

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ \text{subject to} \quad &\mathbf{x} \in G \,, \end{aligned}$$

  where $G \subset \mathbb{R}^d$ is called the **feasible set**.

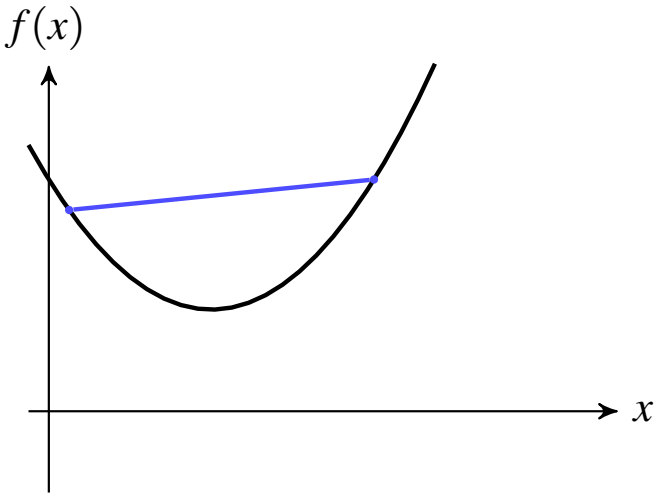
- The set G is often defined by equations, e.g.

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

  The condition “subject to ...” is called a **constraint**. The constraint $g(\mathbf{x}) = 0$ above is an **equality constraint**, whereas one of the form $g(\mathbf{x}) \geq 0$ is an **inequality constraint**.

Recall: Convex Functions

A function f is **convex** if every line segment between function values lies above the graph of f. It is **strictly convex** if every such line lies strictly above the graph.



Useful criterion

- A twice differentiable function is convex if its Hessian is positive semidefinite everywhere (or $f'' \geq 0$ in one dimension).
- It is strictly convex if the Hessian is positive definite everywhere (or $f'' > 0$).

These conditions are sufficient, but not necessary.

Implications for optimization

If f is convex, then:

- $\nabla f(\mathbf{x}) = 0$ is a sufficient criterion for a minimum (in $\mathbb{R}$: $f'(x) = 0$).
- Local minima are global.
- If f is **strictly convex** ($f'' > 0$ or H_f positive definite), there is at most one minimum, which then is global and local.

Objective

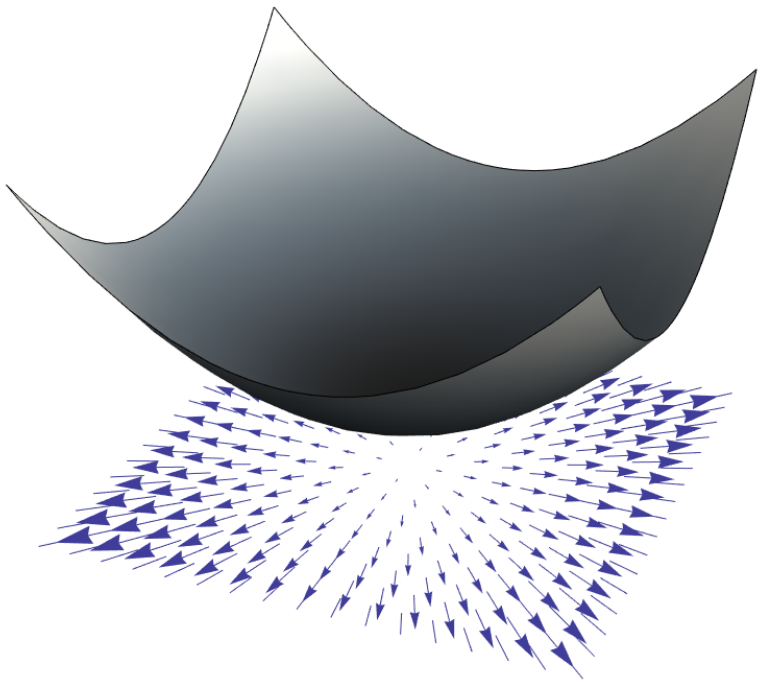
$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{subject to } g(\mathbf{x}) = 0 \end{aligned}$$

Idea

- The feasible set is the set of points $\mathbf{x}$ which satisfy $g(\mathbf{x}) = 0$,

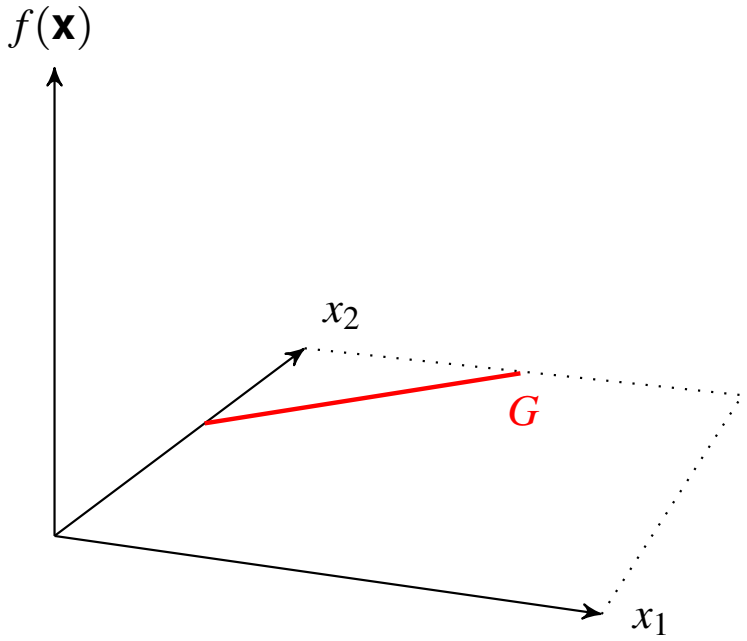
  $$G := \{\mathbf{x} \mid g(\mathbf{x}) = 0\} \ .$$

  If g is reasonably smooth, G is a smooth surface in $\mathbb{R}^d$.
- We restrict the function f to this surface and call the restricted function f_g.
- The constrained optimization problem says that we are looking for the minimum of f_g.



$$f(\mathbf{x}) = x_1^2 + x_2^2$$

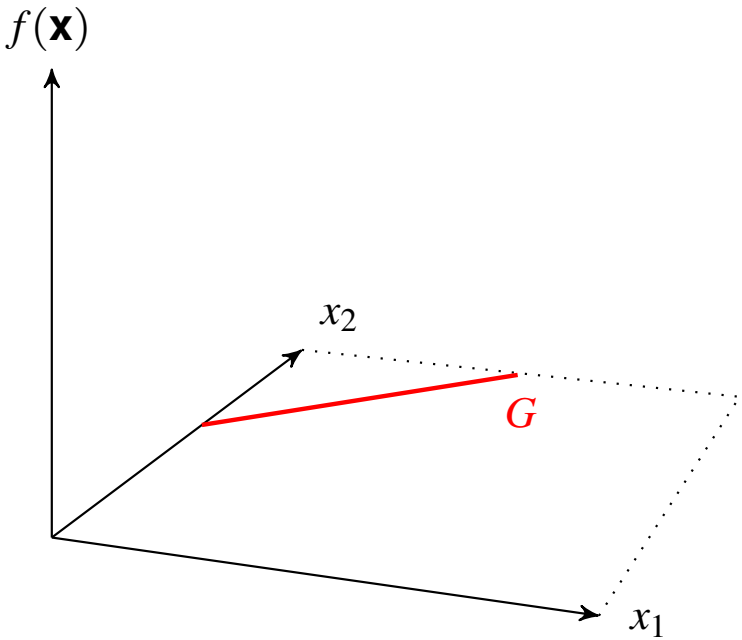
The blue arrows are the gradients $\nabla f(\mathbf{x})$ at various values of $\mathbf{x}$.

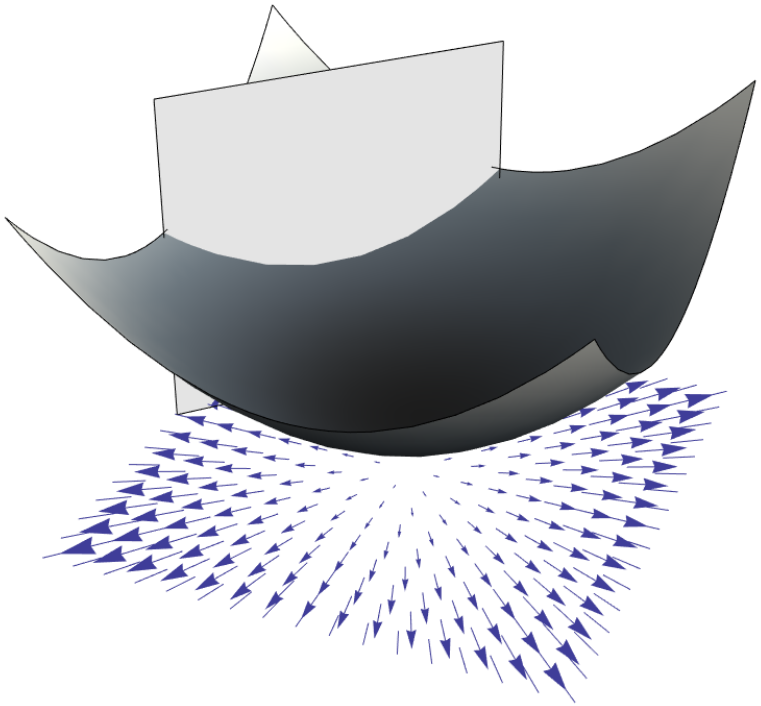


Constraint g.

Here, g is linear, so the graph of g is a (sloped) affine plane. The intersection of the plane with the x_1-x_2-plane is the set G of all points $\mathbf{x}$ with $g(\mathbf{x}) = 0$.

Lagrange Optimization





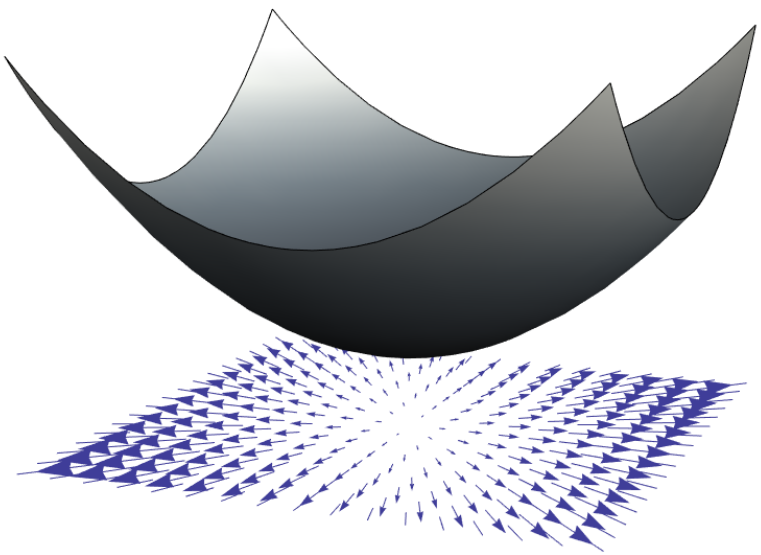
- We can make the function f_g given by the constraint $g(\mathbf{x}) = 0$ visible by placing a plane vertically through G. The graph of f_g is the intersection of the graph of f with the plane.
- Here, f_g has parabolic shape.
- The gradient of f at the miniumum of f_g is *not* 0.

Fact

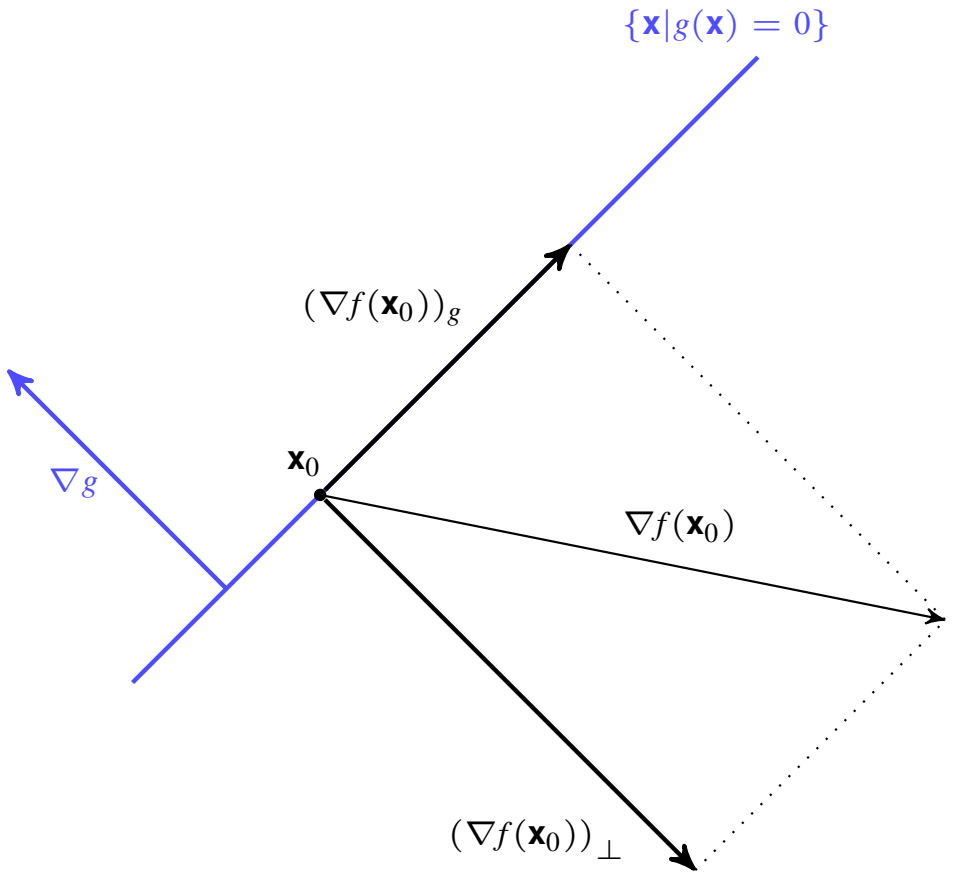
Gradients are orthogonal to contour lines.

Intuition

- The gradient points in the direction in which f grows most rapidly.
- Contour lines are sets along which f does not change.

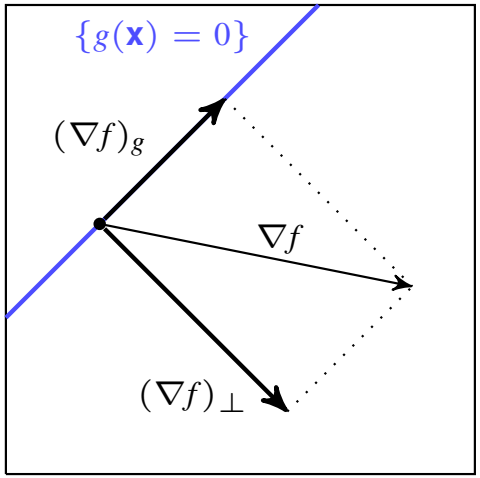


The Crucial Bit



Idea

- Decompose ∇f into a component $(\nabla f)_g$ in the set $\{\mathbf{x} \mid g(\mathbf{x}) = 0\}$ and a remainder $(\nabla f)_\perp$.
- The two components are orthogonal.
- If f_g is minimal within $\{\mathbf{x} \mid g(\mathbf{x}) = 0\}$, the component within the set vanishes.
- The remainder need not vanish.



Consequence

- We need a criterion for $(\nabla f)_g = 0$.

Solution

- If $(\nabla f)_g = 0$, then ∇f is orthogonal to the set $g(\mathbf{x}) = 0$.
- Since gradients are orthogonal to contours, and the set is a contour of g, ∇g is also orthogonal to the set.
- Hence: At a minimum of f_g, the gradients are parallel: $\nabla f + \lambda \nabla g = 0$ for some scalar λ (assuming $\nabla g \neq 0$ there).
- $\lambda = 0$ is possible: this happens exactly if $\nabla f = 0$, i.e. the point is also a stationary point of f without the constraint.

Solution

The constrained optimization problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g(\mathbf{x}) = 0 \end{aligned}$$

is solved by solving the equation system

$$\begin{aligned} \nabla f(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) &= 0 \\ g(\mathbf{x}) &= 0 \end{aligned}$$

The vectors ∇f and ∇g are d-dimensional, so the system contains $d + 1$ equations for the $d + 1$ variables $x_1, \ldots, x_d, \lambda$.

Objective

For a function f and a convex function g, solve

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{subject to } g(\mathbf{x}) \leq 0 \end{aligned}$$

i.e. we replace $g(\mathbf{x}) = 0$ as previously by $g(\mathbf{x}) \leq 0$. This problem is called an optimization problem with **inequality constraint**.

Feasible set

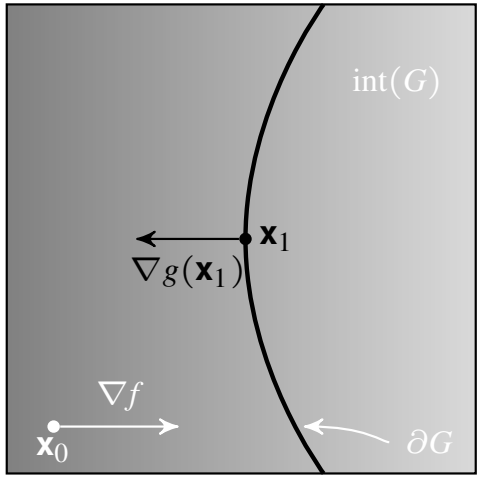
We again write G for the set of all points which satisfy the constraint,

$$G := \{\mathbf{x} \,|\, g(\mathbf{x}) \leq 0\} \ .$$

G is often called the **feasible set** (the same name is used for equality constraints).

Case distinction

1. The location $\mathbf{x}$ of the minimum can be in the *interior* of G
2. $\mathbf{x}$ may be on the *boundary* of G.



lighter shade = larger value of f

Decomposition of G

$$G = \text{int}(G) \cup \partial G = \text{ interior } \cup \text{ boundary}$$

The interior is given by $g(\mathbf{x}) < 0$, the boundary by $g(\mathbf{x}) = 0$.

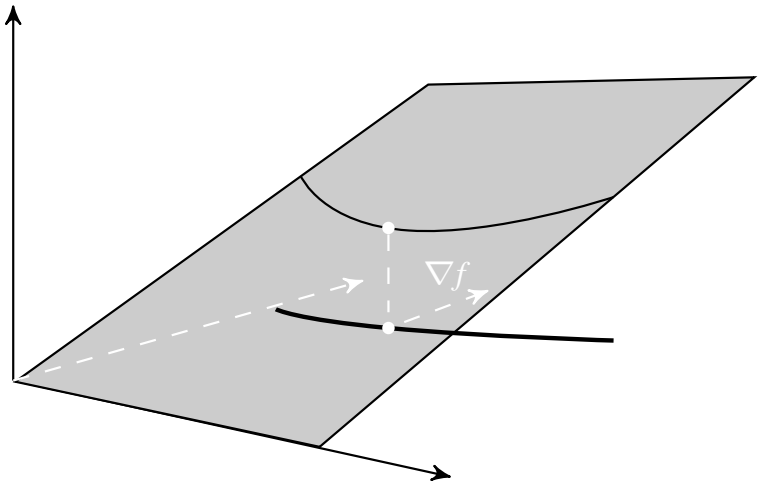
Criteria for minimum

1. **In interior:** $f_g = f$ and hence $\nabla f_g = \nabla f$. We have to solve a standard optimization problem with criterion $\nabla f = 0$.
2. **On boundary:** Here, $\nabla f_g \neq \nabla f$. Since $g(\mathbf{x}) = 0$, the geometry of the problem is the same as we have discussed for equality constraints, with criterion $\nabla f = \lambda \nabla g$.

   However: In this case, the sign of λ matters.

Observation

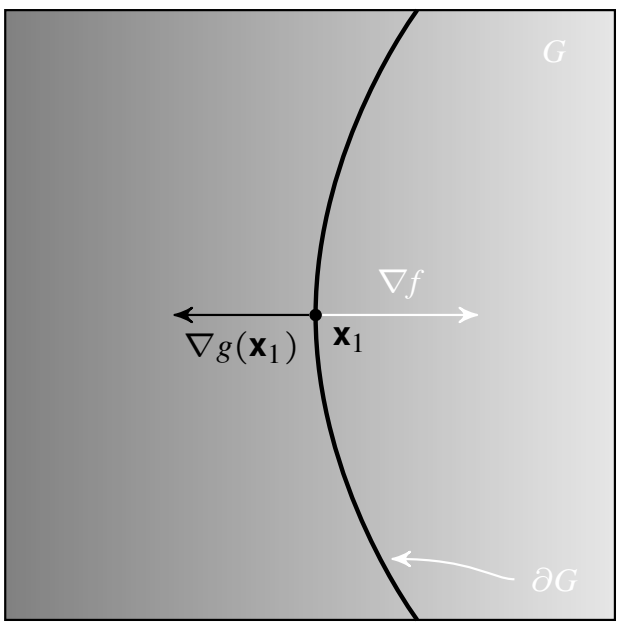
- An extremum on the boundary is a minimum if ∇f points *into* G.
- Otherwise, it is not a minimum (and may be maximum instead).



Criterion for minimum on boundary

Since ∇g points *away* from G (since g increases away from G), ∇f and ∇g have to point in opposite directions:

$$\nabla f = \lambda \nabla g \qquad \text{with } \lambda \leq 0$$



Convention

To make the sign of λ explicit, we constrain λ to non-negative values and instead write:

$$\nabla f = -\lambda \nabla g$$
$$\text{s.t. } \lambda \geq 0$$

Combined problem

$$\begin{aligned} & \nabla f = -\lambda \nabla g \\ \text{s.t.} \quad & g(\mathbf{x}) \leq 0 \\ & \lambda = 0 \text{ if } \mathbf{x} \in \operatorname{int}(G) \\ & \lambda \geq 0 \text{ if } \mathbf{x} \in \partial G \end{aligned}$$

Can we get rid of the "if $\mathbf{x} \in \cdot$" distinction?

Yes: Note that $g(\mathbf{x}) < 0$ holds if $\mathbf{x}$ is in the interior, and $g(\mathbf{x}) = 0$ on the boundary.
Hence, at least one of λ or $g(\mathbf{x})$ is always 0.

That means we can substitute

$$\begin{aligned} & \lambda = 0 \text{ if } \mathbf{x} \in \operatorname{int}(G) \\ & \lambda \geq 0 \text{ if } \mathbf{x} \in \partial G \end{aligned}$$

by

$$\lambda \cdot g(\mathbf{x}) = 0 \qquad \text{and} \qquad \lambda \geq 0 \; .$$

Combined solution

The optimization problem with inequality constraints

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{subject to } g(\mathbf{x}) \leq 0 \end{aligned}$$

can be solved by solving

$$\left.\begin{aligned} \nabla f(\mathbf{x}) &= -\lambda \nabla g(\mathbf{x}) \\ \text{s.t.} \qquad \lambda g(\mathbf{x}) &= 0 \\ g(\mathbf{x}) &\leq 0 \\ \lambda &\geq 0 \end{aligned}\right\} \longleftarrow \begin{array}{l} \text{system of } d+1 \text{ equations for } d+1 \\ \text{variables } x_1, \ldots, x_d, \lambda \end{array}$$

These conditions are known as the **Karush-Kuhn-Tucker** (or **KKT**) conditions.

Haven't we made the problem more difficult?

- To simplify the minimization of f for $g(\mathbf{x}) \leq 0$, we have made f more complicated and added a variable and two constraints. Well done.
- However: In the original problem, we *do not know how to minimize* f, since the usual criterion $\nabla f = 0$ does not work.
- By adding λ and additional constraints, we have reduced the problem to solving a system of equations.

Summary: Conditions

Condition	Ensures that...	Purpose
$\nabla f(\mathbf{x}) = -\lambda \nabla g(\mathbf{x})$	If $\lambda = 0$: ∇f is 0	Opt. criterion inside G
	If $\lambda > 0$: ∇f is anti-parallel to ∇g	Opt. criterion on boundary
$\lambda g(\mathbf{x}) = 0$	$\lambda = 0$ in interior of G	Distinguish cases int(G) and ∂G
$\lambda \geq 0$	∇f cannot flip to orientation of ∇g	Optimum on ∂G is minimum

Why Should g be Convex?

More precisely

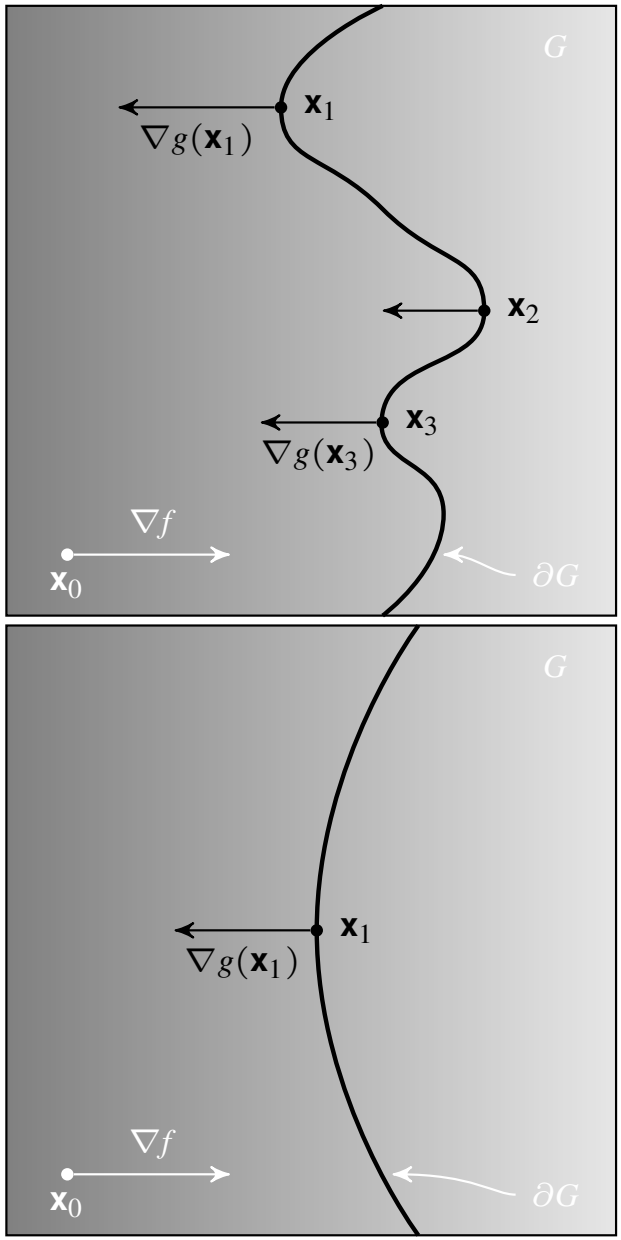
If g is a convex function, then $G = \{\mathbf{x} \mid g(\mathbf{x}) \leq 0\}$ is a convex set. Why do we require convexity of G?

Problem

If G is not convex, the KKT conditions do not guarantee that $\mathbf{x}$ is a minimum. (The conditions still hold, i.e. if G is not convex, they are necessary conditions, but not sufficient.)

Example (Figure)

- f is a linear function (lighter colour = larger value)
- ∇f is identical everywhere
- If G is not convex, there can be several points (here: $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$) which satisfy the KKT conditions. Only $\mathbf{x}_1$ minimizes f on G.
- If G is convex, such problems cannot occur.



Numerical methods for constrained problems

Once we have transformed our problem using Lagrange multipliers, we still have to solve a problem of the form

$$\begin{aligned} & \nabla f(\mathbf{x}) = -\lambda \nabla g(\mathbf{x}) \\ \text{s.t.} \quad & \lambda g(\mathbf{x}) = 0 \quad \text{and} \quad g(\mathbf{x}) \leq 0 \quad \text{and} \quad \lambda \geq 0 \end{aligned}$$

numerically.

Idea

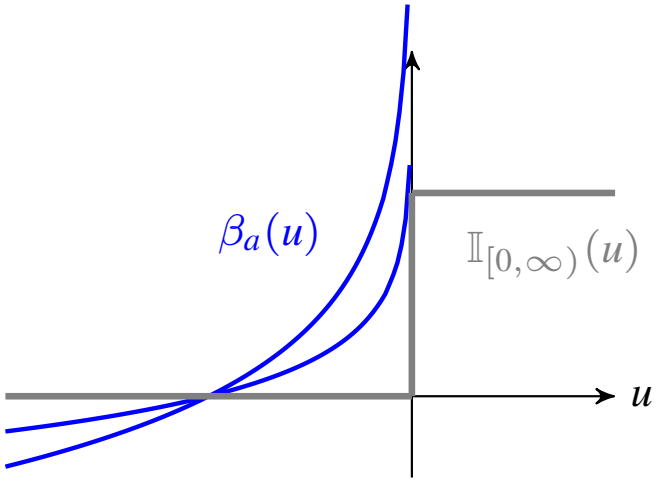
A constraint in the problem

$$\min f(\mathbf{x}) \qquad \text{s.t.} \quad g(\mathbf{x}) < 0$$

can be expressed as an indicator function:

$$\min f(\mathbf{x}) + const. \cdot \mathbb{I}_{[0,\infty)}(g(\mathbf{x}))$$

The constant must be chosen large enough to enforce the constraint.



Problem: The indicator function is piece-wise constant and not differentiable at 0. Newton or gradient descent are not applicable.

Barrier function

A **barrier function** approximates $\mathbb{I}_{[0,\infty)}$ by a smooth function, e.g.

$$\beta_a(u) := -\frac{1}{a}\log(-u) \ .$$

Interior point methods

We can (approximately) solve

$$\min f(\mathbf{x}) \text{ s.t.} \quad g_i(\mathbf{x}) < 0 \quad \text{for } i = 1, \ldots, m$$

by solving

$$\min f(\mathbf{x}) + \sum_{i=1}^{m} \beta_a(g_i(\mathbf{x})) \ .$$

with one barrier term $\beta_a(g_i(\mathbf{x}))$ for each constraint g_i.
We do not have to adjust a multiplicative constant since $\beta_a(u) \to \infty$ as $u \nearrow 0$.

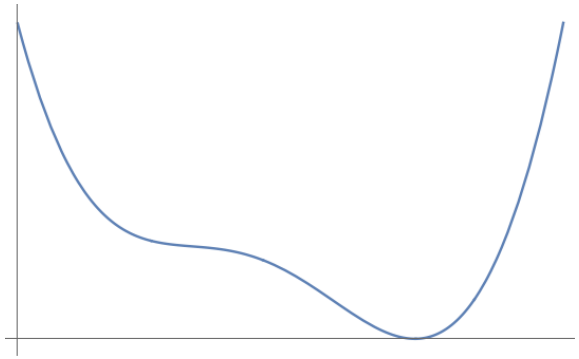
Constrained problems: General solution strategy

1. Convert constraints into barrier terms.
2. Minimize the resulting, unconstrained problem numerically (usually with Newton), increasing a.

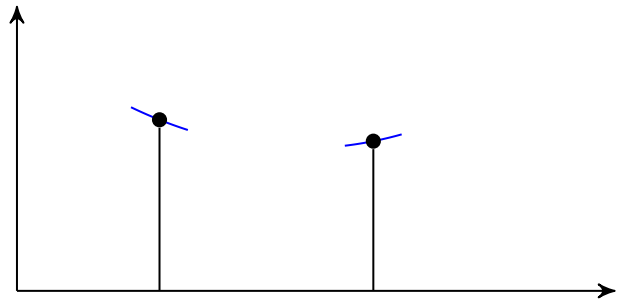
Lagrange optimization and KKT theory explain why this converges to the constrained optimum.

RELEVANCE OF CONVEXITY

A common textbook claim is that convexity is important for optimization because every local minimum of a convex function is global. That is correct, but not really the whole story.



Not convex, but unique minimum and well-suited for numerical optimization.



Recall that a numerical optimization algorithm can only query a function at points.

Convexity allows us to draw *global* conclusions from *local* properties.

Convexity as a generalization of linearity

- Constant functions: If we know $f(\mathbf{x})$ at a single point $\mathbf{x}$, we know f.
- Linear functions: If we know $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$ at a single point, we know f.
- Convex functions: If we know $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$ at a single point, we know "on which side of $\mathbf{x}$" the minimum occurs (that is: in which half-space).

There are various other global properties of convex functions that are determined by their behaviour on the neighbourhood of a single point, some of which are much deeper and more surprising than the one above.