# Optimization for Machine Learning in Python

Dr. Moritz Wolter

July 27, 2022

High-Performance Computing and Analytics Lab, Uni Bonn

# Overview

Introduction

The derivative

Optimization in a single dimension

Optimization in many dimensions

# Introduction

# Optimization

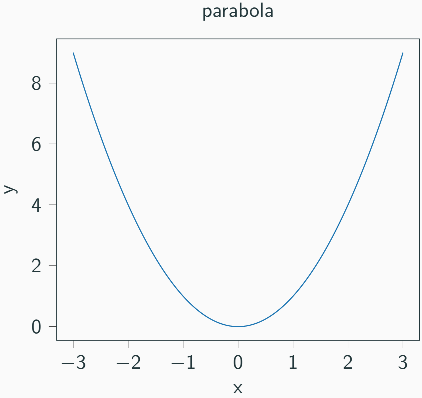
Traditionally, optimization means minimizing using a cost function $f(x)$. Given the cost, we must find the cheapest point $x^*$ on the function, or in other words,

$$x^* = \min_{x \in \mathbb{R}} f(x) \tag{1}$$

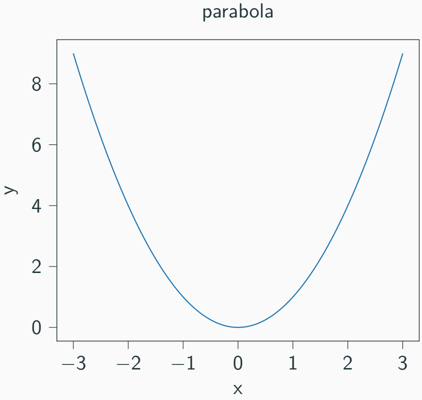
# Functions

Functions are mathematical mappings. Consider for example, the quadratic function, $f(x) : \mathbb{R} \rightarrow \mathbb{R}$:

$$f(x) = x^2 \tag{2}$$



# Where is the minimum?




In this case, we immediately see it's at zero. To find it via an iterative process, we require derivate information.

# Summary

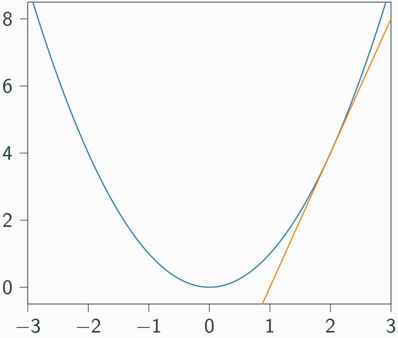
- Functions assign a value to each input.
- We seek an iterative way to find the smallest value.
- Doing so requires derivates.

# The derivative

# The derivative

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{3}$$

parabola with derivative at two



# Derivation of the parabola derivative

$$
\begin{aligned}
\lim_{h\to 0} \frac{(x+h)^2 - x^2}{h} &= \lim_{h\to 0} \frac{x^2 + 2xh + h^2 - x^2}{h} && (4) \\
&= \lim_{h\to 0} \frac{2xh + h^2}{h} && (5) \\
&= \lim_{h\to 0} \frac{h(2x+h)}{h} && (6) \\
&= \lim_{h\to 0} 2x + h && (7) \\
&= 2x && (8)
\end{aligned}
$$

Derivation of the parabola derivative

$$\begin{aligned}
\lim_{h\to 0}\frac{(x+h)^2-x^2}{h} &= \lim_{h\to 0}\frac{x^2+2xh+h^2-x^2}{h} && (4)\\
&= \lim_{h\to 0}\frac{2xh+h^2}{h} && (5)\\
&= \lim_{h\to 0}\frac{h(2x+h)}{h} && (6)\\
&= \lim_{h\to 0} 2x+h && (7)\\
&= 2x && (8)
\end{aligned}$$

Derive on the board:

$$\begin{aligned}
\lim_{h\to 0}\frac{(x+h)^2-x^2}{h} &= \lim_{h\to 0}\frac{x^2+2xh+h^2-x^2}{h} && (9)\\
&= \lim_{h\to 0}\frac{2xh+h^2}{h} && (10)\\
&= \lim_{h\to 0}\frac{h(2x+h)}{h} && (11)\\
&= \lim_{h\to 0} 2x+h && (12)\\
&= 2x && (13)
\end{aligned}$$

## Summary

- A function is differentiable if the limit of the difference quotient exists.
- For any point on a differentiable function, the derivative provides a tangent slope.
- We will exclusively work with differentiable functions in this course.

# Optimization in a single dimension

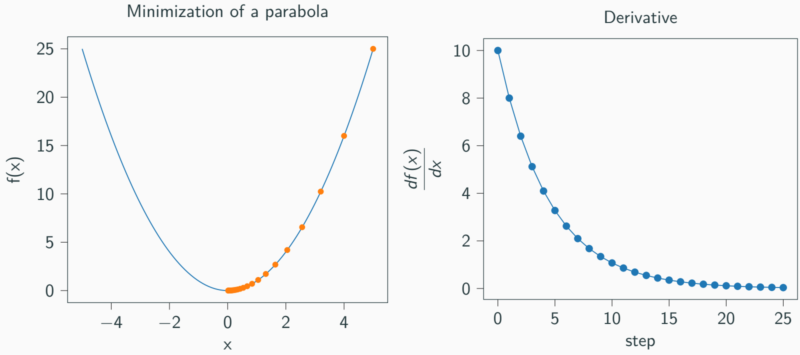
# Steepest descent

To find a minumum, we descent along the gradient, with $n$ denoting the step number, $\epsilon \in \mathbb{R}$ the step size and $\frac{df}{dx}$ the derivate of $f$ along $x \in \mathbb{R}$:

$$x_n = x_{n-1} - \epsilon \cdot \frac{df}{dx}. \tag{14}$$

# Steepest descent on the parabola

Working with the initial position $x_0 = 5$ and a step size of $\epsilon = 0.1$ for 25 steps leads to:



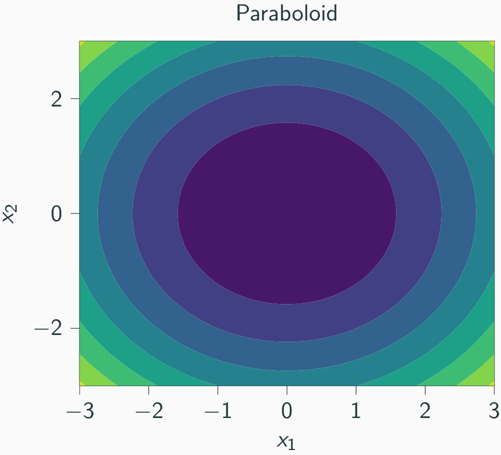
## Summary

- Following the negative derivative iteratively got us to the minimum.
- At points of interest, the first derivate is zero.

# Optimization in many dimensions

# The two-dimensional paraboloid

$$f(x_1, x_2) = x_1^2 + x_2^2 \tag{15}$$



# The gradient

The gradient lists partial derivatives with respect to all inputs in a vector. For a function $f : \mathbb{R}^n \to \mathbb{R}$ of $n$ variables the gradient $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}. \tag{16}$$

## The gradient

The gradient lists partial derivatives with respect to all inputs in a vector. For a function $f : \mathbb{R}^n \to \mathbb{R}$ of $n$ variables the gradient $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}. \tag{16}$$

- Gradients point in the steepest ascent direction.
- To find the gradient, we must compute the partial derivate with respect to every input.
- A vector collects all derivates.

# Computing the gradient of the paraboloid

$$\nabla f(x_1, x_2) = \nabla(x_1^2 + x_2^2) \tag{17}$$

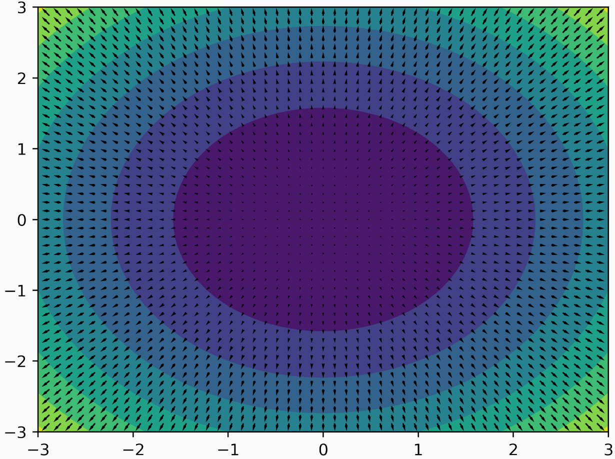
$$= \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix} \tag{18}$$

# Gradients at points

For every point $\mathbf{p} = (x_1, x_2, \ldots, x_n)$ we can write

$$\nabla f(\mathbf{p}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{p}) \\ \frac{\partial f}{\partial x_2}(\mathbf{p}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{p}) \end{pmatrix}. \tag{19}$$

# Gradients on the Paraboloid



# Gradient descent

Initial position: $x_0 = [2.9, -2.9]$,
Gradient step size: $\epsilon = 0.025$

$$x_n = x_{n-1} - \epsilon \cdot \nabla f(\mathbf{x}) \tag{20}$$

$n$ denotes the step number, $\nabla$ the gradient operator, and $f(\mathbf{x})$ a vector valued function.

## Gradient descent on the Paraboloid

Paraboloid Optimization

# The Rosenbrock test function

$$f(x_1, x_2) = (a - x_1)^2 + b(x_2 - x_1^2)^2 \tag{21}$$

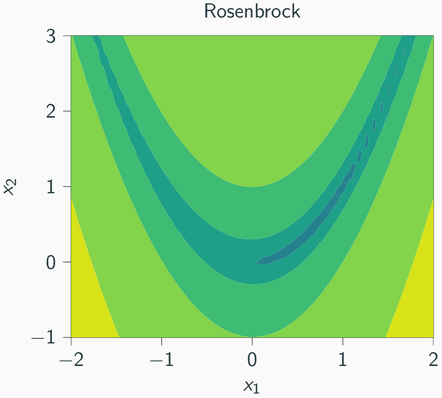



**Figure:** Rosenbrock function with a=1 and b=100 .

## The gradient of the Rosenbrock function

Recall the Rosenbrock function:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2 \tag{22}$$

$$\nabla f(x, y) = \begin{pmatrix} -2a + 2x - 4byx + 4bx^3 \\ 2by - 2bx^2 \end{pmatrix} \tag{23}$$

**The gradient of the Rosenbrock function**

Recall the Rosenbrock function:

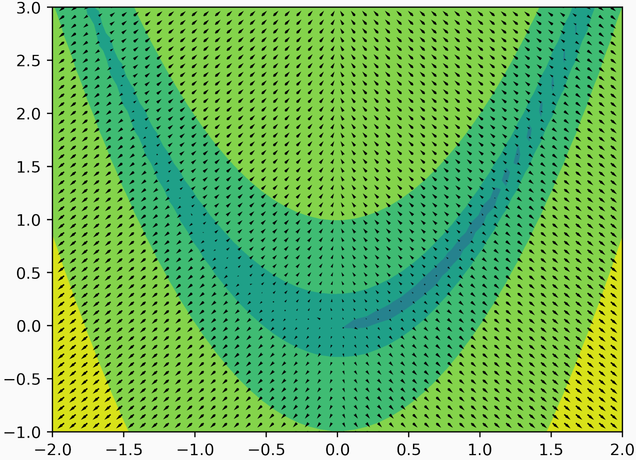
$$f(x, y) = (a - x)^2 + b(y - x^2)^2 \tag{22}$$

$$\nabla f(x, y) = \begin{pmatrix} -2a + 2x - 4byx + 4bx^3 \\ 2by - 2bx^2 \end{pmatrix} \tag{23}$$

On the board, derive:

$$\begin{aligned} f(x, y) &= (a - x)^2 + b(y - x^2)^2 && (24) \\ &= a^2 - 2ax + x^2 + b(y^2 - 2yx^2 + x^4) && (25) \\ &= a^2 - 2ax + x^2 + by^2 - 2byx^2 + bx^4 && (26) \\ \Rightarrow \frac{\partial f(x, y)}{\partial x} &= -2a + 2x - 4byx + 4bx^3 && (27) \\ \Rightarrow \frac{\partial f(x, y)}{\partial y} &= 2by - 2bx^2 && (28) \end{aligned}$$

# Gradients on the Rosenbrock function



# Gradient descent

Initial position: $x_0 = [0.1, 3.]$,
Gradient step size: $\epsilon = 0.01$

$$\mathbf{x}_n = \mathbf{x}_{n-1} - \epsilon \cdot \nabla f(\mathbf{x}) \tag{29}$$

$n$ denotes the step number, $\nabla$ the gradient operator, and $f(\mathbf{x})$ a vector valued function.

# Gradient descent on the Rosenbrock function

Rosenbrock Optimization

# Motivating Momentum

- The standard gradient descent approach gets stuck.
- What if we could somehow use a history of recent gradient information?

# Gradient descent with momentum

Initial position: $x_0 = [0.1, 3.]$,
Gradient step size: $\epsilon = 0.01$,
Momentum parameter: $\alpha = 0.8$

$$\mathbf{v} = \alpha \mathbf{v}_{n-1} - \epsilon \cdot \nabla f(\mathbf{x}) \tag{30}$$

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{v} \tag{31}$$

$\mathbf{v}$ denotes the velocity vector, $n$ the step number, $\nabla$ the gradient operator, and $f(\mathbf{x})$ a vector-valued function.

# Gradient descent with momentum

Rosenbrock Optimization

# Summary

- Gradient descent works in high-dimensional spaces!
- On the Rosenbrock function, we required momentum to find the minimum.
- Momentum adds the notion of inertia, which can help overcome local minima in some cases.
- Just like in the 1d case, the gradient equals zero at local minima and saddle points.