

Optimization for Machine Learning in Python

Dr. Moritz Wolter

August 4, 2022

High-Performance Computing and Analytics Lab, Uni Bonn

Overview

Introduction

The derivative

Optimization in a single dimension

Optimization in many dimensions

Introduction

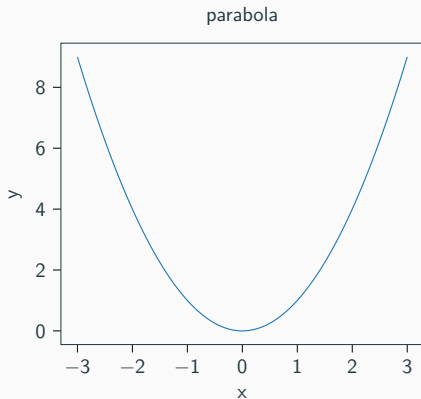
Traditionally, optimization means minimizing using a cost function $f(x)$. Given the cost, we must find the cheapest point x^* on the function, or in other words,

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

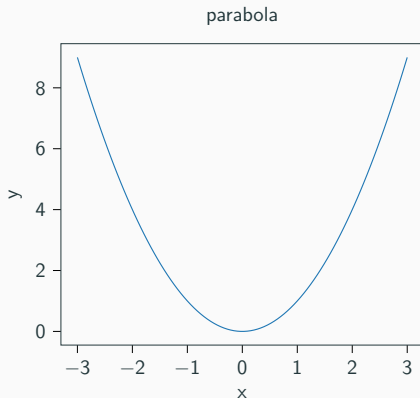
Functions

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

$$f(x) = x^2 \quad (2)$$



Where is the minimum?



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

Summary

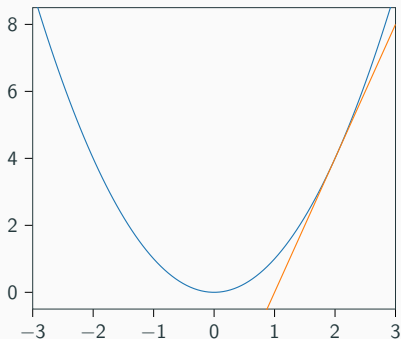
- Functions assign a value to each input.
- We seek an iterative way to find the smallest value.
- Doing so requires derivatives.

The derivative

The derivative

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

parabola with derivative at two



Derivation of the parabola derivative

$$\lim_{h \rightarrow 0} \frac{(x + h)^2 - x^2}{h} = \lim_{h \rightarrow 0} \frac{x^2 + 2xh + h^2 - x^2}{h} \quad (4)$$

$$= \lim_{h \rightarrow 0} \frac{2xh + h^2}{h} \quad (5)$$

$$= \lim_{h \rightarrow 0} \frac{h(2x + h)}{h} \quad (6)$$

$$= \lim_{h \rightarrow 0} 2x + h \quad (7)$$

$$= 2x \quad (8)$$

Optimization for Machine Learning in Python

└ The derivative

└ Derivation of the parabola derivative

$$\lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2}{h} = \lim_{h \rightarrow 0} \frac{x^2 + 2xh + h^2 - x^2}{h} \quad (4)$$

$$= \lim_{h \rightarrow 0} \frac{2xh + h^2}{h} \quad (5)$$

$$= \lim_{h \rightarrow 0} \frac{h(2x + h)}{h} \quad (6)$$

$$= \lim_{h \rightarrow 0} 2x + h \quad (7)$$

$$= 2x \quad (8)$$

Derive on the board.

Derivate of a parabola:

$$\lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2}{h} = \lim_{h \rightarrow 0} \frac{x^2 + 2xh + h^2 - x^2}{h} \quad (9)$$

$$= \lim_{h \rightarrow 0} \frac{2xh + h^2}{h} \quad (10)$$

$$= \lim_{h \rightarrow 0} \frac{h(2x + h)}{h} \quad (11)$$

$$= \lim_{h \rightarrow 0} 2x + h \quad (12)$$

$$= 2x \quad (13)$$

The derivate of a polynomial

What is the derivative of the function $f(x) = x^n$?

$$\frac{df(x)}{dx} = nx^{n-1} \quad (14)$$

Optimization for Machine Learning in Python

└ The derivative

└ The derivate of a polynomial

What is the derivative of the function $f(x) = x^2$?

$$\frac{df(x)}{dx} = 2x^{2-1}$$

(14)

Derivate of a polynomial $f(x) = x^n$ [DFO20]:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} = \lim_{h \rightarrow 0} \frac{(x+h)^n - x^n}{h} \quad (15)$$

$$= \lim_{h \rightarrow 0} \frac{\sum_{i=0}^n \binom{n}{i} x^{n-1} h^i - x^n}{h} \quad (16)$$

$$= \lim_{h \rightarrow 0} \frac{\sum_{i=1}^n \binom{n}{i} x^{n-1} h^i}{h} \quad (17)$$

$$= \lim_{h \rightarrow 0} \sum_{i=1}^n \binom{n}{i} x^{n-1} h^{i-1} \quad (18)$$

$$= \lim_{h \rightarrow 0} \left(\binom{n}{1} x^{n-1} \right) + \sum_{i=2}^n i \binom{n}{i} x^{n-i} h^{i-1} \quad (19)$$

$$= \frac{n!}{1!(n-1)!} x^{n-1} = nx^{n-1}. \quad (20)$$

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.

Differentiation Rules [DFO20]

$$\text{Product Rule: } (g(x)h(x))' = g'(x)h(x) + g(x)h'(x) \quad (21)$$

$$\text{Quotient Rule: } \left(\frac{g(x)}{h(x)}\right)' = \frac{g'(x)h(x) - g(x)h'(x)}{(h(x))^2} \quad (22)$$

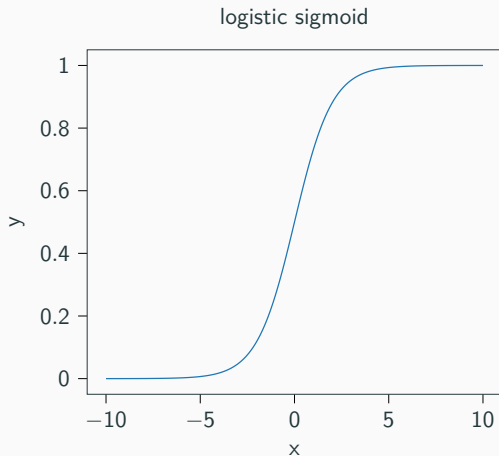
$$\text{Sum Rule: } (g(x) + h(x))' = g'(x) + h'(x) \quad (23)$$

$$\text{Chain Rule: } (g(h(x)))' = g'(h(x))h'(x) \quad (24)$$

The logistic sigmoid [GBC16]

The sigmoid function $\sigma(x)$ is a common activation function.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (25)$$



Using the Quotient Rule i

$$\frac{d\sigma(x)}{dx} = \frac{d}{dx} \frac{1}{1 + e^{-x}} \quad (26)$$

$$= \frac{d}{dx} \frac{1}{1 + e^{-x}} \cdot 1 \quad (27)$$

$$= \frac{d}{dx} \frac{1}{1 + e^{-x}} \cdot \frac{e^x}{e^x} \quad (28)$$

$$= \frac{d}{dx} \frac{e^x}{e^x + 1} \quad (29)$$

$$g(x) = e^x, h(x) = e^x + 1 \quad (30)$$

Using the Quotient Rule ii

$$\frac{d\sigma(x)}{dx} = \frac{e^x \cdot (e^x + 1) - e^x \cdot e^x}{(e^x + 1)^2} \quad (31)$$

$$= \frac{e^x \cdot e^x + e^x - e^x \cdot e^x}{(e^x + 1)^2} \quad (32)$$

$$= \frac{e^x}{(e^x + 1)^2} \quad (33)$$

$$= \frac{e^x}{(e^x + 1)} \frac{1}{(e^x + 1)} \quad (34)$$

$$= \frac{e^x}{(e^x + 1)} \left(\frac{1 + e^x - e^x}{(e^x + 1)} \right) \quad (35)$$

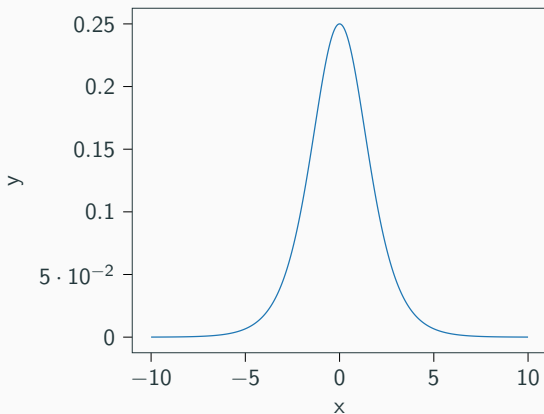
$$= \frac{e^x}{(e^x + 1)} \left(\frac{1 + e^x}{(e^x + 1)} - \frac{e^x}{(e^x + 1)} \right) \quad (36)$$

$$= \frac{e^x}{(e^x + 1)} \left(1 - \frac{e^x}{(e^x + 1)} \right) \quad (37)$$

$$= \sigma(x)(1 - \sigma(x)) \quad (38) \quad 13$$

The derivative of the sigmoidal function

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x)) \quad (39)$$



Using the Chain Rule

How do we best differentiate $f(x) = \sigma(ax + b)$?

$$\frac{df(x)}{dx} = \frac{d\sigma(ax + b)}{dx} \quad (40)$$

$$(41)$$

Chain Rule: $(g(h(x)))' = g'(h(x))h'(x)$

$$g(x) = \sigma(x), h(x) = ax + b \quad (42)$$

$$\Rightarrow \sigma(ax + b)(1 - \sigma(ax + b))(a) \quad (43)$$

Optimization in a single dimension

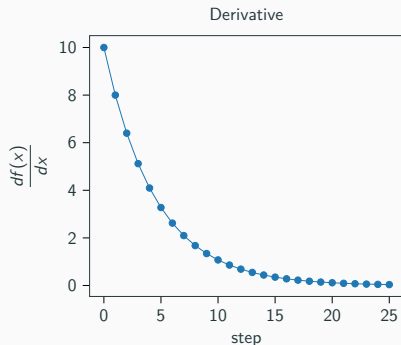
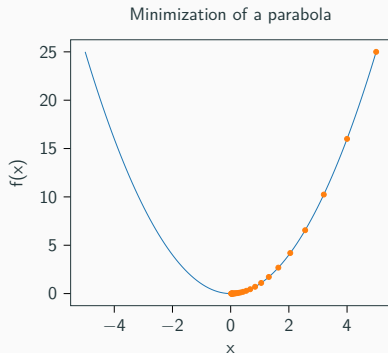
Steepest descent

To find a minimum, 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 $x \in \mathbb{R}$:

$$x_n = x_{n-1} - \epsilon \cdot \frac{df}{dx}. \quad (44)$$

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:



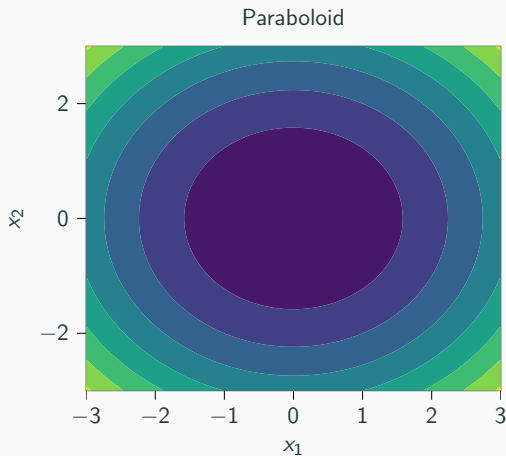
Summary

- 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

$$f(x_1, x_2) = x_1^2 + x_2^2 \quad (45)$$



The gradient

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

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}. \quad (46)$$

Optimization for Machine Learning in Python

└ Optimization in many dimensions

└ The gradient

- 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 derivatives.

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

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}. \quad (46)$$

Computing the gradient of the paraboloid

$$\nabla f(x_1, x_2) = \nabla(x_1^2 + x_2^2) \quad (47)$$

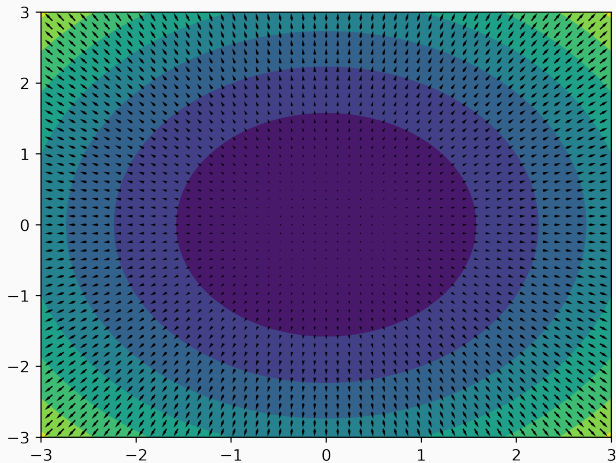
$$= \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix} \quad (48)$$

Gradients at points

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

$$\nabla f(\mathbf{p}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{p}) \\ \frac{\partial f}{\partial x_2}(\mathbf{p}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{p}) \end{pmatrix}. \quad (49)$$

Gradients on the Paraboloid



Gradient descent

Initial position: $x_0 = [2.9, -2.9]$,

Gradient step size: $\epsilon = 0.025$

$$x_n = x_{n-1} - \epsilon \cdot \nabla f(\mathbf{x}) \quad (50)$$

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

Gradient descent on the Paraboloid

Paraboloid Optimization

The Rosenbrock test function

$$f(x_1, x_2) = (a - x_1)^2 + b(x_2 - x_1^2)^2 \quad (51)$$

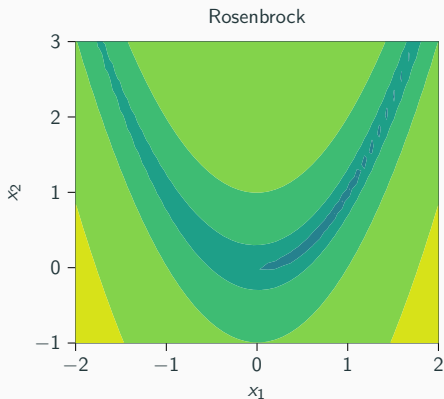


Figure: Rosenbrock function with $a=1$ and $b=100$.

The gradient of the Rosenbrock function

Recall the Rosenbrock function:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2 \quad (52)$$

$$\nabla f(x, y) = \begin{pmatrix} -2a + 2x - 4byx + 4bx^3 \\ 2by - 2bx^2 \end{pmatrix} \quad (53)$$

Optimization for Machine Learning in Python

└ Optimization in many dimensions

└ The gradient of the Rosenbrock function

Recall the Rosenbrock function:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2 \quad (52)$$

$$\nabla f(x, y) = \begin{pmatrix} -2a + 2x - 4byx + 4bx^3 \\ 2by - 2bx^2 \end{pmatrix} \quad (53)$$

On the board, derive:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2 \quad (54)$$

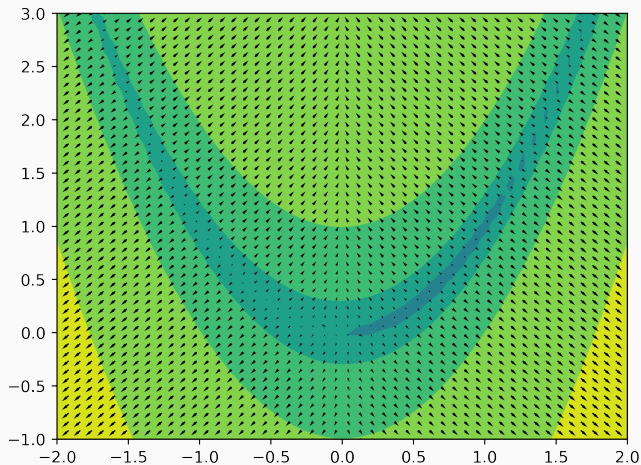
$$= a^2 - 2ax + x^2 + b(y^2 - 2yx^2 + x^4) \quad (55)$$

$$= a^2 - 2ax + x^2 + by^2 - 2byx^2 + bx^4 \quad (56)$$

$$\Rightarrow \frac{\partial f(x, y)}{\partial x} = -2a + 2x - 4byx + 4bx^3 \quad (57)$$

$$\Rightarrow \frac{\partial f(x, y)}{\partial y} = 2by - 2bx^2 \quad (58)$$

Gradients on the Rosenbrock function



Gradient descent

Initial position: $\mathbf{x}_0 = [0.1, 3.]$,

Gradient step size: $\epsilon = 0.01$

$$\mathbf{x}_n = \mathbf{x}_{n-1} - \epsilon \cdot \nabla f(\mathbf{x}) \quad (59)$$

n denotes the step number, ∇ 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: $\mathbf{x}_0 = [0.1, 3.]$,

Gradient step size: $\epsilon = 0.01$,

Momentum parameter: $\alpha = 0.8$

$$\mathbf{v} = \alpha \mathbf{v}_{n-1} - \epsilon \cdot \nabla f(\mathbf{x}) \quad (60)$$

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{v} \quad (61)$$

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

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.

- Mathematics for machine learning, [DFO20, Chapter 5, Vector Calculus]
- Deep learning, [WN+99, Chapter 8.2, Automatic Differentiation]
- Numerical optimization, [GBC16, Chapter 8, Optimization for Training Deep Models]

References

- [DFO20] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.
- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [WN+99] Stephen Wright, Jorge Nocedal, et al. “Numerical optimization.” In: *Springer Science* 35.67-68 (1999), p. 7.