Mathematical Preliminaries II

Machine Learning 1 UE (INP.33761UF)

Join here: `fbr.io/ml1p3`

Thomas Wedenig

Mar 20, 2024

Institute of Theoretical Computer Science
Graz University of Technology, Austria

Organization

- **No in-person lecture** on April 10th
  - Appointment is already cancelled in TUGonline
- Session will be **pre-recorded** and will be available on TUbe (on April 10th)
- Any questions/comments $\rightarrow$ TC forum

Differential Calculus

- Let $f : \mathbb{R} \to \mathbb{R}$ be a **univariate function**
- If it exists, **the derivative of** f w.r.t. x is defined as

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

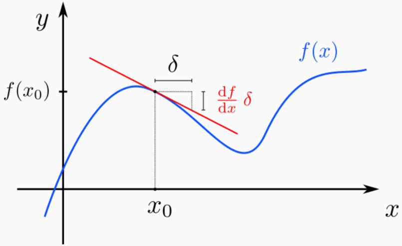
Intuition

- I know $f(x)$ for some x
- **I slightly change the input** (from x to e.g. $x + 0.00001$)
- How will the **output** $f(x + 0.00001)$ change (in proportion to the change in input)?
- i.e., **how sensitive** is f to changes to its inputs (at particular points)?

Best Linear Approximation

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

- "**Instantaneous**" **rate of change**
- $f'(x)$ is the **slope of the tangent line** going through x
- This is the best **local linear approximation**

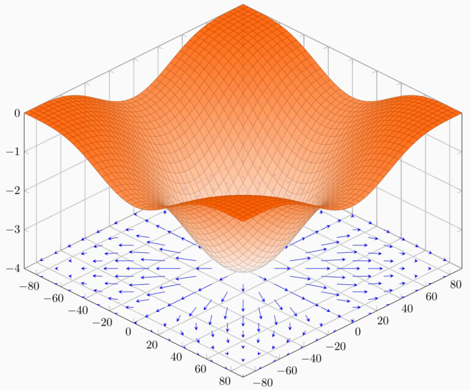


- Let $f(x_1, x_2, \ldots, x_D)$ be a **multi-variate scalar-valued** function ($f : \mathbb{R}^D \to \mathbb{R}$)

- The **gradient** of f w.r.t. $\mathbf{x} = (x_1, \ldots, x_D)^T$ at some point $\mathbf{x}$ is defined as

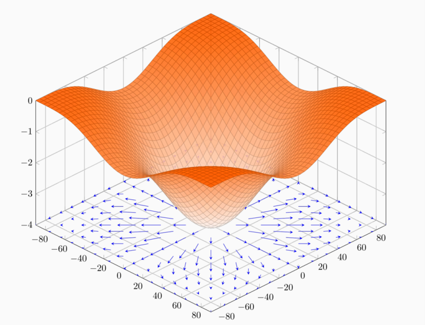
$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_D} \end{pmatrix}$$

- **$\nabla f(\mathbf{x})$ points in the direction of steepest ascent**



https://commons.wikimedia.org/wiki/File:3d-gradient-cos.svg

Why care about Gradients?

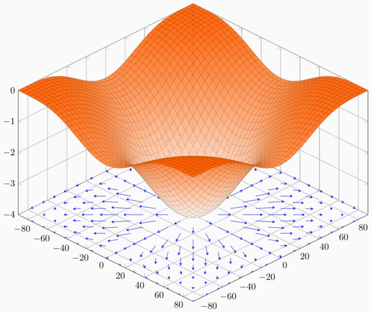


https://commons.wikimedia.org/wiki/File:3d-gradient-cos.svg

Question ?

ML folks **care a lot about gradients.**

But why? 🤔



https://commons.wikimedia.org/wiki/File:3d-gradient-cos.svg

Question ?

ML folks **care a lot about gradients.**

But why? 🤔

Optimization !

Often in ML, we are given $f : \mathbb{R}^D \to \mathbb{R}$ and seek

$$\mathrm{x}^* = \underset{\mathrm{x}}{\operatorname{argmin}} f(\mathrm{x})$$

$\rightarrow \nabla f(\mathrm{x})$ **is useful in finding the minimum**

- Find $\nabla f(\mathbf{x})$ for

$$f(\mathbf{x}) = \|\mathbf{x}\|_2^2 = \mathbf{x}^T\mathbf{x}, \qquad \mathbf{x} = (x_1, \ldots, x_N)^T$$

- Let's compute a **single partial derivative** (w.r.t. some x_k)
- Since $f(\mathbf{x}) = \sum_{i=1}^{N} x_i^2$, we have for some $k \in \{1, \ldots, N\}$:

$$\frac{\partial f(\mathbf{x})}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{i=1}^{N} x_i^2 = \sum_{i=1}^{N} \frac{\partial}{\partial x_k} x_i^2 = 2x_k.$$

- Thus, $\nabla f(\mathbf{x}) = (2x_1,\ 2x_2,\ \ldots, 2x_N)^T = 2\mathbf{x}$

- Let $f(\mathbf{x})$ be a **vector-valued** function ($f : \mathbb{R}^D \to \mathbb{R}^M$), defined as

$$f(\mathbf{x}) = (f_1(\mathbf{x}),\ f_2(\mathbf{x}),\ \ldots,\ f_M(\mathbf{x}))^T$$

  where $f_i(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$.

- The **Jacobian matrix** of f at some point $\mathbf{x}$ is

$$J_f(\mathbf{x}) = \begin{pmatrix} \nabla f_1(\mathbf{x})^T \\ \nabla f_2(\mathbf{x})^T \\ \vdots \\ \nabla f_M(\mathbf{x})^T \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_D} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_D} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_M(\mathbf{x})}{\partial x_1} & \frac{\partial f_M(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_M(\mathbf{x})}{\partial x_D} \end{pmatrix} \in \mathbb{R}^{M \times D}$$

Chain Rule in 1D

If $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$, then

$$(f \circ g)'(x) = f'(g(x))g'(x)$$

Chain Rule in arbitrary dimensions

If $g : \mathbb{R}^D \to \mathbb{R}^E$ and $f : \mathbb{R}^E \to \mathbb{R}^F$, then

$$J_{f \circ g}(\mathbf{x}) = J_f(g(\mathbf{x}))\, J_g(\mathbf{x})$$

- Jacobians are **the most general form of** “derivatives”
  - Gradients are (transposed) Jacobians
  - Scalar derivatives are 1×1 Jacobians
- **Training Neural Networks $\approx$ applying the Jacobian chain rule** !
- This is called **Backpropagation**
  - More on this later in the semester
- In practice, **Autodiff frameworks** keep track of Jacobians in the background
  - Examples include *Tensorflow, PyTorch, JAX*
  - Especially important in **“Deep Learning”**

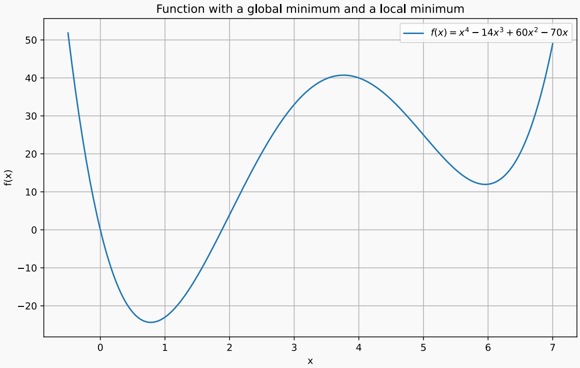
Optimization

Unconstrained Optimization

Let $f : \mathbb{R}^n \to \mathbb{R}$. Find $\mathbf{x}^* \in \mathbb{R}^n$ s.t.

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} f(\mathbf{x})$$

- If $\mathbf{x}^*$ exists, it is called a **global minimizer**
- In general, there could be **multiple local minima**

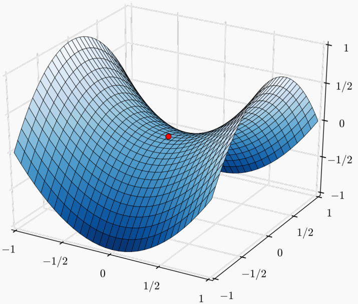


Question ?

Let f be differentiable everywhere and assume $\nabla f(\mathbf{x}) = \mathbf{0}$ for some $\mathbf{x}$.
Is $\mathbf{x}$ guaranteed to be a **local minimum/maximum**?

Question ?

Let f be differentiable everywhere and assume $\nabla f(\mathbf{x}) = \mathbf{0}$ for some $\mathbf{x}$.
Is $\mathbf{x}$ guaranteed to be a **local minimum/maximum**?



https://commons.wikimedia.org/wiki/File:Saddle_point.svg

- **No**, it could also be a **saddle point**
- When $\nabla f(\mathbf{x}) = \mathbf{0}$ we call $\mathbf{x}$ a **stationary point**, which is either
  - Local (and possibly global) minimum/maximum
  - Saddle Point

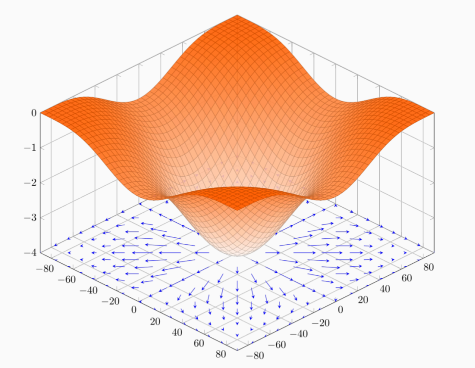
- If we set $\nabla f(\mathbf{x}) = \mathbf{0}$ and "solve" for $\mathbf{x}$ analytically, we have **candidate solutions**, right?
- In ML, we **usually do not have analytical solutions** to $\nabla f(\mathbf{x}) = \mathbf{0}$
  - Except, for example, **Linear Regression** with **Least Squares Loss**
- Even if we have access to analytical solutions, they sometimes **do not scale well to large datasets**
- **However:** We have **iterative optimization methods** !

Gradient Descent (GD)

1. Select initial point $\mathbf{x}^{(0)}$ (e.g., randomly)
2. while $\|\nabla f(\mathbf{x}^{(t)})\|_2 > \varepsilon$ (for small $\varepsilon > 0$):

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \nabla f(\mathbf{x}^{(t)})$$

with η_t denoting the **learning rate (step size)**



https://commons.wikimedia.org/wiki/File:3d-gradient-cos.svg

- There exist many variants of this
- Gradient Descent works **very well in high dimensions**
- Ubiquitous in ML
  - Training Neural Networks, Logistic Regression, Linear Regression etc.
- In practice, we usually use **Stochastic Gradient Descent**
  - There, we don't have access to the gradient $\nabla f(\mathbf{x})$, but to a **noisy estimate** of it
- Choice of **step size** is crucial

- There exist many variants of this
- Gradient Descent works **very well in high dimensions**
- Ubiquitous in ML
  - Training Neural Networks, Logistic Regression, Linear Regression etc.
- In practice, we usually use **Stochastic Gradient Descent**
  - There, we don't have access to the gradient $\nabla f(\mathbf{x})$, but to a **noisy estimate** of it
- Choice of **step size** is crucial

Importance of gradient-based optimization

The success of ML/AI hinges on the fact that first-order optimization works so well in high dimensions !

- For some ML concepts, **constrained optimization** is useful
  - Support Vector Machines, Principal Component Analysis
- Simplified problem statement:

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad \text{subject to} \quad g(\mathbf{x}) = 0$$

- Objective function $f : \mathbb{R}^n \to \mathbb{R}$, constraint function $g : \mathbb{R}^n \to \mathbb{R}$
- **Langrangian function** $L : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ defined as

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

- Find stationary points $(\mathbf{x}^*, \lambda^*)$ by setting $\nabla_{\mathbf{x},\lambda} L(\mathbf{x}^*, \lambda^*) = \mathbf{0}$

- For some $A \in \mathbb{R}^{m\times n}$, solve

$$\min_{\mathbf{x}} \|A\mathbf{x}\|_2^2 \qquad \text{subject to} \quad \|\mathbf{x}\|_2^2 = 1$$

- The constraint reads $g(\mathbf{x}) = \|\mathbf{x}\|_2^2 - 1 = 0$ and thus, the Lagrangian is

$$L(\mathbf{x}, \lambda) = \|A\mathbf{x}\|_2^2 + \lambda(\|\mathbf{x}\|_2^2 - 1)$$

$$(A x)^T (A x)$$

$$\nabla_x \{ x^T A^T A x \} = 2 A^T A x$$

- We compute

$$\begin{pmatrix} \nabla_{\mathbf{x}} L(\mathbf{x}, \lambda) \\ \nabla_{\lambda} L(\mathbf{x}, \lambda) \end{pmatrix} = \begin{pmatrix} 2A^T A\mathbf{x} + \lambda 2\mathbf{x} \\ \|\mathbf{x}\|_2^2 - 1 \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}$$

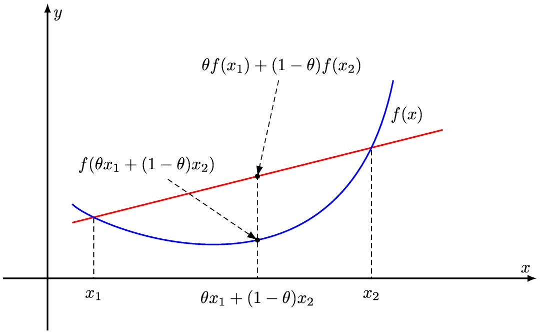
- Therefore, for a stationary $\mathbf{x}^*$ we have $A^T A\mathbf{x}^* = -\lambda\mathbf{x}^*$
- $\mathbf{x}^*$ is the **eigenvector** of $A^T A$ with the **smallest** eigenvalue

Convex Functions

- A function $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ and $\theta \in [0, 1]$ we have

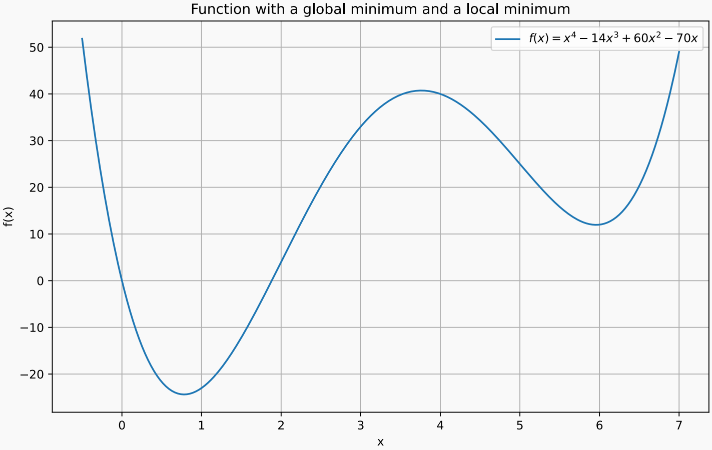
$$f(\theta \mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \le \theta f(\mathbf{x}_1) + (1-\theta) f(\mathbf{x}_2)$$

- For every two points the **line connecting them is above (or on) the function graph**

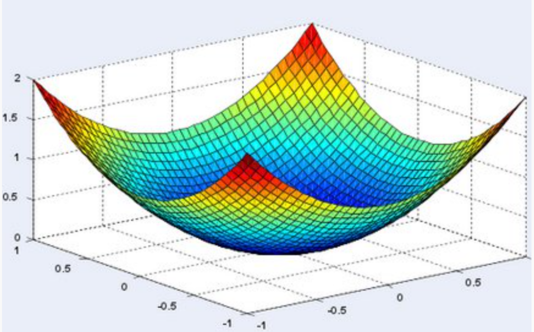


https://tisp.indigits.com/convex_sets/convex_functions

Is this a convex function?

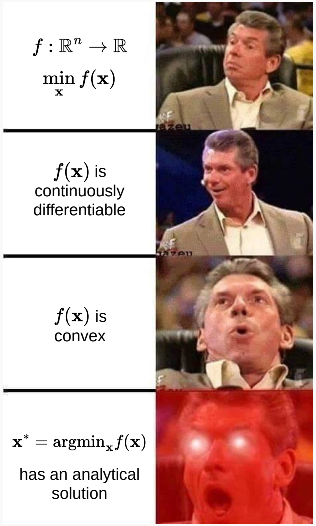


- Assume f is **convex**
- **Every local minimum** is a **global minimum** 😍
- Also, there are no **saddle points** 😍



https://www.quora.com/Are-all-quadratic-programming-problems-convex

Function Properties



Optimization - Question Time

`fbr.io/ml1p3`