Mathematical Preliminaries I

Machine Learning 1 UE (INP.33761UF)

Join here: `fbr.io/ml1p2`

Thomas Wedenig

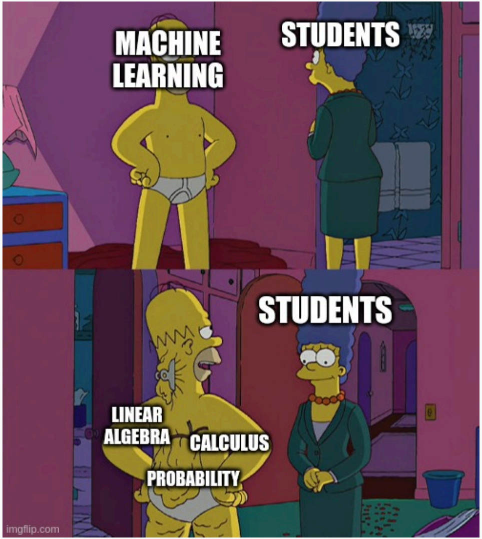
Mar 13, 2024

Institute of Theoretical Computer Science
Graz University of Technology, Austria

Organization

- I will notify you **in class** whenever the syllabus changes
- You can search groups in the **Search Groups** forum on TC

Mathematics in ML



Linear Algebra

- Shows up almost **everywhere** in ML
- **Extremely important** from a computational perspective
  - Linear regression $\Rightarrow$ Matrix Inversion,
  - Principal Component Analysis $\Rightarrow$ Finding Eigenvectors
  - Neural Networks $\Rightarrow$ Matrix Multiplication

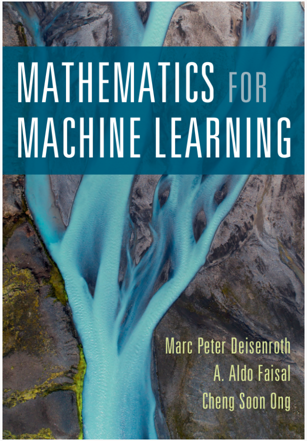
Calculus

- Minimizing continuous, smooth functions (**loss functions**)
- Will often be **high-dimensional** $\rightarrow$ Multivariate Calculus (Matrix Calculus)

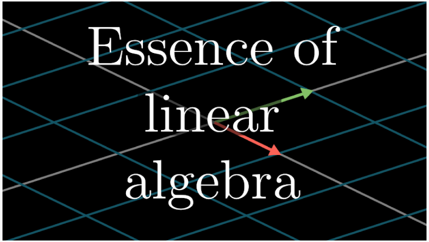
Probability Theory

- "**Optimal Reasoning under Uncertainty**" (Real world $\Rightarrow$ Noisy data/predictions)

Recommended Material



- Mathematics for Machine Learning (Deisenroth et al.)
  - `https://mml-book.github.io`



- 3Blue1Brown: Essence of Linear Algebra
  - Intuitive, visuals-first introduction
  - `https://youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab`

Linear Algebra

Dot Product

- Given vectors

$$\mathbf{x} = (x_1, x_2, \ldots, x_N)^T \in \mathbb{R}^N, \qquad \mathbf{y} = (y_1, y_2, \ldots, y_N)^T \in \mathbb{R}^N$$

- We define the **dot product** as

$$\mathbf{x}^T\mathbf{y} = \mathbf{y}^T\mathbf{x} = \sum_{i=1}^{N} x_i y_i$$

- NumPy: `x.T @ y`

Important Norms

For $\mathbf{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$, we define

- The ℓ_2 norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^N x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$.
  - Thus, $\|\mathbf{x}\|_2^2 = \sum_{i=1}^N x_i^2 = \mathbf{x}^T\mathbf{x}$ **!**
- The ℓ_1 norm $\|\mathbf{x}\|_1 = \sum_{i=1}^N |x_i|$

Important Norms

For $\mathbf{x} = (x_1, \dots, x_N)^T \in \mathbb{R}^N$, we define

- The ℓ_2 norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^N x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$.
  - Thus, $\|\mathbf{x}\|_2^2 = \sum_{i=1}^N x_i^2 = \mathbf{x}^T\mathbf{x}$ **!**
- The ℓ_1 norm $\|\mathbf{x}\|_1 = \sum_{i=1}^N |x_i|$

NumPy

- `np.linalg.norm(x, ord=p)`
- $p = 1$ or $p = 2$ in this case

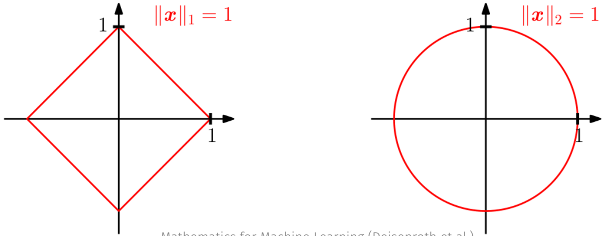
Important Norms

For $\mathbf{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$, we define

- The ℓ_2 norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^N x_i^2} = \sqrt{\mathbf{x}^T \mathbf{x}}$.
  - Thus, $\|\mathbf{x}\|_2^2 = \sum_{i=1}^N x_i^2 = \mathbf{x}^T \mathbf{x}$ **!**
- The ℓ_1 norm $\|\mathbf{x}\|_1 = \sum_{i=1}^N |x_i|$

NumPy

- `np.linalg.norm(x, ord=p)`
- $p = 1$ or $p = 2$ in this case



Mathematics for Machine Learning (Deisenroth et al.)

- Matrix $A \in \mathbb{R}^{M \times N}$
  - Can be seen as **collection** of N column vectors $\in \mathbb{R}^M$

Matrix-vector product

$$\mathbf{y} = A\mathbf{x}, \qquad \mathbf{x} \in \mathbb{R}^N, \mathbf{y} \in \mathbb{R}^M$$

- NumPy: `y = A @ x`
- A is a **linear map** that transforms $\mathbf{x} \in \mathbb{R}^N$ into $\mathbf{y} \in \mathbb{R}^M$

Linear Map

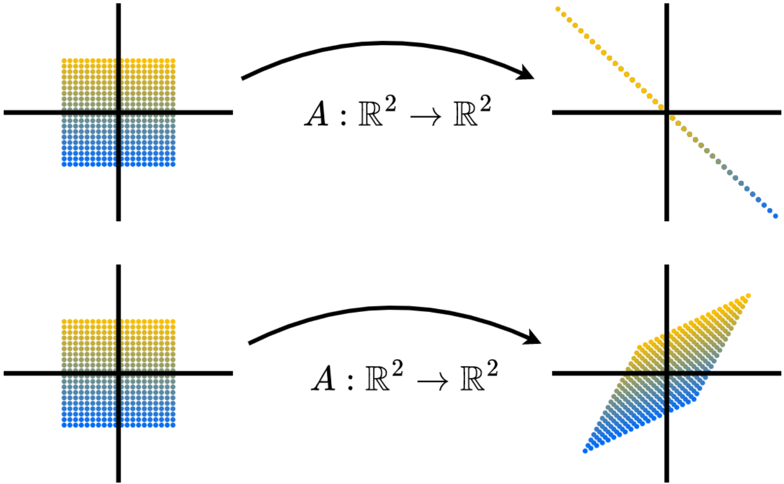
A map $f : \mathbb{R}^N \to \mathbb{R}^M$ is **linear** iff

- $f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$
- $f(\lambda \mathbf{x}) = \lambda f(\mathbf{x}), \ \lambda \in \mathbb{R}$

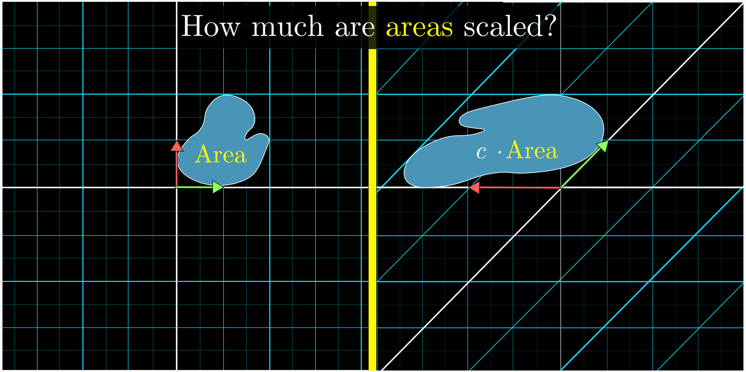
We omit the parenthesis:

- $A(\mathbf{x} + \mathbf{y}) = A\mathbf{x} + A\mathbf{y}$
- $A(\lambda \mathbf{x}) = \lambda A\mathbf{x}, \ \lambda \in \mathbb{R}$

Visualizing Matrix-vector multiplication



Adapted from Mathematics for Machine Learning (Deisenroth et al.)

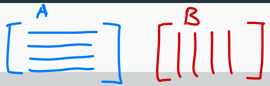


https://www.3blue1brown.com/lessons/determinant

- Let $A \in \mathbb{R}^{N \times N}$ be a **square matrix** (A maps from $\mathbb{R}^N$ to $\mathbb{R}^N$)
- **The determinant** $\det(A) \in \mathbb{R}$ quantifies the **change of volume** when applying A

Matrix-matrix product

- Matrix $A \in \mathbb{R}^{M \times N}$ and matrix $B \in \mathbb{R}^{N \times K}$



Matrix-matrix product

$$C = AB \in \mathbb{R}^{M \times K} \qquad C_{ij} = \sum_k A_{ik} B_{kj}$$

- NumPy: `C = A @ B`
- $AB\mathbf{x}$ means that $\mathbf{x}$ is **first** transformed according to B, **then** the output is transformed according to A

- In general, $AB \neq BA$ **!**
- Distributive: $A(B + C) = AB + AC$ for matching A, B, C
- Associative: $A(BC) = (AB)C$ for matching A, B, C
  - The **number of sums and products** might be very different between $A(BC)$ and $(AB)C$

- **Symmetric** matrix A:

$$A = A^T, \qquad \text{for example } A = \begin{pmatrix} 3 & -1 & 6 \\ -1 & 2 & 9 \\ 6 & 9 & 1 \end{pmatrix}$$

- **Identity** matrix I, e.g. $I \in \mathbb{R}^{3\times 3}$:

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad I\mathbf{x} = \mathbf{x} \;\; \forall \mathbf{x} \in \mathbb{R}^3$$

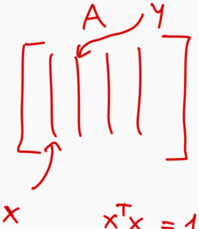
- For **some square matrices** A the **inverse** A^{-1} exists:

$$A^{-1}A = AA^{-1} = I$$

- A **square matrix** $A \in \mathbb{R}^{N \times N}$ is called **orthonormal** if

$$A^T A = AA^T = I$$

- Sometimes just called **Orthogonal Matrix**
- Columns are **pairwise orthogonal to each other**
  - i.e., dot product between **different columns** is 0
- All columns have **unit ℓ_2 norm**
  - i.e., dot product between **same column** is 1 (squared ℓ_2 norm)
- **The transpose is its inverse** **!**



$x^T x = 1 = \|x\|_2^2$

$x^T y = 0$

Useful Identities

- $(AB)^{-1} = B^{-1}A^{-1}$
- $(A+B)^{-1} \neq A^{-1} + B^{-1}$
- $\left(A^T\right)^T = A$
- $(A+B)^T = A^T + B^T$
- $(AB)^T = B^T A^T$

- Consider

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

- Matrix-vector product **computes a linear combination of column vectors**
- We can **express any vector** $\in \mathbb{R}^3$ as a **linear combination** of **these particular vectors**

- Consider

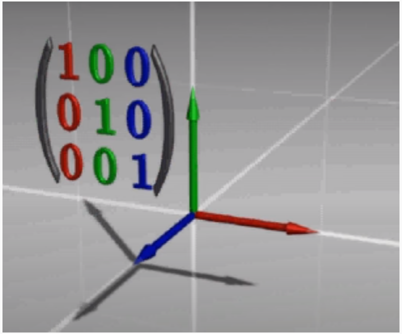
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

- Matrix-vector product **computes a linear combination of column vectors**
- We can **express any vector** $\in \mathbb{R}^3$ as a **linear combination** of **these particular vectors**

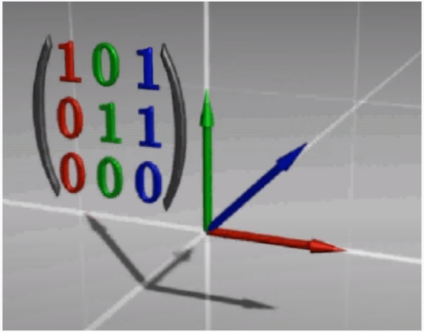
Question ?

Is there a vector $\mathbf{x} \in \mathbb{R}^3$ that **cannot be expressed as**

$$\mathbf{x} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$



Dan Knights, `https://www.youtube.com/watch?v=RkEVuZ0x5mI`



Dan Knights, `https://www.youtube.com/watch?v=RkEVuZ0x5mI`

$$\operatorname{span}\left(\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}\right) = \mathbb{R}^3$$

$$\operatorname{span}\left(\begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}1\\1\\0\end{pmatrix}\right) = U \subset \mathbb{R}^3$$

$$\text{Let } A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

- The **columns of** A are **linearly dependent** ❗

$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = 1 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + 1 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

- **Rank of matrix:** maximum number of linearly independent columns (or rows)
- $\text{rank}(A) = 2$

- For $A \in \mathbb{R}^{M \times N}$

$$\text{rank}(A) < \min(M, N) \iff \det(A) = 0 \iff A^{-1} \text{ does not exist}$$

- For example:

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}}_{\mathbf{u}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \underbrace{\begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}}_{\mathbf{v}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- Both $\mathbf{u}$ and $\mathbf{v}$ ($\mathbf{u} \neq \mathbf{v}$) are **mapped to the zero vector** 😱
- We have $\mathbf{u} \neq \mathbf{v}$, but $A\mathbf{u} = A\mathbf{v}$ and therefore, A **cannot be invertible**

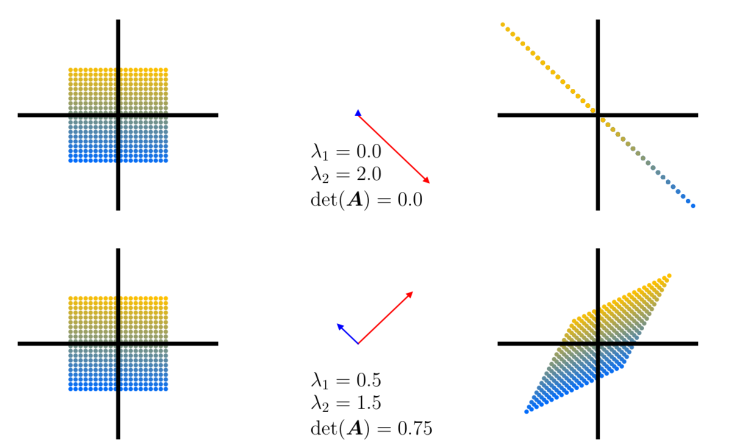
- Given A, find $\mathbf{v} \neq \mathbf{0}$ such that
$$A\mathbf{v} = \lambda\mathbf{v}$$
- A only scales $\mathbf{v}$, **it does not change its direction** !
- $\mathbf{v}$ is called an **eigenvector** of A
- The scaling factor λ is called the corresponding **eigenvalue**

- Given A, find $\mathbf{v} \neq \mathbf{0}$ such that
$$A\mathbf{v} = \lambda\mathbf{v}$$
- A only scales $\mathbf{v}$, **it does not change its direction** ❗
- $\mathbf{v}$ is called an **eigenvector** of A
- The scaling factor λ is called the corresponding **eigenvalue**

Practical Considerations 💻

- How to find eigenvectors/eigenvalues for a given A? 🤔
- For comically small matrices (i.e., smaller than 4×4 😂): Find roots of the characteristic polynomial $\det(A - \lambda I)$
- For all practical problems: **iterative methods**
  - e.g., NumPy: `np.linalg.eig(A)`

Note: $\det(A) = \prod_i \lambda_i$



Mathematics for Machine Learning (Deisenroth et al.)

- Given a **symmetric matrix** $A \in \mathbb{R}^{N \times N}$, we say that A is **positive definite** iff

$$\forall \mathbf{x} \neq \mathbf{0} \in \mathbb{R}^N : \mathbf{x}^T A \mathbf{x} > 0$$

- A weaker condition is **positive semidefiniteness**:

$$\forall \mathbf{x} \in \mathbb{R}^N : \mathbf{x}^T A \mathbf{x} \geq 0$$

Question ?

- Given a matrix $X \in \mathbb{R}^{N \times M}$, show that $X^T X$ is **symmetric and positive semidefinite**
- **Hints:** $(AB)^T = B^T A^T$ and $\|\mathbf{v}\|_2^2 \geq 0$ for all $\mathbf{v}$

Question ?

- Given a matrix $X \in \mathbb{R}^{N \times M}$, show that $X^T X$ is **symmetric and positive semidefinite**
- **Hints:** $(AB)^T = B^T A^T$ and $\|\mathbf{v}\|_2^2 \geq 0$ for all $\mathbf{v}$

- **Symmetry** follows from $(AB)^T = B^T A^T$:

$$\left(X^T X\right)^T = X^T X$$

- $X^T X$ is **postive semidefinite**:

$$\text{To show: } \forall \mathbf{z} \in \mathbb{R}^N : \mathbf{z}^T (X^T X) \mathbf{z} \geq 0$$

- Note that

$$\mathbf{z}^T (X^T X) \mathbf{z} = (\mathbf{z}^T X^T)(X\mathbf{z}) = (X\mathbf{z})^T (X\mathbf{z}) = \|X\mathbf{z}\|_2^2 \geq 0$$

Linear Algebra - Question Time

`fbr.io/ml1p2`

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

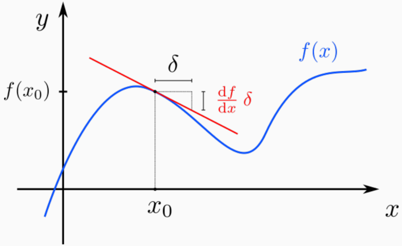
$$\frac{df}{dx}(x) = f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

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)?

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

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

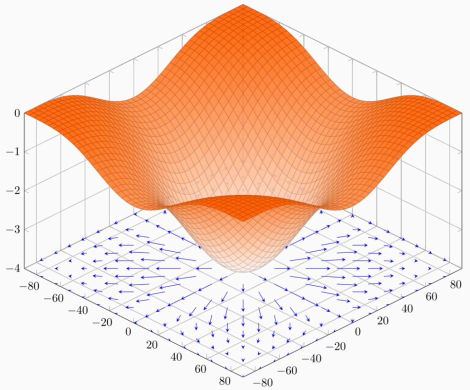


- Let $f(x_1, x_2, \ldots, x_D)$ be a **multi-variate scalar-valued** function ($f : \mathbb{R}^D \to \mathbb{R}$)

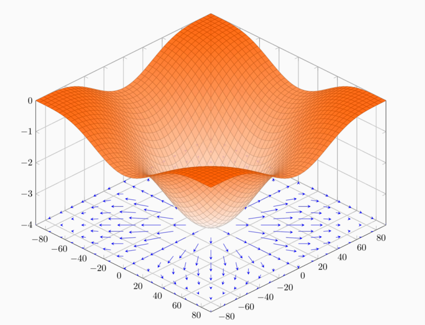
- The **gradient** of f w.r.t. $\mathbf{x} = (x_1, \ldots, x_D)^T$ at some point $\mathbf{x}$ is defined as

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

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



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

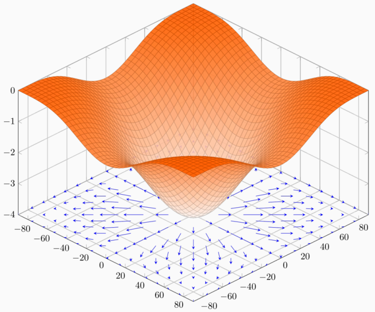


https://commons.wikimedia.org/wiki/File:3d-gradient-cos.svg

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

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

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

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

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

- 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 $k \in \{1, \dots, N\}$:

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

- Thus, $\nabla f(\mathbf{x}) = (2x_1,\ 2x_2,\ \dots, 2x_N)^T = 2\mathbf{x}$
- Answer to question: $\underbrace{\|\alpha \underline{x}\|_2^2}_{f(\underline{x})} = (\alpha \underline{x})^T(\alpha \underline{x}) = \alpha^2 \underline{x}^T\underline{x} = \alpha^2 \|\underline{x}\|_2^2$

$\Rightarrow \nabla_{\underline{x}} f(\underline{x}) = \alpha^2 \cdot 2\underline{x}$

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

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

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

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

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

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 in Practice

- 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”**

Differential Calculus - Question Time

`fbr.io/ml1p2`