Deep Learning

Tutorial 1 – Vector Calculus

Naama Pearl
Autonomous Vision Group
University of Tübingen

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Tübingen AI Center
tuebingen.ai

ellis
European Laboratory for Learning and Intelligent Systems

Outline

- Quiz
- Organization
- Derivatives in 1D
- Partial derivatives and gradients
- Jacobians
- Working problems

Quiz

Tutorials and Exercise Sheets

Tutorials

- Every **Wednesday, 16:15–17:45**, same room as the lecture (after a 30 min break)
- Tutors: Yong Cao, Glenn Angrabeit, Naama Pearl
- Hands-on and interactive:
  - A short **quiz** at the beginning
  - Background you need for the sheets (math, Python, NumPy, …)
  - **Working problems**: you solve, then we go through the solutions together
  - Introducing a new sheet, or discussing the solutions of the last one
- **Ask questions!** In the tutorial, in the Ilias forum, or at the weekly helpdesk

Exercise sheets

- **6 sheets**, one every **two weeks**, on the course website
- Each sheet has **pen & paper** problems and **coding** problems (Python notebooks)
- You solve them **at home**
- **Nothing is handed in.** We publish the solutions and discuss them in the tutorial
- Solving the sheets on your own is good **preparation for the exam**

Date	**Tutorial**	**Sheet**
14.10. (today)	1: Vector calculus	**Sheet 1 out**
21.10.	2: Python, NumPy and coding the sheet	work on sheet 1
28.10.	3	**Solutions 1**, sheet 2 out

Exercise sheet 1

Sheet 1 (PDF)

Exercise	You need	Covered in
Pen & paper		
1. Derivatives	derivatives, Jacobians, chain rule	tutorial 1 (today)
2. Linear regression	linear regression, partial derivatives	lecture 1, tutorial 1 (today)
3. Computation graphs	computation graphs, backpropagation	lecture 2 (next week)
4. Logistic regression	logistic regression, chain rule	lecture 1 or 2
Coding		
1. Vector derivative	NumPy, broadcasting	tutorial 2 (next week)
2. Linear regression	NumPy, the coding framework	lecture 2, tutorial 2 (next week)
3. Logistic regression	NumPy, the coding framework	lecture 2, tutorial 2 (next week)

- Ask in the **forum** or come to the **helpdesk**
- **Solutions** are published and discussed in the tutorial on **28.10.**

Bring your laptop

Fully charged. Tutorial 2 is a coding tutorial

Setup for tutorial 2

- Tutorial 2 is a **coding** tutorial: Python, NumPy and the coding part of sheet 1
- **Bring your laptop** (charged)
- **Before next week**, set up the Python environment for sheet 1:
  1. Install `conda`:
     `https://docs.conda.io/projects/conda/en/latest/user-guide/install/`
  2. In the sheet's `code` folder: `conda env create -f environment.yml`
  3. `conda activate deep_learning_ex_1` and then `jupyter lab`

  If step 2 is stuck at *Solving environment*, try adding `--solver libmamba` (it can still take a few minutes)
- Problems with the setup? Come to the helpdesk or ask in the forum **before** the tutorial

Derivatives in 1D

What is a derivative?

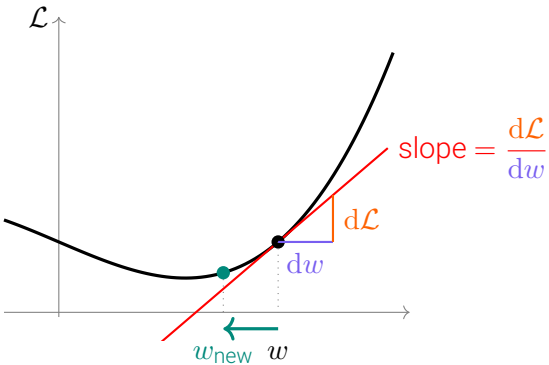
- The **derivative** of $f\colon \mathbb{R} \to \mathbb{R}$ at a point x measures how sensitive the output is to a small change of the input:

$$\frac{\text{change of output}}{\text{change of input}} \quad \longrightarrow \quad f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- In deep learning: how much does the **loss** change if we change a **weight** a little?
  $\mathrm{d}\mathcal{L}/\mathrm{d}w$

The derivative is the slope

We can visualize the derivative as the slope of the function at a point. For the loss $\mathcal{L}$ at the weight w, the slope is $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w}$.



- Here the slope is **positive**: the loss grows with w
- So we push the weight to the **left** (make w smaller) to decrease the loss
- Gradient descent: $w_{\text{new}} = w - \eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w}$
- With a negative slope the step goes to the right

Shape of the derivative

Before computing the derivative:

1. What is the **output** dimension of the function?
2. What is the **input** dimension of the variable we differentiate with respect to?
3. What is the **shape of the derivative**?

Throughout the tutorial: output dimensions in red, input dimensions in blue.

Dimensions in 1D

- $f\colon \mathbb{R}^1 \to \mathbb{R}^1$: scalar input, scalar output
- The derivative is a scalar

$$f\colon \mathbb{R}^1 \to \mathbb{R}^1 \qquad x \in \mathbb{R}^1 \qquad \Rightarrow \qquad f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}(x) \in \mathbb{R}^1$$

Example: derivative from the definition

$$f(x) = x^2, \quad f\colon \mathbb{R}^1 \to \mathbb{R}^1 \quad \Rightarrow \quad f'(x) \in \mathbb{R}^1$$

$$\begin{aligned}
f'(x) &= \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} \\
&= \lim_{h\to 0} \frac{(x+h)^2 - x^2}{h} && \text{plug in } f \\
&= \lim_{h\to 0} \frac{x^2 + 2xh + h^2 - x^2}{h} && \text{expand} \\
&= \lim_{h\to 0} \frac{2xh + h^2}{h} && x^2 - x^2 = 0 \\
&= \lim_{h\to 0} (2x + h) && \text{divide by } h \\
&= 2x && \text{let } h \to 0
\end{aligned}$$

Derivative rules

Elementary functions

$f(x)$	$f'(x)$
c	0
x^n	$n\,x^{n-1}$
e^x	e^x
$\log x$	$\frac{1}{x}$
$\sin x$	$\cos x$
$\cos x$	$-\sin x$

Combinations

Sum: $(u+v)' = u' + v'$

Constant factor: $(c\,u)' = c\,u'$

Product: $(u\,v)' = u'\,v + u\,v'$

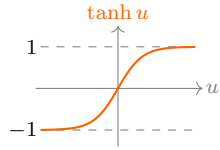
Quotient: $\left(\frac{u}{v}\right)' = \frac{u'\,v - u\,v'}{v^2}$

Chain: $\big(f(g(x))\big)' = f'(g(x))\,g'(x)$

The **chain rule** is the core of how backpropagation computes gradients.

Derivative of a neuron (chain rule)

$$f(w) = \tanh(w\,x + b), \quad x, b \text{ fixed}, \quad f\colon \mathbb{R}^1 \to \mathbb{R}^1 \quad \Rightarrow \quad \frac{\mathrm{d}f}{\mathrm{d}w} \in \mathbb{R}^1$$



$$\begin{aligned} f(w) &= \tanh\big(w\,x + b\big) \\ &= g\big(u(w)\big) \quad \text{with} \quad g(u) = \tanh u, \quad u(w) = w\,x + b \qquad \text{activation of a linear layer} \\ \frac{\mathrm{d}f}{\mathrm{d}w} &= g'\big(u\big)\,u'(w) \qquad \text{chain rule} \\ &= \big(1 - \tanh^2(w\,x + b)\big) \cdot x \qquad \tanh'(u) = 1 - \tanh^2 u,\ u'(w) = x \end{aligned}$$

- In backpropagation, $\mathrm{d}\mathcal{L}/\mathrm{d}f$ comes from the layers above, and we multiply by the **local** derivatives: $\dfrac{\mathrm{d}\mathcal{L}}{\mathrm{d}w} = \dfrac{\mathrm{d}\mathcal{L}}{\mathrm{d}f}\,\dfrac{\mathrm{d}f}{\mathrm{d}u}\,\dfrac{\mathrm{d}u}{\mathrm{d}w}$

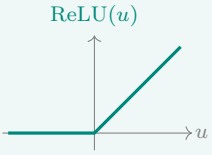
Derivative of a ReLU neuron

Exercise

Same neuron, now with a **ReLU** activation (x, b fixed):

$$f(w) = \mathrm{ReLU}(w\,x + b)$$

$$\mathrm{ReLU}(u) = \max(0, u) = \begin{cases} 0 & u < 0 \\ u & u \geq 0 \end{cases}$$



1. Compute $\dfrac{\mathrm{d}f}{\mathrm{d}w}$.

Reminder (chain rule): $f = g\big(u(w)\big) \;\Rightarrow\; \dfrac{\mathrm{d}f}{\mathrm{d}w} = g'(u)\;u'(w)$

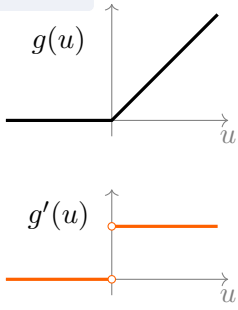
Solution: derivative of a ReLU neuron

$$f(w) = \max(0,\ w\,x + b), \quad x, b \text{ fixed}, \quad f\colon \mathbb{R}^1 \to \mathbb{R}^1 \quad \Rightarrow \quad \frac{\mathrm{d}f}{\mathrm{d}w} \in \mathbb{R}^1$$

$$f(w) = g\big(u(w)\big), \quad g(u) = \max(0, u), \quad u(w) = w\,x + b$$

$$\frac{\mathrm{d}f}{\mathrm{d}w} = g'(u)\,u'(w) = g'(u) \cdot x$$

$$= \begin{cases} 1 \cdot x = x & w\,x + b > 0 \\ 0 \cdot x = 0 & w\,x + b < 0 \end{cases}$$



- Same chain rule as for $\tanh$, only g' changes
- At $w\,x + b = 0$: not defined
  (in practice, frameworks use 0)

Partial Derivatives and Gradients

Partial derivatives of a neuron

$$f(w,b) = \tanh(w\,x + b), \quad x \text{ fixed}, \qquad f\colon \mathbb{R}^2 \to \mathbb{R}^1 \qquad \Rightarrow \qquad \nabla_{(w,b)}\, f \in \mathbb{R}^2$$

$$\frac{\partial f}{\partial w} = \frac{\partial}{\partial w}\tanh(w\,x + b) = \big(1 - \tanh^2(w\,x + b)\big) \cdot x$$ as before: b is a constant

$$\frac{\partial f}{\partial b} = \frac{\partial}{\partial b}\tanh(w\,x + b) = \big(1 - \tanh^2(w\,x + b)\big) \cdot 1$$ now w is a constant

$$\nabla_{(w,b)}\, f = \begin{pmatrix} \frac{\partial f}{\partial w} \\ \frac{\partial f}{\partial b} \end{pmatrix} = \big(1 - \tanh^2(w\,x + b)\big) \begin{pmatrix} x \\ 1 \end{pmatrix} \in \mathbb{R}^2$$ stack them in a column

- When we computed the derivative w.r.t. the weight w before, we kept b fixed. This was a **partial derivative** (written ∂ instead of d)
- The **gradient** has the shape of the parameters (w, b)

Partial derivatives and the gradient

- Let $f\colon \mathbb{R}^n \to \mathbb{R}^1$ depend on several inputs $x_1, \ldots, x_n$
- The **partial derivative** $\frac{\partial f}{\partial x_j}$: differentiate w.r.t. x_j and treat all other variables as constants
- The **gradient** collects all n partial derivatives in a column, in the **shape of** $\mathbf{x}$:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{x}) \end{pmatrix} \in \mathbb{R}^n$$

$f\colon \mathbb{R}^n \to \mathbb{R}^1 \qquad \mathbf{x} \in \mathbb{R}^n \qquad \Rightarrow \qquad \nabla_{\mathbf{x}} f \in \mathbb{R}^n$ (same shape as $\mathbf{x}$)

- In deep learning: the gradient of the loss w.r.t. the weights has the shape of the weights, so we can use it to update them: $\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla_{\mathbf{w}} \mathcal{L}$

Partial derivatives and gradient

Exercise

A neuron with two inputs ($\mathbf{x}$ fixed):

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \qquad f(\mathbf{w}) = \tanh(\mathbf{w}^\top \mathbf{x})$$

1. Shapes: $f\colon \mathbb{R}^{?} \to \mathbb{R}^{?}$, $\nabla_{\mathbf{w}} f \in ?$
2. Compute $\frac{\partial f}{\partial w_1}$, $\frac{\partial f}{\partial w_2}$ and $\nabla_{\mathbf{w}} f$.

Reminder: $\tanh'(u) = 1 - \tanh^2 u$

Solution: $f(\mathbf{w}) = \tanh(\mathbf{w}^\top \mathbf{x})$

$\mathbf{w}, \mathbf{x} \in \mathbb{R}^2, \quad f : \mathbb{R}^? \to \mathbb{R}^? \quad \Rightarrow \quad \nabla_{\mathbf{w}} f \in ?$ What are the shapes?

$$f(\mathbf{w}) = \tanh\left(w_1 x_1 + w_2 x_2\right)$$ $\mathbf{w}^\top \mathbf{x}$ in components

$$\frac{\partial f}{\partial w_1} = \left(1 - \tanh^2(\mathbf{w}^\top \mathbf{x})\right) \cdot x_1$$ chain rule, w_2 constant

$$\frac{\partial f}{\partial w_2} = \left(1 - \tanh^2(\mathbf{w}^\top \mathbf{x})\right) \cdot x_2$$ chain rule, w_1 constant

$$\nabla_{\mathbf{w}} f = \left(1 - \tanh^2(\mathbf{w}^\top \mathbf{x})\right) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \left(1 - \tanh^2(\mathbf{w}^\top \mathbf{x})\right) \mathbf{x} \in \mathbb{R}^2$$ stack in a column, shape of $\mathbf{w}$ ✓

The gradient w.r.t. the weights is the input $\mathbf{x}$, times a scalar.

Gradients of Vector Expressions

How to differentiate vector expressions

For $\tanh(\mathbf{w}^\top \mathbf{x})$ we wrote $\mathbf{w}^\top \mathbf{x}$ in components. This works for any scalar expression of a vector, like $\mathbf{w}^\top \mathbf{x}$, $\mathbf{w}^\top \mathbf{w}$, $\mathbf{x}^\top \mathbf{A}\mathbf{x}$, …

1. **Shapes:** f is a scalar, $\mathbf{x} \in \mathbb{R}^n \Rightarrow \nabla_{\mathbf{x}} f \in \mathbb{R}^n$ (shape of $\mathbf{x}$)
2. **Components:** write f as a sum of scalars, e.g. $\mathbf{w}^\top \mathbf{x} = \sum_i w_i x_i$
3. **Scalar derivative:** compute $\frac{\partial f}{\partial x_j}$ with the 1D rules (everything else is a constant)
4. **Reassemble:** put the n entries into a column and check the shape

Sanity check: for $n = 1$ the result must reduce to the 1D derivative.

Example: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$

$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}, \quad \mathbf{w}, \mathbf{x} \in \mathbb{R}^n, \quad f : \mathbb{R}^? \to \mathbb{R}^? \quad \Rightarrow \quad \nabla_{\mathbf{x}} f \in ?$ What are the shapes?

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i \, x_i$$ components

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{n} w_i \, \frac{\partial x_i}{\partial x_j}$$ scalar derivative w.r.t. one x_j

$$= \sum_{i=1}^{n} w_i \cdot \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} = w_j$$ only the term $i = j$ survives

$$\nabla_{\mathbf{x}} f = \begin{pmatrix} w_1 & \cdots & w_n \end{pmatrix}^\top = \mathbf{w} \in \mathbb{R}^n$$ reassemble as a column, shape ✓

Sanity check $n = 1$: $f(x) = wx \Rightarrow f'(x) = w$ ✓

Gradients of $\mathbf{w}^\top \mathbf{x}$ and $\mathbf{w}^\top \mathbf{w}$

Exercise

Let $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$.

1. The same expression as a function of $\mathbf{w}$, $f(\mathbf{w}) = \mathbf{w}^\top \mathbf{x}$:
   compute $\nabla_{\mathbf{w}} f$
2. $g(\mathbf{w}) = \mathbf{w}^\top \mathbf{w} = \|\mathbf{w}\|_2^2$:
   compute $\nabla_{\mathbf{w}} g$

Previous example: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i \, x_i$$

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{n} w_i \frac{\partial x_i}{\partial x_j} = \sum_{i=1}^{n} w_i \cdot \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} = w_j$$

$$\nabla_{\mathbf{x}} f = \begin{pmatrix} w_1 & \cdots & w_n \end{pmatrix}^\top = \mathbf{w}$$

Solution: gradients of $\mathbf{w}^\top \mathbf{x}$ and $\mathbf{w}^\top \mathbf{w}$

$\mathbf{w}, \mathbf{x} \in \mathbb{R}^n, \quad f, g \in \mathbb{R}^? \quad \Rightarrow \quad \nabla_{\mathbf{w}} f,\ \nabla_{\mathbf{w}}\, g \in ?$ What are the shapes?

$$f = \sum_{i=1}^{n} w_i\, x_i$$ components

$$\frac{\partial f}{\partial w_j} = \sum_{i=1}^{n} \frac{\partial w_i}{\partial w_j}\, x_i = \sum_{i=1}^{n} \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \cdot x_i = x_j$$ now $\mathbf{x}$ is the constant

$$\nabla_{\mathbf{w}} f = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}^\top = \mathbf{x}$$ (before: $\nabla_{\mathbf{x}} f = \mathbf{w}$)

$$g = \sum_{i=1}^{n} w_i\, w_i = \sum_{i=1}^{n} w_i^2$$ components

$$\frac{\partial g}{\partial w_j} = \sum_{i=1}^{n} 2w_i \frac{\partial w_i}{\partial w_j} = \sum_{i=1}^{n} 2w_i \cdot \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} = 2w_j$$ chain rule for w_i^2, only the term $i = j$ survives

$$\nabla_{\mathbf{w}}\, g = \begin{pmatrix} 2w_1 & \cdots & 2w_n \end{pmatrix}^\top = 2\mathbf{w}$$

Jacobians and the Chain Rule in Matrix Form

A function with several outputs

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 x_2 \\ x_1 + x_2^2 \\ e^{x_1} \end{pmatrix}, \quad \mathbf{f}\colon \mathbb{R}^{?} \to \mathbb{R}^{?} \quad \Rightarrow \quad \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \in \, ?$$

What are the shapes?

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} \end{pmatrix} = \begin{pmatrix} x_2 & x_1 \\ 1 & 2x_2 \\ e^{x_1} & 0 \end{pmatrix}$$

- Each output gets one row with its partial derivatives: row 1 for $f_1 = x_1 x_2$, row 2 for f_2, row 3 for f_3
- Stacking the rows gives a 3×2 matrix (3 outputs, 2 inputs): the **Jacobian**

Derivative of a vector-to-vector function: the Jacobian

In general, let $\mathbf{f}\colon \mathbb{R}^n \to \mathbb{R}^m$, i.e., n inputs and m outputs $\mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\big)^\top$.

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}) = \mathbf{J}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

- Row i belongs to output f_i: all partial derivatives of f_i $\quad \Rightarrow m$ rows
- Column j belongs to input x_j: how all outputs react to x_j $\quad \Rightarrow n$ columns
- Entry (i, j) is an ordinary scalar derivative: $\frac{\partial f_i}{\partial x_j}$

Gradient and Jacobian

Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$

- "If I change the input a little, how much does the output change?"
- $\mathrm{d}\mathbf{f} \approx \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathrm{d}\mathbf{x}$
- Shape: #outputs × #inputs (for a scalar f: a row)
- Used in the **chain rule**

Gradient $\nabla_{\mathbf{x}} f$ (scalar f only)

- All partial derivatives $\frac{\partial f}{\partial x_j}$, arranged in the **shape of** $\mathbf{x}$
- So it is easy to update $\mathbf{x}$: $\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla_{\mathbf{x}} f$
- Used in **gradient descent**

For a scalar f, they hold the same numbers, so $\nabla_{\mathbf{x}} f = \left(\frac{\partial f}{\partial \mathbf{x}}\right)^{\top}$.

Example: $f(\mathbf{x}) = \mathbf{w}^{\top}\mathbf{x}$: $\frac{\partial f}{\partial \mathbf{x}} = \mathbf{w}^{\top}$ (row), $\nabla_{\mathbf{x}} f = \mathbf{w}$ (column)

Shapes of derivatives

function	w.r.t.	derivative	name
$f \in \mathbb{R}^1$	$x \in \mathbb{R}^1$	$\frac{\partial f}{\partial x} \in \mathbb{R}^1$	derivative
$f \in \mathbb{R}^1$	$\mathbf{x} \in \mathbb{R}^n$	$\nabla_{\mathbf{x}} f \in \mathbb{R}^n$	gradient (shape of $\mathbf{x}$)
$\mathcal{L} \in \mathbb{R}^1$	$\mathbf{W} \in \mathbb{R}^{m \times n}$	$\nabla_{\mathbf{W}} \mathcal{L} \in \mathbb{R}^{m \times n}$	gradient (shape of $\mathbf{W}$)
$\mathbf{f} \in \mathbb{R}^m$	$\mathbf{x} \in \mathbb{R}^n$	$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \in \mathbb{R}^{m \times n}$	Jacobian

- There is **one number for every pair (output, input)**

Shape and first row of a Jacobian

Exercise

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 x_2 x_3 \\ x_1 + x_3 \end{pmatrix}$$

1. What is the shape of $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$?
2. Compute its first row.

Solution: Jacobian of $\mathbf{f}$

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 x_2 x_3 \\ x_1 + x_3 \end{pmatrix}, \quad \mathbf{f}: \mathbb{R}^{?} \to \mathbb{R}^{?} \quad \Rightarrow \quad \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \in \mathbf{?}$$

What is the shape?

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \end{pmatrix} = \begin{pmatrix} x_2 x_3 & x_1 x_3 & x_1 x_2 \\ 1 & 0 & 1 \end{pmatrix}$$

- 2 outputs (rows), 3 inputs (columns) $\Rightarrow 2 \times 3$
- Row 1: partial derivatives of $f_1 = x_1 x_2 x_3$ (the other two variables are constants)
- Row 2: partial derivatives of $f_2 = x_1 + x_3$ (it does not depend on x_2)

Example: least squares with the chain rule

$$\mathcal{L}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2, \quad \mathbf{X} \in \mathbb{R}^{N \times n}, \ \mathbf{w} \in \mathbb{R}^n, \ \mathbf{y} \in \mathbb{R}^N, \quad \mathcal{L}\colon \mathbb{R}^{?} \to \mathbb{R}^{?} \quad \Rightarrow \quad \nabla_{\mathbf{w}} \mathcal{L} \in ?$$

What are the shapes?

$$\mathbf{u} = \mathbf{X}\mathbf{w} - \mathbf{y} \in \mathbb{R}^N, \qquad \mathcal{L} = \mathbf{u}^\top \mathbf{u}$$

inner (vector), outer (scalar)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{w}} = \underbrace{2\mathbf{u}^\top}_{1 \times N} \underbrace{\mathbf{X}}_{N \times n} \in \mathbb{R}^{1 \times n}$$

chain rule (details on the next slide)

$$\nabla_{\mathbf{w}} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{w}}\right)^\top = \mathbf{X}^\top 2(\mathbf{X}\mathbf{w} - \mathbf{y}) \in \mathbb{R}^n$$

the gradient is the row as a column

Same as in lecture 1: $\nabla_{\mathbf{w}} \mathcal{L} = 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y}$ ✓

Least squares: details

$$\mathcal{L}(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2, \quad \mathbf{X} \in \mathbb{R}^{N \times n}, \ \mathbf{w} \in \mathbb{R}^n, \ \mathbf{y} \in \mathbb{R}^N, \quad \mathcal{L}\colon \mathbb{R}^n \to \mathbb{R}^1 \quad \Rightarrow \quad \nabla_{\mathbf{w}} \mathcal{L} \in \mathbb{R}^n$$

$$\mathbf{u} = \mathbf{X}\mathbf{w} - \mathbf{y} \in \mathbb{R}^N, \qquad \mathcal{L} = \mathbf{u}^\top \mathbf{u}$$

inner (vector), outer (scalar)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = 2\mathbf{u}^\top \in \mathbb{R}^{1 \times N}$$

as for $\mathbf{w}^\top \mathbf{w}$, written as a row (1 output, N inputs)

$$\frac{\partial \mathbf{u}}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \begin{pmatrix} \mathbf{x}_1^\top \mathbf{w} - y_1 \\ \vdots \\ \mathbf{x}_N^\top \mathbf{w} - y_N \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_N^\top \end{pmatrix} = \mathbf{X} \in \mathbb{R}^{N \times n}$$

row i: one sample, as for $\mathbf{w}^\top \mathbf{x}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{w}} = \underbrace{2\mathbf{u}^\top}_{1 \times N} \underbrace{\mathbf{X}}_{N \times n} \in \mathbb{R}^{1 \times n}$$

chain rule: multiply, the inner N matches

$$\nabla_{\mathbf{w}} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{w}} \right)^\top = \mathbf{X}^\top 2(\mathbf{X}\mathbf{w} - \mathbf{y}) \in \mathbb{R}^n$$

the gradient is this row as a column (shape of $\mathbf{w}$)

Chain rule for vector-to-vector functions

In general, let $\mathbf{f}\colon \mathbb{R}^n \to \mathbb{R}^m$, $\mathbf{g}\colon \mathbb{R}^m \to \mathbb{R}^p$ and $\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{f}(\mathbf{x}))$, so $\mathbf{h}\colon \mathbb{R}^n \to \mathbb{R}^p$. Then

$$\underbrace{\frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\mathbf{x})}_{p\times n} = \underbrace{\frac{\partial \mathbf{g}}{\partial \mathbf{y}}(\mathbf{f}(\mathbf{x}))}_{p\times m} \underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})}_{m\times n}$$

- The 1D chain rule, but now a **matrix product**
- The inner dimension m (output of $\mathbf{f}$ = input of $\mathbf{g}$) always matches
- The result has the output dimension of $\mathbf{g}$ and the input dimension of $\mathbf{f}$
- Scalar loss ($p = 1$) with gradients, as for least squares: $\underbrace{\nabla_{\mathbf{x}} h}_{n\times 1} = \underbrace{\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^\top}_{n\times m} \underbrace{\nabla_{\mathbf{y}} g}_{m\times 1}$, i.e.

  $\frac{\partial h}{\partial x_j} = \sum_{i=1}^m \frac{\partial g}{\partial y_i}\frac{\partial f_i}{\partial x_j}$ (multivariate chain rule, lecture 2)

Bring your laptop

Fully charged. Tutorial 2 is a coding tutorial

Summary

- Derivative = how much the output changes for a small change of the input. Its sign tells gradient descent which way to move
- Partial derivative: differentiate w.r.t. one variable, all others are constants
- Scalar function: the gradient $\nabla_{\mathbf{x}} f \in \mathbb{R}^n$ is in the shape of $\mathbf{x}$
- Differentiating vector expressions: shapes $\rightarrow$ components $\rightarrow$ scalar derivative $\rightarrow$ reassemble
- Shape of a Jacobian:

$$\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m \qquad \Rightarrow \qquad \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \in \mathbb{R}^{m \times n}$$ (rows = outputs, columns = inputs)

- Chain rule = product of Jacobians, the inner dimensions match: $(p \times n) = (p \times m)(m \times n)$. Scalar output: $\nabla_{\mathbf{x}} h = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^\top \nabla_{\mathbf{y}} g$