Slides: Marc Deisenroth (Imperial College London/Prowler.io), 2017

Mathematics for Machine Learning

Avishkar Bhoopchand, *DeepMind* avishkar@google.com

Cynthia Mulenga, *Mwabu* cynthiammulenga@gmail.com

Daniela Massiceti, *Univ Oxford* daniela@robots.ox.ac.uk

Kendi Muchungi, *Africa Nazarenne Univ* kmuchungi@anu.ac.ke

DEEP LEARNING INDABA

Stellenbosch University

9th September 2018

Before we begin...

Introduce yourselves to your neighbours!

(you'll be working with them throughout this lecture)

Before we begin...

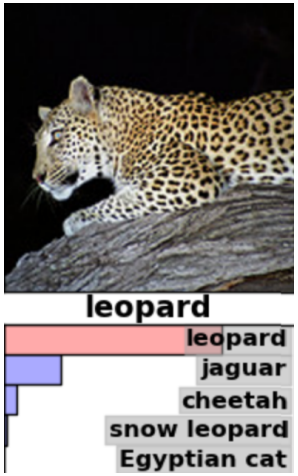
Colab notebook with all examples in this lecture:
https://colab.research.google.com/github/deep-learning-indaba/indaba-2018/blob/master/Mathematics_for_Machine_Learning_Examples.ipynb

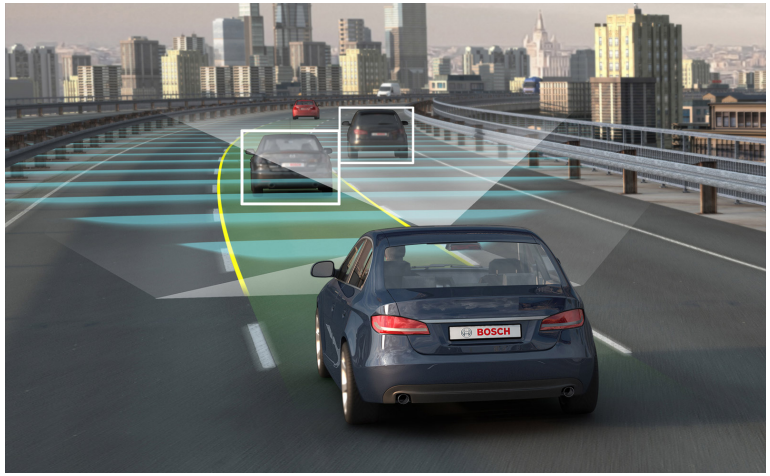
What is Machine Learning?

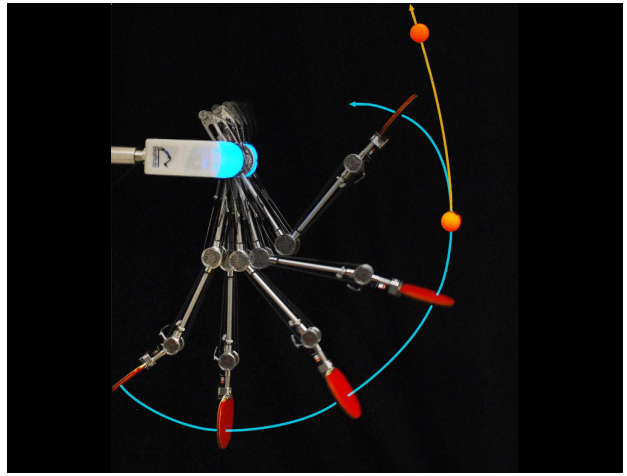
Improving a machine's performance at some task with experience

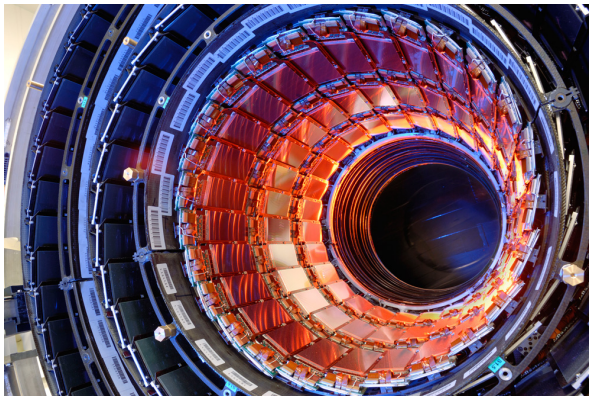
- Define task **T** - classification, regression
- Establish performance measure **P** - top-5/top-1 classification error rate, Mean Square Error (MSE)
- Provide some experience **E** - training examples

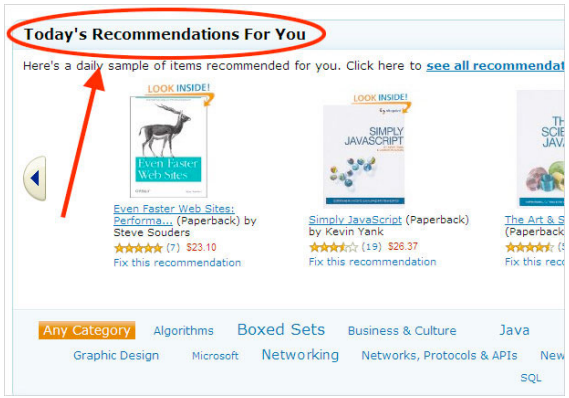
Applications of Machine Learning

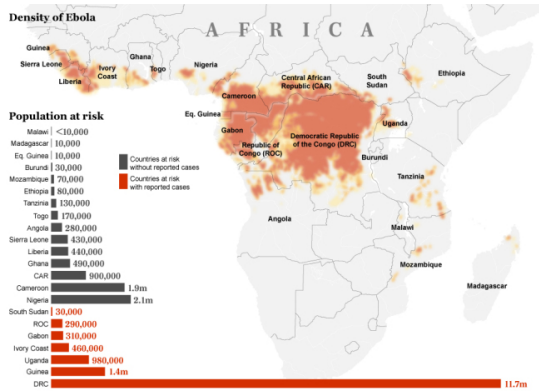




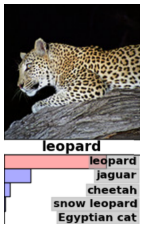


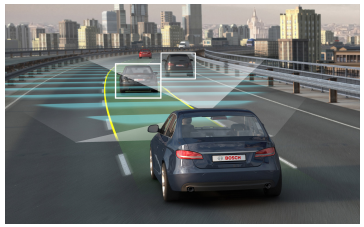
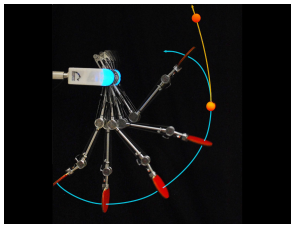
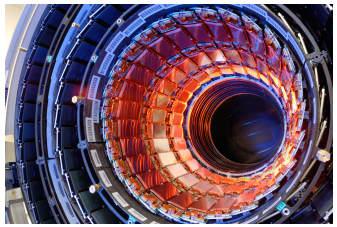
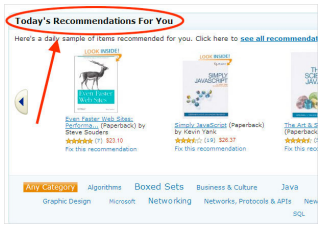


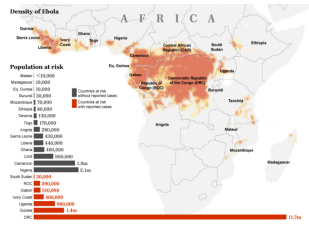




Mathematical Concepts in Machine Learning

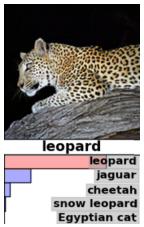


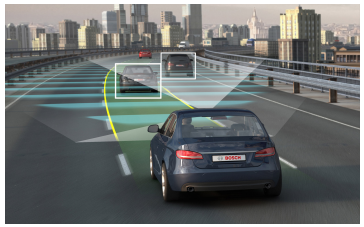
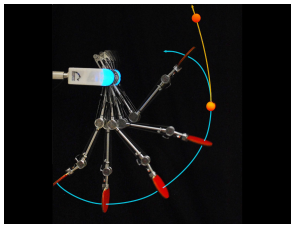
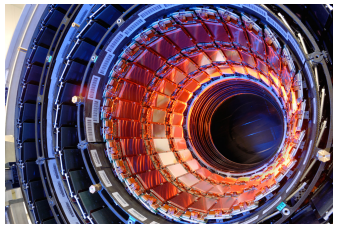
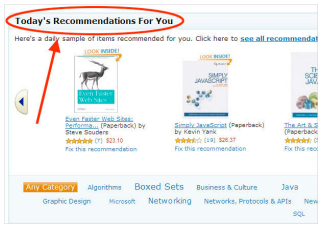





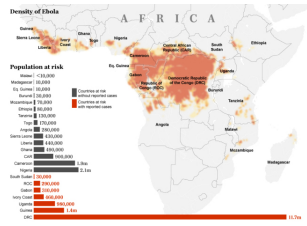


- Probability theory and Bayesian inference
- Linear algebra and matrix decomposition
- Differentiation
- Optimisation
- Integration
- Functional analysis

Mathematical Concepts in Machine Learning





- **Probability theory** and Bayesian inference
- Linear algebra and matrix decomposition
- **Differentiation**
- Optimisation
- Integration
- Functional analysis

Outline

Key Concepts in Probability
- Probability density function
- Sum & Product Rules
- Bayes' Theorem

Differentiation
- Where it is used in ML?
- Types of differentiation
- Gradients in Neural Networks

Overview

Key Concepts in Probability

- Probability density function
- Sum & Product Rules
- Bayes' Theorem

Differentiation

- Where it is used in ML?
- Types of differentiation
- Gradients in Neural Networks

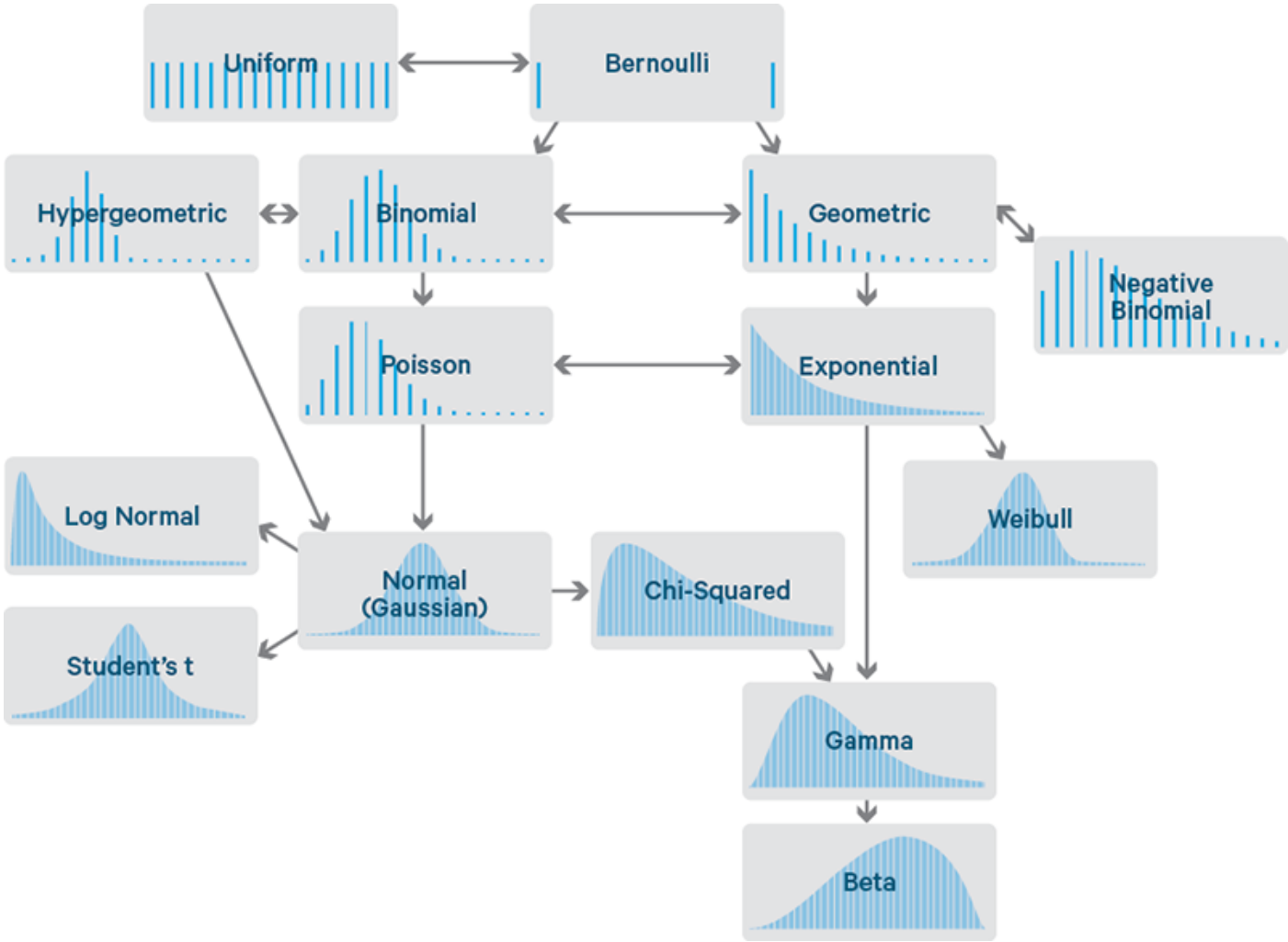
Probability density functions

A function $p : \mathbb{R}^D \to \mathbb{R}$ is called a probability density function if

1. Its integral exists,
2. $\forall \boldsymbol{x} \in \mathbb{R}^D : p(\boldsymbol{x}) \geqslant 0$ and
3. $\int_{\mathbb{R}^D} p(\boldsymbol{x}) d\boldsymbol{x} = 1$ *

* Here, $\boldsymbol{x} \in \mathbb{R}^D$ is a (continuous) random variable. For discrete random variables, the integral is replaced with a sum.

Probability density functions



\blue{http://www.math.wm.edu/~leemis/chart/UDR/UDR.html}

Sum & Product Rules

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \qquad \text{Sum rule/Marginalization property}$$

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}) \qquad \text{Product rule}$$

- $p(\boldsymbol{x}, \boldsymbol{y})$ is the joint probability distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$
- $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$ are the marginal probability distributions of $\boldsymbol{x}$ and $\boldsymbol{y}$
- $p(\boldsymbol{y}|\boldsymbol{x})$ is the conditional probability distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$

Sum & Product Rules

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \qquad \text{Sum rule/Marginalization property}$$

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}) \qquad \text{Product rule}$$

- $p(\boldsymbol{x}, \boldsymbol{y})$ is the joint probability distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$
- $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$ are the marginal probability distributions of $\boldsymbol{x}$ and $\boldsymbol{y}$
- $p(\boldsymbol{y}|\boldsymbol{x})$ is the conditional probability distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$

Sum & Product Rules

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \qquad \text{Sum rule/Marginalization property}$$

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}) \qquad \text{Product rule}$$

- $p(\boldsymbol{x}, \boldsymbol{y})$ is the joint probability distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$
- $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$ are the marginal probability distributions of $\boldsymbol{x}$ and $\boldsymbol{y}$
- $p(\boldsymbol{y}|\boldsymbol{x})$ is the conditional probability distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$

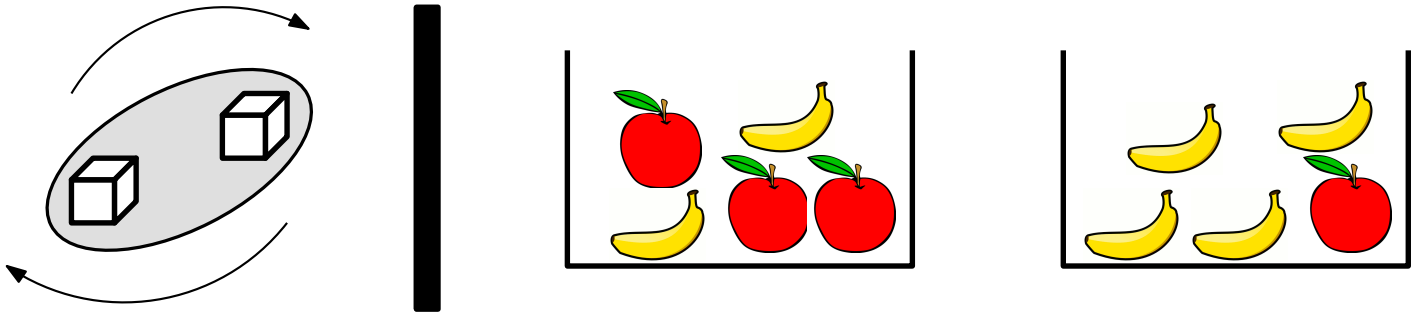
Sum & Product Rules

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} \qquad \text{Sum rule/Marginalization property}$$

$$p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}) \qquad \text{Product rule}$$

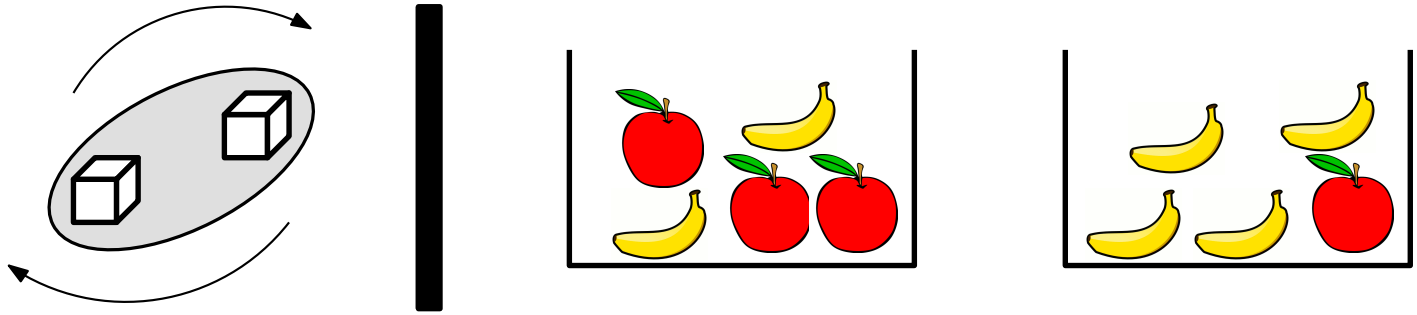
- $p(\boldsymbol{x}, \boldsymbol{y})$ is the joint probability distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$
- $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$ are the marginal probability distributions of $\boldsymbol{x}$ and $\boldsymbol{y}$
- $p(\boldsymbol{y}|\boldsymbol{x})$ is the conditional probability distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$

Illustration: Bayesian Inference



Two cartons $\boldsymbol{x} = \{c_1, c_2\}$ with fruit in each $\boldsymbol{y} = \{\text{banana}, \text{apple}\}$
Say we pick a banana ($\boldsymbol{y} = \text{banana}$)
We want to estimate $p(\boldsymbol{x} = c_1 | \boldsymbol{y} = \text{banana})$

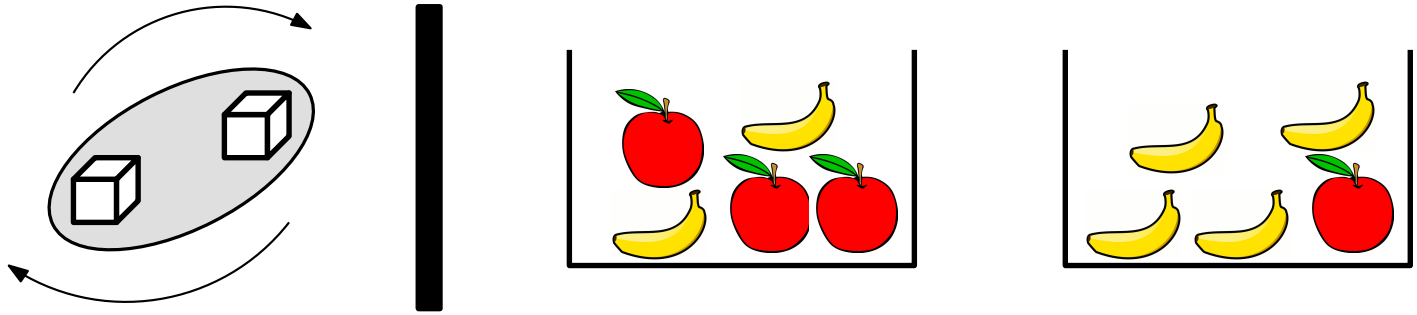
Illustration: Bayesian Inference



$$p(c_1) = 1/2 = p(c_2)$$

Prior

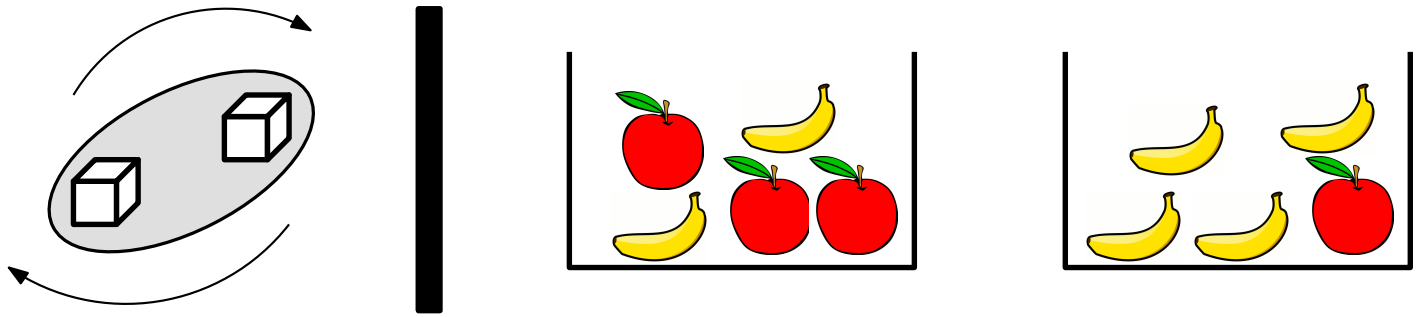
Illustration: Bayesian Inference



$p(c_1) = 1/2 = p(c_2)$ **Prior**

$p(\text{banana}|c_1) = 2/5\,, \quad p(\text{banana}|c_2) = 4/5$ **Likelihood**

Illustration: Bayesian Inference



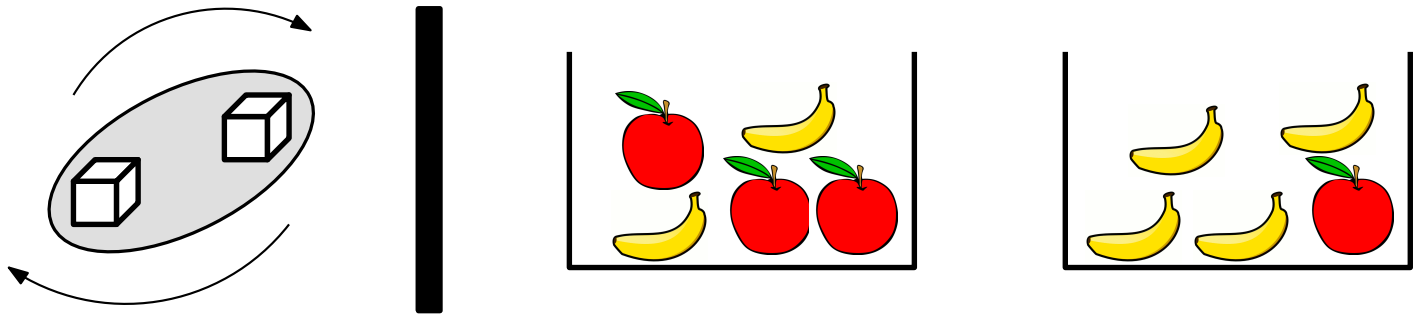
$p(c_1) = 1/2 = p(c_2)$ **Prior**

$p(\text{banana}|c_1) = 2/5\,, \quad p(\text{banana}|c_2) = 4/5$ **Likelihood**

$p(\text{banana}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) =$ **Evidence**

$p(\text{banana}|c_1)p(c_1) + p(\text{banana}|c_2)p(c_2) = 6/10 = 3/5$

Illustration: Bayesian Inference



$p(c_1) = 1/2 = p(c_2)$ **Prior**

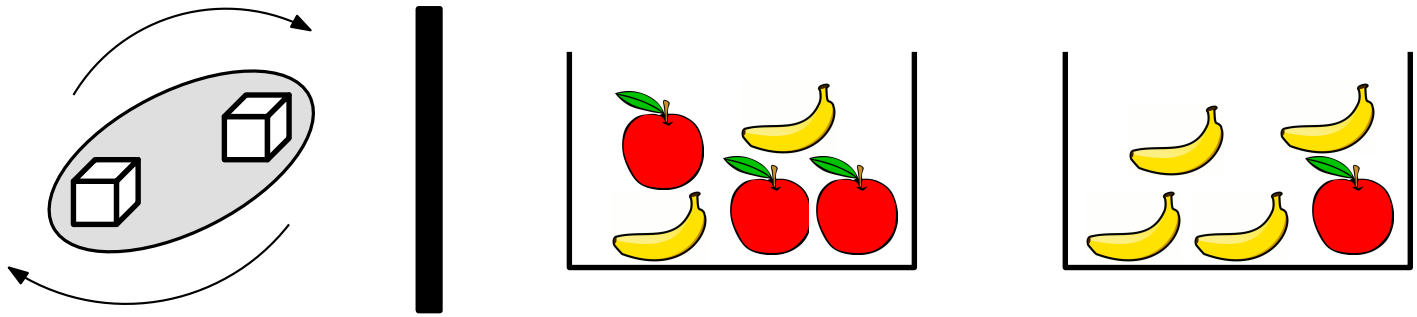
$p(\text{banana}|c_1) = 2/5\,, \quad p(\text{banana}|c_2) = 4/5$ **Likelihood**

$p(\text{banana}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) =$ **Evidence**

$p(\text{banana}|c_1)p(c_1) + p(\text{banana}|c_2)p(c_2) = 6/10 = 3/5$

What is $p(\boldsymbol{x} = c_1|\boldsymbol{y} = \text{banana})$? **Posterior**

Illustration: Bayesian Inference



$p(c_1) = 1/2 = p(c_2)$ **Prior**

$p(\text{banana}|c_1) = 2/5\,, \quad p(\text{banana}|c_2) = 4/5$ **Likelihood**

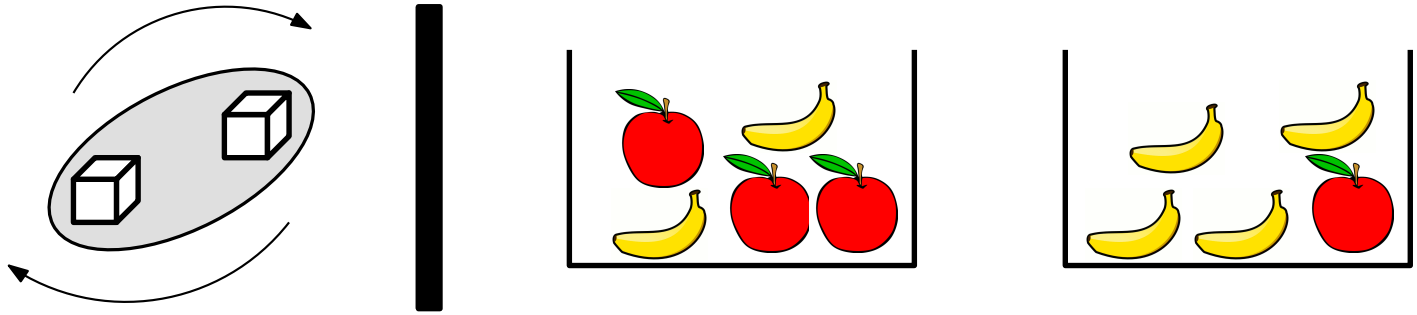
$p(\text{banana}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) =$ **Evidence**

$p(\text{banana}|c_1)p(c_1) + p(\text{banana}|c_2)p(c_2) = 6/10 = 3/5$

What is $p(\boldsymbol{x} = c_1|\boldsymbol{y} = \text{banana})$? **Posterior**

Bayes' Theorem: $\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$

Illustration: Bayesian Inference



$p(c_1) = 1/2 = p(c_2)$ **Prior**

$p(\text{banana}|c_1) = 2/5\,, \quad p(\text{banana}|c_2) = 4/5$ **Likelihood**

$p(\text{banana}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) =$ **Evidence**

$p(\text{banana}|c_1)p(c_1) + p(\text{banana}|c_2)p(c_2) = 6/10 = 3/5$

What is $p(\boldsymbol{x} = c_1|\boldsymbol{y} = \text{banana})$? **Posterior**

Bayes' Theorem: $\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence}}$

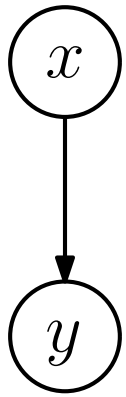
$$p(\boldsymbol{x} = c_1|\boldsymbol{y} = \text{banana}) = \frac{p(\text{banana}|c_1)p(c_1)}{p(\text{banana})} = \frac{2/5 * 1/2}{3/5} = 1/3$$

Bayes' Theorem

Bayes' Theorem (Probabilistic Inverse)

$$p(\boldsymbol{x}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x})\, p(\boldsymbol{x})}{p(\boldsymbol{y})}, \qquad \boldsymbol{x} : \text{hypothesis}, \quad \boldsymbol{y} : \text{measurement}$$

- Posterior belief
- Prior belief
- Likelihood (measurement model)
- Marginal likelihood (normalisation constant)



Overview

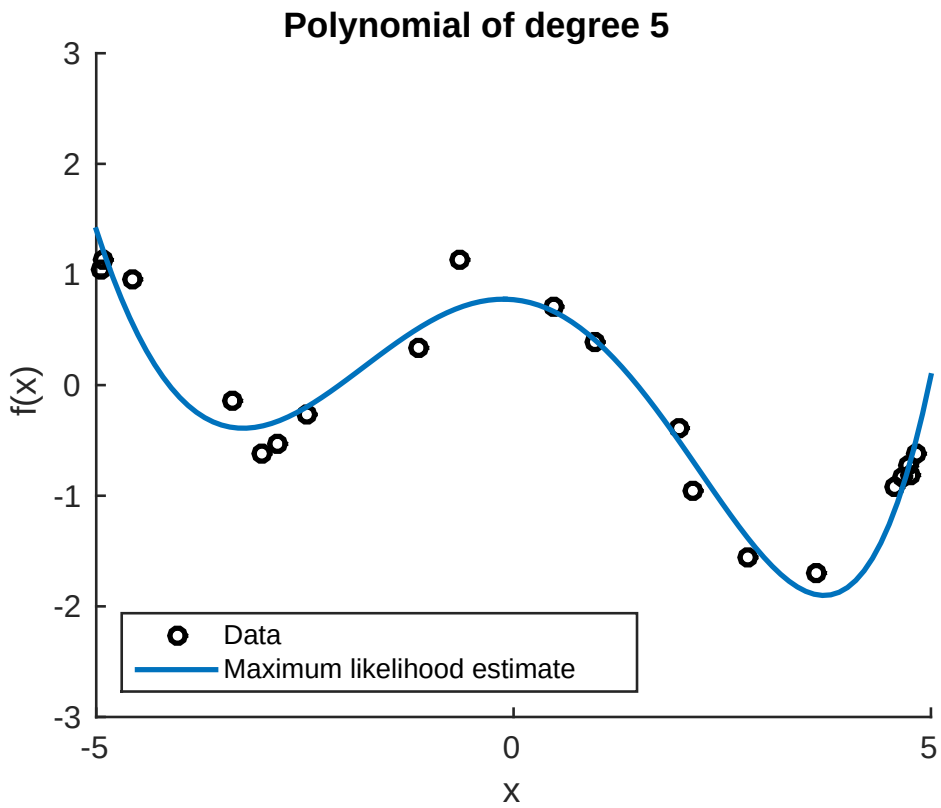
Key Concepts in Probability

- Probability density function
- Sum & Product Rules
- Bayes' Theorem

Differentiation

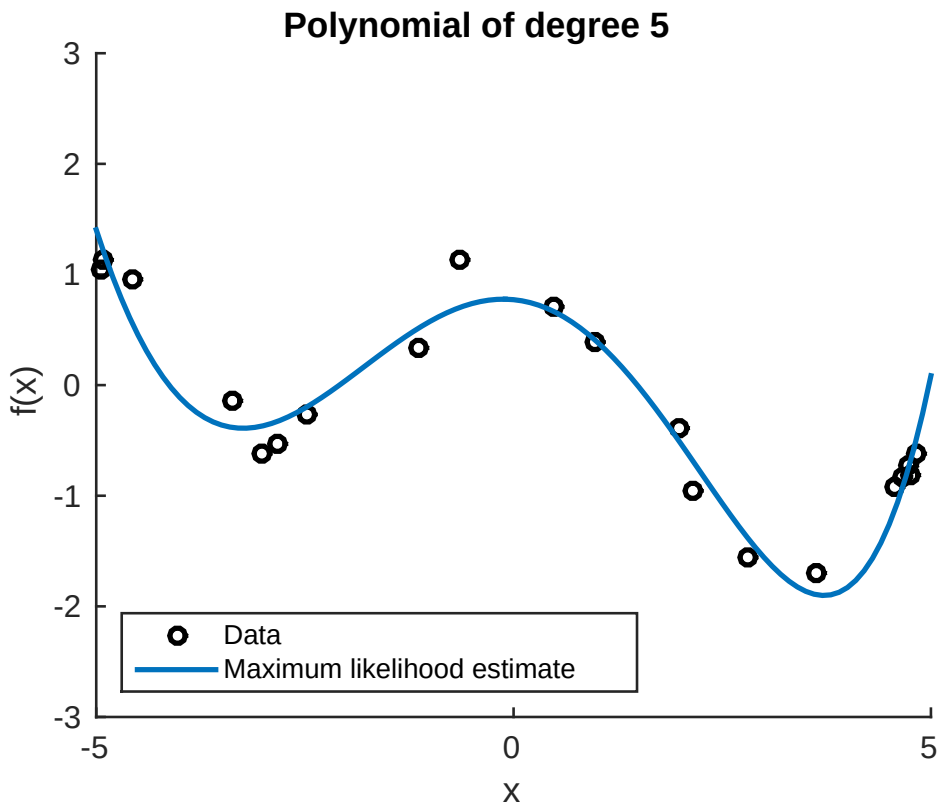
- Where it is used in ML?
- Types of differentiation
- Gradients in Neural Networks

Scenario 1: Find optimal parameters to fit a 1D curve



- **Given**: N data pairs (x_i, y_i) (black dots) where x_i is input & y_i is output/target

Scenario 1: Find optimal parameters to fit a 1D curve



- **Given**: N data pairs (x_i, y_i) (black dots) where x_i is input & y_i is output/target
- **Goal**: Learn a model f (blue curve) that best fits the data

$$f(x_i) \approx y_i$$

Scenario 1: Find optimal parameters to fit a 1D curve

- Model f is D-degree polynomial parametrised by $\boldsymbol{\theta}$, or $f_{\boldsymbol{\theta}}$

$$f_{\boldsymbol{\theta}}(x) = \theta_0 x^0 + \theta_1 x^1 + \theta_2 x^2 + \cdots + \theta_{D-1} x^{D-1} + \theta_D x^D$$

Scenario 1: Find optimal parameters to fit a 1D curve

- Model f is D-degree polynomial parametrised by $\boldsymbol{\theta}$, or $f_{\boldsymbol{\theta}}$

$$f_{\boldsymbol{\theta}}(x) = \theta_0 x^0 + \theta_1 x^1 + \theta_2 x^2 + \cdots + \theta_{D-1} x^{D-1} + \theta_D x^D$$

$$= \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \cdots & \theta_{D-1} & \theta_D \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ \cdots \\ x^{D-1} \\ x^D \end{bmatrix}$$

Scenario 1: Find optimal parameters to fit a 1D curve

- Model f is D-degree polynomial parametrised by $\boldsymbol{\theta}$, or $f_{\boldsymbol{\theta}}$

$$f_{\boldsymbol{\theta}}(x) = \theta_0 x^0 + \theta_1 x^1 + \theta_2 x^2 + \cdots + \theta_{D-1} x^{D-1} + \theta_D x^D$$

$$= \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \cdots & \theta_{D-1} & \theta_D \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ \cdots \\ x^{D-1} \\ x^D \end{bmatrix}$$

$$= \boldsymbol{\theta}^\top \boldsymbol{x} \qquad \boldsymbol{x}, \boldsymbol{\theta} \in \mathbb{R}^{D+1}$$

Scenario 1: Find optimal parameters to fit a 1D curve

- Model f is D-degree polynomial parametrised by $\boldsymbol{\theta}$, or $f_{\boldsymbol{\theta}}$

$$f_{\boldsymbol{\theta}}(x) = \theta_0 x^0 + \theta_1 x^1 + \theta_2 x^2 + \cdots + \theta_{D-1} x^{D-1} + \theta_D x^D$$

$$= \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \cdots & \theta_{D-1} & \theta_D \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ \cdots \\ x^{D-1} \\ x^D \end{bmatrix}$$

$$= \boldsymbol{\theta}^\top \boldsymbol{x} \qquad \boldsymbol{x}, \boldsymbol{\theta} \in \mathbb{R}^{D+1}$$

- Training data: N pairs (x_i, y_i)
- Training the model means finding parameters $\boldsymbol{\theta}^*$, such that

$$f_{\boldsymbol{\theta}^*}(x_i) \approx y_i$$

Scenario 1: Find optimal parameters to fit a 1D curve

- Model f is D-dimensional polynomial parametrised by $\boldsymbol{\theta}$, or $f_{\boldsymbol{\theta}}$

$$f_{\boldsymbol{\theta}}(x) = \theta_0 x^0 + \theta_1 x^1 + \theta_2 x^2 + \cdots + \theta_{D-1} x^{D-1} + \theta_D x^D$$

$$= \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \cdots & \theta_{D-1} & \theta_D \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ \cdots \\ x^{D-1} \\ x^D \end{bmatrix}$$

$$= \boldsymbol{\theta}^\top \boldsymbol{x} \qquad \boldsymbol{x}, \boldsymbol{\theta} \in \mathbb{R}^{D+1}$$

- Define a loss function, $L_{\boldsymbol{\theta}}(x, y) = \sum_{i=1}^{N} (y_i - f_{\boldsymbol{\theta}}(x_i))^2$
- Minimisation of loss function is typically based on gradient descent

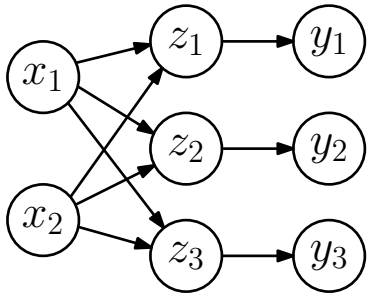
  ⏩ **Differentiation** required to compute gradients of $L_{\boldsymbol{\theta}}$ w.r.t $\boldsymbol{\theta}$

Scenario 2: Find optimal parameters of a Neural Network

$$\boldsymbol{x} \in \mathbb{R}^2, \boldsymbol{A} \in \mathbb{R}^{2\times 3}, \boldsymbol{b} \in \mathbb{R}^3$$

$$\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \in \mathbb{R}^3$$

$$\boldsymbol{y} = \sigma(\boldsymbol{z}) \in \mathbb{R}^3$$

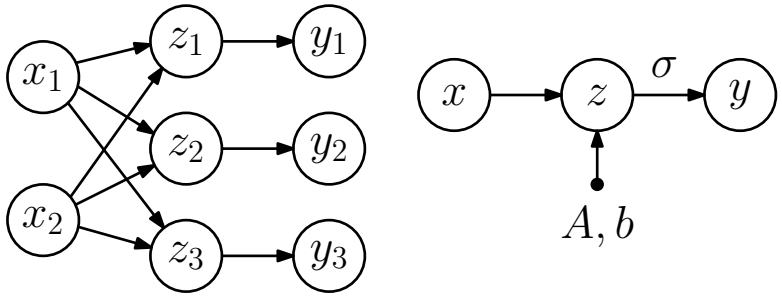


Scenario 2: Find optimal parameters of a Neural Network

$$\boldsymbol{x} \in \mathbb{R}^2, \boldsymbol{A} \in \mathbb{R}^{2\times 3}, \boldsymbol{b} \in \mathbb{R}^3$$

$$\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \in \mathbb{R}^3$$

$$\boldsymbol{y} = \sigma(\boldsymbol{z}) \in \mathbb{R}^3$$



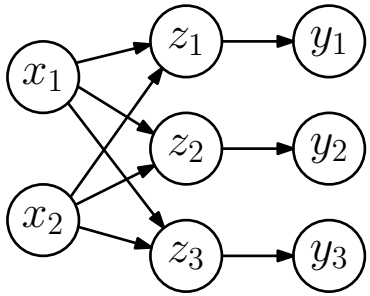
Scenario 2: Find optimal parameters of a Neural Network

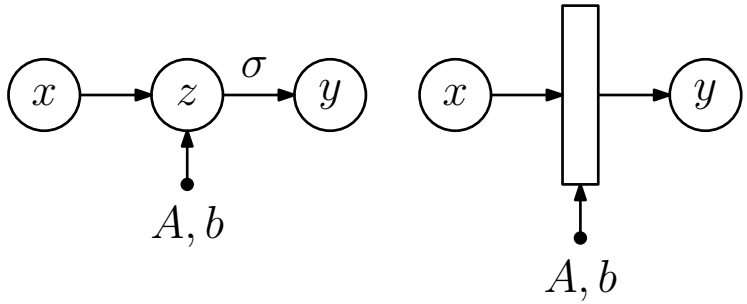
$$\boldsymbol{x} \in \mathbb{R}^2, \boldsymbol{A} \in \mathbb{R}^{2\times 3}, \boldsymbol{b} \in \mathbb{R}^3$$

$$\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \in \mathbb{R}^3$$

$$\boldsymbol{y} = \sigma(\boldsymbol{z}) \in \mathbb{R}^3$$

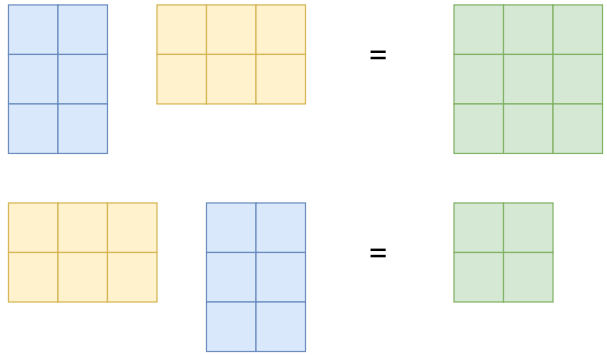
$$\boldsymbol{y} = f_{\boldsymbol{A},\boldsymbol{b}}(\boldsymbol{x})$$





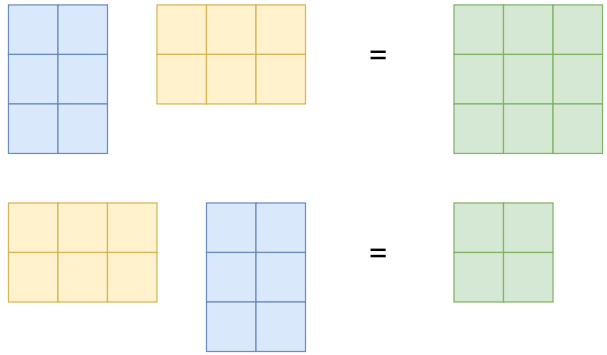
Background: Matrix Multiplication

- Matrix multiplication is not commutative, i.e., $\boldsymbol{AB} \neq \boldsymbol{BA}$



Background: Matrix Multiplication

- Matrix multiplication is not commutative, i.e., $\boldsymbol{AB} \neq \boldsymbol{BA}$

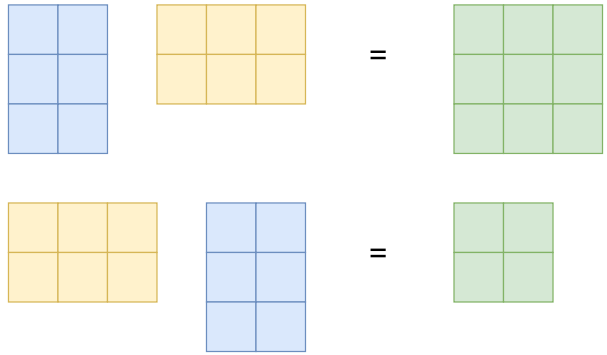


- When multiplying matrices, the "neighbouring" dimensions have to fit:

$$\underbrace{\boldsymbol{A}}_{n \times k} \underbrace{\boldsymbol{B}}_{k \times m} = \underbrace{\boldsymbol{C}}_{n \times m}$$

Background: Matrix Multiplication

- Matrix multiplication is not commutative, i.e., $\boldsymbol{AB} \neq \boldsymbol{BA}$



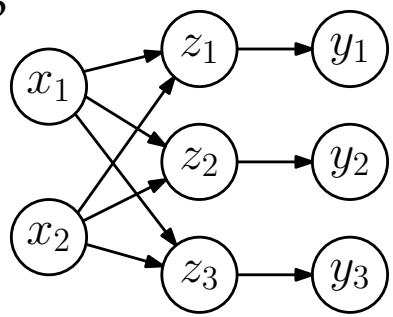
- When multiplying matrices, the "neighbouring" dimensions have to fit:

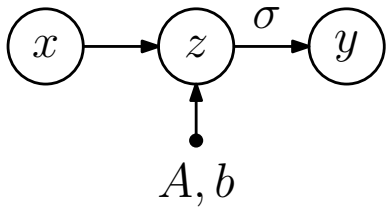
$$\underbrace{\boldsymbol{A}}_{n \times k} \underbrace{\boldsymbol{B}}_{k \times m} = \underbrace{\boldsymbol{C}}_{n \times m}$$

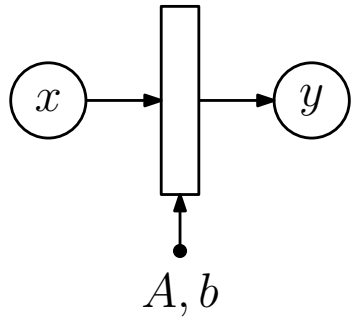
	`import numpy as np`
$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$	`y = A.dot(x)`
$y_i = \sum_j A_{ij} x_j$	`y = np.einsum('ij, j', A, x)`
$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$	`C = A.dot(B)`
$C_{ij} = \sum_k A_{ik} B_{kj}$	`C = np.einsum('ik, kj', A, B)`

Scenario 2: Find optimal parameters of a Neural Network

$$\boldsymbol{x} \in \mathbb{R}^2, \boldsymbol{A} \in \mathbb{R}^{2\times 3}, \boldsymbol{b} \in \mathbb{R}^3$$
$$\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \in \mathbb{R}^3$$
$$\boldsymbol{y} = \sigma(\boldsymbol{z}) \in \mathbb{R}^3$$
$$\boldsymbol{y} = f_{\boldsymbol{A},\boldsymbol{b}}(\boldsymbol{x})$$





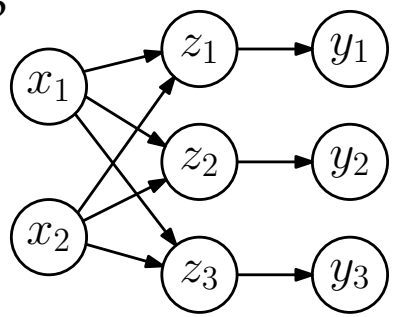


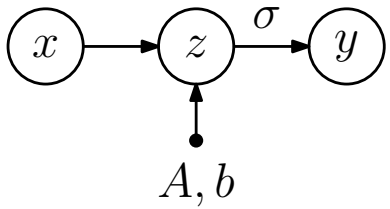
- Training data: N pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$
- Training the NN means finding parameters $\boldsymbol{A}^*, \boldsymbol{b}^*$, such that

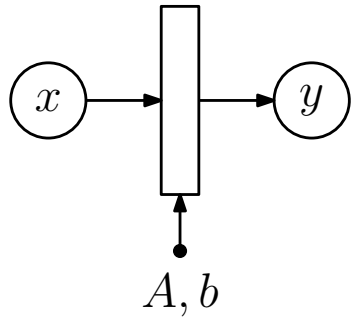
$$f_{\boldsymbol{A}^*,\boldsymbol{b}^*}(\boldsymbol{x}_i) \approx \boldsymbol{y}_i$$

Scenario 2: Find optimal parameters of a Neural Network

$$\boldsymbol{x} \in \mathbb{R}^2, \boldsymbol{A} \in \mathbb{R}^{2\times 3}, \boldsymbol{b} \in \mathbb{R}^3$$
$$\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \in \mathbb{R}^3$$
$$\boldsymbol{y} = \sigma(\boldsymbol{z}) \in \mathbb{R}^3$$
$$\boldsymbol{y} = f_{\boldsymbol{A},\boldsymbol{b}}(\boldsymbol{x})$$







- Training data: N pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$
- Training the NN means finding parameters $\boldsymbol{A}^*, \boldsymbol{b}^*$, such that

$$f_{\boldsymbol{A}^*,\boldsymbol{b}^*}(\boldsymbol{x}_i) \approx \boldsymbol{y}_i$$

- Define a loss function, $L_{\boldsymbol{A},\boldsymbol{b}}(\boldsymbol{x}, \boldsymbol{y})$
- Minimisation of loss function is typically based on gradient descent

  ⏩ **Differentiation** required to compute gradients of $L_{\boldsymbol{A},\boldsymbol{b}}$ w.r.t neural network parameters $\boldsymbol{A}, \boldsymbol{b}$

Types of differentiation

1. Scalar differentiation: $f : \mathbb{R} \to \mathbb{R}$

   $y \in \mathbb{R}$ w.r.t. $x \in \mathbb{R}$

2. Multivariate case: $f : \mathbb{R}^N \to \mathbb{R}$

   $y \in \mathbb{R}$ w.r.t. vector $\boldsymbol{x} \in \mathbb{R}^N$

3. Vector fields: $f : \mathbb{R}^N \to \mathbb{R}^M$

   vector $\boldsymbol{y} \in \mathbb{R}^M$ w.r.t. vector $\boldsymbol{x} \in \mathbb{R}^N$

4. General derivatives: $f : \mathbb{R}^{M \times N} \to \mathbb{R}^{P \times Q}$

   matrix $\boldsymbol{y} \in \mathbb{R}^{P \times Q}$ w.r.t. matrix $\boldsymbol{x} \in \mathbb{R}^{M \times N}$

Rules

- Sum Rule

$$\big(f(x) + g(x)\big)' = f'(x) + g'(x) = \frac{df}{dx} + \frac{dg}{dx}$$

Rules

- Sum Rule

$$\big(f(x)+g(x)\big)' = f'(x)+g'(x) = \frac{df}{dx}+\frac{dg}{dx}$$

- Product Rule

$$\big(f(x)g(x)\big)' = f'(x)g(x)+f(x)g'(x) = \frac{df}{dx}g(x)+f(x)\frac{dg}{dx}$$

Rules

- Sum Rule

$$\big(f(x)+g(x)\big)' = f'(x)+g'(x) = \frac{df}{dx}+\frac{dg}{dx}$$

- Product Rule

$$\big(f(x)g(x)\big)' = f'(x)g(x)+f(x)g'(x) = \frac{df}{dx}g(x)+f(x)\frac{dg}{dx}$$

- Chain Rule

$$(g\circ f)'(x) = \big(g(f(x))\big)' = g'(f(x))f'(x) = \frac{dg}{df}\frac{df}{dx}$$

Rules

- Sum Rule

$$\big(f(x)+g(x)\big)' = f'(x)+g'(x) = \frac{df}{dx}+\frac{dg}{dx}$$

- Product Rule

$$\big(f(x)g(x)\big)' = f'(x)g(x)+f(x)g'(x) = \frac{df}{dx}g(x)+f(x)\frac{dg}{dx}$$

- Chain Rule

$$(g\circ f)'(x) = \big(g(f(x))\big)' = g'(f(x))f'(x) = \frac{dg}{df}\frac{df}{dx}$$

- Quotient Rule

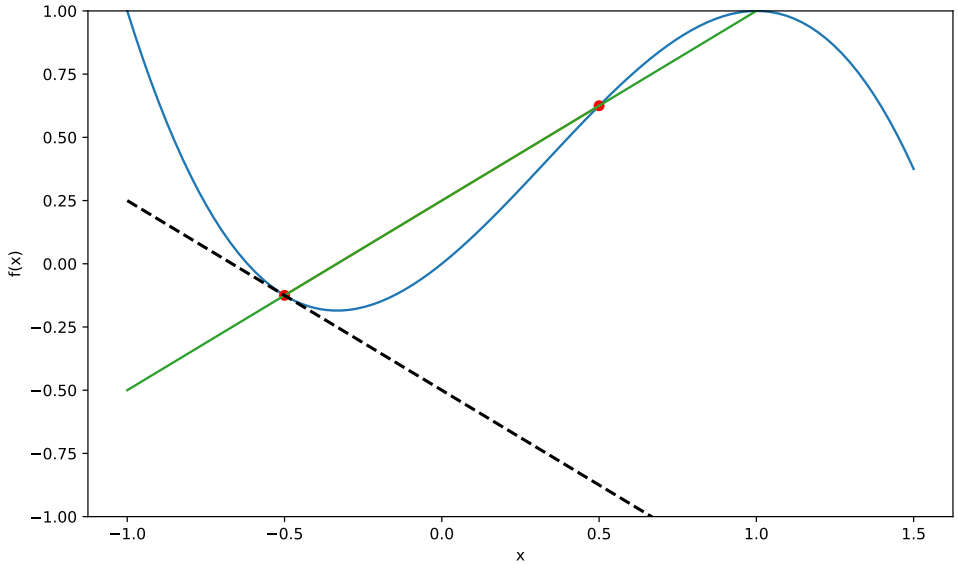
$$\left(\frac{f(x)}{g(x)}\right)' = \frac{f(x)'g(x)-f(x)g(x)'}{(g(x))^2} = \frac{\frac{df}{dx}g(x)-f(x)\frac{dg}{dx}}{(g(x))^2}$$

Scalar Differentiation $f : \mathbb{R} \to \mathbb{R}$

- Derivative defined as the limit of the difference quotient

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

  - Slope of the secant line through $f(x)$ and $f(x+h)$



Some examples

$$
\begin{aligned}
f(x) &= x^n & f'(x) &= nx^{n-1} \\
f(x) &= \sin(x) & f'(x) &= \cos(x) \\
f(x) &= \tanh(x) & f'(x) &= 1 - \tanh^2(x) \\
f(x) &= \exp(x) & f'(x) &= \exp(x) \\
f(x) &= \log(x) & f'(x) &= \tfrac{1}{x}
\end{aligned}
$$

Example: Scalar Chain Rule

$$(g \circ f)'(x) = \big(g(f(x))\big)' = g'(f(x))f'(x) = \frac{dg}{df}\frac{df}{dx}$$

Beginner

$$g(z) = 6z + 3$$
$$z = f(x) = -2x + 5$$
$$(g \circ f)'(x) =$$

Advanced

$$g(z) = \tanh(z)$$
$$z = f(x) = x^n$$
$$(g \circ f)'(x) =$$

Work it out with your neighbours

Example: Scalar Chain Rule

$$(g \circ f)'(x) = \big(g(f(x))\big)' = g'(f(x))f'(x) = \frac{dg}{df}\frac{df}{dx}$$

Beginner

$$\begin{aligned} g(z) &= 6z + 3 \\ z &= f(x) = -2x + 5 \\ (g \circ f)'(x) &= \underbrace{(6)}_{dg/df}\underbrace{(-2)}_{df/dx} \\ &= -12 \end{aligned}$$

Advanced

$$\begin{aligned} g(z) &= \tanh(z) \\ z &= f(x) = x^n \\ (g \circ f)'(x) &= \underbrace{(1 - \tanh^2(x^n))}_{dg/df}\underbrace{nx^{n-1}}_{df/dx} \end{aligned}$$

Multivariate differentiation $f : \mathbb{R}^N \to \mathbb{R}$

$$y = f(\boldsymbol{x}), \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^N$$

- Partial derivative (change one coordinate at a time):

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_N) - f(\boldsymbol{x})}{h}$$

Multivariate differentiation $f : \mathbb{R}^N \to \mathbb{R}$

$$y = f(\boldsymbol{x}), \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^N$$

- Partial derivative (change one coordinate at a time):

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_N) - f(\boldsymbol{x})}{h}$$

- Jacobian vector (gradient) collects all partial derivatives:

$$\frac{df}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{x_N} \end{bmatrix} \in \mathbb{R}^{1 \times N}$$

Note: This is a row vector.

Example: Multivariate differentiation

Beginner

$$f : \mathbb{R}^2 \to \mathbb{R}$$
$$f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in \mathbb{R}$$

Advanced

$$f : \mathbb{R}^2 \to \mathbb{R}$$
$$f(x_1, x_2) = (x_1 + 2x_2^3)^2 \in \mathbb{R}$$

Partial derivatives?

Work it out with your neighbours

Example: Multivariate differentiation

Beginner

$$f : \mathbb{R}^2 \to \mathbb{R}$$
$$f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in \mathbb{R}$$

Advanced

$$f : \mathbb{R}^2 \to \mathbb{R}$$
$$f(x_1, x_2) = (x_1 + 2x_2^3)^2 \in \mathbb{R}$$

Partial derivatives

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 x_2 + x_2^3$$
$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 + 3x_1 x_2^2$$

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2(x_1 + 2x_2^3) \overbrace{(1)}^{\frac{d}{dx_1}(x_1+2x_2^3)}$$
$$\frac{\partial f(x_1, x_2)}{\partial x_2} = 2(x_1 + 2x_2^3) \underbrace{(6x_2^2)}_{\frac{d}{dx_2}(x_1+2x_2^3)}$$

Example: Multivariate differentiation

Beginner

$f : \mathbb{R}^2 \to \mathbb{R}$

$f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in \mathbb{R}$

Advanced

$f : \mathbb{R}^2 \to \mathbb{R}$

$f(x_1, x_2) = (x_1 + 2x_2^3)^2 \in \mathbb{R}$

Partial derivatives

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 x_2 + x_2^3$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 + 3x_1 x_2^2$$

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2(x_1 + 2x_2^3) \underbrace{(1)}_{\frac{d}{dx_1}(x_1 + 2x_2^3)}$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = 2(x_1 + 2x_2^3) \underbrace{(6x_2^2)}_{\frac{d}{dx_2}(x_1 + 2x_2^3)}$$

Gradient $\quad \frac{df}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(x_1, x_2)}{\partial x_1} & \frac{\partial f(x_1, x_2)}{\partial x_2} \end{bmatrix} \in \mathbb{R}^{1 \times 2}$

$$\frac{df}{d\boldsymbol{x}} = \begin{bmatrix} 2x_1 x_2 + x_2^3 & x_1^2 + 3x_1 x_2^2 \end{bmatrix}$$

$$\frac{df}{d\boldsymbol{x}} = \begin{bmatrix} 2(x_1 + 2x_2^3) & 12(x_1 + 2x_2^3)x_2^2 \end{bmatrix}$$

Multivariate Differentiation Rules

- Sum Rule

$$\big(f(\boldsymbol{x}) + g(\boldsymbol{x})\big)' = f'(\boldsymbol{x}) + g'(\boldsymbol{x}) = \frac{df}{d\boldsymbol{x}} + \frac{dg}{d\boldsymbol{x}}$$

- Product Rule

$$\big(f(\boldsymbol{x})g(\boldsymbol{x})\big)' = f'(\boldsymbol{x})g(\boldsymbol{x}) + f(\boldsymbol{x})g'(\boldsymbol{x}) = \frac{df}{d\boldsymbol{x}}g(\boldsymbol{x}) + f(\boldsymbol{x})\frac{dg}{d\boldsymbol{x}}$$

- Chain Rule

$$(g \circ f)'(\boldsymbol{x}) = \big(g(f(\boldsymbol{x}))\big)' = g'(f(\boldsymbol{x}))f'(\boldsymbol{x}) = \frac{dg}{df}\frac{df}{d\boldsymbol{x}}$$

- Quotient Rule

$$\left(\frac{f(\boldsymbol{x})}{g(\boldsymbol{x})}\right)' = \frac{f(\boldsymbol{x})'g(\boldsymbol{x}) - f(\boldsymbol{x})g(\boldsymbol{x})'}{(g(\boldsymbol{x}))^2} = \frac{\frac{df}{d\boldsymbol{x}}g(\boldsymbol{x}) - f(\boldsymbol{x})\frac{dg}{d\boldsymbol{x}}}{(g(\boldsymbol{x}))^2}$$

Example: Multivariate Chain Rule

- Consider the function

$$L(\boldsymbol{e}) = \tfrac{1}{2}\|\boldsymbol{e}\|^2 = \tfrac{1}{2}\boldsymbol{e}^\top \boldsymbol{e}$$

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M \times N}, \boldsymbol{e}, \boldsymbol{y} \in \mathbb{R}^M$$

- Compute $dL/d\boldsymbol{x}$. What is the dimension/size of $dL/d\boldsymbol{x}$?

Example: Multivariate Chain Rule

- Consider the function

$$L(\boldsymbol{e}) = \tfrac{1}{2}\|\boldsymbol{e}\|^2 = \tfrac{1}{2}\boldsymbol{e}^\top \boldsymbol{e}$$

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{e}, \boldsymbol{y} \in \mathbb{R}^M$$

- Compute $dL/d\boldsymbol{x}$. What is the dimension/size of $dL/d\boldsymbol{x}$?

$$\frac{dL}{d\boldsymbol{x}} = \frac{dL}{d\boldsymbol{e}}\frac{d\boldsymbol{e}}{d\boldsymbol{x}}$$

$$\frac{dL}{d\boldsymbol{e}} = \boldsymbol{e}^\top \in \mathbb{R}^{1\times M}$$

$$\frac{d\boldsymbol{e}}{d\boldsymbol{x}} = -\boldsymbol{A} \in \mathbb{R}^{M\times N}$$

$$\Rightarrow \frac{dL}{d\boldsymbol{x}} = \boldsymbol{e}^\top(-\boldsymbol{A}) = -(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x})^\top \boldsymbol{A} \in \mathbb{R}^{1\times N}$$

Vector field differentiation $f : \mathbb{R}^N \to \mathbb{R}^M$

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} f_1(x_1, \ldots, x_N) \\ \vdots \\ f_M(x_1, \ldots, x_N) \end{bmatrix}$$

Vector field differentiation $f : \mathbb{R}^N \to \mathbb{R}^M$

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} f_1(x_1, \ldots, x_N) \\ \vdots \\ f_M(x_1, \ldots, x_N) \end{bmatrix}$$

- Jacobian matrix (collection of all partial derivatives)

$$\begin{bmatrix} \frac{dy_1}{d\boldsymbol{x}} \\ \vdots \\ \frac{dy_M}{d\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_i} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & \vdots & & \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_i} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix} \in \mathbb{R}^{M \times N}$$

Example: Vector field differentiation

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}, \qquad \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{A} \in \mathbb{R}^{M \times N}, \quad \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

- Compute the gradient $d\boldsymbol{f}/d\boldsymbol{x}$
  - Dimension of $d\boldsymbol{f}/d\boldsymbol{x}$:

Example: Vector field differentiation

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}, \qquad \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{A} \in \mathbb{R}^{M\times N}, \quad \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

- Compute the gradient $d\boldsymbol{f}/d\boldsymbol{x}$
  - Dimension of $d\boldsymbol{f}/d\boldsymbol{x}$: Since $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^M$, $d\boldsymbol{f}/d\boldsymbol{x} \in \mathbb{R}^{M\times N}$

Example: Vector field differentiation

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}, \qquad \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{A} \in \mathbb{R}^{M\times N}, \quad \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

- Compute the gradient $d\boldsymbol{f}/d\boldsymbol{x}$
  - Dimension of $d\boldsymbol{f}/d\boldsymbol{x}$: Since $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^M$, $d\boldsymbol{f}/d\boldsymbol{x} \in \mathbb{R}^{M\times N}$
  - Gradient:

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{N} A_{ij}x_j \qquad \Rightarrow \frac{\partial f_i}{\partial x_j} = A_{ij}$$

Example: Vector field differentiation

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}, \qquad \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{A} \in \mathbb{R}^{M\times N}, \quad \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

- Compute the gradient $d\boldsymbol{f}/d\boldsymbol{x}$
  - Dimension of $d\boldsymbol{f}/d\boldsymbol{x}$: Since $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^M$, $d\boldsymbol{f}/d\boldsymbol{x} \in \mathbb{R}^{M\times N}$
  - Gradient:

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{N} A_{ij}x_j \qquad \Rightarrow \frac{\partial f_i}{\partial x_j} = A_{ij}$$

$$\Rightarrow \frac{d\boldsymbol{f}}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix} = \begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & & \vdots \\ A_{M1} & \cdots & A_{MN} \end{bmatrix} = \boldsymbol{A}$$

Chain Rule

$$\frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}\big(g(f(\boldsymbol{x}))\big) = \frac{\partial g}{\partial f}\frac{\partial f}{\partial \boldsymbol{x}}$$

Example: Combined Chain Rule

- Consider $f : \mathbb{R}^2 \to \mathbb{R}, \quad \boldsymbol{x} : \mathbb{R} \to \mathbb{R}^2$

$$f(\boldsymbol{x}) = f(x_1, x_2) = x_1^2 + 2x_2\,,$$

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$$

Example: Combined Chain Rule

- Consider $f : \mathbb{R}^2 \to \mathbb{R}, \quad \boldsymbol{x} : \mathbb{R} \to \mathbb{R}^2$

$$f(\boldsymbol{x}) = f(x_1, x_2) = x_1^2 + 2x_2\,,$$

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$$

- What are the dimensions of $df/d\boldsymbol{x}$ and $d\boldsymbol{x}/dt$?

Example: Combined Chain Rule

- Consider $f : \mathbb{R}^2 \to \mathbb{R}, \quad \boldsymbol{x} : \mathbb{R} \to \mathbb{R}^2$

$$f(\boldsymbol{x}) = f(x_1, x_2) = x_1^2 + 2x_2\,,$$

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$$

- What are the dimensions of $df/d\boldsymbol{x}$ and $d\boldsymbol{x}/dt$?

1×2 and 2×1

- Compute the gradient df/dt using the chain rule:

Work it out with your neighbours

Example: Combined Chain Rule

- Consider $f : \mathbb{R}^2 \to \mathbb{R}, \quad \boldsymbol{x} : \mathbb{R} \to \mathbb{R}^2$

$$f(\boldsymbol{x}) = f(x_1, x_2) = x_1^2 + 2x_2\,,$$

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$$

- What are the dimensions of $df/d\boldsymbol{x}$ and $d\boldsymbol{x}/dt$?

1×2 and 2×1

- Compute the gradient df/dt using the chain rule:

$$\frac{df}{dt} = \frac{df}{d\boldsymbol{x}}\frac{d\boldsymbol{x}}{dt} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix} = \begin{bmatrix} 2\sin t & 2 \end{bmatrix} \begin{bmatrix} \cos t \\ -\sin t \end{bmatrix}$$

$$= 2\sin t \cos t - 2\sin t = 2\sin t(\cos t - 1)$$

5 MINUTE BREAK

Derivatives with respect to Matrices

- Recall: function $f : \mathbb{R}^D \to \mathbb{R}^E$ has a gradient that is $E \times D$-matrix

  # target dimensions $\times$ # input dimensions

  with

$$\frac{df}{dx} \in \mathbb{R}^{E \times D}, \qquad df[e,d] = \frac{\partial f_e}{\partial x_d}$$

Derivatives with respect to Matrices

- Recall: function $f : \mathbb{R}^D \to \mathbb{R}^E$ has a gradient that is $E \times D$-matrix

  $$\#\text{ target dimensions} \times \#\text{ input dimensions}$$

  with

  $$\frac{df}{dx} \in \mathbb{R}^{E \times D}, \qquad df[e,d] = \frac{\partial f_e}{\partial x_d}$$

- This generalises to when the inputs (D) or targets (E) are **matrices**

Derivatives with respect to Matrices

- Recall: function $f : \mathbb{R}^D \to \mathbb{R}^E$ has a gradient that is $E \times D$-matrix

  $$\#\text{ target dimensions} \times \#\text{ input dimensions}$$

  with

  $$\frac{df}{dx} \in \mathbb{R}^{E \times D}, \qquad df[e,d] = \frac{\partial f_e}{\partial x_d}$$

- This generalises to when the inputs (D) or targets (E) are **matrices**
- Function $f : \mathbb{R}^{M \times N} \to \mathbb{R}^{P \times Q}$, has a gradient that is a $(P \times Q) \times (M \times N)$ object (tensor)

$$\frac{df}{dX} \in \mathbb{R}^{(P \times Q) \times (M \times N)}, \qquad df[p,q,m,n] = \frac{\partial f_{pq}}{\partial X_{mn}}$$

Example 1: Derivatives w.r.t Matrices

$$\boldsymbol{f} = \boldsymbol{A}\boldsymbol{x}, \quad \boldsymbol{f} \in \mathbb{R}^M, \boldsymbol{A} \in \mathbb{R}^{M \times N}, \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

$$\frac{d\boldsymbol{f}}{d\boldsymbol{A}} \in \mathbb{R}^{?}$$

Example 1: Derivatives w.r.t Matrices

$$\boldsymbol{f} = \boldsymbol{A}\boldsymbol{x}, \quad \boldsymbol{f} \in \mathbb{R}^M, \boldsymbol{A} \in \mathbb{R}^{M \times N}, \boldsymbol{x} \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

$$\frac{d\boldsymbol{f}}{d\boldsymbol{A}} \in \mathbb{R}^{\#\text{ target dim} \times \#\text{ input dim}} = M \times (M \times N)$$

$$\frac{d\boldsymbol{f}}{d\boldsymbol{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial \boldsymbol{A}} \\ \vdots \\ \frac{\partial f_M}{\partial \boldsymbol{A}} \end{bmatrix}, \quad \frac{\partial f_i}{\partial \boldsymbol{A}} \in \mathbb{R}^{1 \times (M \times N)}$$

Example 2: Derivatives w.r.t Matrices

$$f_i = \sum_{j=1}^{N} A_{ij} x_j, \quad i = 1, \ldots, M$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_i(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{i1}x_1 + A_{i2}x_2 & \cdots & + A_{i(N-1)}x_{N-1} + A_{iN}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

$$\frac{\partial f_i}{\partial A_{iq}} = ? \qquad \frac{\partial f_i}{\partial A_{i,:}} = ? \qquad \frac{\partial f_i}{\partial A_{k\neq i,:}} = ? \qquad \frac{\partial f_i}{\partial \boldsymbol{A}} = ?$$

Example 2: Derivatives w.r.t Matrices

$$f_i = \sum_{j=1}^{N} A_{ij} x_j, \quad i = 1, \ldots, M$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_i(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{i1}x_1 + A_{i2}x_2 & \cdots & + A_{i(N-1)}x_{N-1} + A_{iN}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

$$\frac{\partial f_i}{\partial A_{iq}} = \underbrace{x_q}_{\in \mathbb{R}} \qquad \frac{\partial f_i}{\partial A_{i,:}} = ? \qquad \frac{\partial f_i}{\partial A_{k \neq i,:}} = ? \qquad \frac{\partial f_i}{\partial \boldsymbol{A}} = ?$$

Example 2: Derivatives w.r.t Matrices

$$f_i = \sum_{j=1}^{N} A_{ij} x_j, \quad i = 1, \ldots, M$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_i(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{i1}x_1 + A_{i2}x_2 & \cdots & + A_{i(N-1)}x_{N-1} + A_{iN}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

$$\frac{\partial f_i}{\partial A_{iq}} = \underbrace{x_q}_{\in \mathbb{R}} \qquad \frac{\partial f_i}{\partial A_{i,:}} = \underbrace{\boldsymbol{x}^\top}_{\in \mathbb{R}^{1\times 1\times N}} \qquad \frac{\partial f_i}{\partial A_{k\neq i,:}} = ? \qquad \frac{\partial f_i}{\partial \boldsymbol{A}} = ?$$

Example 2: Derivatives w.r.t Matrices

$$f_i = \sum_{j=1}^{N} A_{ij} x_j, \quad i = 1, \ldots, M$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_i(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & +A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{i1}x_1 + A_{i2}x_2 & \cdots & +A_{i(N-1)}x_{N-1} + A_{iN}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & +A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

$$\frac{\partial f_i}{\partial A_{iq}} = \underbrace{x_q}_{\in \mathbb{R}} \qquad \frac{\partial f_i}{\partial A_{i,:}} = \underbrace{\boldsymbol{x}^\top}_{\in \mathbb{R}^{1\times 1\times N}} \qquad \frac{\partial f_i}{\partial A_{k\neq i,:}} = \underbrace{\boldsymbol{0}^\top}_{\in \mathbb{R}^{1\times 1\times N}} \qquad \frac{\partial f_i}{\partial \boldsymbol{A}} = ?$$

Example 2: Derivatives w.r.t Matrices

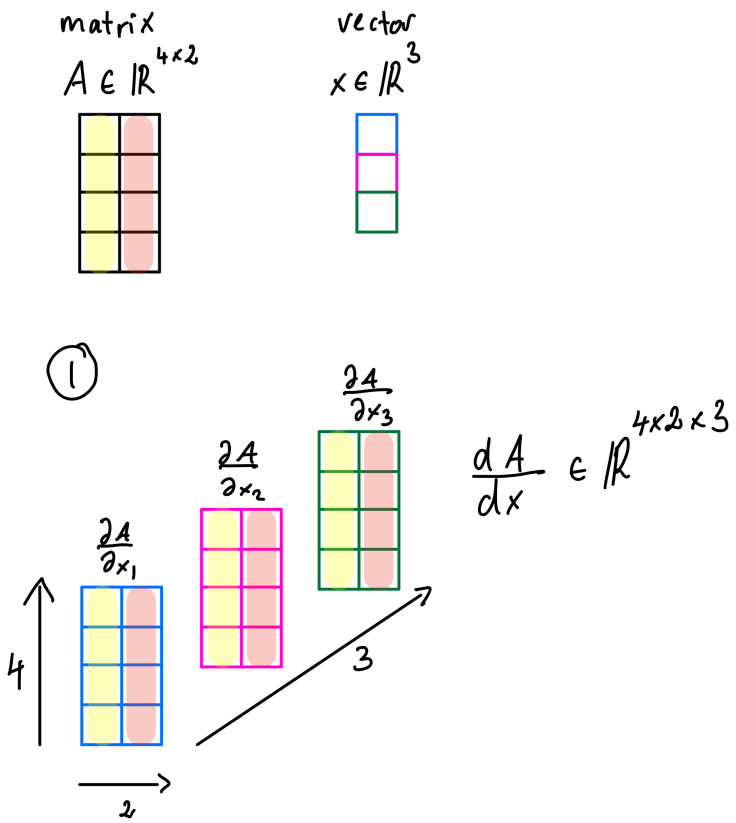
$$f_i = \sum_{j=1}^{N} A_{ij} x_j, \quad i = 1, \ldots, M$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_i(\boldsymbol{x}) \\ \vdots \\ f_M(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + & \cdots & + A_{1(N-1)}x_{N-1} + A_{1N}x_N \\ \vdots & \vdots & \vdots \\ A_{i1}x_1 + A_{i2}x_2 & \cdots & + A_{i(N-1)}x_{N-1} + A_{iN}x_N \\ \vdots & \vdots & \vdots \\ A_{M1}x_1 + A_{M2}x_2 + & \cdots & + A_{M(N-1)}x_{N-1} + A_{MN}x_N \end{bmatrix}$$

$$\frac{\partial f_i}{\partial A_{iq}} = \underbrace{x_q}_{\in \mathbb{R}} \qquad \frac{\partial f_i}{\partial A_{i,:}} = \underbrace{\boldsymbol{x}^\top}_{\in \mathbb{R}^{1\times 1\times N}} \qquad \frac{\partial f_i}{\partial A_{k\neq i,:}} = \underbrace{\mathbf{0}^\top}_{\in \mathbb{R}^{1\times 1\times N}} \qquad \frac{\partial f_i}{\partial \boldsymbol{A}} = \underbrace{\begin{bmatrix} \mathbf{0}^\top \\ \vdots \\ \boldsymbol{x}^\top \\ \vdots \\ \mathbf{0}^\top \end{bmatrix}}_{\in \mathbb{R}^{1\times(M\times N)}}$$

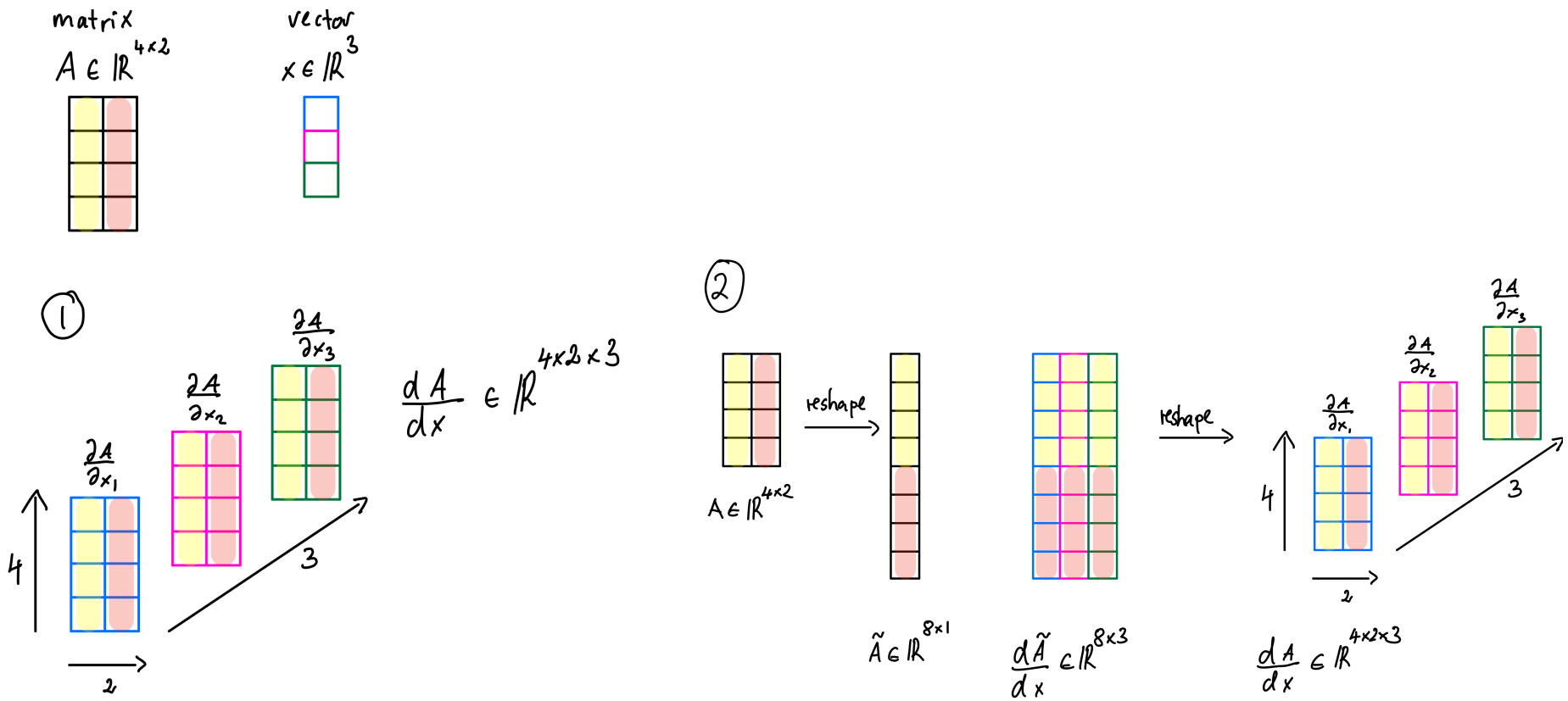
Example 3: Derivatives w.r.t Higher-Order Tensors

- Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{4\times 2}$ whose entries depend on a vector $\boldsymbol{x} \in \mathbb{R}^3$
- We can compute $d\boldsymbol{A}(\boldsymbol{x})/d\boldsymbol{x} \in \mathbb{R}^{4\times 2\times 3}$ in two equivalent ways:

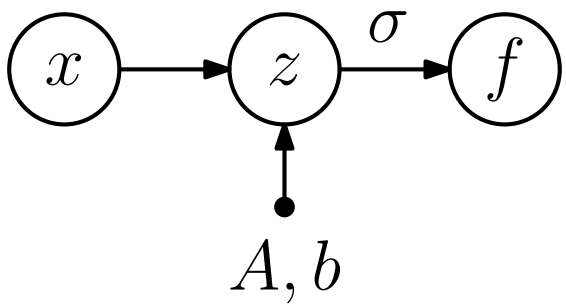


Example 3: Derivatives w.r.t Higher-Order Tensors

- Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{4\times 2}$ whose entries depend on a vector $\boldsymbol{x} \in \mathbb{R}^3$
- We can compute $d\boldsymbol{A}(\boldsymbol{x})/d\boldsymbol{x} \in \mathbb{R}^{4\times 2\times 3}$ in two equivalent ways:



Gradients of a Single-Layer Neural Network



$$\boldsymbol{f} = \tanh(\underbrace{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}_{=:\boldsymbol{z} \in \mathbb{R}^M}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M \times N}, \boldsymbol{b} \in \mathbb{R}^M$$

Gradients of a Single-Layer Neural Network

$$\boldsymbol{f} = \tanh(\underbrace{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}_{=:\boldsymbol{z}\in\mathbb{R}^M}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{b} \in \mathbb{R}^M$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{b}} =$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{A}} =$$

Gradients of a Single-Layer Neural Network

$$\boldsymbol{f} = \tanh(\underbrace{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}_{=:\boldsymbol{z}\in\mathbb{R}^M}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{b} \in \mathbb{R}^M$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{b}} = \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}}_{M\times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}}_{M\times M} \in \mathbb{R}^{M\times M}$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{A}} =$$

Gradients of a Single-Layer Neural Network

$$\boldsymbol{f} = \tanh(\underbrace{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}_{=:\boldsymbol{z}\in\mathbb{R}^M}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{b} \in \mathbb{R}^M$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{b}} = \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}}_{M\times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}}_{M\times M} \in \mathbb{R}^{M\times M}$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{A}} = \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}}_{M\times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{A}}}_{M\times(M\times N)} \in \mathbb{R}^{M\times(M\times N)}$$

Gradients of a Single-Layer Neural Network

$$\boldsymbol{f} = \tanh(\underbrace{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}_{=:\boldsymbol{z}\in\mathbb{R}^M}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{b} \in \mathbb{R}^M$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{b}} = \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}}_{M\times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}}_{M\times M} \in \mathbb{R}^{M\times M}$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{A}} = \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}}_{M\times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{A}}}_{M\times(M\times N)} \in \mathbb{R}^{M\times(M\times N)}$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} = \underbrace{\operatorname{diag}(1 - \tanh^2(\boldsymbol{z}))}_{\in \mathbb{R}^{M\times M}}$$

Gradients of a Single-Layer Neural Network

$$\boldsymbol{f} = \tanh(\underbrace{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}_{=:\boldsymbol{z} \in \mathbb{R}^M}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M \times N}, \boldsymbol{b} \in \mathbb{R}^M$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{b}} = \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}}_{M \times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}}_{M \times M} \in \mathbb{R}^{M \times M}$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{A}} = \underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}}_{M \times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{A}}}_{M \times (M \times N)} \in \mathbb{R}^{M \times (M \times N)}$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} = \underbrace{\operatorname{diag}(1 - \tanh^2(\boldsymbol{z}))}_{\in \mathbb{R}^{M \times M}} \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}} = \underbrace{\boldsymbol{I}}_{\in \mathbb{R}^{M \times M}}$$

Gradients of a Single-Layer Neural Network

$$f = \tanh(\underbrace{\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}_{=:\boldsymbol{z}\in\mathbb{R}^M}) \in \mathbb{R}^M, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{b} \in \mathbb{R}^M$$

$$\frac{\partial f}{\partial \boldsymbol{b}} = \underbrace{\frac{\partial f}{\partial \boldsymbol{z}}}_{M\times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}}_{M\times M} \in \mathbb{R}^{M\times M}$$

$$\frac{\partial f}{\partial \boldsymbol{A}} = \underbrace{\frac{\partial f}{\partial \boldsymbol{z}}}_{M\times M} \underbrace{\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{A}}}_{M\times(M\times N)} \in \mathbb{R}^{M\times(M\times N)}$$

$$\frac{\partial f}{\partial \boldsymbol{z}} = \underbrace{\operatorname{diag}(1 - \tanh^2(\boldsymbol{z}))}_{\in\mathbb{R}^{M\times M}} \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}} = \underbrace{\boldsymbol{I}}_{\in\mathbb{R}^{M\times M}} \quad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{A}} = \underbrace{\begin{bmatrix} \boldsymbol{x}^\top & \cdot & \boldsymbol{0}^\top & \cdot & \boldsymbol{0}^\top \\ \cdot & & \cdot & & \cdot \\ \boldsymbol{0}^\top & \cdot & \boldsymbol{x}^\top & \cdot & \boldsymbol{0}^\top \\ \cdot & & \cdot & & \cdot \\ \boldsymbol{0}^\top & \cdot & \boldsymbol{0}^\top & \cdot & \boldsymbol{x}^\top \end{bmatrix}}_{\in\mathbb{R}^{M\times(M\times N)}}$$

```
import numpy as np
def NN(x, A, b):
    M, N = A.shape
    z = A.dot(x) + b
    f = np.tanh(z)
    # partial derivatives
    dfdz = 1-f**2
    dzdx = A
    dzdb = np.eye(M)
    dzdA = np.zeros((M, M, N))
    for i in range(M):
        dzdA[i,i,:]  = x.T
    # gradients
    dfdx = np.einsum('il, lj', dfdz, dzdx)
    dfdb = np.einsum('il, lj', dfdz, dzdb)
    dfdA = np.einsum('il, ljk', dfdz, dzdA)
    return f, dfdA, dfdb, dfdx
```

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}[i,j] = \sum_{l=1}^{M} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}[i,l] \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}[l,j]$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{b}}[i,j] = \sum_{l=1}^{M} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}[i,l] \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}[l,j]$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{A}}[i,j,k] = \sum_{l=1}^{M} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}[i,l] \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{A}}[l,j,k$$

Putting Things Together

- Inputs $\boldsymbol{x} \in \mathbb{R}^N$

Putting Things Together

- Inputs $\boldsymbol{x} \in \mathbb{R}^N$
- Observed outputs $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{z}) + \boldsymbol{\epsilon} \in \mathbb{R}^M, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$

Putting Things Together

- Inputs $\boldsymbol{x} \in \mathbb{R}^N$
- Observed outputs $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{z}) + \boldsymbol{\epsilon} \in \mathbb{R}^M, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$
- Train single-layer neural network with

$$f_{\boldsymbol{\theta}}(\boldsymbol{z}) = \tanh(\boldsymbol{z}) \in \mathbb{R}^M, \quad \boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \in \mathbb{R}^M, \quad \boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{b}\}$$

Putting Things Together

- Inputs $\boldsymbol{x} \in \mathbb{R}^N$
- Observed outputs $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{z}) + \boldsymbol{\epsilon} \in \mathbb{R}^M, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$
- Train single-layer neural network with

$$f_{\boldsymbol{\theta}}(\boldsymbol{z}) = \tanh(\boldsymbol{z}) \in \mathbb{R}^M, \quad \boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \in \mathbb{R}^M, \quad \boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{b}\}$$

- Find $\boldsymbol{A}^*, \boldsymbol{b}^*$, such that the squared loss

$$L(\boldsymbol{\theta}) = \tfrac{1}{2}\|\boldsymbol{e}\|^2 \in \mathbb{R}, \quad \boldsymbol{e} = \boldsymbol{y} - \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{z}) \in \mathbb{R}^M$$

  is minimised

Putting Things Together

Partial derivatives:

$$\frac{\partial L}{\partial \boldsymbol{A}} = \frac{\partial L}{\partial \boldsymbol{e}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{f}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{A}}$$

$$\frac{\partial L}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{e}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{f}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}}$$

$$\frac{\partial L}{\partial \boldsymbol{e}} = \underbrace{\boldsymbol{e}^\top}_{\in \mathbb{R}^{1\times M}} \qquad \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{f}} = \underbrace{-\boldsymbol{I}}_{\in \mathbb{R}^{M\times M}} \qquad \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} = \underbrace{\operatorname{diag}(1-\tanh^2(\boldsymbol{z}))}_{\in \mathbb{R}^{M\times M}}$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{A}} = \underbrace{\begin{bmatrix} \boldsymbol{x}^\top & \cdot & \boldsymbol{0}^\top & \cdot & \boldsymbol{0}^\top \\ \cdot & & \cdot & & \cdot \\ \boldsymbol{0}^\top & \cdot & \boldsymbol{x}^\top & \cdot & \boldsymbol{0}^\top \\ \cdot & & \cdot & & \cdot \\ \boldsymbol{0}^\top & \cdot & \boldsymbol{0}^\top & \cdot & \boldsymbol{x}^\top \end{bmatrix}}_{\in \mathbb{R}^{M\times(M\times N)}} \qquad \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}} = \underbrace{\boldsymbol{I}}_{\in \mathbb{R}^{M\times M}}$$

Gradient descent for neural network optimisation

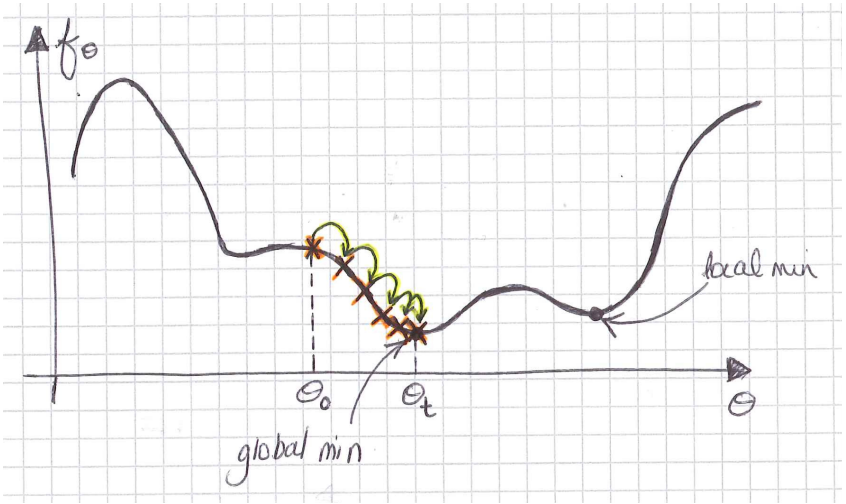
Goal: Find the local optimum of loss $L(\boldsymbol{\theta}^*) \in \mathbb{R}$ where $\boldsymbol{\theta}^* = \{\boldsymbol{A}^*, \boldsymbol{b}^*\}$

Gradient descent algorithm:

1. Initialise neural network parameters, $\boldsymbol{\theta}_0 = \{\boldsymbol{A}_0, \boldsymbol{b}_0\}$
2. Compute gradient of L w.r.t. parameters, $\frac{\partial L}{\partial \boldsymbol{\theta}_0}$
3. Update initial guess $\boldsymbol{\theta}_0$ using chain rule:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \Delta\boldsymbol{\theta}_i = \boldsymbol{\theta}_i - \gamma(\frac{\partial L}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i))^\top$$

- Note: parameter update is the direction of steepest **de**scent



Gradient descent for neural network optimisation

Goal: Find the local optimum of loss $L(\boldsymbol{\theta}^*) \in \mathbb{R}$ where $\boldsymbol{\theta}^* = \{\boldsymbol{A}^*, \boldsymbol{b}^*\}$

Gradient descent algorithm:

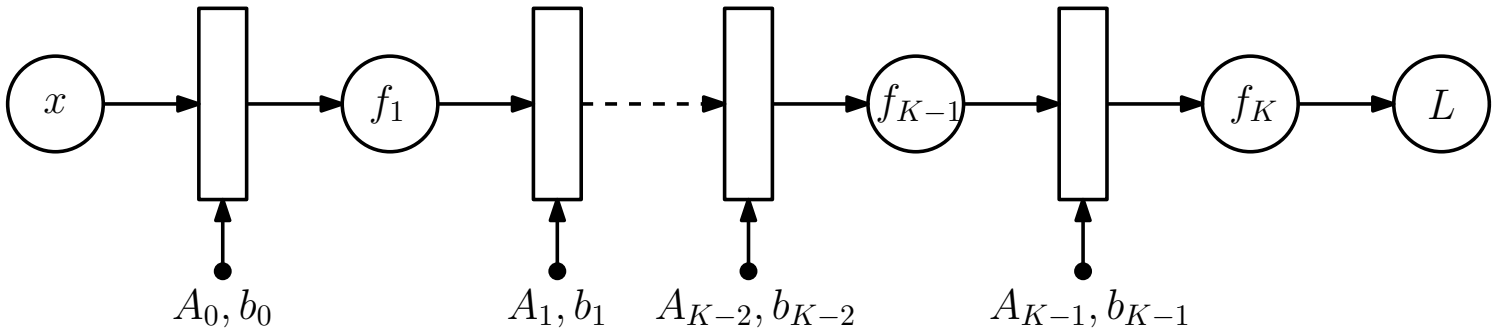
1. Initialise neural network parameters, $\boldsymbol{\theta}_0 = \{\boldsymbol{A}_0, \boldsymbol{b}_0\}$
2. Compute gradient of L w.r.t. parameters, $\frac{\partial L}{\partial \boldsymbol{\theta}_0}$
3. Update initial guess $\boldsymbol{\theta}_0$ using chain rule:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \Delta\boldsymbol{\theta}_i = \boldsymbol{\theta}_i - \gamma(\frac{\partial L}{\partial \boldsymbol{\theta}_i}(\boldsymbol{\theta}_i))^\top$$

- Note: parameter update is the direction of steepest **de**scent

For suitable step-size γ, the sequence $L(\boldsymbol{\theta}_0) \geqslant L(\boldsymbol{\theta}_1) \geqslant \cdots$ converges to a local minimum $L(\boldsymbol{\theta}^*)$.

Gradients of a Multi-Layer Neural Network

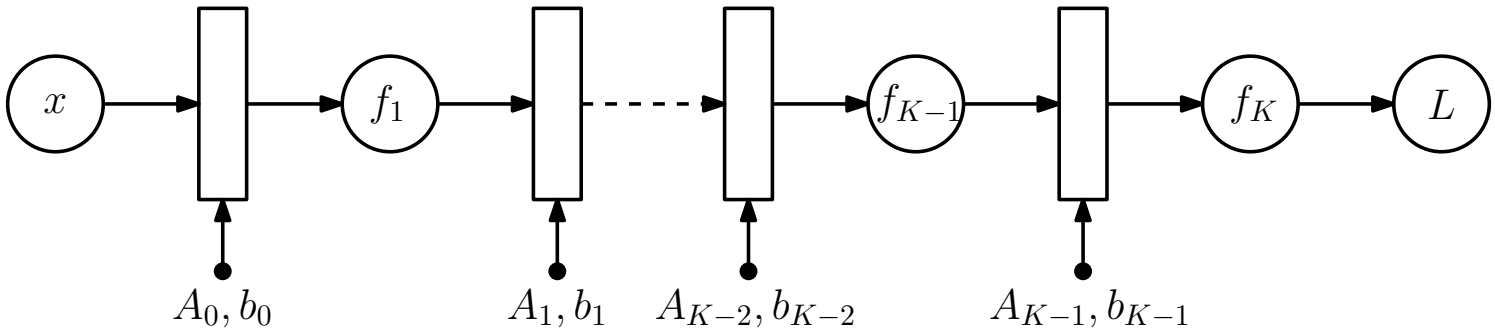


- Inputs $\boldsymbol{x}$, observed outputs $\boldsymbol{y}$
- Train K-layer neural network with

$$\boldsymbol{f}_0 = \boldsymbol{x}$$

$$\boldsymbol{f}_i = \sigma_i(\boldsymbol{A}_{i-1}\boldsymbol{f}_{i-1} + \boldsymbol{b}_{i-1}), \quad i = 1, \ldots, K$$

Gradients of a Multi-Layer Neural Network



- Inputs $\boldsymbol{x}$, observed outputs $\boldsymbol{y}$
- Train K-layer neural network with

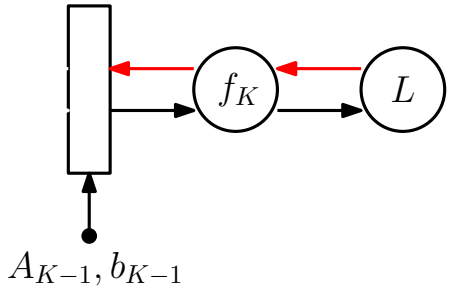
$$\boldsymbol{f}_0 = \boldsymbol{x}$$
$$\boldsymbol{f}_i = \sigma_i(\boldsymbol{A}_{i-1}\boldsymbol{f}_{i-1} + \boldsymbol{b}_{i-1}), \quad i = 1, \ldots, K$$

- Find $\boldsymbol{A}_k^*, \boldsymbol{b}_k^*$ for $k = 0, \ldots, K-1$, such that the squared loss

$$L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{f}_{K,\boldsymbol{\theta}}(\boldsymbol{x})\|^2$$

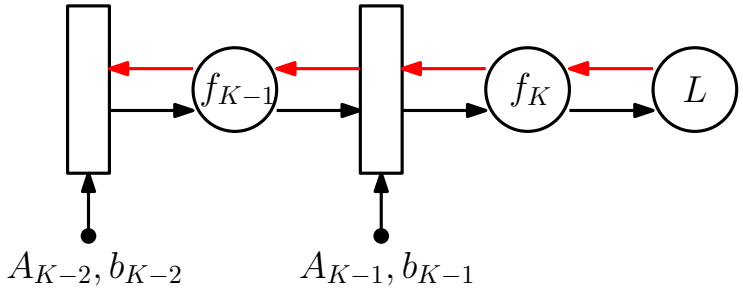
is minimised, where $\boldsymbol{\theta} = \{\boldsymbol{A}_k, \boldsymbol{b}_k\}, \quad k = 0, \ldots, K-1$

Gradients of a Multi-Layer Neural Network



$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}}$$

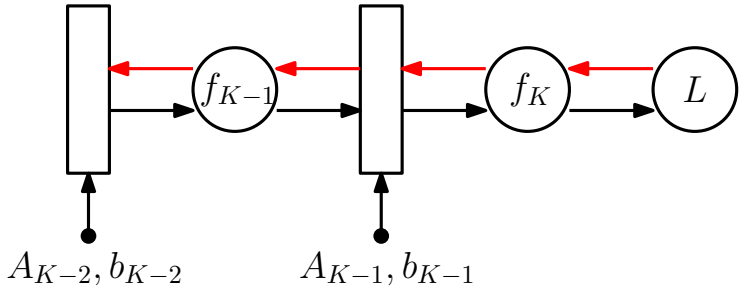
Gradients of a Multi-Layer Neural Network



$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}}$$

Gradients of a Multi-Layer Neural Network

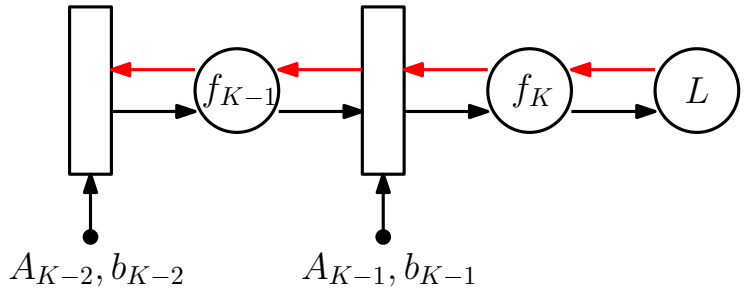


$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \boxed{\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{f}_{K-2}} \frac{\partial \boldsymbol{f}_{K-2}}{\partial \boldsymbol{\theta}_{K-3}}}$$

Gradients of a Multi-Layer Neural Network



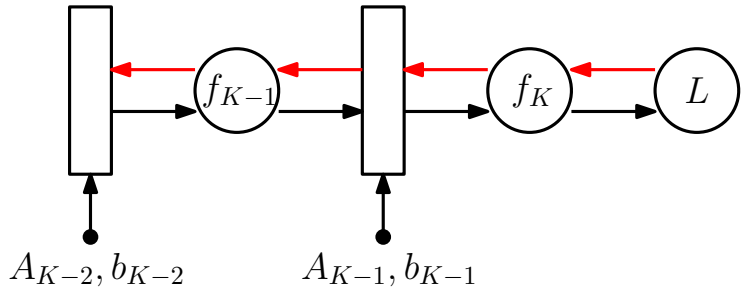
$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \boxed{\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{f}_{K-2}} \frac{\partial \boldsymbol{f}_{K-2}}{\partial \boldsymbol{\theta}_{K-3}}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_i} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \cdots \boxed{\frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}} \frac{\partial \boldsymbol{f}_{i+1}}{\partial \boldsymbol{\theta}_i}}$$

Gradients of a Multi-Layer Neural Network



$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \boxed{\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{f}_{K-2}} \frac{\partial \boldsymbol{f}_{K-2}}{\partial \boldsymbol{\theta}_{K-3}}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_i} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \cdots \boxed{\frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}} \frac{\partial \boldsymbol{f}_{i+1}}{\partial \boldsymbol{\theta}_i}}$$

- Intermediate derivatives are stored during the forward pass

Training NNs as Maximum Likelihood Estimation

- $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- **Goal**: Maximise log-likelihood of training data $(\boldsymbol{X}, \boldsymbol{Y})$:
  $\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X})$

Training NNs as Maximum Likelihood Estimation

- $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- **Goal**: Maximise log-likelihood of training data $(\boldsymbol{X}, \boldsymbol{Y})$:
  $\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_n|\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \mathcal{N}(\boldsymbol{y}_n = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{I})$

Training NNs as Maximum Likelihood Estimation

- $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- **Goal**: Maximise log-likelihood of training data $(\boldsymbol{X}, \boldsymbol{Y})$:
  $\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_n|\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \mathcal{N}(\boldsymbol{y}_n = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{I})$
- Then:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_n|\boldsymbol{x}_n) = \frac{N}{2} \log(2\pi) - \frac{1}{2}||\boldsymbol{y} - f_{\boldsymbol{\theta}}(\boldsymbol{x})||^2$$

Training NNs as Maximum Likelihood Estimation

- $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- **Goal**: Maximise log-likelihood of training data $(\boldsymbol{X}, \boldsymbol{Y})$:
  $\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_n|\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \mathcal{N}(\boldsymbol{y}_n = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{I})$
- Then:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_n|\boldsymbol{x}_n) = \tfrac{N}{2}\log(2\pi) - \tfrac{1}{2}||\boldsymbol{y} - f_{\boldsymbol{\theta}}(\boldsymbol{x})||^2$$

- Find $\boldsymbol{\theta}^*$ by minimising the negative log-likelihood:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}[-\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X})] = \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{y} - f_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2$$

$$= \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

⏩ **Equivalent to minimising loss of neural network**

Training NNs as Maximum Likelihood Estimation

- $\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- **Goal**: Maximise log-likelihood of training data $(\boldsymbol{X}, \boldsymbol{Y})$:
  $\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}) = \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_n|\boldsymbol{x}_n) = \sum_{n=1}^{N} \log \mathcal{N}(\boldsymbol{y}_n = f_{\boldsymbol{\theta}}(\boldsymbol{x}_n), \boldsymbol{I})$
- Then:
  $$\log p_{\boldsymbol{\theta}}(\boldsymbol{y}_n|\boldsymbol{x}_n) = \tfrac{N}{2} \log(2\pi) - \tfrac{1}{2}||\boldsymbol{y} - f_{\boldsymbol{\theta}}(\boldsymbol{x})||^2$$
- Find $\boldsymbol{\theta}^*$ by minimising the negative log-likelihood:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} [-\log p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X})] = \arg\min_{\boldsymbol{\theta}} \tfrac{1}{2} \sum_{n=1}^{N} \|\boldsymbol{y} - f_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2$$

$$= \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

⏩ **Equivalent to minimising loss of neural network**

- ML estimation can lead to overfitting (interprets noise as signal)

Example: Linear Regression with Polynomials

- Linear regression with a polynomial of order M:

$$y = f_{\boldsymbol{\theta}}(x) + \epsilon\,, \quad \epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$$

$$f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_M x^M = \sum_{i=0}^{M} \theta_i x^i$$

Example: Linear Regression with Polynomials

- Linear regression with a polynomial of order M:

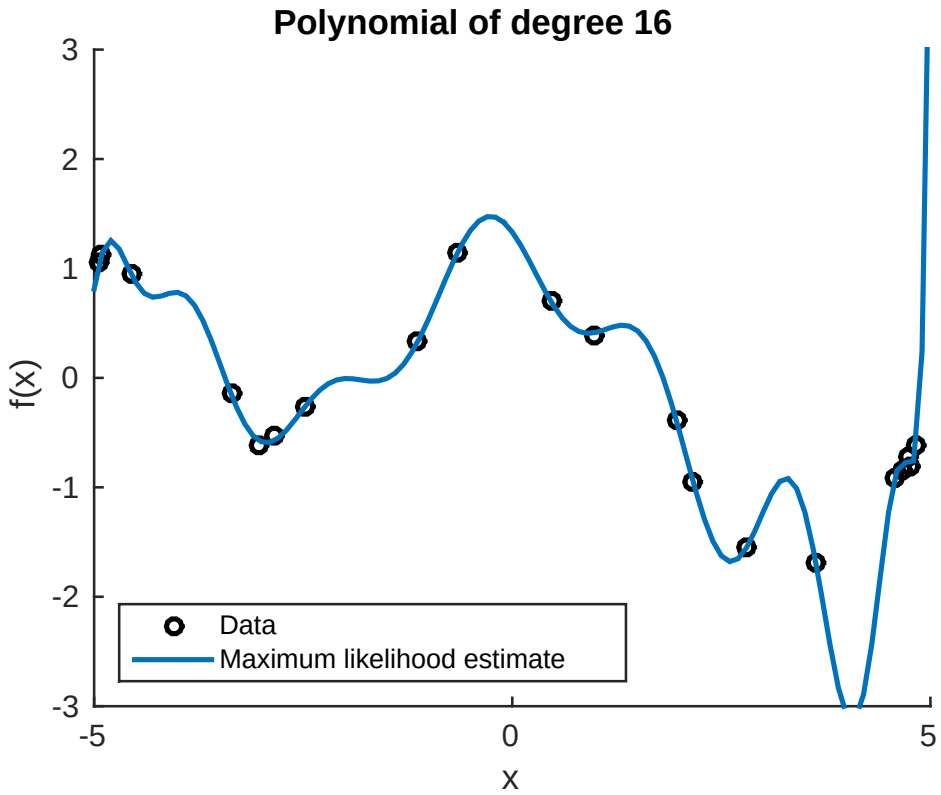
$$y = f_{\boldsymbol{\theta}}(x) + \epsilon\,, \quad \epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$$

$$f_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_M x^M = \sum_{i=0}^{M} \theta_i x^i$$

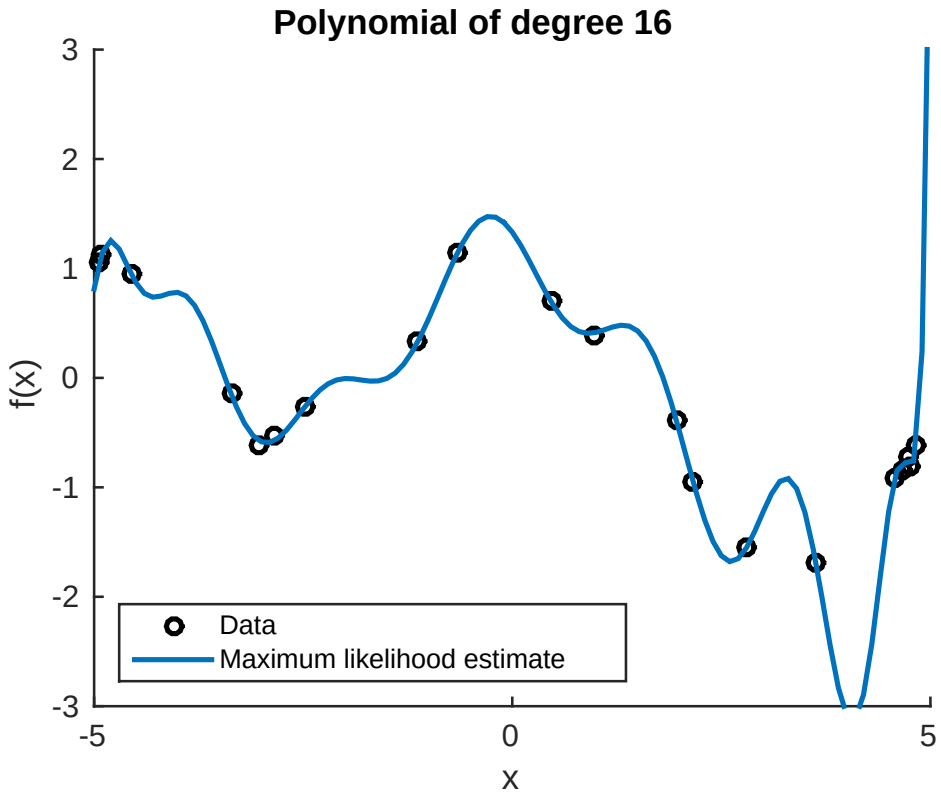
- Given inputs x_i and corresponding (noisy) observations y_i, $i = 1, \ldots, N$, find parameters $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_M]^\top$ that minimise the squared loss (equivalently: maximise the likelihood)

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{N} (y_i - f_{\boldsymbol{\theta}}(x_i))^2$$

Example: Linear Regression with Polynomials

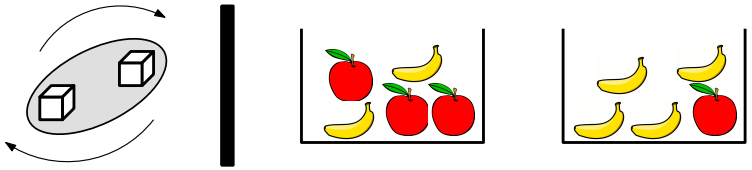


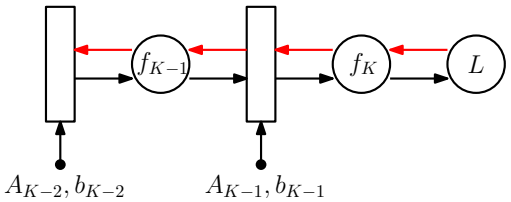
Example: Linear Regression with Polynomials

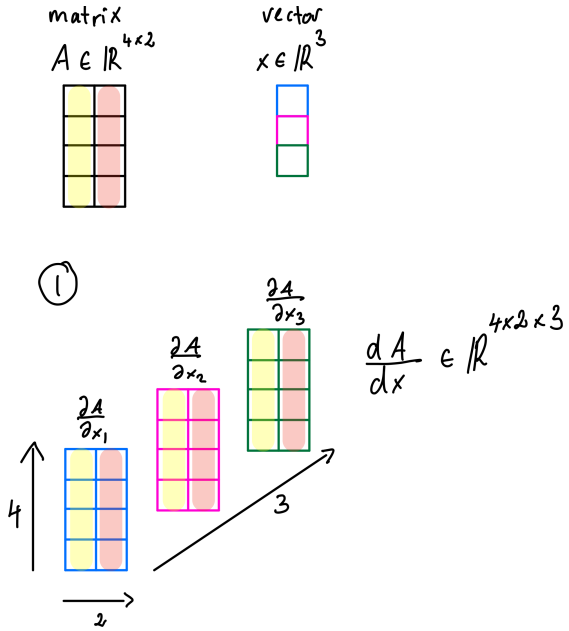


- Overfitting!
- Regularization, model selection etc. necessary
  - ⏩ Tutorials later this week

Summary







- Key concepts in probability - PDFs, Sum & Product Rules, Bayes
- **Differentiation** for optimising parameters of ML models
  - Vector calculus
  - Chain rule
  - Single- & Multi-layer NNs
  - Maximum Likelihood Estimation Equivalence

Useful Resources

- Indaba 2017 slides/videos
- Textbooks:
  - Mathematics for ML - Marc Deisenroth, Aldo Faisal, Cheng Soon Ong
  - Deep Learning - Yoshua Bengio, Ian Goodfellow, Aaron Courville
  - Matrix Cookbook - Kaare Petersen, Michael Pedersen
- Online courses:
  - fast.ai Deep Learning by Jeremy Howard
  - deeplearning.ai/coursera Deep Learning Specialisation by Andrew Ng
  - Udacity Machine Learning Nanodegree Programs
  - Coursera Machine Learning by Stanford University
- Blogs, podcasts:
  - Reddit ML, KDnuggets, TWiML&AI, Talking Machines

More textbooks, online courses, papers, datasets and other resources at `github.com/ChristosChristofidis/awesome-deep-learning`

Rest of the week...

Monday

Deep Learning Fundamentals - Moustapha Cisse
Convolutional Models - Naila Murray

Tuesday

Probabilistic Thinking - Yabebal Fantaye
Recurrent Neural Networks - Kyunghyun Cho

Wednesday

Generative Models - Asja Fischer
Reinforcement Learning - Katja Hofmann

Thursday - Parallel Sessions

Computer Vision, Non-Recurrent Sequence Models, AI for Africa, Generative Modelling and Healthcare, Natural Language Processing, Life of a ML Start-up

Friday - More Parallel Sessions

Reinforcement Learning, ML in Production, AI Ethics & Policy


Learn. Connect. Strengthen.
Enjoy.
DEEP
LEARNING
INDABA
Stellenbosch University
9th September 2018