Applied Deep Learning

Dr. Philippe Blaettchen
Bayes Business School (formerly Cass)

www.bayes.city.ac.uk

Learning objectives of today

Goals: Understand the key concepts of linear algebra and calculus relevant to deep learning

- Basic definitions
- Taking derivatives
- Typical operations on vectors and matrices

How will we do this?

- Pick up where the videos left off
- Not a comprehensive review, but focusing on the most relevant concepts for understanding deep learning
- Introduction how to implement concepts in Python
- Building our very own logistic regression algorithm (the simplest neural network)

Why are we even doing this?

- Straight up: this will be the most tedious class for most of you
- However, deep learning, at the very core, functions with fundamental linear algebra operations and gradient descent algorithms (calculus!)
- Hence, we need to learn the gist of these concepts in order to:
    - Understand what is happening “behind the scenes” of neural networks
    - Build intuition about how models can be adjusted and improved – even many of the out-of-the-box tools to tune your networks don’t make sense if you don’t know what a gradient descent algorithm is
    - Create confidence in handling ultra-high dimensional data with millions of observations without having to rely on visual inspection
    - …

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Linear algebra – definitions

Scalars, Vectors, Matrices, and Tensors

- Scalar: a single number
  - Integers (-1,0,1,2,…), real numbers (0.319375, 1.17, π), rational numbers $\left(\frac{integer}{integer}\right)$

Scalars, Vectors, Matrices, and Tensors

- Scalar: a single number
  - Integers (-1,0,1,2,…), real numbers (0.319375, 1.17, π), rational numbers $\left(\frac{integer}{integer}\right)$

- Vector: One-dimensional array of scalars

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  - Entries can be real numbers, binary, integer, …

Scalars, Vectors, Matrices, and Tensors

- Scalar: a single number
  - Integers (-1,0,1,2,…), real numbers (0.319375, 1.17, π), rational numbers $\left(\frac{integer}{integer}\right)$
- Vector: One-dimensional array of scalars

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  - Entries can be real numbers, binary, integer, …
- Matrix: Two-dimensional array of numbers

$$\begin{array}{cc} & \text{Row} \\ \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}. & \\ \text{Column} & \end{array}$$

Scalars, Vectors, Matrices, and Tensors

- Scalar: a single number
  - Integers (-1,0,1,2,…), real numbers (0.319375, 1.17, π), rational numbers $\left(\frac{integer}{integer}\right)$

- Vector: One-dimensional array of scalars

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  - Entries can be real numbers, binary, integer, …

- Matrix: Two-dimensional array of numbers

Row

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}.$$

Column

- Tensor: Any array of numbers
  - Could have zero dimensions (scalar), one dimension (vector), two dimensions (matrix)
  - Could also have three or more dimensions

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

A typical use of matrices in deep learning

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & \boxed{a_{nm}} \end{bmatrix} \longleftrightarrow$$

Pixel's intensity value

Source: Ivanovic

In Python

- We generally use Numpy to work with vectors, matrices, and sometimes tensors
- Later, we'll also see the TensorFlow-specific implementation of tensors

Linear algebra – typical operations

Matrix transpose

- Essentially a mirror image across the main diagonal

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \boldsymbol{A}^\top = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

$$(\boldsymbol{A}^\top)_{i,j} = A_{j,i}.$$

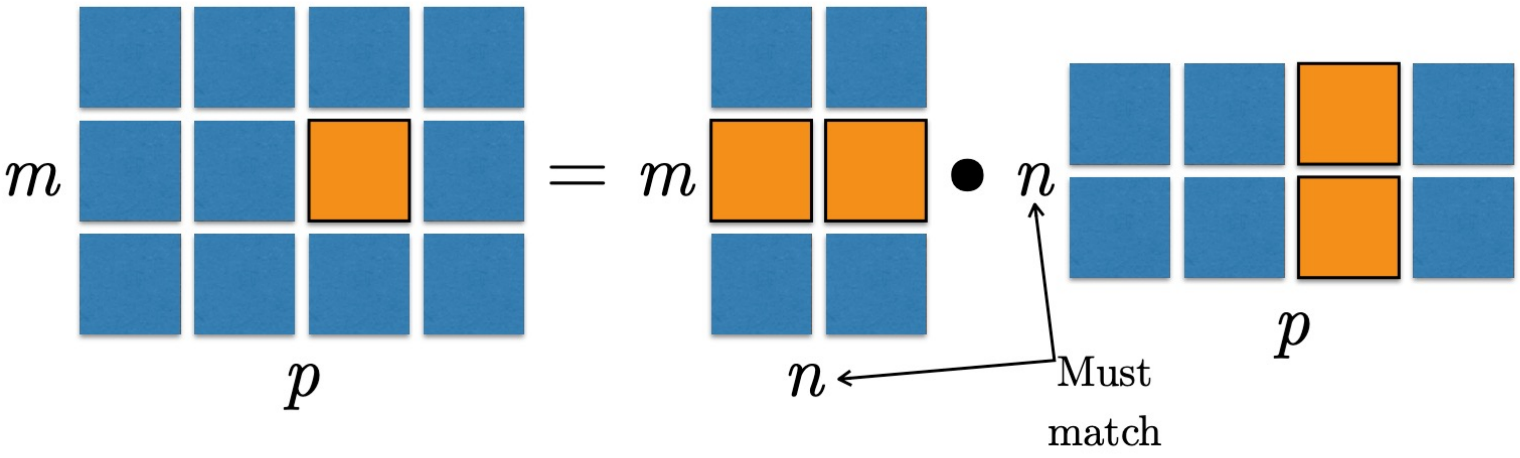
- We call a matrix symmetric if $\boldsymbol{A} = \boldsymbol{A}^T$

Source: Goodfellow

Matrix multiplication

$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}.$$

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}.$$



Source: Goodfellow

In Python

Linear algebra – norms

Norms

- Functions f that measures the “length” of a vector $\boldsymbol{x}$

- Such functions need to fulfill four conditions:

  - f needs to return non-negative values only (there is no negative length!)
  - The only vector that has length zero should be the 0-vector
  - The length “scales”: For all $\boldsymbol{x} \in \mathbb{R}^n, \alpha \in \mathbb{R}$, $f(\alpha\boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$
  - The triangle inequality needs to hold: For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, f(\boldsymbol{x}+\boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Some commonly used norms

- L^p norm:
  - $\|\boldsymbol{x}\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$
  - Most commonly used norm: $\|\boldsymbol{x}\|_2 = \sqrt{\sum_i x_i^2}$ (L^2 norm or “Euclidian” norm)
  - Quite common as well: $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$ (L^1 norm)
- An extreme case: the max-norm $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Norms defined for matrices

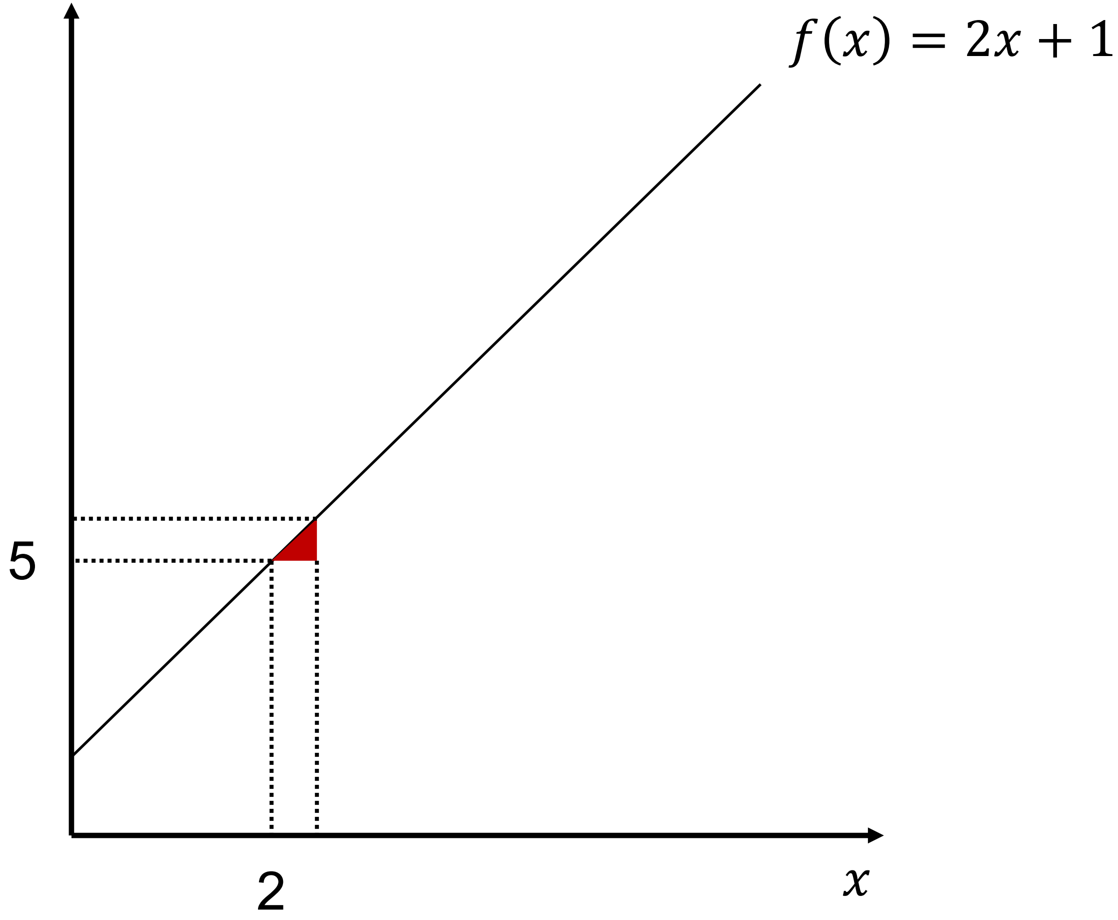
- There are many matrix norms, but we will only need one: the Frobenius norm

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} A_{i,j}^2}$$

- Note the similarity with the L^2 norm for vectors

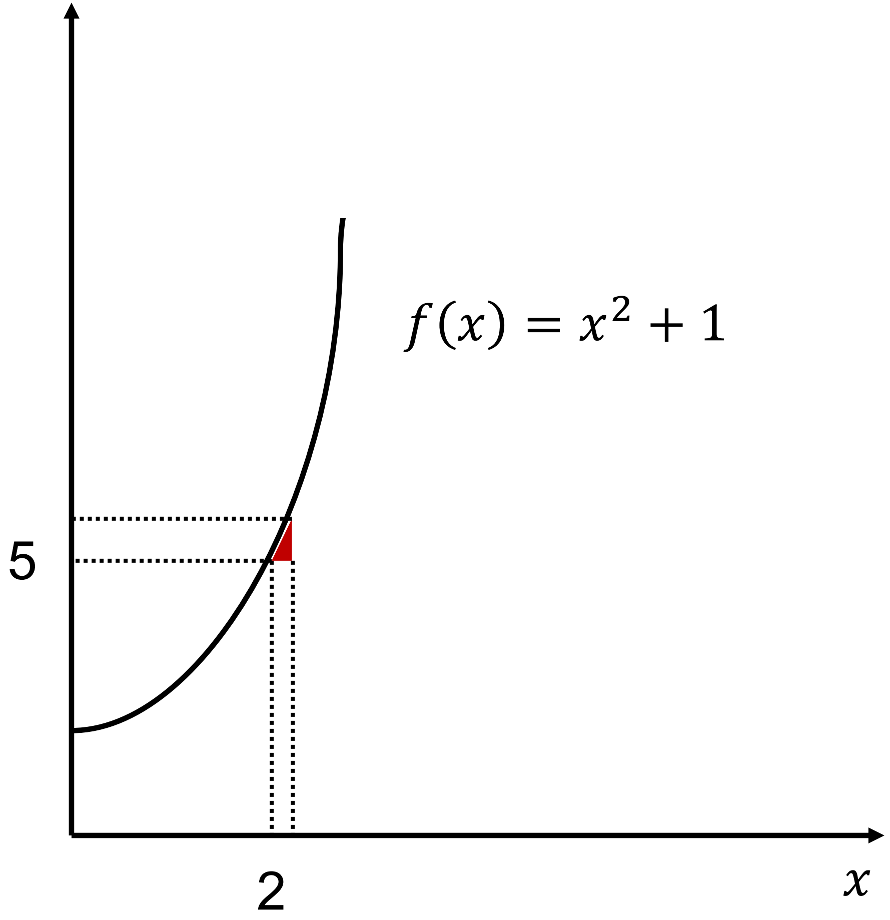
A refresher on calculus

What is a derivative?



Slope (derivative) of $f(x)$ at 2 is $\frac{\text{“}height\text{”}}{\text{“}width\text{”}}$

What is a derivative?



Slope (derivative) of $f(x)$ at 2 is $\frac{\text{“}height\text{”}}{\text{“}width\text{”}}$

A few important derivatives

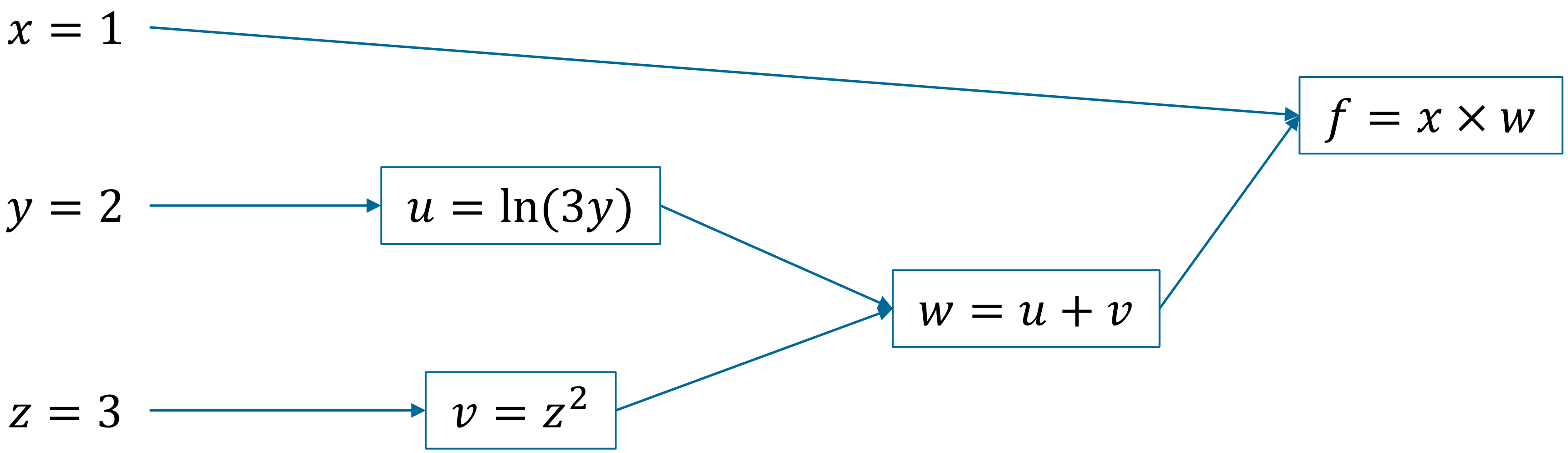
$f(x)$	$f'(x) = \frac{df(x)}{dx} = \frac{d}{dx}f(x)$
1	
x	
x^2	
x^3	
$\sqrt{x}$	
$\ln(x)$	
e^x	

Some rules about handling derivatives

$h(x)$	$h'(x)$	Example
$c\, f(x)$	$c\, f'(x)$	$h(x) = 18\, x^k$
$f(x) + g(x)$	$f'(x) + g'(x)$	$h(x) = \log(x) - x^2 + 5$
$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$	$h(x) = 2e^x x$
$\dfrac{1}{f(x)}$	$-\dfrac{f'(x)}{f(x)^2}$	$h(x) = \dfrac{1}{\ln(x)}$
$\dfrac{f(x)}{g(x)}$	$\dfrac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$	$h(x) = \dfrac{\ln(x)}{x^2}$
$f\big(g(x)\big)$	$f'\big(g(x)\big)g'(x)$	$h(x) = e^{x^2}$

Using a computation graph for derivatives

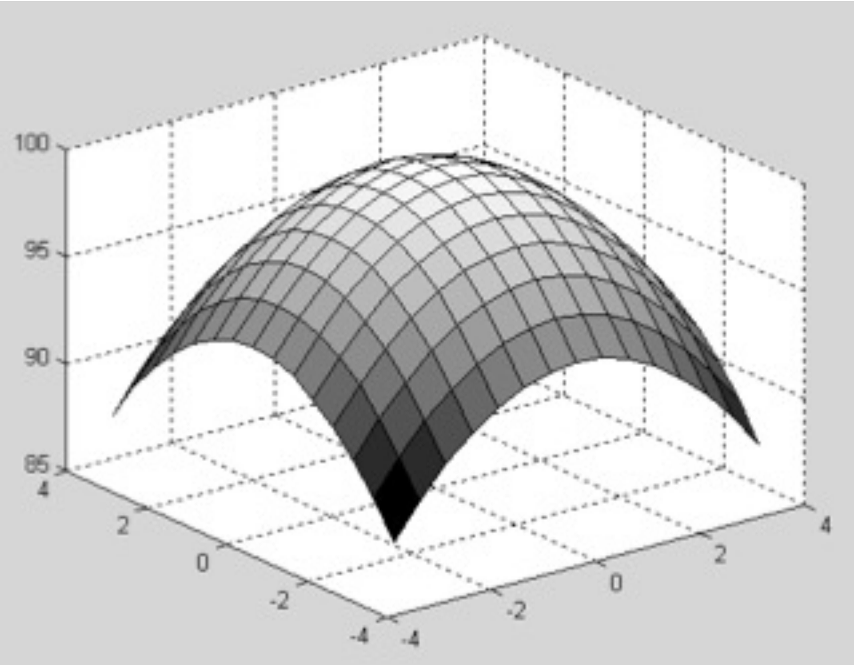
- We can use the “chain rule” to simplify the search for derivatives
- Say, we want to compute the derivatives of $f = x(\ln(3y) + z^2)$ to x, y, z at $x = 1, y = 2, z = 3$

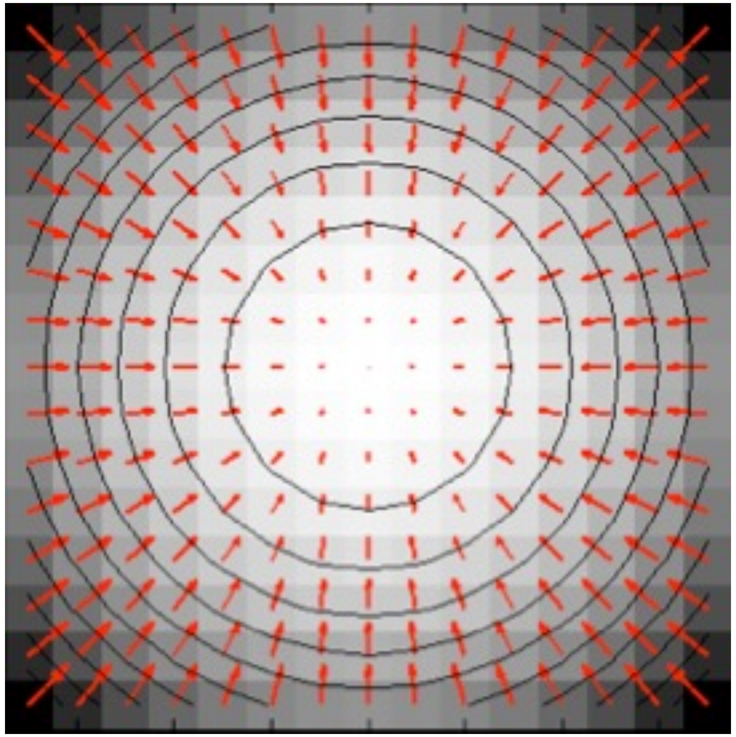


BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Putting it together – gradient descent

The gradient





Source: Collins

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

What are we seeing here?

- Say, we have a function f, taking as input a vector $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

- The gradient (with respect to $\boldsymbol{x}$) is the vector pointing in the direction of fastest increase:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix}$$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

This naturally extends to functions that take as input a matrix

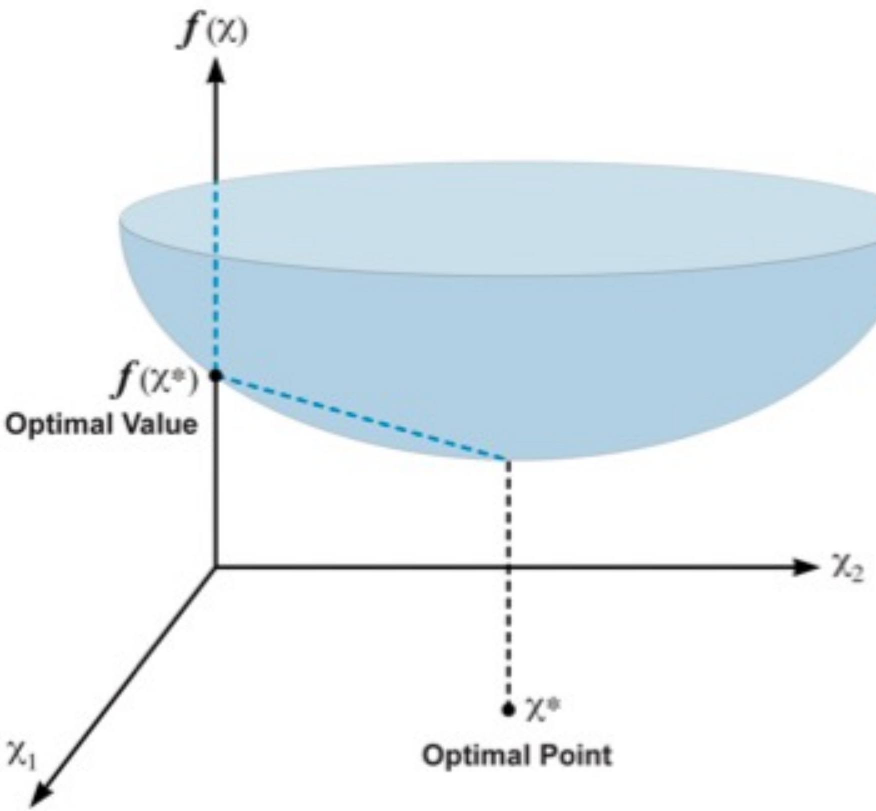
- Say, now we have a function f, taking as input a matrix $\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix}$

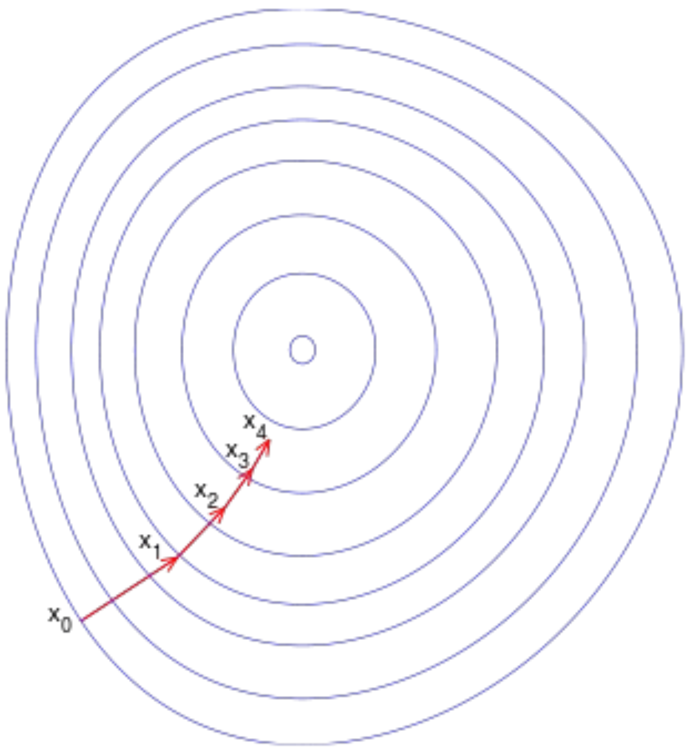
- The gradient (with respect to $\boldsymbol{A}$) is now also a matrix $\nabla_{\boldsymbol{A}} f(\boldsymbol{A}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{A})}{\partial a_{1,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{1,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{1,m}} \\ \frac{\partial f(\boldsymbol{A})}{\partial a_{2,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{2,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{2,m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(\boldsymbol{A})}{\partial a_{n,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{n,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{n,m}} \end{bmatrix}$

Why do we need the gradient? A thought experiment:

- You wake up somewhere in the mountain
- Your goal is to reach the lowest point as quickly as possible
- You have no map, GPS, and can only see a few meters ahead because of trees and fog
- How do you do it?

Gradient descent – the idea





The algorithm

- Take small steps
- For each step, go downhill in the locally steepest descent direction
- Repeat until you are on a flat surface

Taking a step back – logistic regression

What do we actually do when training a logistic regression model?

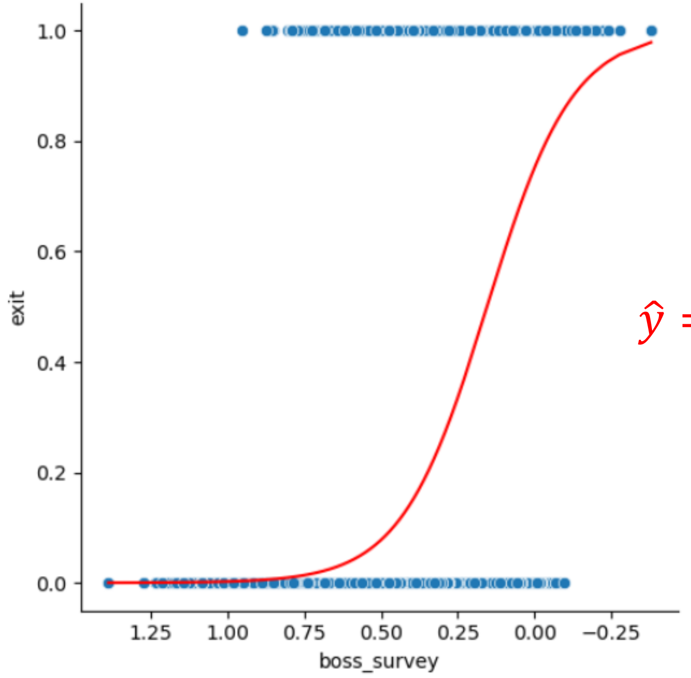
- We are given values $(\boldsymbol{x}^{(i)}, y^{(i)})$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \{0,1\}$
- Our prediction $\hat{y}^{(i)}$ should reflect the probability that $y^{(i)} = 1$: $\hat{y}^{(i)} = P\left(y^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}\right)$

- We model this probability, using the sigmoid function:

What do we actually do when training a logistic regression model?

- We are given values $(\boldsymbol{x}^{(i)}, y^{(i)})$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \{0,1\}$
- Our prediction $\hat{y}^{(i)}$ should reflect the probability that $y^{(i)} = 1$: $\hat{y}^{(i)} = P\left(y^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}\right)$

- We model this probability, using the sigmoid function:



$$\hat{y} = \frac{1}{1 + e^{-(\boldsymbol{x} \cdot \boldsymbol{w} + \boldsymbol{b})}}$$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

The optimization part

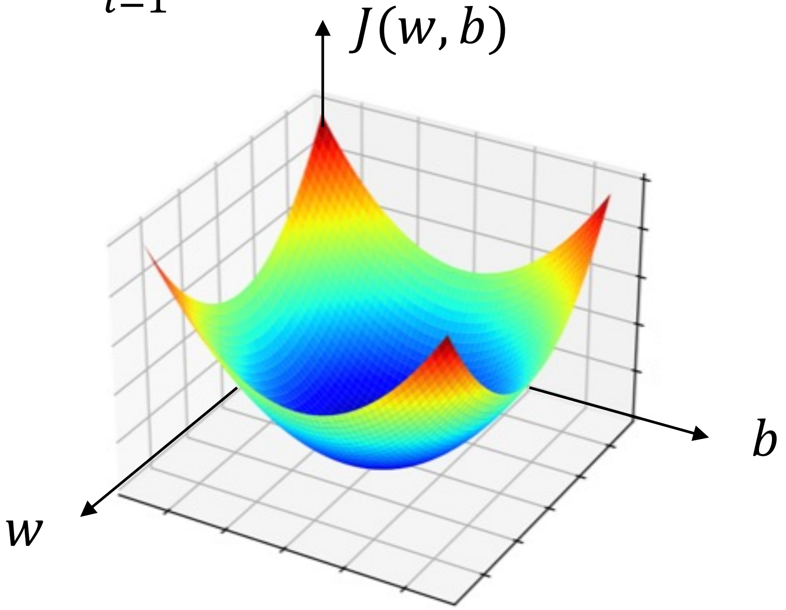
- Remember that $\boldsymbol{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$
- To get to the "right" model, we optimize our parameters $\boldsymbol{w}, b$ so that the $\hat{y}^{(i)}$s are "as close as possible" to the y^is
- What we do is to minimize the "cost-function" $J(\boldsymbol{w}, b)$, where $\hat{y}^{(i)} = \frac{1}{1+e^{-\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+\boldsymbol{b}\right)}}$:

$$J(\boldsymbol{w}, b) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} \ln \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \ln\left(1 - \hat{y}^{(i)}\right)\right]$$

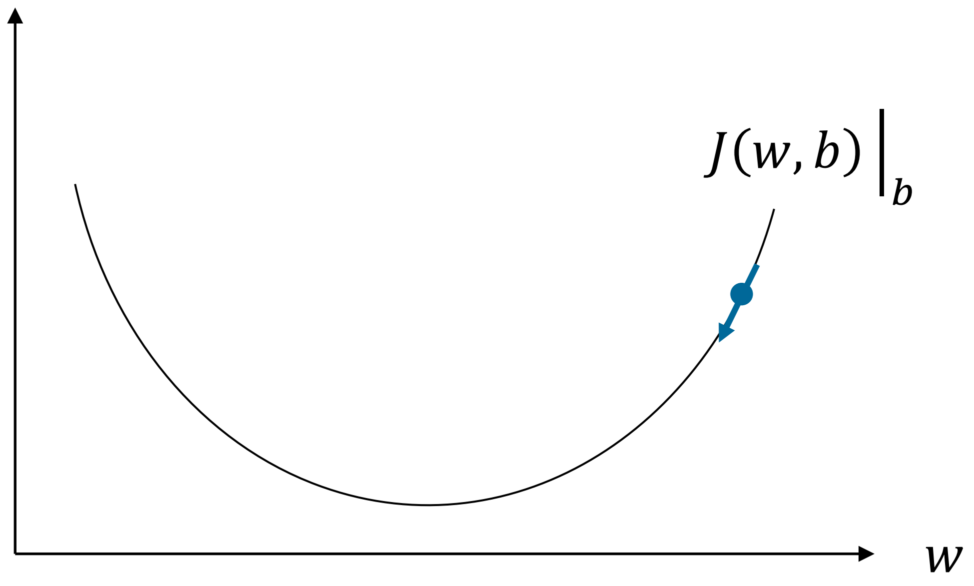
The optimization part

- Remember that $\boldsymbol{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$
- To get to the "right" model, we optimize our parameters $\boldsymbol{w}, b$ so that the $\hat{y}^{(i)}$s are "as close as possible" to the y^is
- What we do is to minimize the "cost-function" $J(\boldsymbol{w}, b)$, where $\hat{y}^{(i)} = \frac{1}{1+e^{-\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+\boldsymbol{b}\right)}}$:

$$J(\boldsymbol{w}, b) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} \ln \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \ln\left(1 - \hat{y}^{(i)}\right)\right]$$



Solving the optimization problem through gradient descent



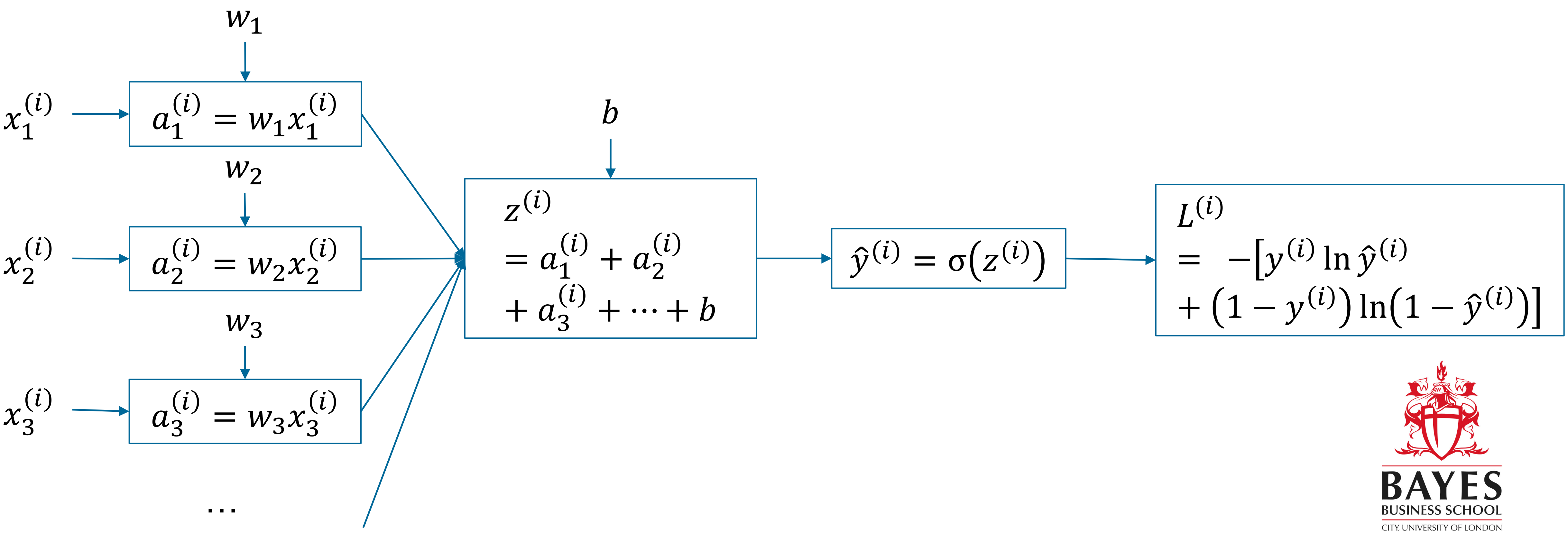
Our first optimization algorithm

1. Decide a “learning rate” α
2. Start with some $\boldsymbol{w}$ and b and compute $J(\boldsymbol{w}, b)$
3. Until J “doesn’t change” anymore:
   - Let $w_1 := w_1 - \alpha \frac{\partial J(\boldsymbol{w},b)}{\partial w_1}$
   - Let $w_2 := w_2 - \alpha \frac{\partial J(\boldsymbol{w},b)}{\partial w_2}$
   - …
   - Let $w_m := w_m - \alpha \frac{\partial J(\boldsymbol{w},b)}{\partial w_m}$
   - Let $b := b - \alpha \frac{\partial J(\boldsymbol{w},b)}{\partial b}$
   - Recompute $J(\boldsymbol{w}, b)$
4. Enjoy the fruits of your labor: you have fit a logistic regression model manually!

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Wait a second, how do we find all those derivatives?

- We can use again the computation graph!
- Recall that $\hat{y}^{(i)} = \frac{1}{1+e^{-\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+\boldsymbol{b}\right)}} = \sigma\left(\boldsymbol{x}^{(i)}\boldsymbol{w} + b\right)$

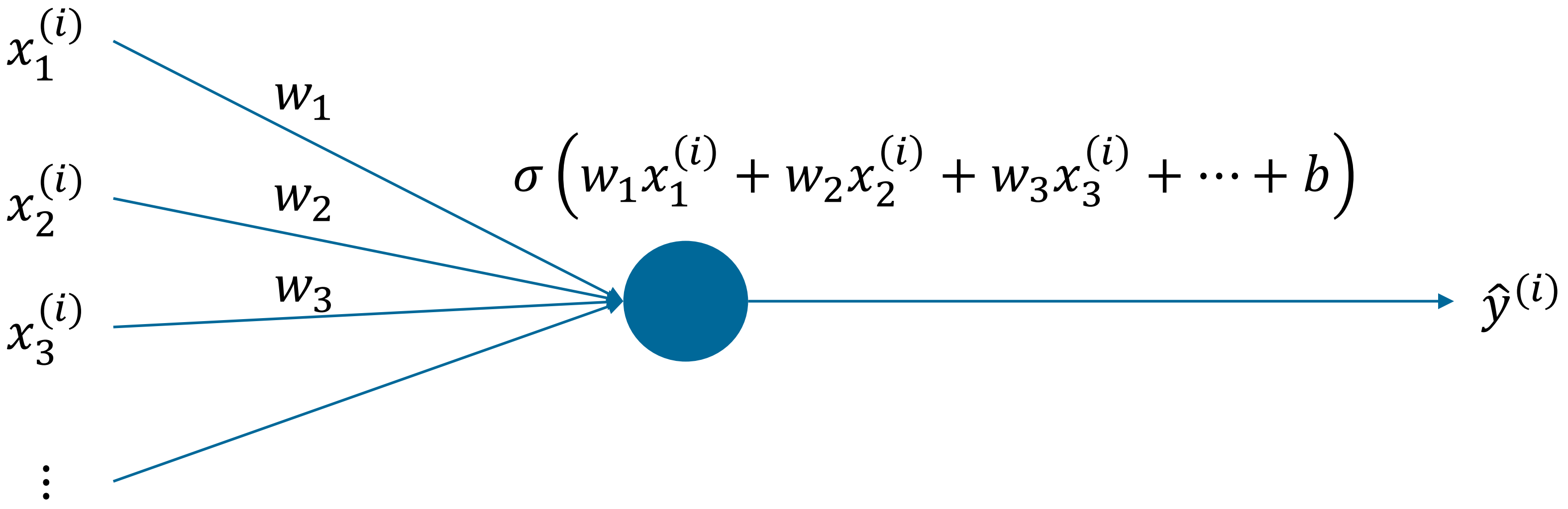


As the same parameters influence all examples, we have to consider one final step

- Recall that $J(\boldsymbol{w}, b) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} \ln \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \ln\left(1 - \hat{y}^{(i)}\right)\right] = \frac{1}{n}\sum_{i=1}^{n} L^{(i)}$
- We have that $\frac{\partial J(\boldsymbol{w},b)}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L^{(i)}}{\partial w_j}$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Schema of a logistic regression



We can now implement a logistic regression

See you next week!

Sources

- Bhaskhar, 2021, Linear Algebra: https://cs229.stanford.edu/notes2021fall/section1notes-linear-algebra-review.pdf
- Collins, 2012, Intensity Surfaces and Gradients: http://www.cse.psu.edu/~rtc12/CSE486/lecture02_6pp.pdf
- Goodfellow, Bengio, Courville, 2016, The Deep Learning Book: http://www.deeplearningbook.org
- Kolter, Do, & Ma, 2020, Linear Algebra Review and Reference: https://cs229.stanford.edu/summer2020/cs229-linalg.pdf
- Ivanovic, 2017, Python Introduction and Linear Algebra Review: https://web.stanford.edu/class/cs231a/section/section1.pdf

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON