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 on how to implement concepts in Python

Linear algebra – definitions

Scalars

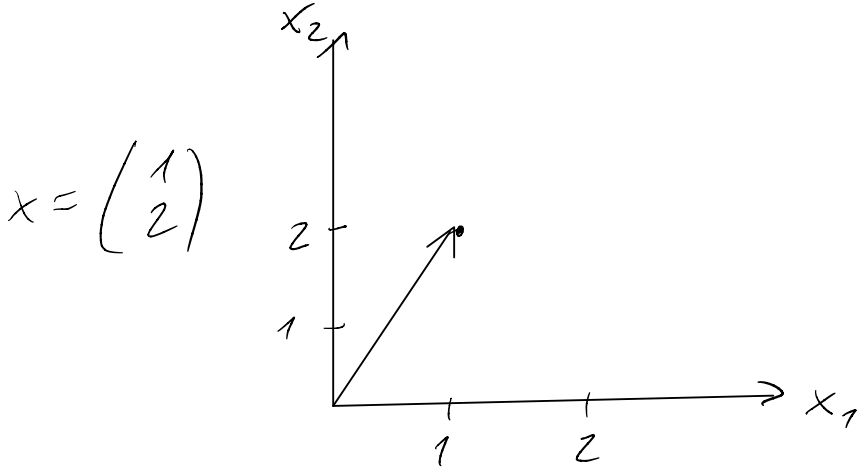
- Scalar: a single number
- Integers (-1,0,1,2,…), real numbers (0.319375, 1.17, π), rational numbers $\left(\frac{integer}{integer}\right)$

$\alpha, t \qquad \alpha \in \mathbb{R}, \; t \in \mathbb{Z}$

Vectors

- A one-dimensional array of numbers:

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

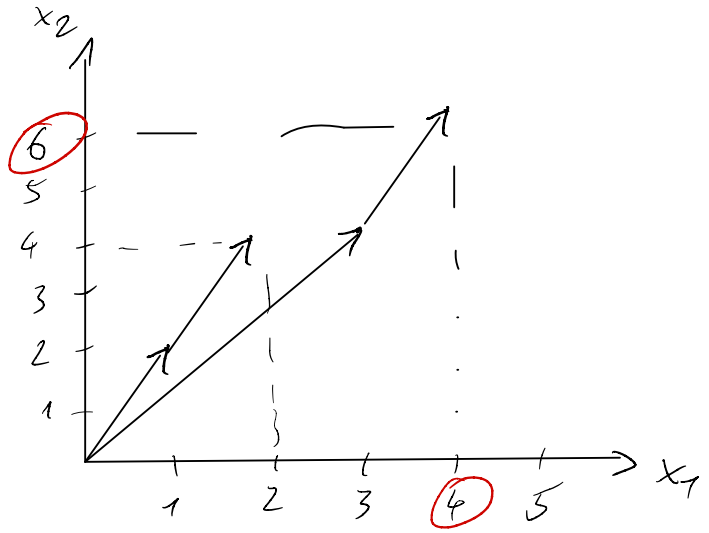


- Entries can be real numbers, binary, integer, …
  → we usually denote a vector with the type of its entries and its size, e.g., $\mathbf{x} \in \mathbb{R}^n$

Vectors

- A one-dimensional array of numbers:

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



- Entries can be real numbers, binary, integer, …
  → we usually denote a vector with the type of its entries and its size, e.g., $\mathbf{x} \in \mathbb{R}^n$

- We can perform elementary algebra on vectors:
  - $\boldsymbol{x} + \boldsymbol{y} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}$ (check dimensions!)
  - $\alpha \boldsymbol{x} = \alpha \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \alpha x_1 \\ \alpha x_2 \\ \vdots \\ \alpha x_n \end{bmatrix}$

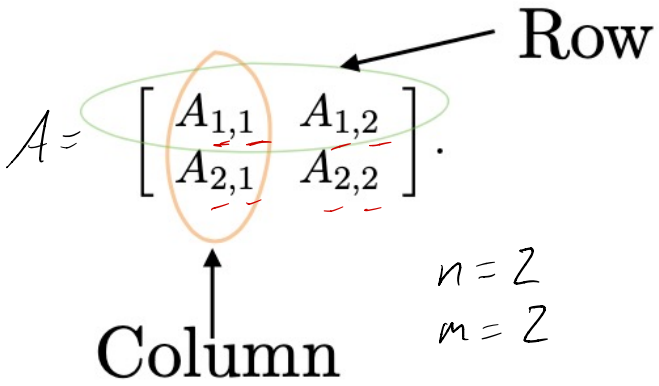
$x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ $y = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ $x + y = \begin{pmatrix} 1+3 \\ 2+4 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \end{pmatrix}$

$\alpha = 2$, $x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ $\alpha x = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Matrices

- A two-dimensional array of numbers

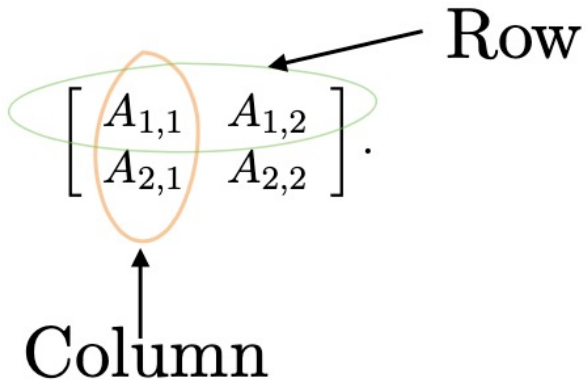


- We again denote it with its type and shape: $A \in \mathbb{R}^{n \times m}$, where n is the number of rows and m is the number of columns

Source: Goodfellow

Matrices

- A two-dimensional array of numbers



- We again denote it with its type and shape: $A \in \mathbb{R}^{n \times m}$, where n is the number of rows and m is the number of columns

$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, $B = \begin{pmatrix} 3 & 0 \\ 1 & 5 \end{pmatrix}$

- We can perform elementary algebra on ~~vectors~~ matrices:
  - $\boldsymbol{A} + \boldsymbol{B} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} + \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} \end{bmatrix}$ (check dimensions!)

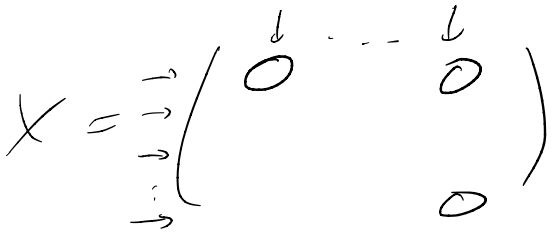
$A + B = \begin{pmatrix} 1+3 & 2+0 \\ 3+1 & 4+5 \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 4 & 9 \end{pmatrix}$

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 3 & 0 & 1 \\ 1 & 5 & 1 \end{pmatrix}$ ✗

  - $\alpha \boldsymbol{A} = \alpha \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} = \begin{bmatrix} \alpha a_{1,1} & \alpha a_{1,2} \\ \alpha a_{2,1} & \alpha a_{2,2} \end{bmatrix}$

Source: Goodfellow

A typical use of matrices in deep learning



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

Pixel's intensity value

Source: Ivanovic

Tensors

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

In Python

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

Linear algebra – typical operations

Trace of a matrix

- Summing up all the entries on the diagonal of the matrix:

$$Tr(\boldsymbol{A}) = \sum_{i=1}^{n} \boldsymbol{A}_{i,i}$$

$$A = \begin{pmatrix} 1 & 2 & 5 \\ 4 & 3 & 0 \\ 2 & 1 & 0 \end{pmatrix}$$

$$Tr(A) = 1 + 3 + 0 = 4$$

- Note that
  - $Tr(\boldsymbol{ABC}) = Tr(\boldsymbol{CAB}) = Tr(\boldsymbol{BCA})$
  - $Tr(\boldsymbol{A} + \boldsymbol{B}) = Tr(\boldsymbol{A}) + Tr(\boldsymbol{B})$

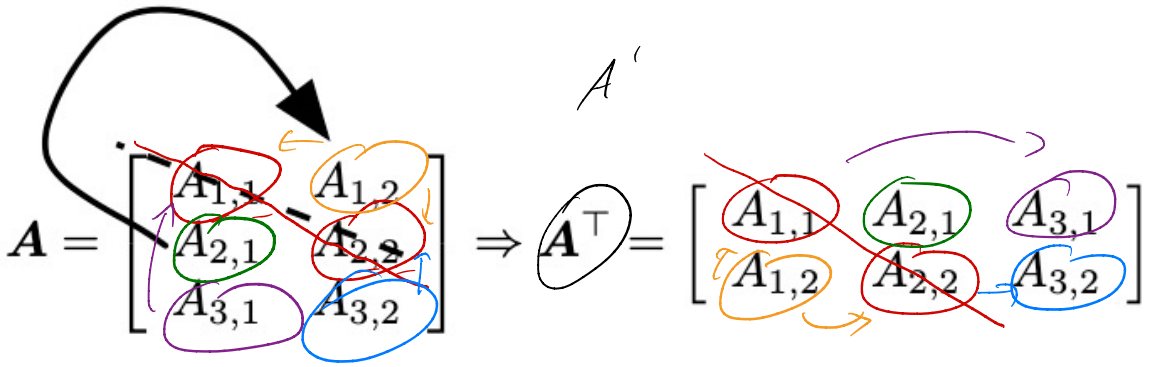
$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \quad A + B = \begin{pmatrix} 2 & 2 \\ 5 & 5 \end{pmatrix}$$

$$Tr(A) = 5 \quad Tr(B) = 2 \quad Tr(A+B) = 7$$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Matrix transpose

- Essentially a mirror image across the main diagonal



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

- Note that:
  - $(\boldsymbol{A}^T)^T = \boldsymbol{A}$
  - $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$
  - $(\boldsymbol{A}+\boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T$

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

Source: Goodfellow

Inner product of vectors (also known as dot product)

- Say $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, then:

$$\boldsymbol{x}^T\boldsymbol{y} = [x_1 \; x_2 \; \dots \; x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i$$

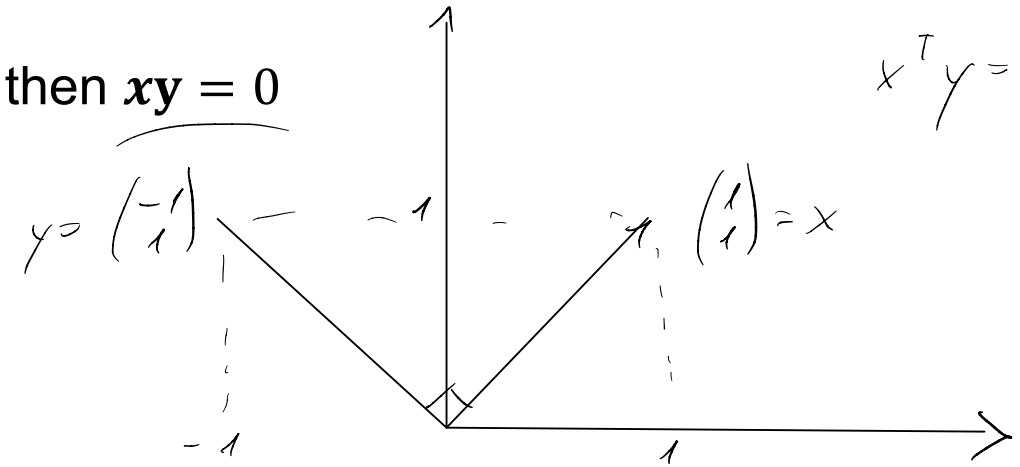
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$x = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad y = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}$$

$$x^T y = (1\;2\;3)\begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = 2 + 0 + 3 = 5$$

- The inner product is a scalar!
- Note: if $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal, then $\boldsymbol{xy} = 0$

$$x^T y = (1\;1)\begin{pmatrix} -1 \\ 1 \end{pmatrix} = -1 + 1 = 0$$



BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Outer product of vectors

- Say $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$ then:

$$\boldsymbol{x}\,\boldsymbol{y}^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} [y_1\ y_2\ \cdots\ y_m] = \begin{bmatrix} x_1y_1 & x_1y_2 & \cdots & x_1y_m \\ x_2y_1 & x_2y_2 & \cdots & x_2y_m \\ \vdots & \vdots & \ddots & \vdots \\ x_ny_1 & x_my_2 & \cdots & x_ny_m \end{bmatrix}$$

- The outer product is a matrix!

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Matrix-vector product

- Say $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ then:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{y} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{1,1}y_1 + a_{1,2}y_2 + \cdots a_{1,m}y_m \\ a_{2,1}y_1 + a_{2,2}y_2 + \cdots a_{2,m}y_m \\ \vdots \\ a_{n,1}y_1 + a_{n,2}y_2 + \cdots a_{n,m}y_m \end{bmatrix}$$

$$= \begin{bmatrix} A_{1,:}\,\boldsymbol{y} \\ A_{2,:}\,\boldsymbol{y} \\ \vdots \\ A_{n,:}\,\boldsymbol{y} \end{bmatrix}$$

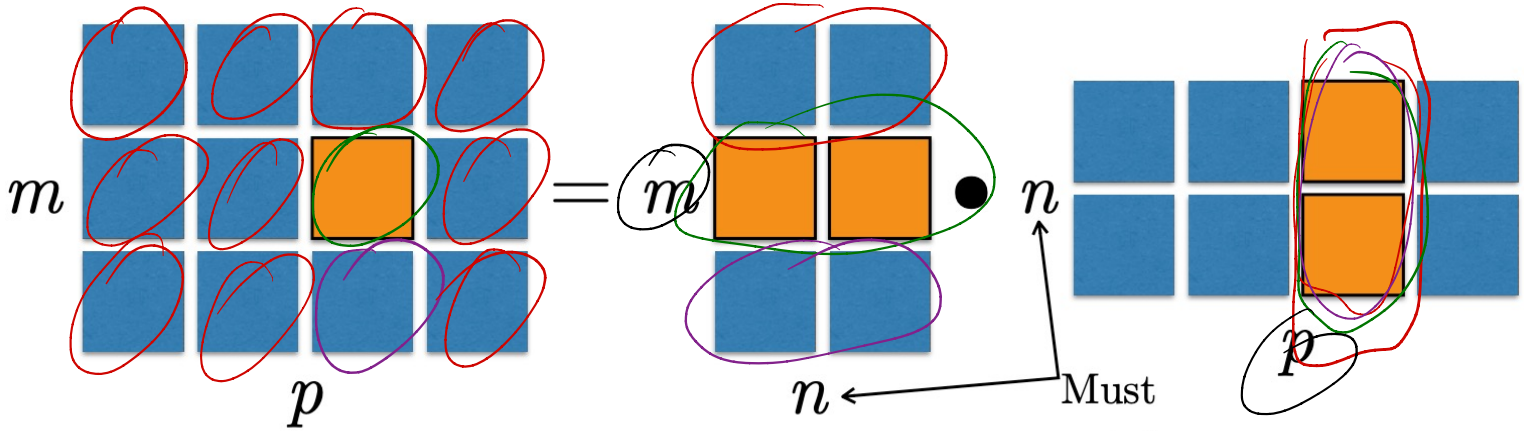
$$= A_{:,1}y_1 + A_{:,2}y_2 + \cdots + A_{:,m}y_m$$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Matrix multiplication

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

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



Source: Goodfellow

A few important properties of matrix multiplication

- Associative: $(\mathbf{AB})\mathbf{C} = \boldsymbol{A}(\boldsymbol{BC})$
- Distributive: $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \boldsymbol{AB} + \boldsymbol{AC}$
- Generally, not commutative: $\boldsymbol{AB} \neq \boldsymbol{BA}$
  (also, just because $\boldsymbol{AB}$ exists, it doesn't mean that $\boldsymbol{BA}$ exists)

$A \in \mathbb{R}^{n \times m}$ $\quad n \neq p$

$B \in \mathbb{R}^{m \times p}$

$\Rightarrow AB \in \mathbb{R}^{n \times p}$

$BA \notin \mathbb{R}^{m}$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Matrix inversion

- A matrix $\boldsymbol{A}^{-1}$ such that $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n$
- Here, $\boldsymbol{I}_n$ is the "identity matrix" of rank n. For example, $\boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  - Why identity matrix? Because $\boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{A}\,\boldsymbol{I}_n = \boldsymbol{A} = \boldsymbol{I}_n\boldsymbol{A}$

Matrix inversion

- A matrix $\boldsymbol{A}^{-1}$ such that $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n$
- Here, $\boldsymbol{I}_n$ is the "identity matrix" of rank n. For example, $\boldsymbol{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  - Why identity matrix? Because $\boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{A}\,\boldsymbol{I}_n = \boldsymbol{A} = \boldsymbol{I}_n\boldsymbol{A}$
- Note that $\boldsymbol{A}^{-1}$ only exists if the number of rows = the number of columns and both rows and columns are **linearly independent**
- A few properties (if the inverses exist):
  - $(\boldsymbol{A}^{-1})^{-1} = \boldsymbol{A}$
  - $(\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$
  - $(\boldsymbol{A}^{-1})^T = (\boldsymbol{A}^T)^{-1} =$ "$\boldsymbol{A}^{-T}$"
  - If $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, (or $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$) we say that $\boldsymbol{A}$ is an orthogonal matrix

Linear independence

- We say that vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m \in \mathbb{R}^n$ are linearly **dependent** if any of the vectors can be written as a linear combination of the others, e.g., if, for some scalars $\alpha_1, \alpha_2, \ldots, \alpha_m \in \mathbb{R}$,

$$\boldsymbol{x}_m = \sum_{j=1}^{m-1} \alpha_j \boldsymbol{x}_j$$

Linear independence

- We say that vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m \in \mathbb{R}^n$ are linearly **dependent** if any of the vectors can be written as a linear combination of the others, e.g., if, for some scalars $\alpha_1, \alpha_2, \ldots, \alpha_m \in \mathbb{R}$,

$$\boldsymbol{x}_m = \sum_{j=1}^{m-1} \alpha_j \boldsymbol{x}_j$$

- If this is **not** the case, we say that the vectors are linearly **independent**

- Going back to our matrix $\boldsymbol{A}$, we say that
  - the column rank is the largest number of columns that are linearly independent
  - the row rank is the largest number of rows that are linearly independent

- It turns out, row rank = column rank = "rank", so also $rank(\boldsymbol{A}) = rank(\boldsymbol{A}^T)$

- $\boldsymbol{A}$ is invertible if it is of "full rank" (that is, $rank(\boldsymbol{A}) = m = n$)

Using matrix inversions to solve systems of equations

- Say you are interested in finding the vector $\boldsymbol{x}$ that solves $\boldsymbol{Ax} = \boldsymbol{b}$
- This can also be written as a system of m equations:

$$\boldsymbol{A}_{1,:}\boldsymbol{x} = b_1$$

$$\boldsymbol{A}_{2,:}\boldsymbol{x} = b_2$$

$$\dots$$

$$\boldsymbol{A}_{m,:}\boldsymbol{x} = b_m$$

- This system can have
  - No solution
  - Many solutions
  - Exactly one solution (if $\boldsymbol{A}$ is invertible): $\boldsymbol{Ax} = \boldsymbol{b}$

$$\boldsymbol{A}^{-1}\boldsymbol{Ax} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

$$\boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

- Note: this way of solving the system is numerically unstable

Source: Goodfellow

Linear algebra – norms

Norms

- Functions f that measures the "length" of a vector $\boldsymbol{x}$
- Such functions need to fulfill four conditions:
  - $f(\boldsymbol{x}) \geq 0$
  - $f(\boldsymbol{x}) = 0 \Leftrightarrow \boldsymbol{x} = 0$
  - For all $\boldsymbol{x} \in \mathbb{R}^n, \alpha \in \mathbb{R}$, $f(\alpha \boldsymbol{x}) = |\alpha| f(\boldsymbol{x})$
  - For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ (triangle inequality)

Some commonly used norms

- L^p norm:
  - $\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$
  - 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 $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$
- For a given norm, we call the vector $\boldsymbol{x}$ with $\|\boldsymbol{x}\| = 1$ the "unit vector"
  - Note: Let $\boldsymbol{y} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$, then $\|\boldsymbol{y}\| = 1$

Norms defined for matrices

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

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

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

Linear algebra – singular value decomposition

Eigenvectors, eigenvalues, and singular values

- If $\boldsymbol{Av} = \lambda \boldsymbol{v}$, where $\boldsymbol{v}$ is a non-zero vector, then we say $\boldsymbol{v}$ is an eigenvector of square-matrix $\boldsymbol{A}$ (with eigenvalue λ)

- Singular values: non-negative square roots of eigenvalues of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$. Say $\sigma_i,\ i = 1, \ldots, m$

- Singular value decomposition: If $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, there exists orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{V} \in \mathbb{R}^{m \times m}$ such that

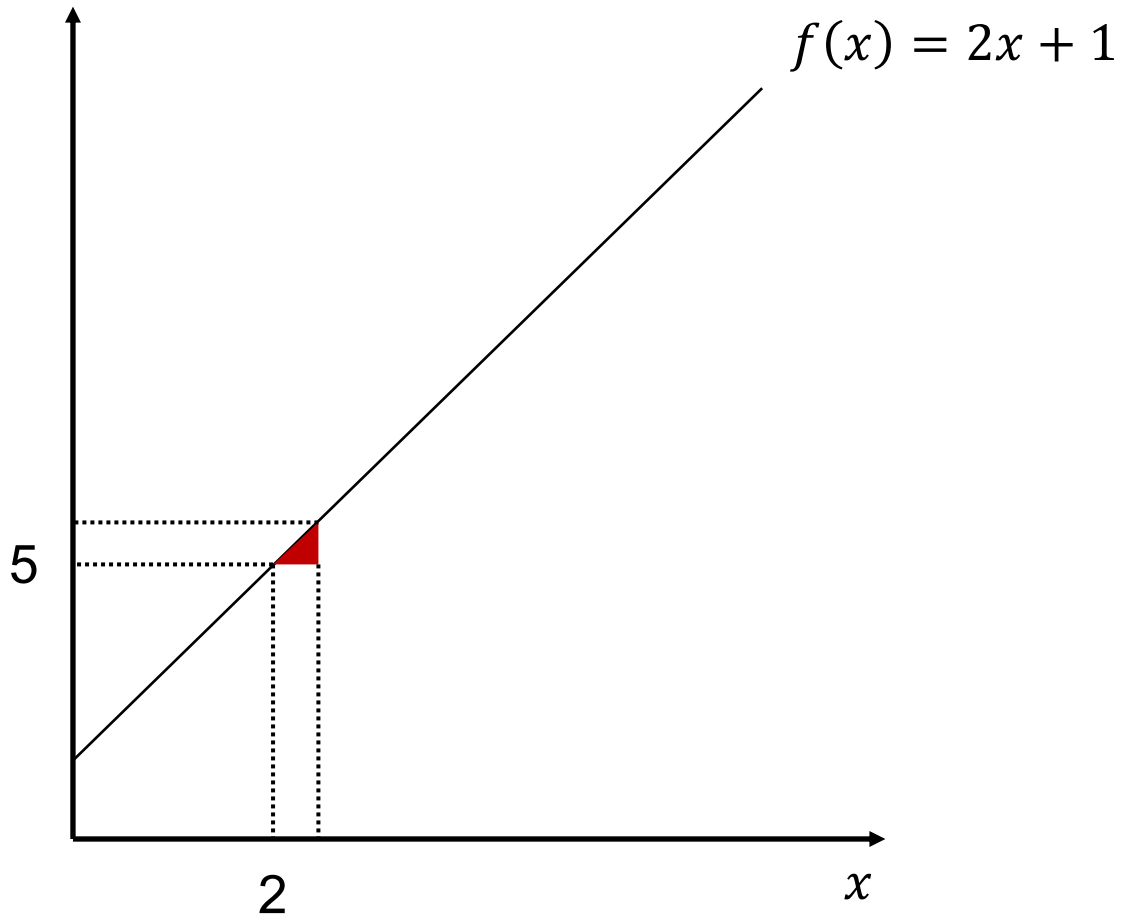
$$U^{-1} A\, V = \Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_m \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrix multiplication and co in Python

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

A refresher on calculus

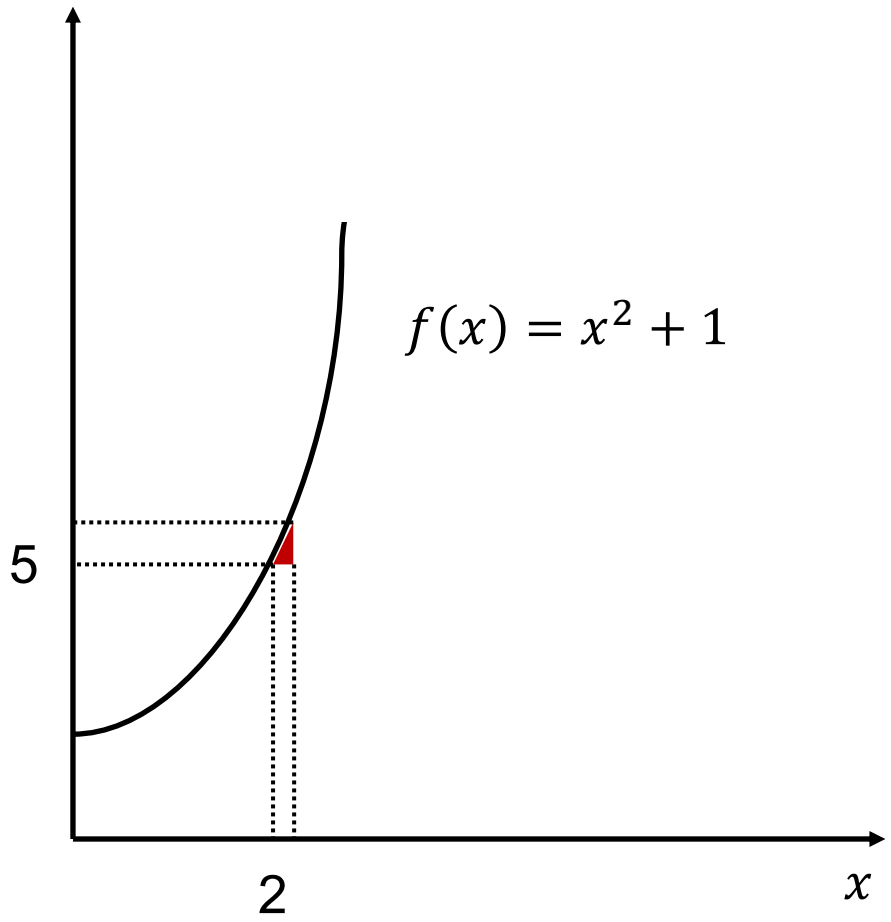
What is a derivative?



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

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

What is a derivative?



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

A few important derivatives

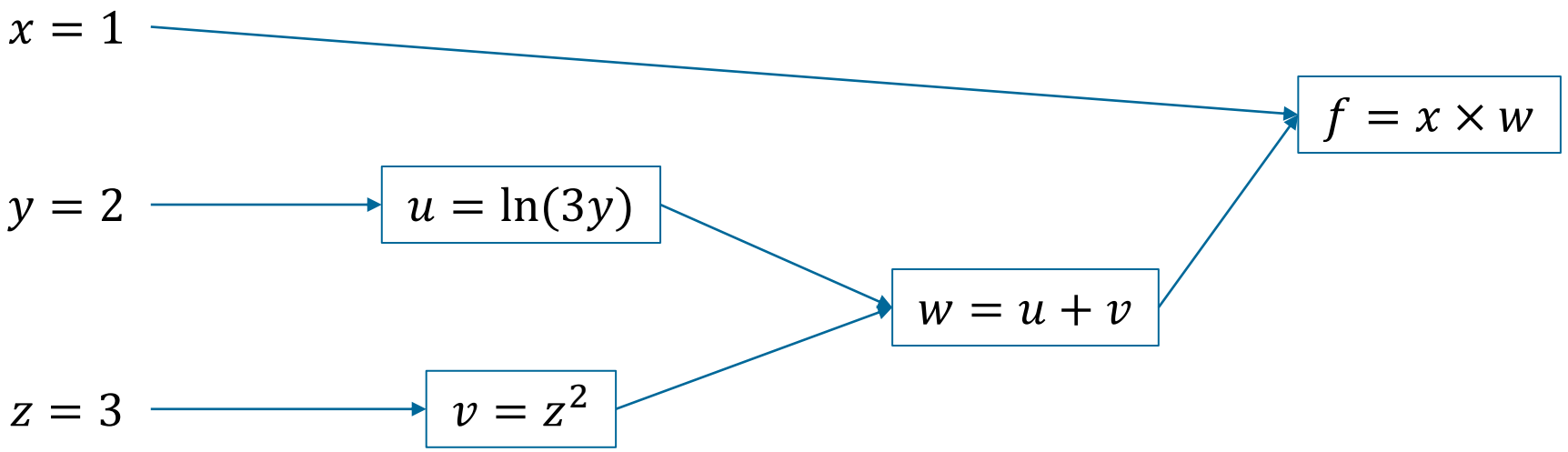
$f(x)$	$f'(x) = \frac{df(x)}{dx} = \frac{d}{dx} f(x)$
1	
x	
x^2	
x^3	
$\sqrt{x}$	
$\ln(x)$	
e^x	

Some rules about handling derivatives

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

Using a computation graph for derivatives

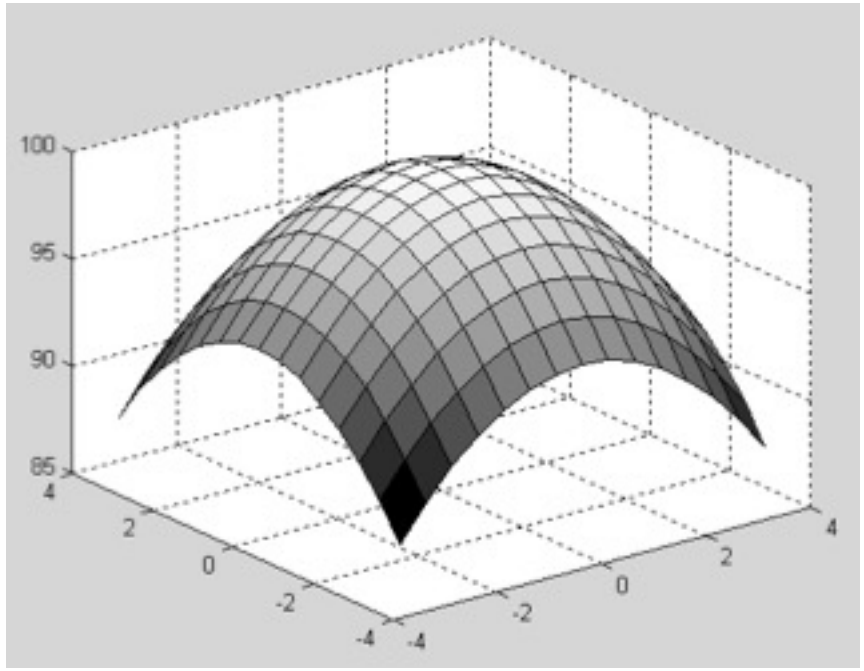
- We can use the “chain rule” to simplify the search for derivatives
- Say, we want to compute the derivatives of $f = x(\ln(3y) + z^2)$ to x, y, z at $x = 1, y = 2, z = 3$

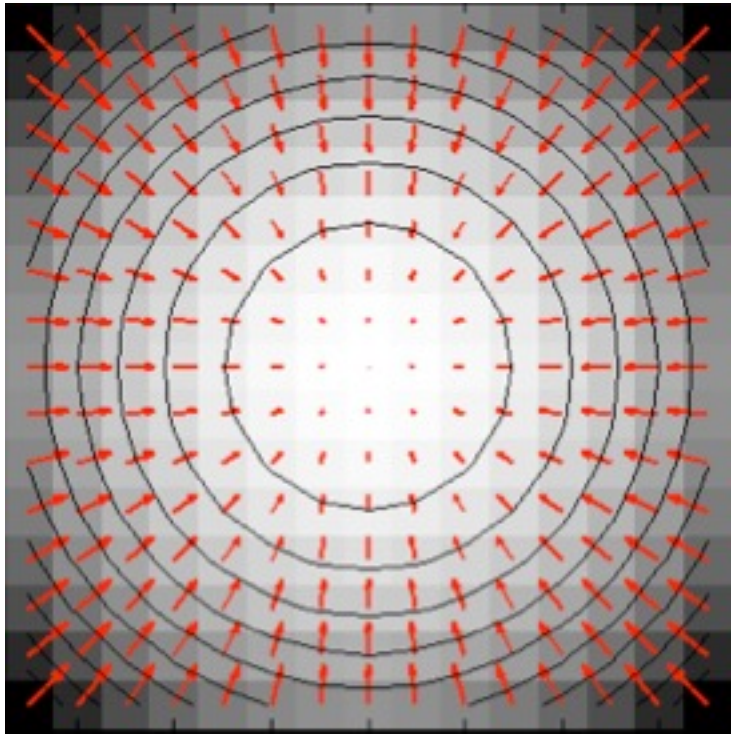


BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Putting it together – matrix calculus

The gradient





Source: Collins

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:

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

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:

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

- Based on the previous ideas, note that:
  - $\nabla_{\boldsymbol{x}}\big(f(\boldsymbol{x}) + g(\boldsymbol{x})\big) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} g(\boldsymbol{x})$
  - For all $\alpha \in \mathbb{R}$, $\nabla_{\boldsymbol{x}}\big(\alpha\, f(\boldsymbol{x})\big) = \alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$
  - If $f(\boldsymbol{x}) = \sum_{i=1}^{n} b_i x_i = \boldsymbol{b}^T \boldsymbol{x}$, then $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{b}$

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 $\nabla_{\boldsymbol{A}} f(\boldsymbol{A}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{A})}{\partial a_{1,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{1,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{1,m}} \\ \frac{\partial f(\boldsymbol{A})}{\partial a_{2,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{2,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{2,m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(\boldsymbol{A})}{\partial a_{n,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{n,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{n,m}} \end{bmatrix}$

Taking a step back – logistic regression

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

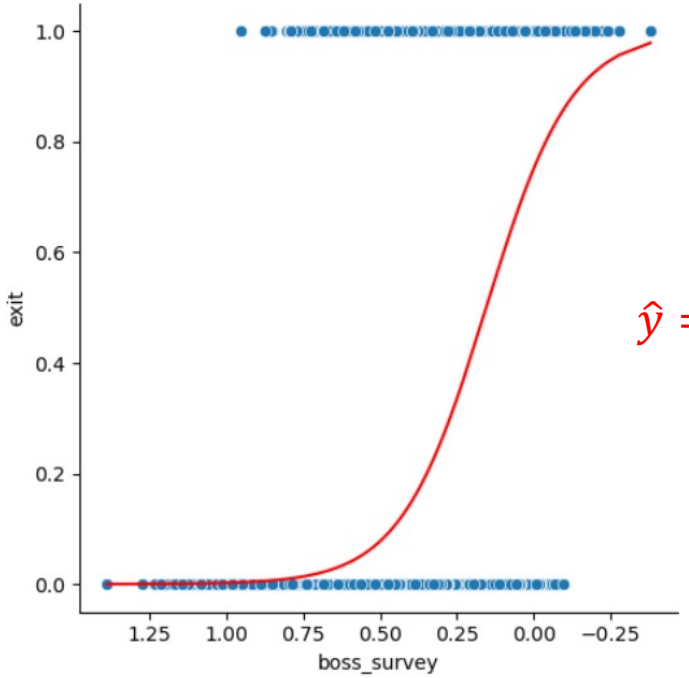
- We are given values $(\boldsymbol{x}^{(i)}, y^{(i)})$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \{0,1\}$
- Our prediction $\hat{y}^{(i)}$ should reflect the probability that $y^{(i)} = 1$: $\hat{y}^{(i)} = P\left(y^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}\right)$

- 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

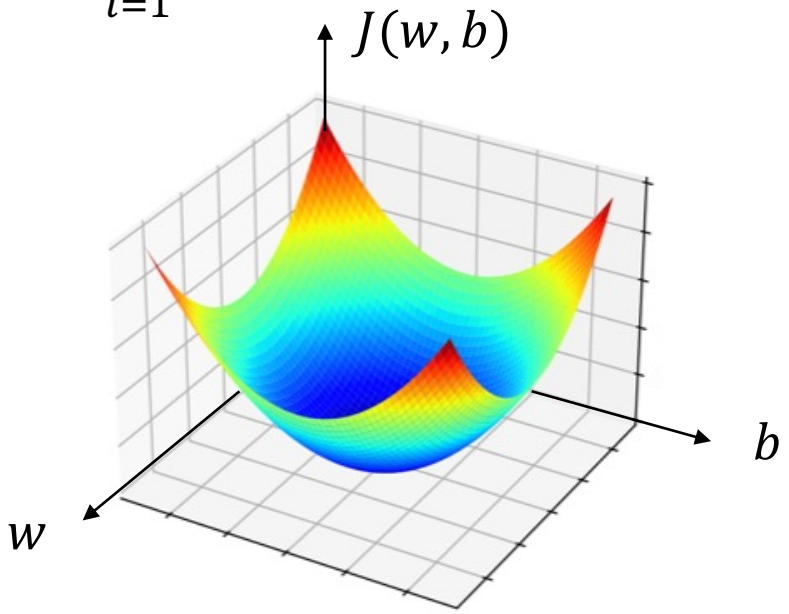
- Remember that $\boldsymbol{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$
- To get to the "right" model, we optimize our parameters $\boldsymbol{w}, b$ so that the $\hat{y}^{(i)}$s are "as close as possible" to the y^is
- What we do is to minimize the "cost-function" $J(\boldsymbol{w}, b)$, where $\hat{y}^{(i)}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

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

The optimization part

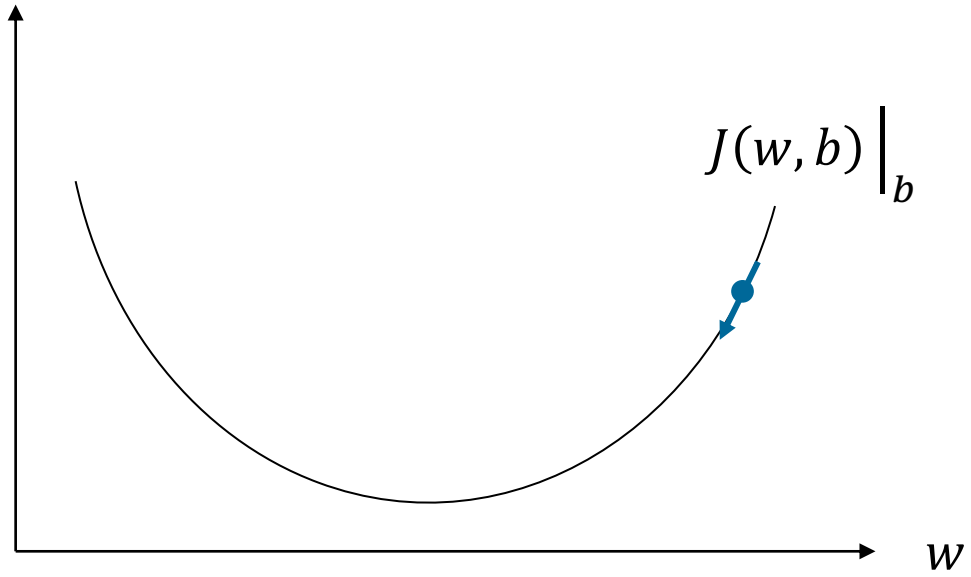
- Remember that $\boldsymbol{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$
- To get to the "right" model, we optimize our parameters $\boldsymbol{w}, b$ so that the $\hat{y}^{(i)}$s are "as close as possible" to the y^is
- What we do is to minimize the "cost-function" $J(\boldsymbol{w}, b)$, where $\hat{y}^{(i)}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

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



BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Solving the optimization problem through gradient descent

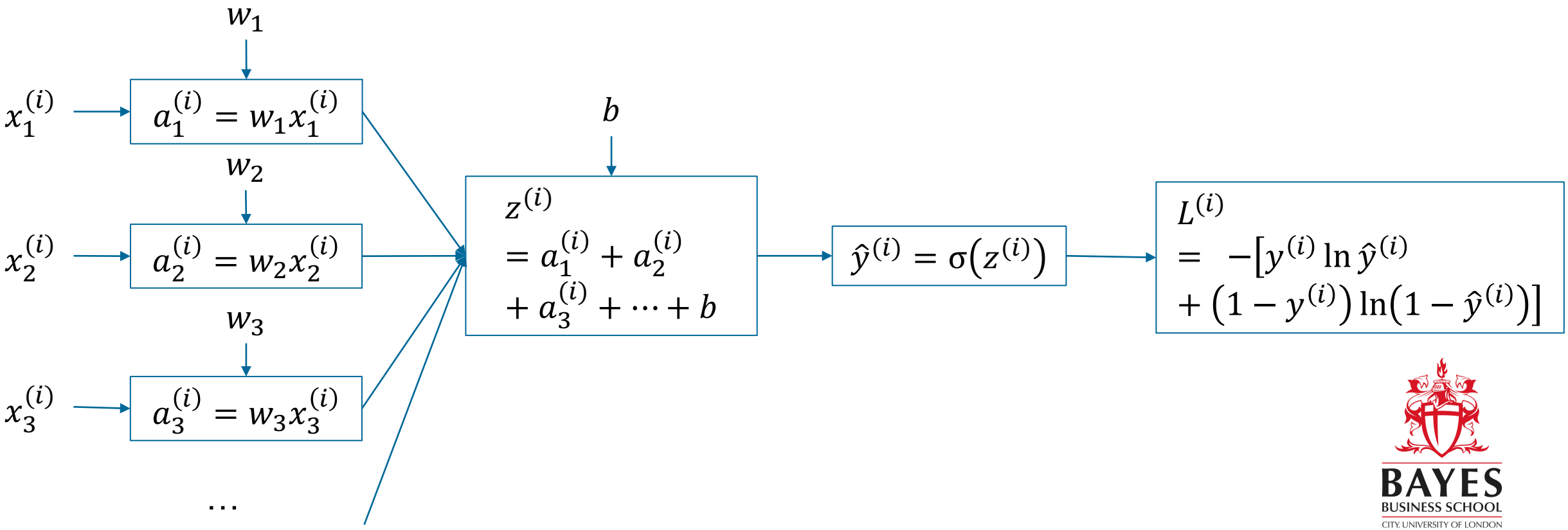


Our first optimization algorithm

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!

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

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



BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

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 $\frac{\partial J(\boldsymbol{w},b)}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L^{(i)}}{\partial w_j}$

We can now implement a logistic regression

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

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