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
  - …

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

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

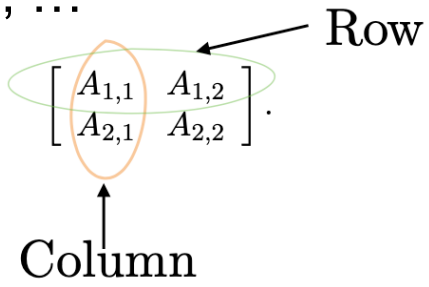
  - Entries can be real numbers, binary, integer, …

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



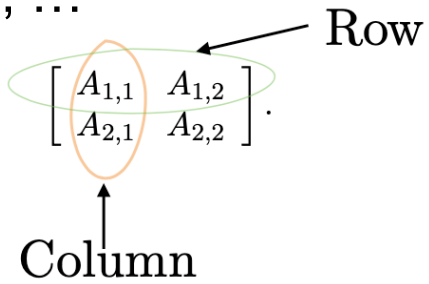
BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

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

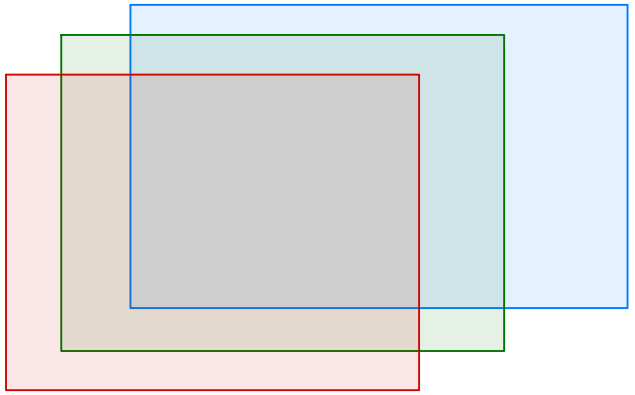


- Tensor: Any array of numbers
  - Could have zero dimensions (scalar), one dimension (vector), two dimensions (matrix)
  - Could also have three or more dimensions

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



Tensor with three dimensions!

Source: Ivanovic

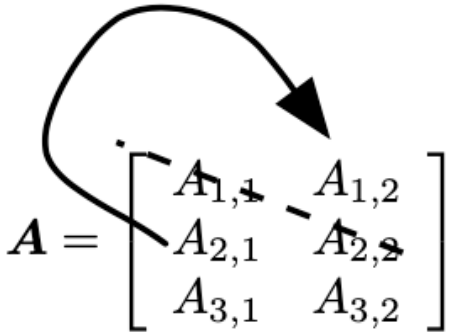
In Python

- 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 $\boldsymbol{A} = \boldsymbol{A}^T$

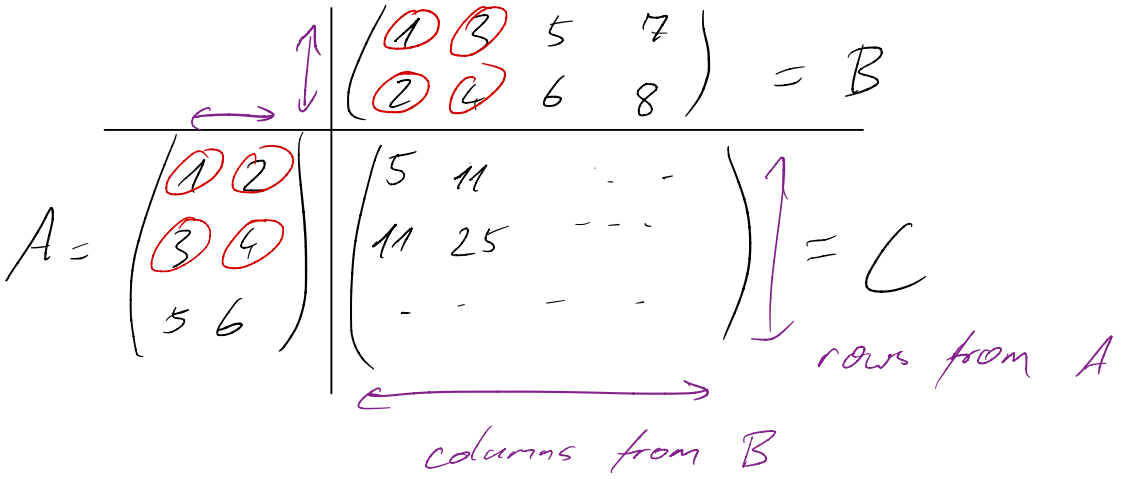
$A = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}$ $A^T = \begin{pmatrix} 1 & 4 \\ 3 & 2 \end{pmatrix}$

Source: Goodfellow

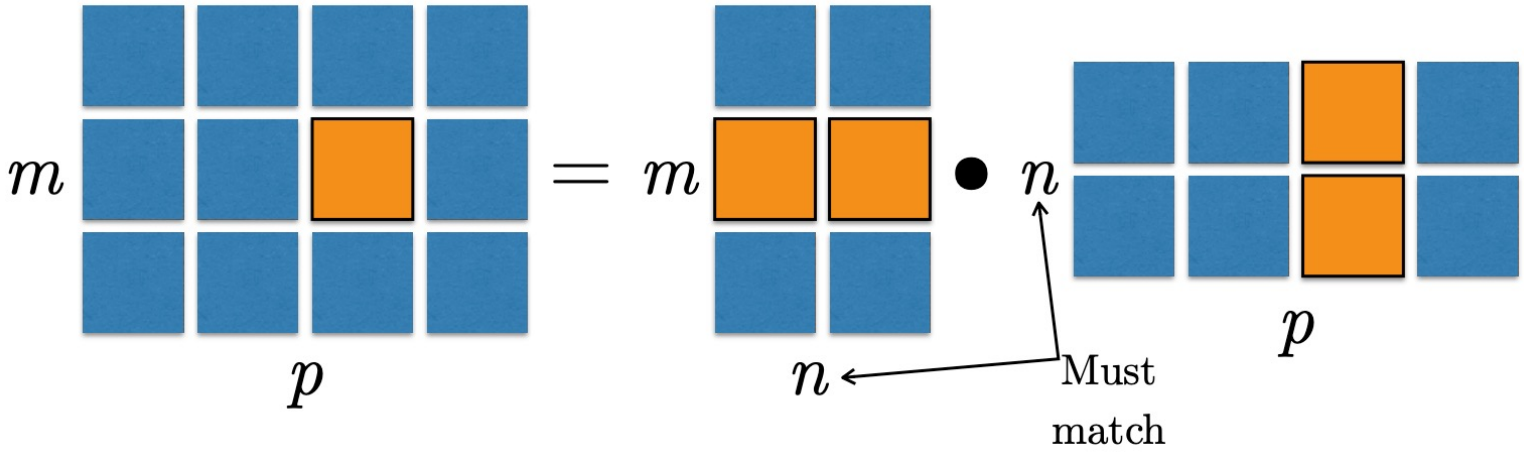
Matrix multiplication

$$\boldsymbol{C} = \boldsymbol{AB}.$$

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



$(3,2) \times (2,4) = (3,4)$



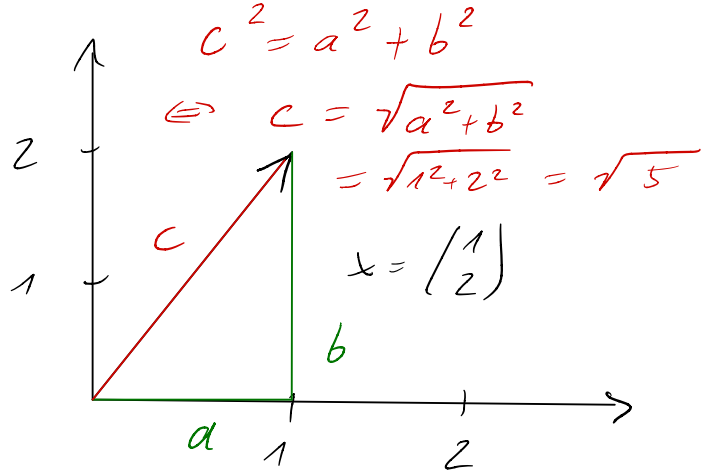
Source: Goodfellow

In Python

BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

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 $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, f(\boldsymbol{x}+\boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$

"default norm":

$$f\begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} = \sqrt{x_1^2 + x_2^2 + \dots}$$

(Euclidean norm)

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Some commonly used norms

- L^p norm:
  - $\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$

    $= \left(\sum_i x_i^2\right)^{\frac{1}{2}}$

  - 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 $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$

Norms defined for matrices

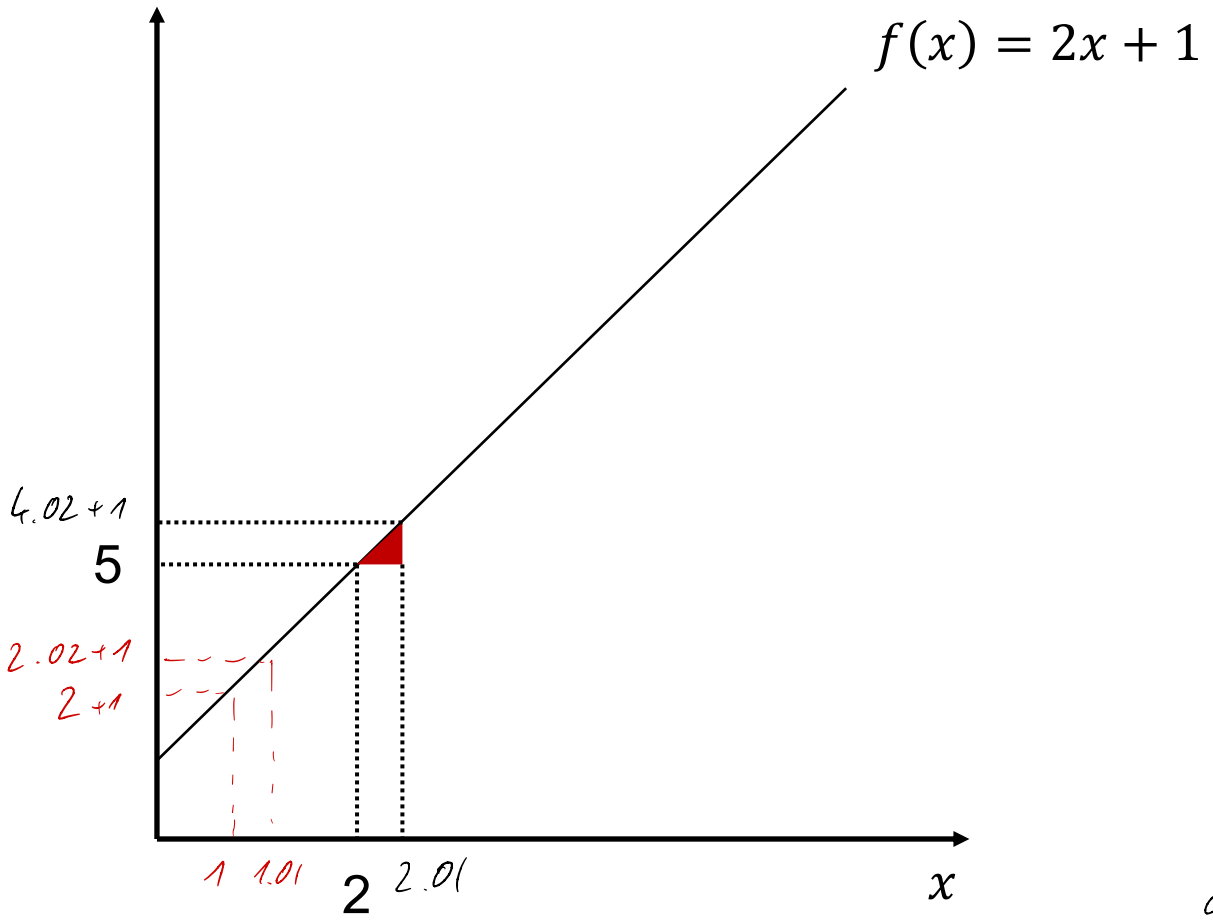
- There are many matrix norms, but we will only need one: the Frobenius norm

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

- 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 $\frac{\text{“height”}}{\text{“width”}}$

$$\frac{5.02 - 5}{2.01 - 2} = \frac{0.2}{0.1} = 2$$

$$\frac{f(2+\varepsilon)}{\varepsilon} \xrightarrow{\varepsilon \to 0} 2$$

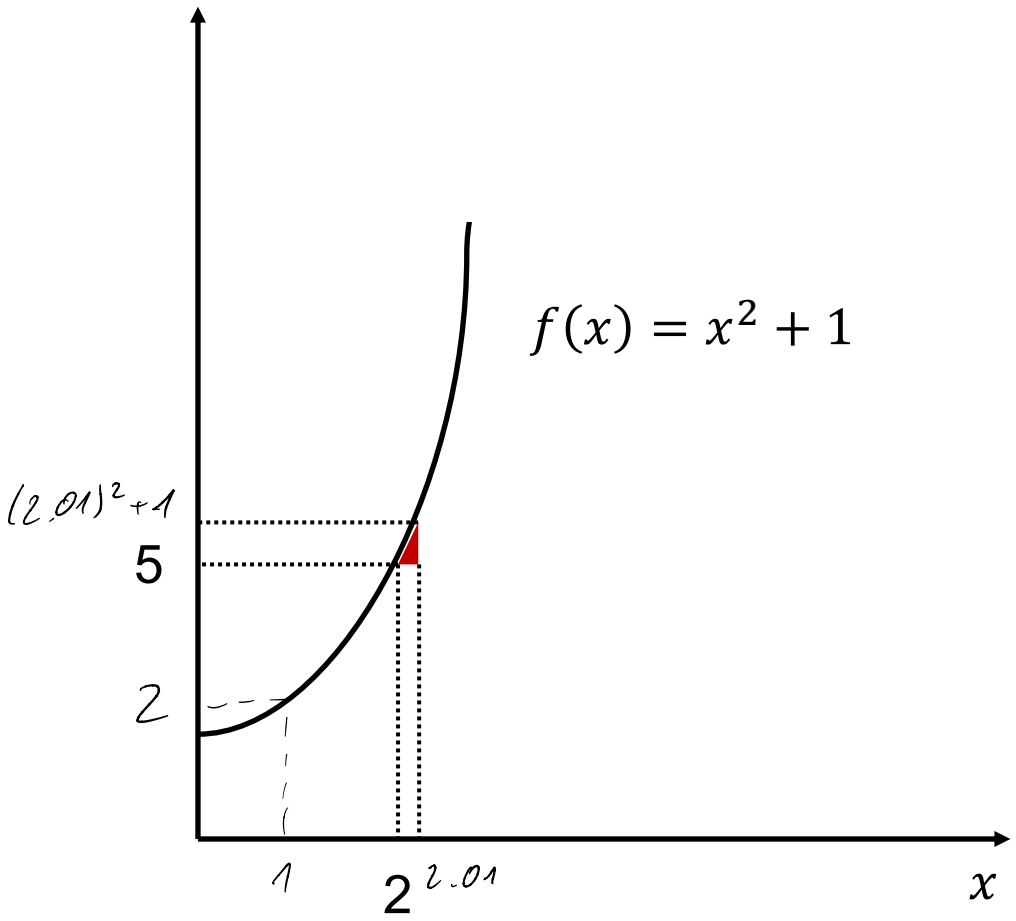
$f'(x)$ at 2

$\frac{df(x)}{dx}$ at 2

at 1: $\frac{3.02 - 3}{1.01 - 1} = \frac{0.2}{0.1} = 2$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

What is a derivative?



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

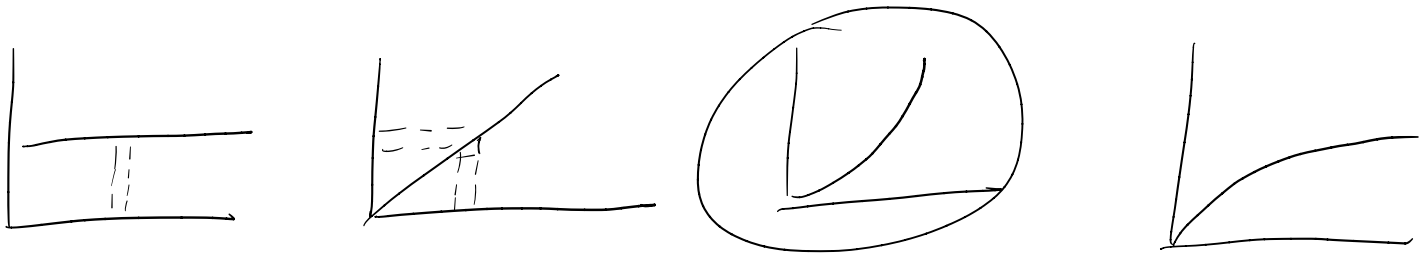
$$\frac{(2.01)^2+1-5}{2.01-2} = \frac{(2.01)^2-4}{0.1} \approx \frac{0.4}{0.1} = 4$$

at 1: $\frac{(1.01)^2+1-2}{1.01-1} \approx \frac{0.2}{0.1} = 2$

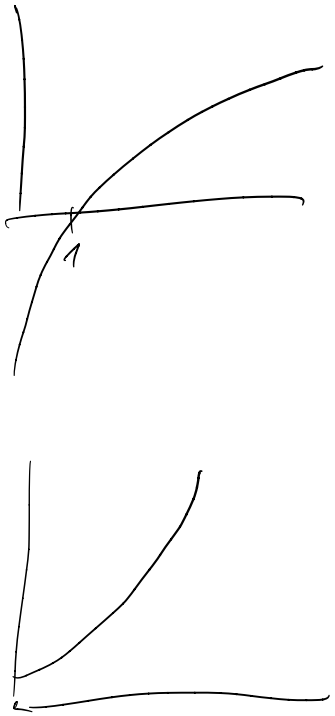
$f'(x)$ for any x: $2x$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

A few important derivatives



$f(x)$	$f'(x) = \frac{df(x)}{dx} = \frac{d}{dx}f(x)$
1	0
x	1
x^2	$2x$
x^3	$3x^2$
$\sqrt{x} = x^{\frac{1}{2}}$	$\frac{1}{2}x^{\frac{1}{2}-1} = \frac{1}{2}x^{-\frac{1}{2}}$
$\ln(x)$	$\frac{1}{x}$
e^x	e^x



Some rules about handling derivatives

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

the engine of any neural network

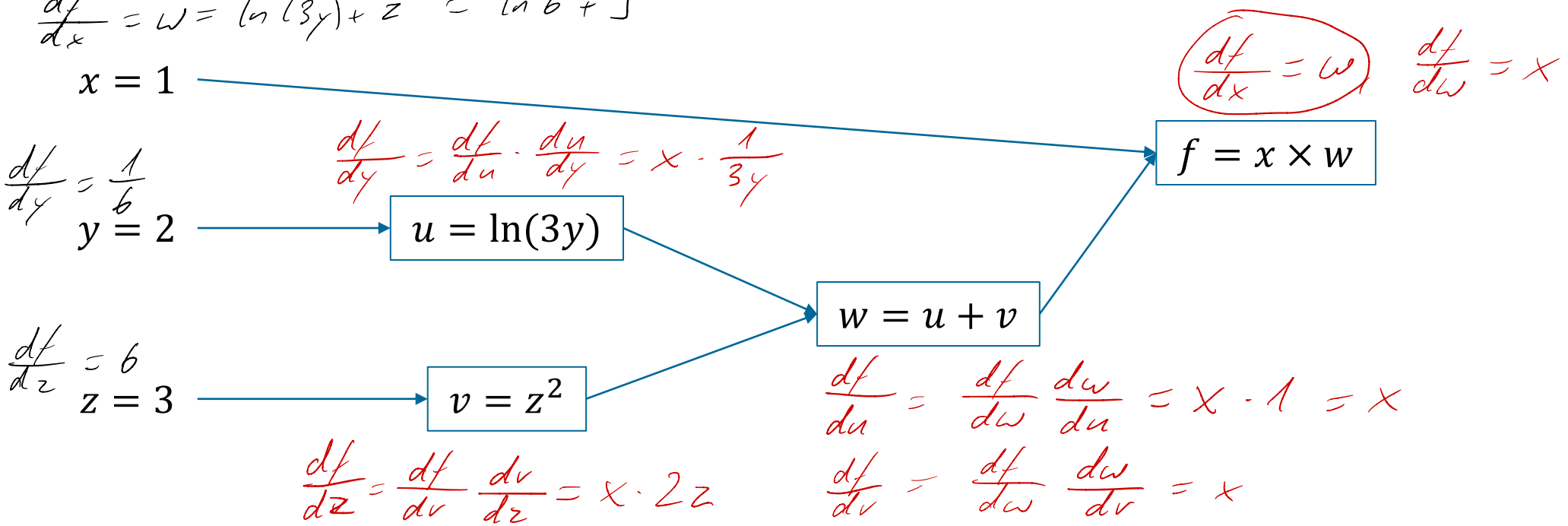
$h(x)$	$h'(x)$	Example
$c\,f(x)$	$c\,f'(x)$	$h(x) = 18\,x^k$ $\quad h'(x) = 18k\,x^{k-1}$
$f(x) + g(x)$	$f'(x) + g'(x)$	$h(x) = \log(x) - x^2 + 5$
$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$	$h(x) = 2\underbrace{e^x}_{f(x)}\underbrace{x}_{g(x)}$ $\quad h'(x) = 2[e^x x + e^x]$
$\frac{1}{f(x)}$	$-\frac{f'(x)}{f(x)^2}$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

$f(y) = e^y \rightsquigarrow f'(y) = e^y$

$g(x) = x^2 \rightsquigarrow g'(x) = 2x$

Using a computation graph for derivatives

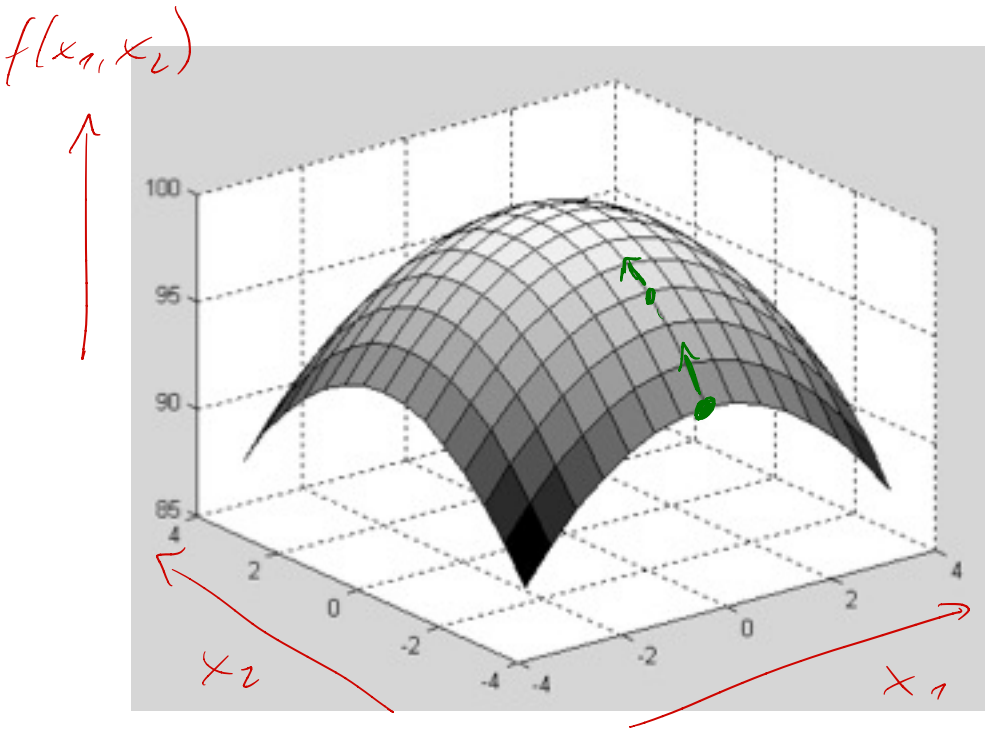
- We can use the “chain rule” to simplify the search for derivatives
- Say, we want to compute the derivatives of $f = x(\ln(3y) + z^2)$ to x, y, z at $x = 1, y = 2, z = 3$

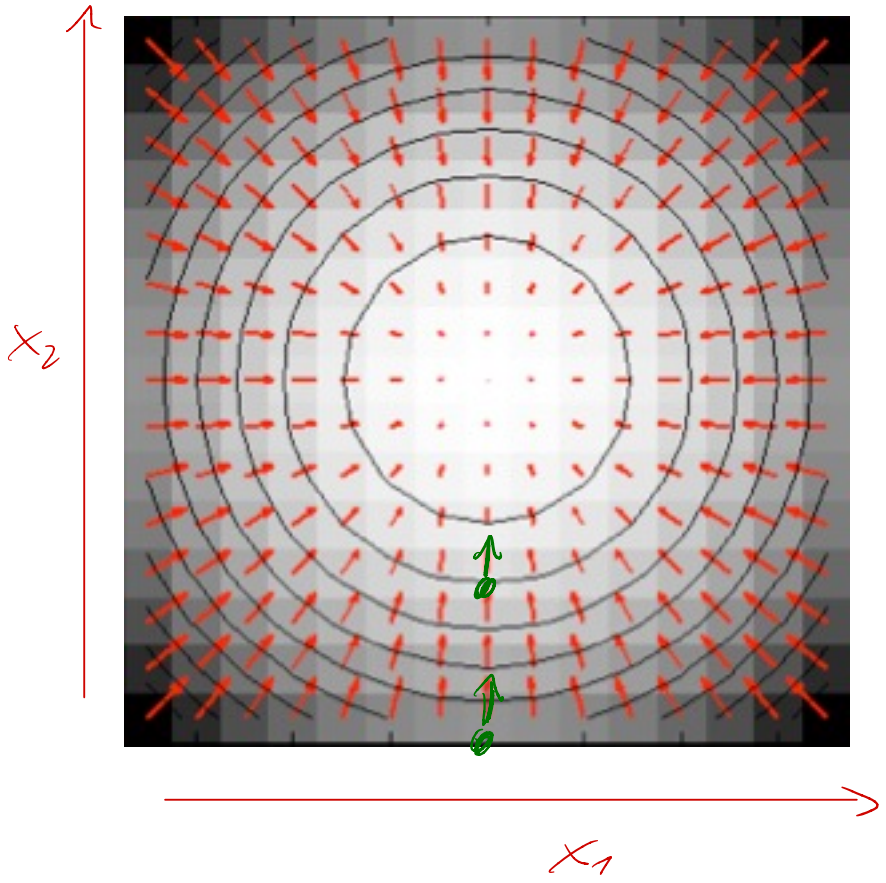


BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Putting it together – gradient descent

The gradient





Source: Collins

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:

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

This naturally extends to functions that take as input a matrix

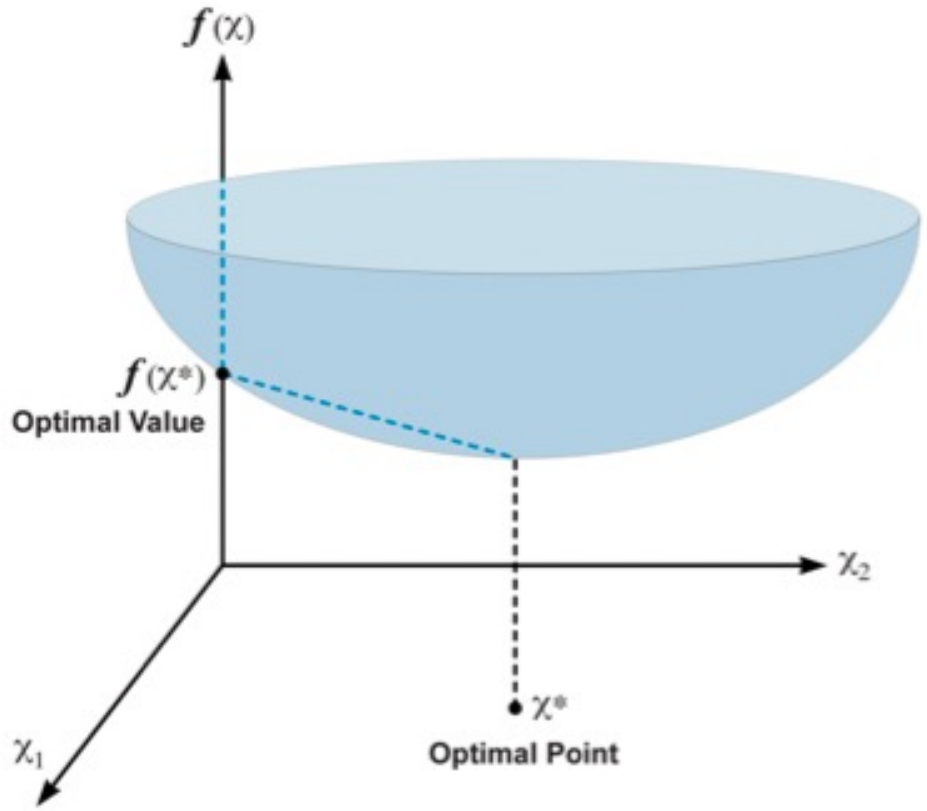
- Say, now we have a function f, taking as input a matrix $\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix}$

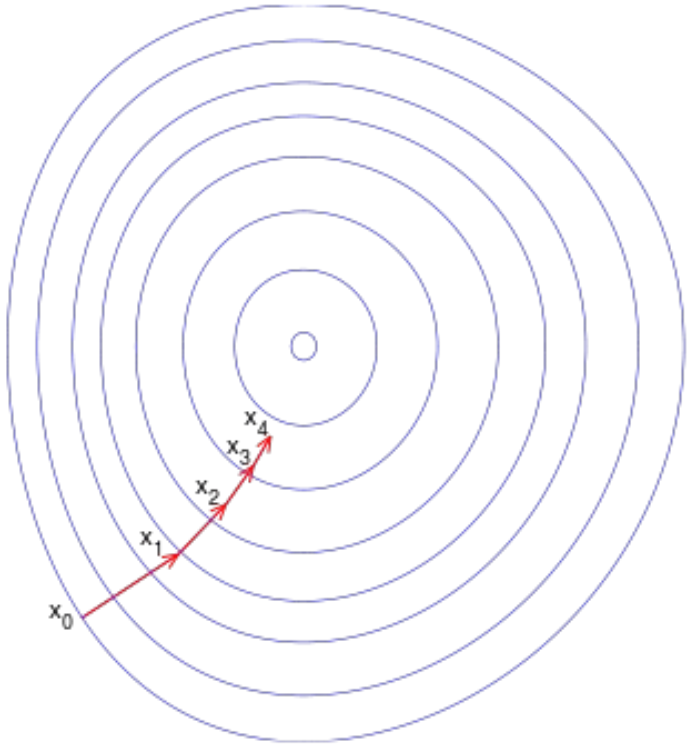
- The gradient (with respect to $\boldsymbol{A}$) is now also a matrix $\nabla_{\boldsymbol{A}} f(\boldsymbol{A}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{A})}{\partial a_{1,1}} & \frac{\partial f(\boldsymbol{A})}{\partial a_{1,2}} & \cdots & \frac{\partial f(\boldsymbol{A})}{\partial a_{1,m}}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

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





BAYES
BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

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?

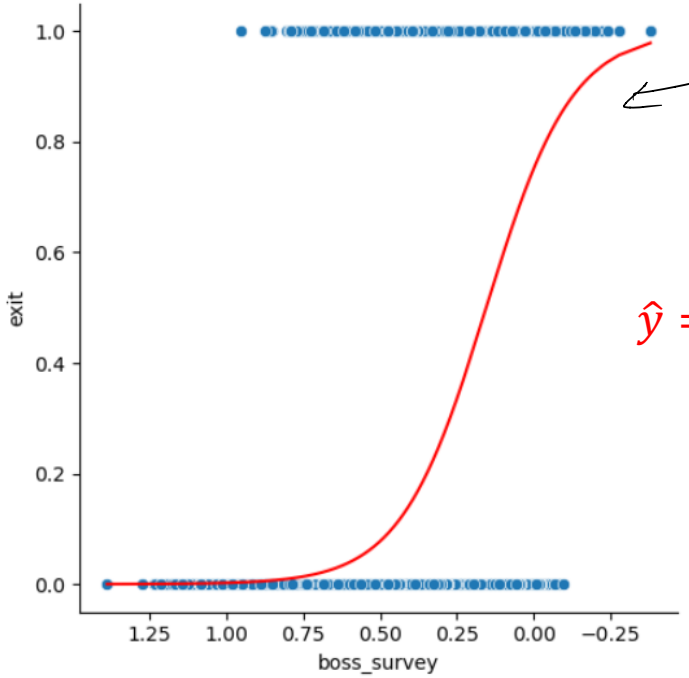
- We are given values $(\boldsymbol{x}^{(i)}, y^{(i)})$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$ and $y^{(i)} \in \{0,1\}$
- Our prediction $\hat{y}^{(i)}$ should reflect the probability that $y^{(i)} = 1$: $\hat{y}^{(i)} = P\left(y^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}\right)$

- 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:



$$(x_{i,1} \ldots x_{i,m}) \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} + b$$

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

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 $\hat{y}^{(i)} = \frac{1}{1+e^{-\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+\boldsymbol{b}\right)}}$:

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

$y^{(i)} = 1$: $1 \times \ln \hat{y}^{(i)} + 0$

$\hat{y}^{(i)} \to 1$: 0

$\hat{y}^{(i)} \to 0$: $+\infty$

$y^{(i)} = 0$: $0 + 1 \times \ln(1 - \hat{y}^{(i)})$

$\hat{y}^{(i)} \to 1$: $+\infty$

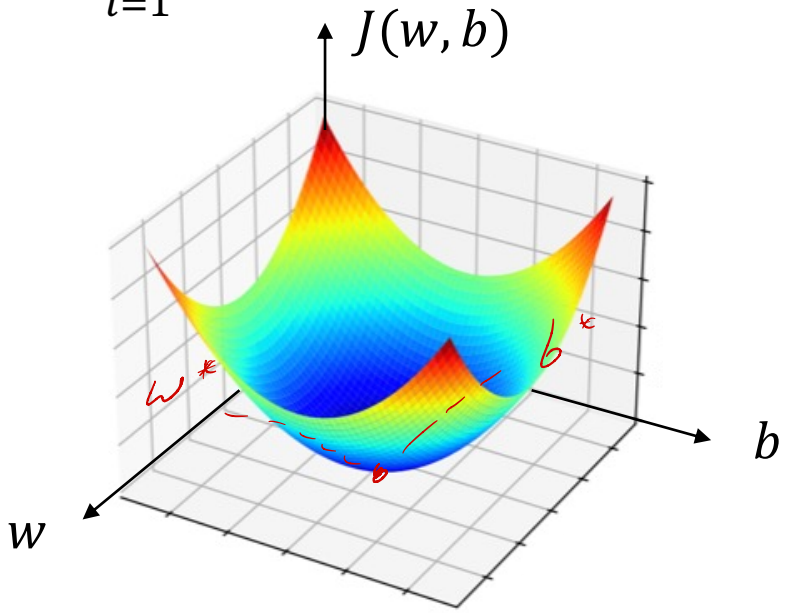
$\hat{y}^{(i)} \to 0$: 0

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

The optimization part

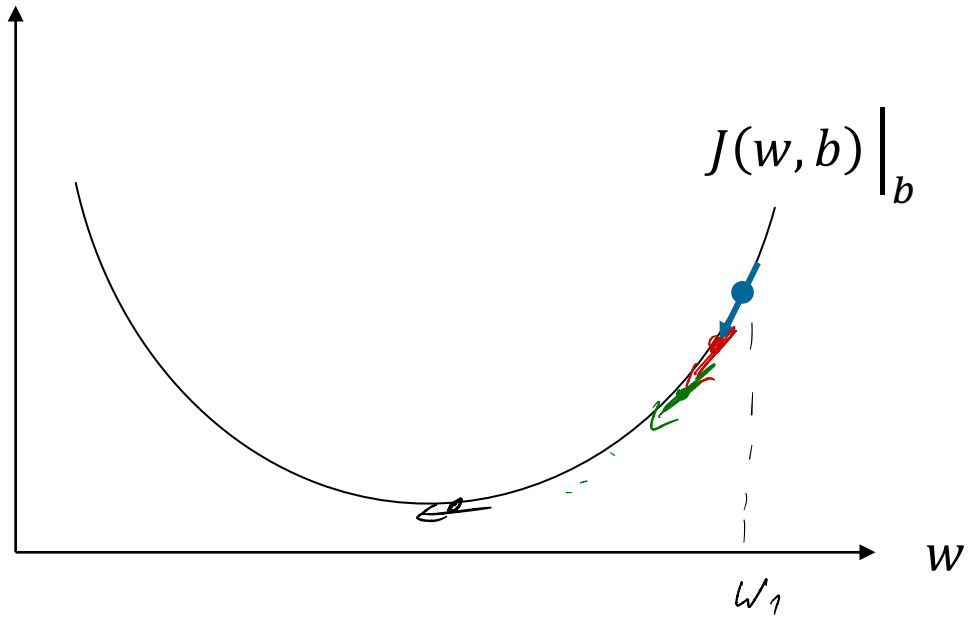
- Remember that $\boldsymbol{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$
- To get to the "right" model, we optimize our parameters $\boldsymbol{w}, b$ so that the $\hat{y}^{(i)}$s are "as close as possible" to the y^is
- What we do is to minimize the "cost-function" $J(\boldsymbol{w}, b)$, where $\hat{y}^{(i)}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

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



BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Solving the optimization problem through gradient descent



$$w^1 \longrightarrow \frac{dJ}{dw}\Big|_{w^1}$$

$$w^2 = w^1 - \alpha \frac{dJ}{dw}\Big|_{w^1}$$

$$w^3 = w^2 - \alpha \frac{dJ}{dw}\Big|_{w^2}$$

$$\vdots$$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

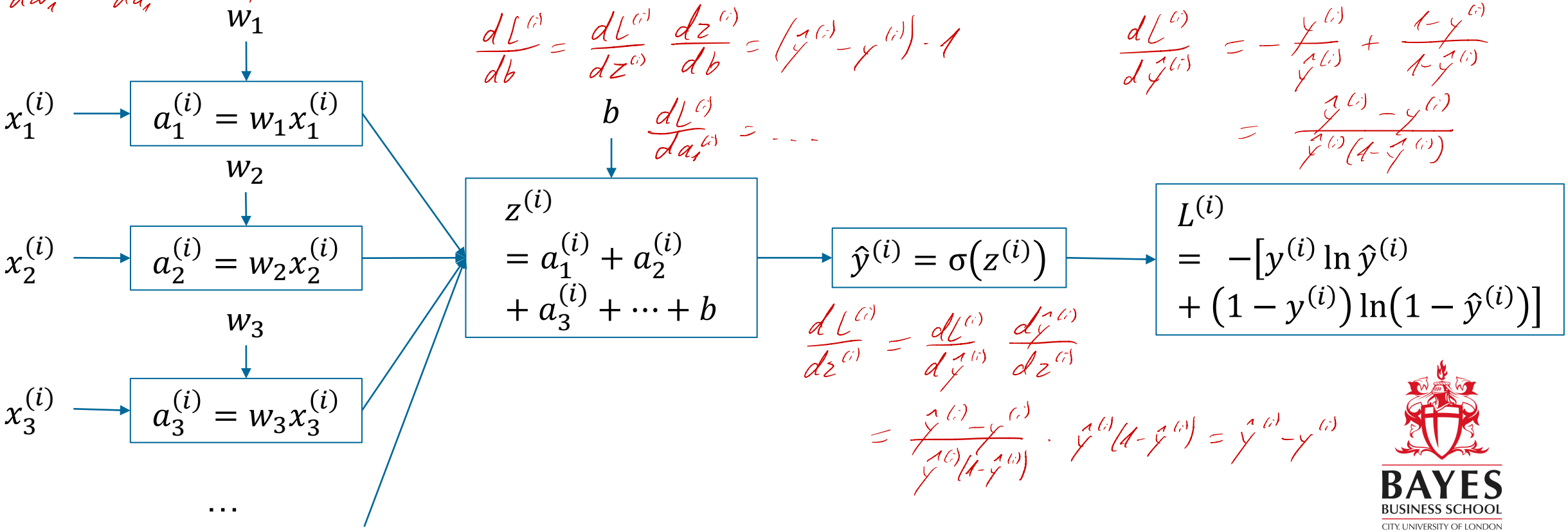
Our first optimization algorithm

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 $\hat{y}^{(i)} = \frac{1}{1+e^{-\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+b\right)}} = \sigma\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+b\right)$

$\frac{dL^{(i)}}{dw_1} = \frac{dL^{(i)}}{da_1^{(i)}} \frac{da_1^{(i)}}{dw_1} = \dots$



$\frac{dL^{(i)}}{db} = \frac{dL^{(i)}}{dz^{(i)}} \frac{dz^{(i)}}{db} = \left(\hat{y}^{(i)} - y^{(i)}\right) \cdot 1$

$\frac{dL^{(i)}}{da_1^{(i)}} = \dots$

$\frac{dL^{(i)}}{d\hat{y}^{(i)}} = -\frac{y^{(i)}}{\hat{y}^{(i)}} + \frac{1-y^{(i)}}{1-\hat{y}^{(i)}} = \frac{\hat{y}^{(i)} - y^{(i)}}{\hat{y}^{(i)}\left(1-\hat{y}^{(i)}\right)}$

$\frac{dL^{(i)}}{dz^{(i)}} = \frac{dL^{(i)}}{d\hat{y}^{(i)}} \frac{d\hat{y}^{(i)}}{dz^{(i)}} = \frac{\hat{y}^{(i)} - y^{(i)}}{\hat{y}^{(i)}\left(1-\hat{y}^{(i)}\right)} \cdot \hat{y}^{(i)}\left(1-\hat{y}^{(i)}\right) = \hat{y}^{(i)} - y^{(i)}$

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

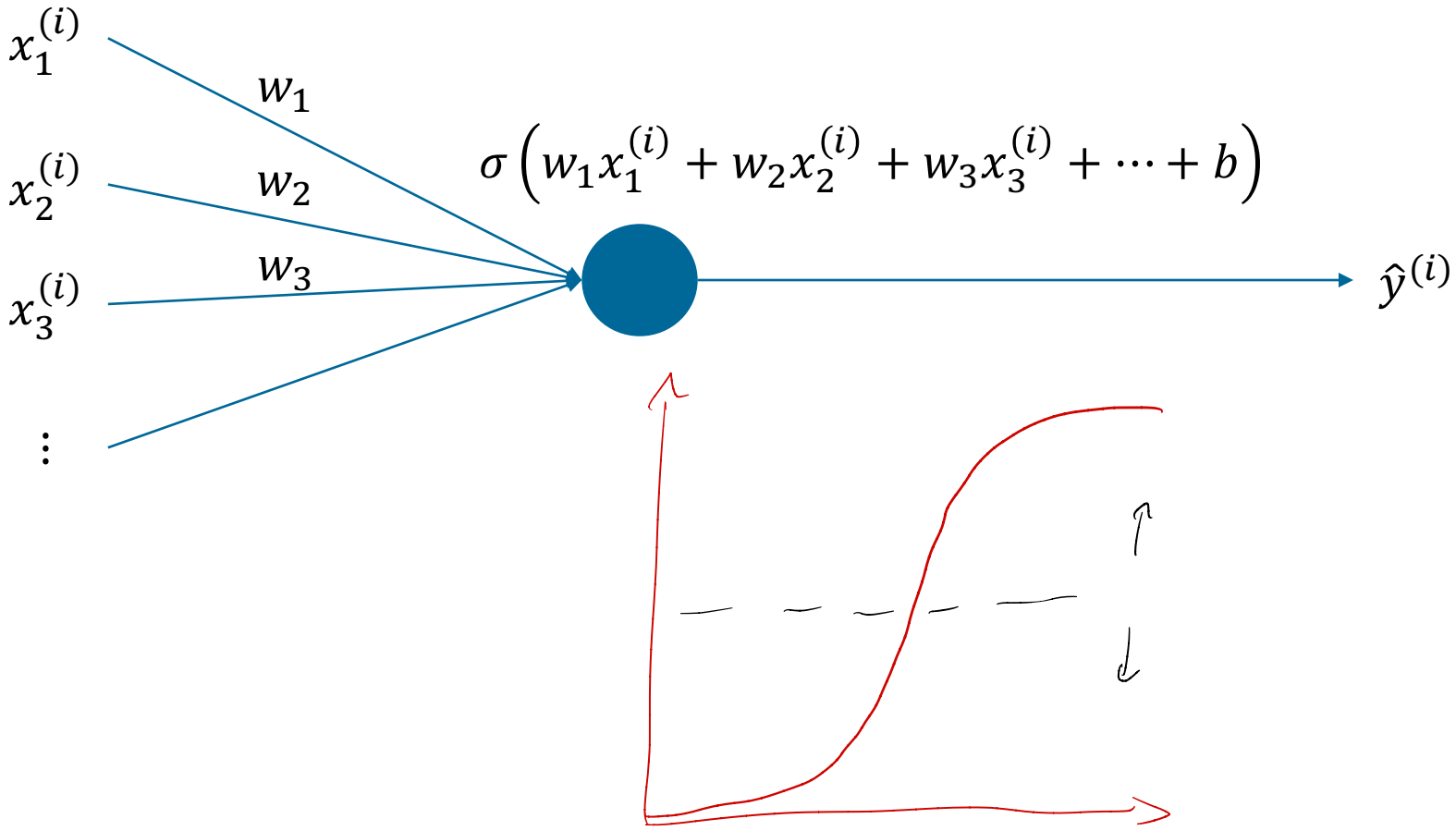
$$\frac{dJ}{dw_1} = \frac{1}{n}\sum_{i=1}^{n}\frac{dL^{(i)}}{dw_1}$$

$$\nabla J_w = \begin{pmatrix} \frac{dJ}{dw_1} \\ \vdots \\ \frac{dJ}{dw_n} \end{pmatrix} = \frac{1}{n} X^T (\hat{y} - y)$$

$$(n,1) = (n,\underline{n}) \times (\underline{n},1)$$

BAYES BUSINESS SCHOOL
CITY, UNIVERSITY OF LONDON

Schema of a logistic regression



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