Vectors, Matrices and beyond!

Week 3, Mathematics and Computational Methods for Complex Systems

Dhruva Venkita Raman

Functions

Recap

Example 1

$$f(x) = x^2$$

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

Domain: all real numbers

Range: all positive real numbers

Example 2

$$+(x, y) = x + y$$

$f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

Domain: all pairs of real numbers

Range: all real numbers

Terminology

Domain: set of possible inputs

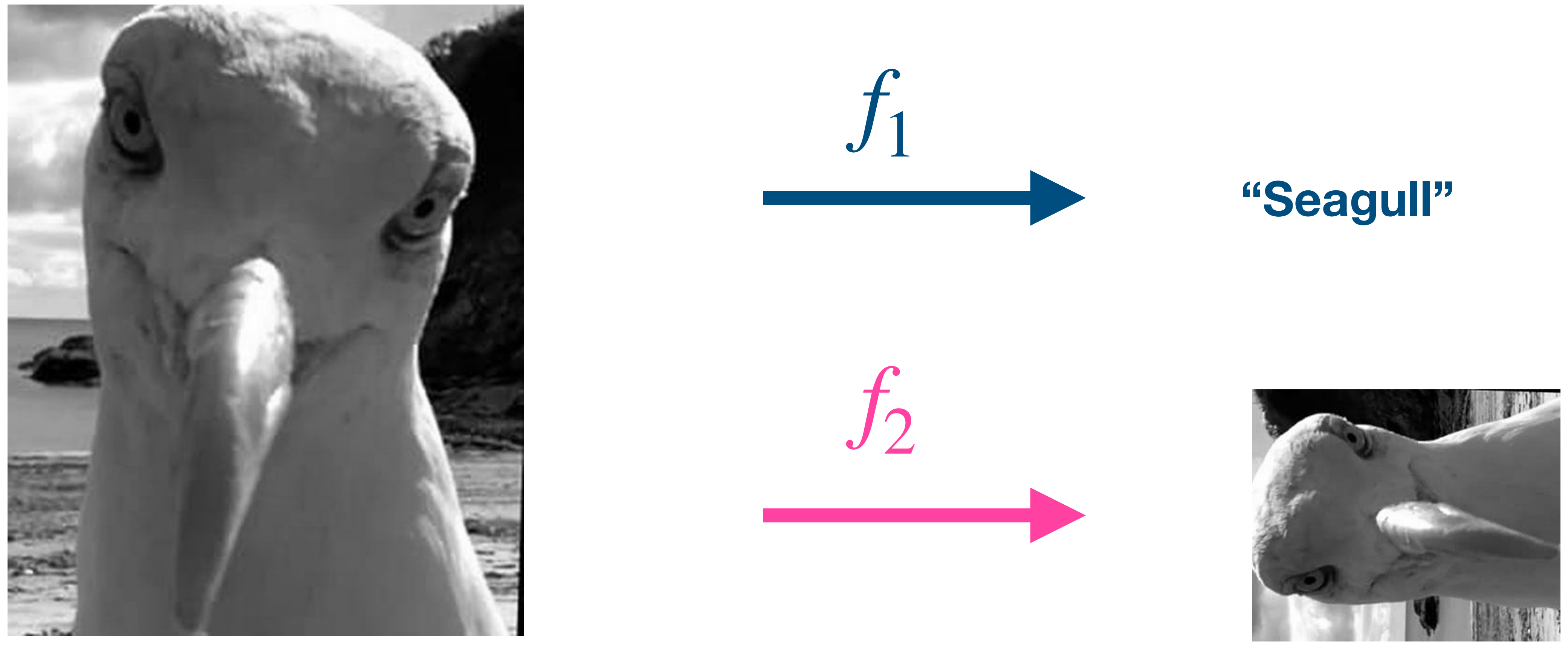
Range: set of possible outputs

Examples express transformations/relationships between numbers

Functions

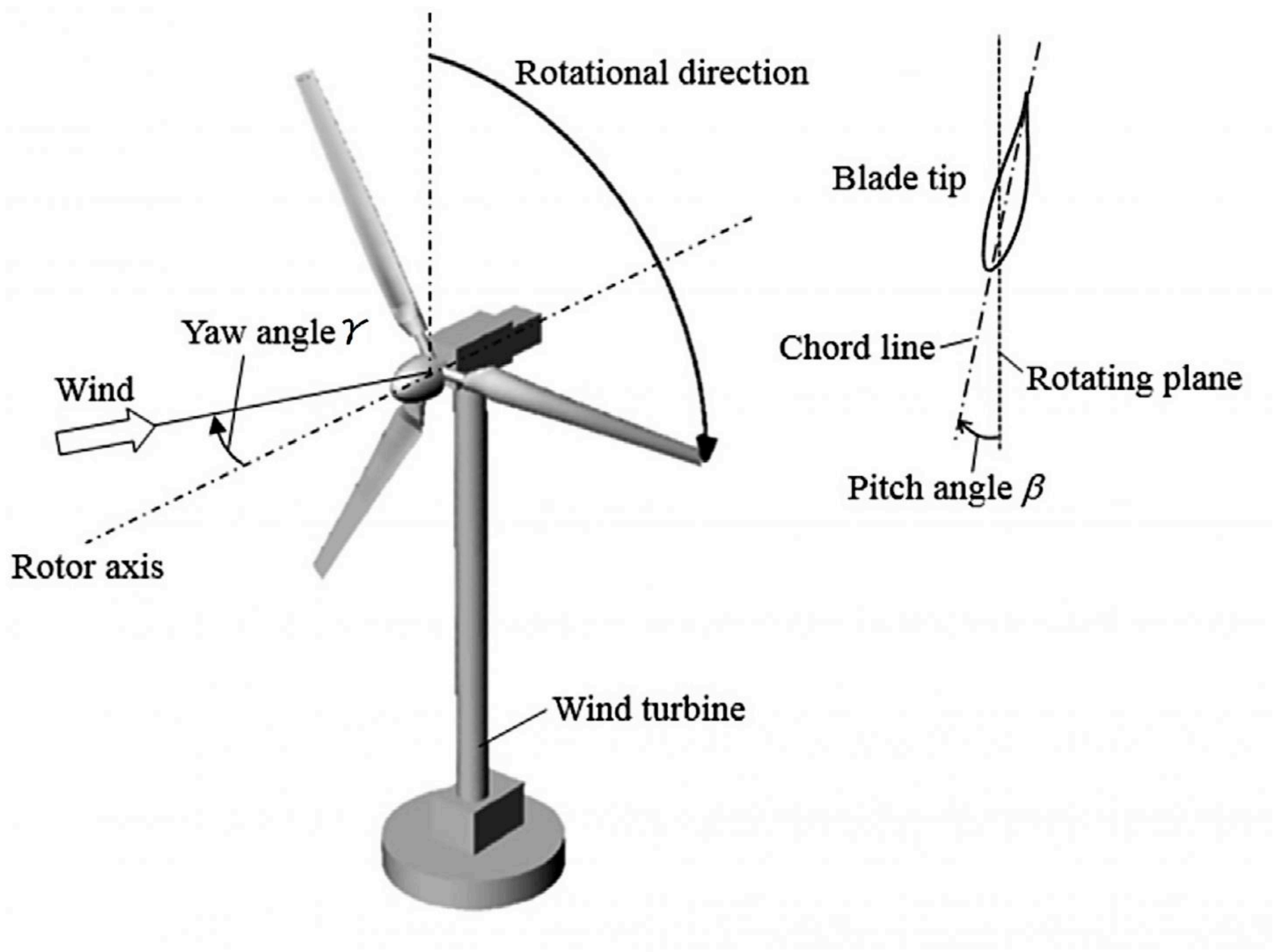
f_1 : images $\rightarrow$ strings (text)

f_2 : images $\rightarrow$ images



Real life: transformations / relationships between more complicated objects

Functions

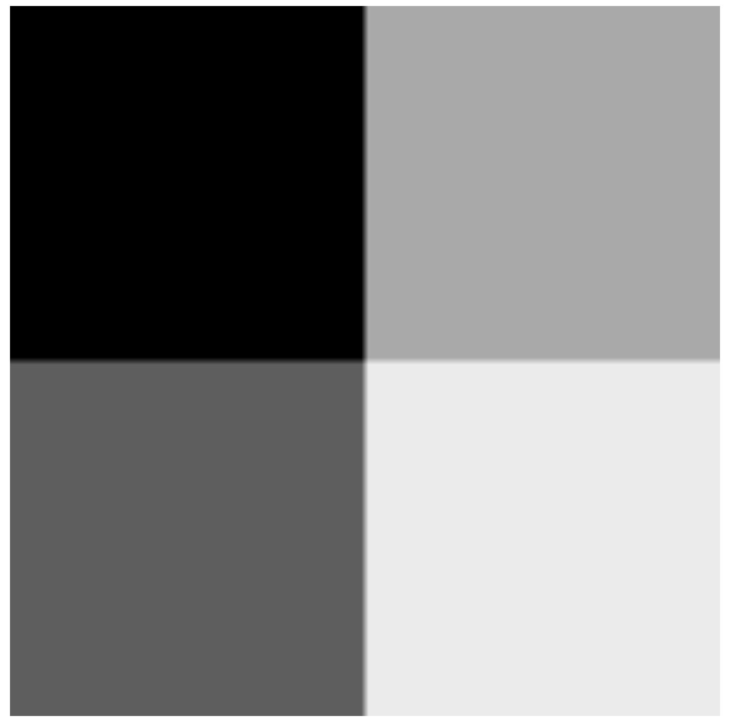


Best angle/pitch/yaw to catch the wind

Real life: transformations / relationships between more complicated objects

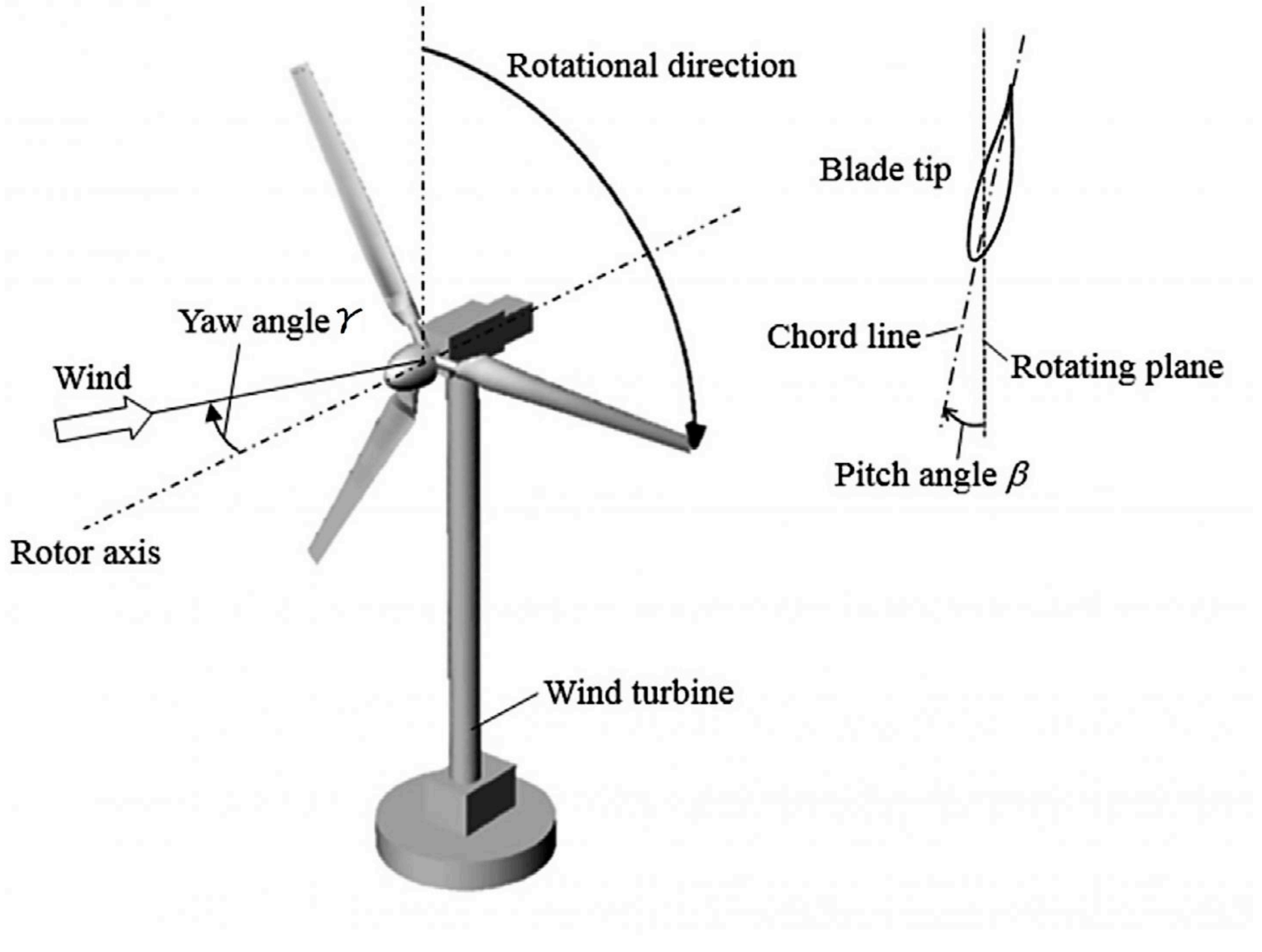
How do we do mathematics on ‘complicated objects’?

Complicated objects often boil down to ~~collections~~ arrays of numbers



$$\begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix}$$

[**Lots of numbers!**]

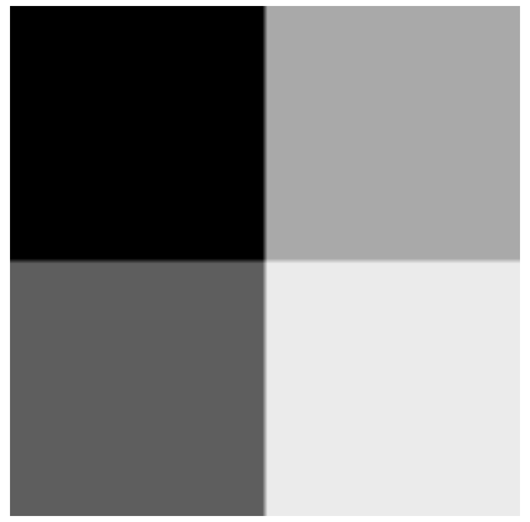


Windspeed,
Wind direction,
Motor torque,
Yaw angle
...

Arrays have a shape

Which we change as convenient!

Image



$$A = \begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix} \longleftrightarrow A^T = \begin{bmatrix} 0.0 & 0.6 \\ 0.3 & 0.9 \end{bmatrix} \longleftrightarrow \mathrm{vec}(A^T) = \begin{bmatrix} 0.0 \\ 0.3 \\ 0.6 \\ 0.9 \end{bmatrix}$$

Numerical representation

0 is black,
1 is white

`numpy.shape(A)`

Transposition:

Rows ⟷ *Columns*

`numpy.transpose(A)`

Vectorisation:

Concatenate columns

`numpy.reshape(A, 4)`

Arrays have a shape

Which we change as convenient!

The number 4

A scalar: no shape

`numpy.shape(4)`

The array [4]

An array containing one element: the number 4

`numpy.shape([4])`

Not the same!!!

Shapes have a dimension

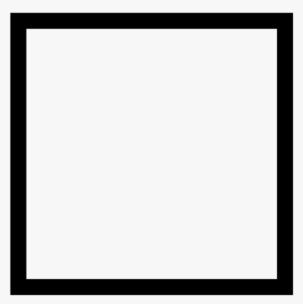
Vector (1d-array)

$$[0.0 \quad 0.3 \quad 0.6 \quad 0.9]$$

Matrix (2d-array)

$$\begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix}$$

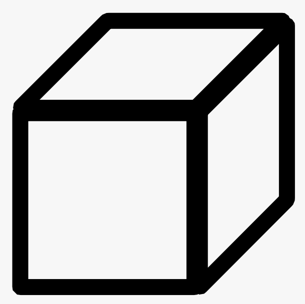
Rows and columns



3-Tensor (3d-array)

$$\left[\begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix} \begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix} \begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix} \right]$$

Rows, columns,….layers?



`A[3,7,2]` *- accessing element in 3d array*

`numpy.reshape([4], (1,1,1,1,1))` *- how many dimensions?*

Shapes have a dimension

Vector (1d-array)

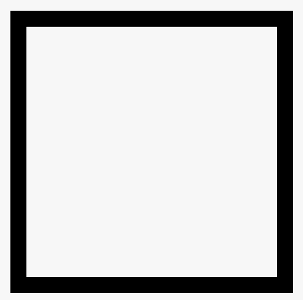
$$[0.0 \quad 0.3 \quad 0.6 \quad 0.9]$$

`A.shape = (4,)`

Matrix (2d-array)

$$\begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix}$$

Rows and columns

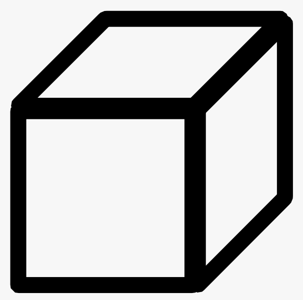


`A.shape = (2,2)`

3-Tensor (3d-array)

$$\left[\begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix} \begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix} \begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix}\right]$$

Rows, columns,….layers?



`A.shape = (2,2,3)`

Dimension is: `len(A.shape)`

Confusing terminology alert

Vector (1d-array)

$$[0.0 \quad 0.3 \quad 0.6 \quad 0.9]$$

Python/Julia: - array dimension is 1
- length is 4

Mathematician: - It's a 4-dimensional vector.
- A vector is a 1d tensor (i.e. array)
- Vector's length depends on its entries

Array indexing alert

Avoid incredible Python frustration!!!

Maths/Julia/Common sense:

$$\begin{array}{ccc} & \text{Column 1} & \text{Column 2} \\ \begin{array}{c}\text{Row 1}\\ \text{Row 2}\end{array} & \multicolumn{2}{c}{\begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix}} \end{array}$$

$$A_{21} = 0.6$$

Python:

$$\begin{array}{ccc} & \text{Column 0} & \text{Column 1} \\ \begin{array}{c}\text{Row 0}\\ \text{Row 1}\end{array} & \multicolumn{2}{c}{\begin{bmatrix} 0.0 & 0.3 \\ 0.6 & 0.9 \end{bmatrix}} \end{array}$$

```
A[1,0] = 0.6
```

Ingredients to do maths on arrays?

Let’s look back to how we do maths on numbers!

Addition/subtraction

is straightforward!

- *Only makes sense on vectors with same shape*

- *Otherwise, just like for numbers!*

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} + \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a+w \\ b+x \\ c+y \\ d+z \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} a-w & b-x \\ c-y & d-z \end{bmatrix}$$

Scaling

is straightforward!

- *Multiply an array (of any shape) by a* *scalar* *(number)*

$$x\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x*a \\ x*b \\ x*c \\ x*d \end{bmatrix}$$

$$-3\begin{bmatrix} 1 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} -3 \\ -12 \\ 6 \end{bmatrix}$$

The magnitude of numbers

i.e. their ***distance*** *from zero*

$$-7 < 2$$

$$|-7| = 7 > |2| = 2$$

$$|-7+6| = 1$$

- adding "big" numbers can result in a "small number

The magnitude/norm of vectors

Option 1: The L1 norm

$$\underline{v} = [2 \quad -3 \quad 7 \quad 1]$$

$$\boxed{\|\underline{v}\|_1} = |2| + |-3| + |7| + |1|$$
$$= 13$$

Double lines for array magnitudes!

The magnitude/norm of vectors

Option 1: The L1 norm

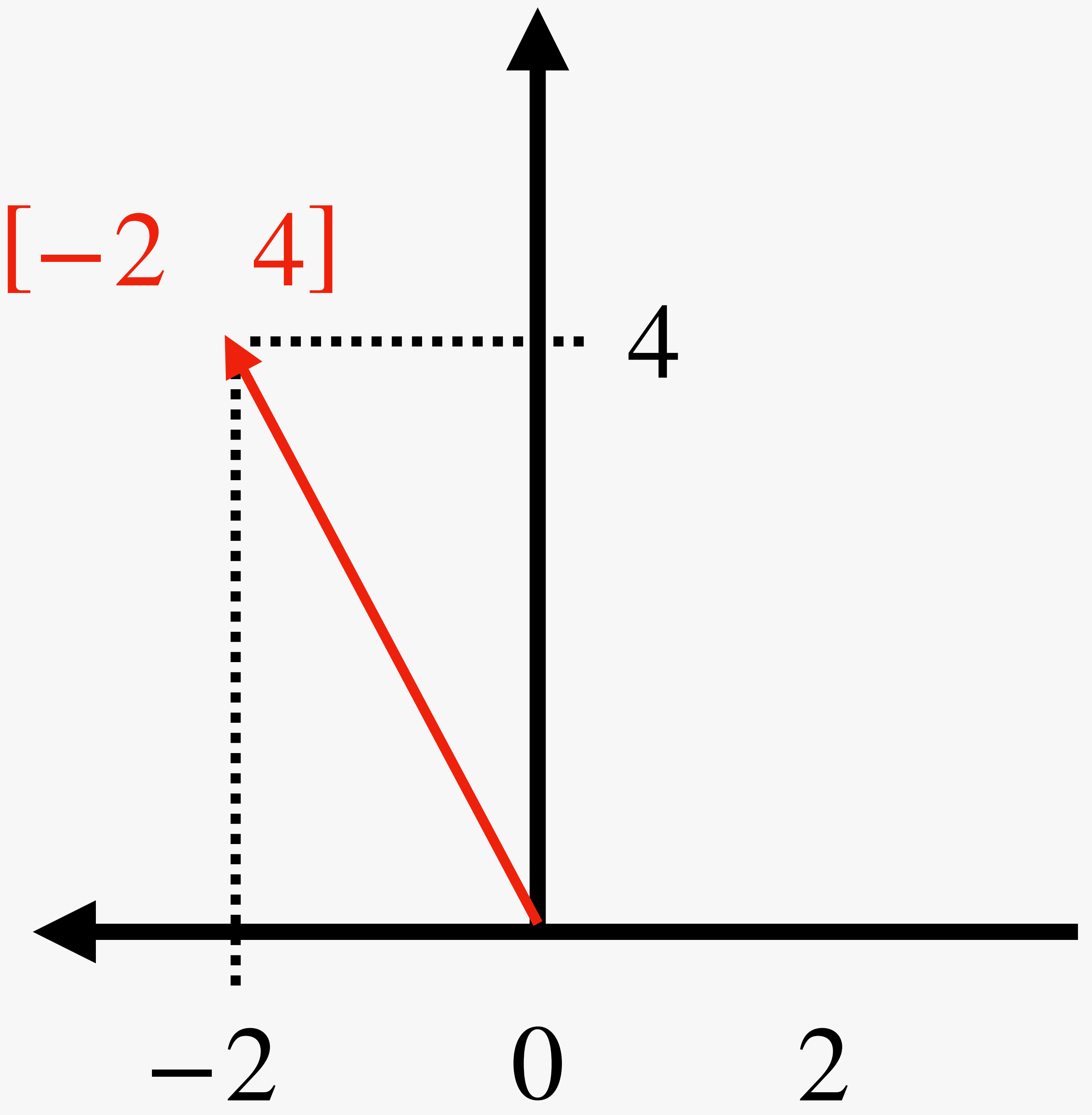
$$\underline{v} = [2 \quad -3 \quad 7 \quad 1]$$

$$\begin{aligned} \|\underline{v}\|_1 &= |2| + |-3| + |7| + |1| \\ &= 13 \end{aligned}$$

Double lines for array magnitudes!

~~Dimension~~ length-2 vectors can be drawn in 2d

$$\underline{a} = [-2 \quad 4]$$



$$\|\underline{a}\|_1 = |-2| + |4| = 6$$

The magnitude/norm of vectors

Option 2: The L2/Euclidean norm

$$\underline{v} = [2 \quad -3 \quad 7 \quad 1]$$

$$\|\underline{v}\|_2 = \sqrt{2^2 + (-3)^2 + 7^2 + 1^2}$$

$$= \sqrt{63} \approx 8$$

L1 and L2 are identical for scalars!

The magnitude/norm of vectors

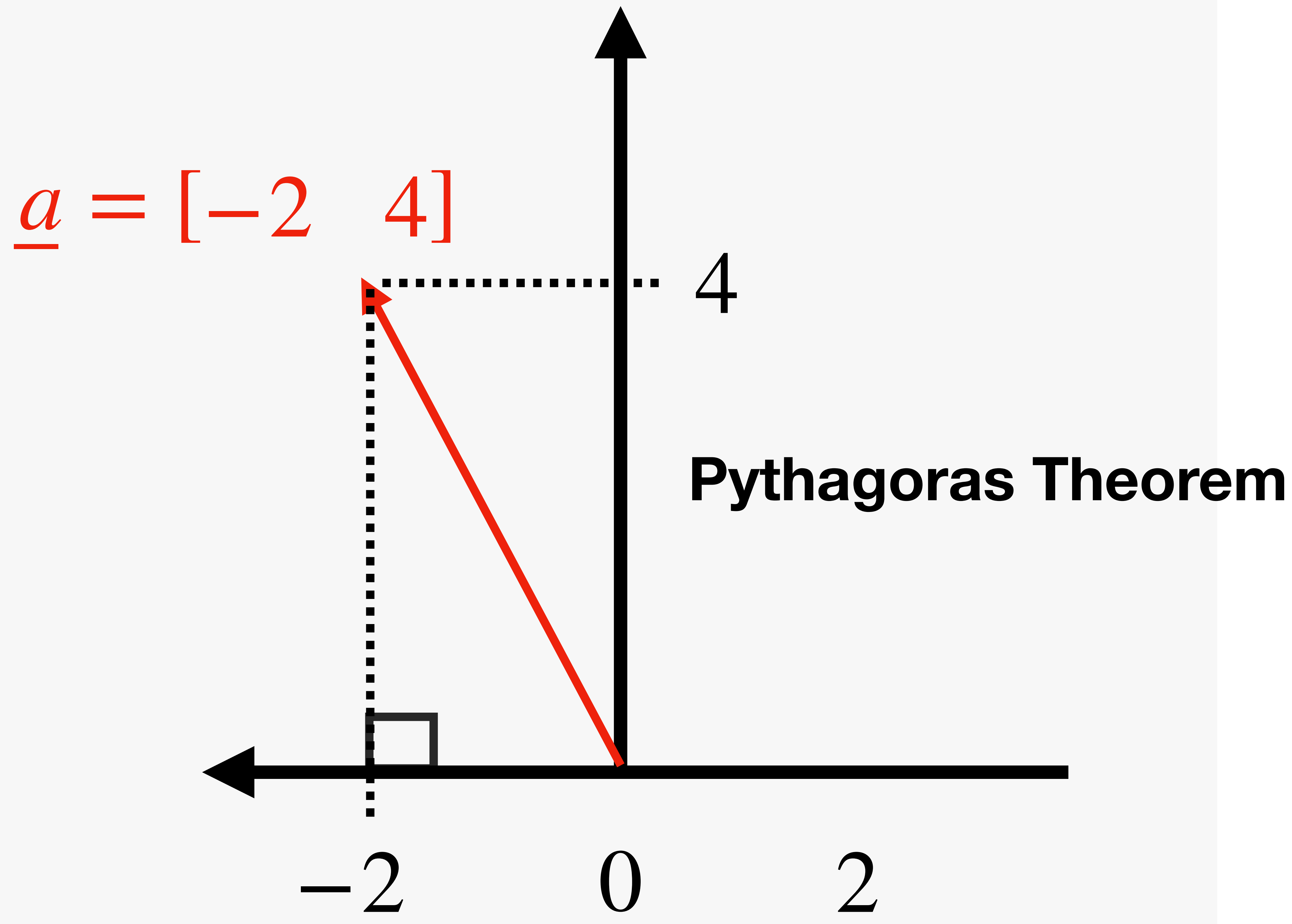
Option 2: The L2/Euclidean norm

$$\underline{v} = [2 \quad -3 \quad 7 \quad 1]$$

$$\|\underline{v}\|_2 = \sqrt{2^2 + (-3)^2 + 7^2 + 1^2}$$

$$= \sqrt{63} \approx 8$$

Length-2 vectors: their literal length



$$\|\underline{a}\|_2 = \sqrt{(-2)^2 + 4^2} \approx 4.5$$

The magnitude/norm of vectors

Option 3: Build your own!

Norm: **-** notion of distance from zero
- any scalar function (i.e. with scalar output) encoding this notion
- Inputs? Arrays of any shape

$$\|\bullet\| : \text{vectors} \rightarrow \text{scalars}$$

e.g. $\|\underline{v}\|_2 = 8$

The magnitude/norm of vectors

Option 3: Build your own!

What makes a *distance?*

$$||\underline{x}|| \geq 0$$

Distances can't be negative

$$||s\underline{x}|| = s||\underline{x}||$$

where s is scalar

Twice the vector means twice the distance

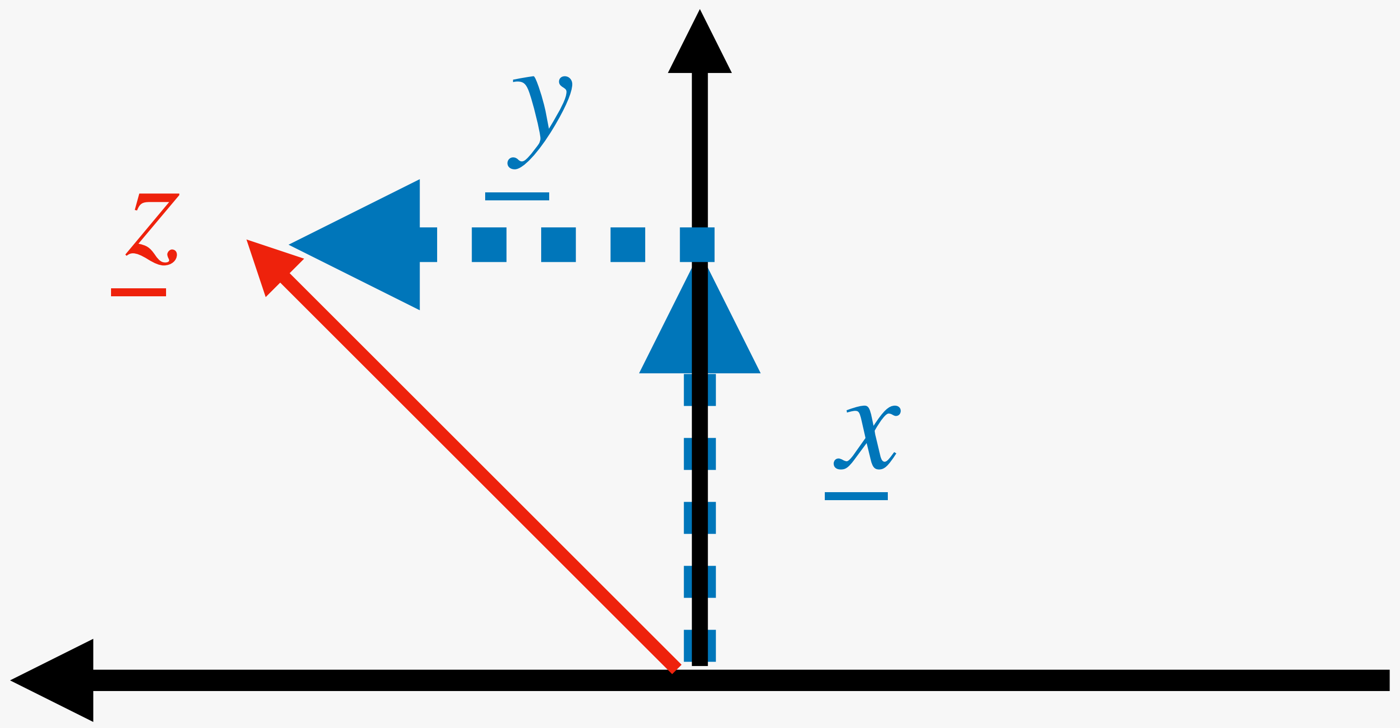
$$\underline{a} = [-2,4]$$
$$3\underline{a} = [-6,12]$$
$$||3\underline{a}|| = 3||\underline{a}||$$

Triangle inequality

$$||\underline{z} + \underline{y}|| \leq ||\underline{x}|| + ||\underline{y}||$$

if $\underline{z} = \underline{x} + \underline{y}$

No shortcuts:
going straight to a destination $\underline{z} = \underline{x} + \underline{y}$ *is shorter than via* $\underline{x}$



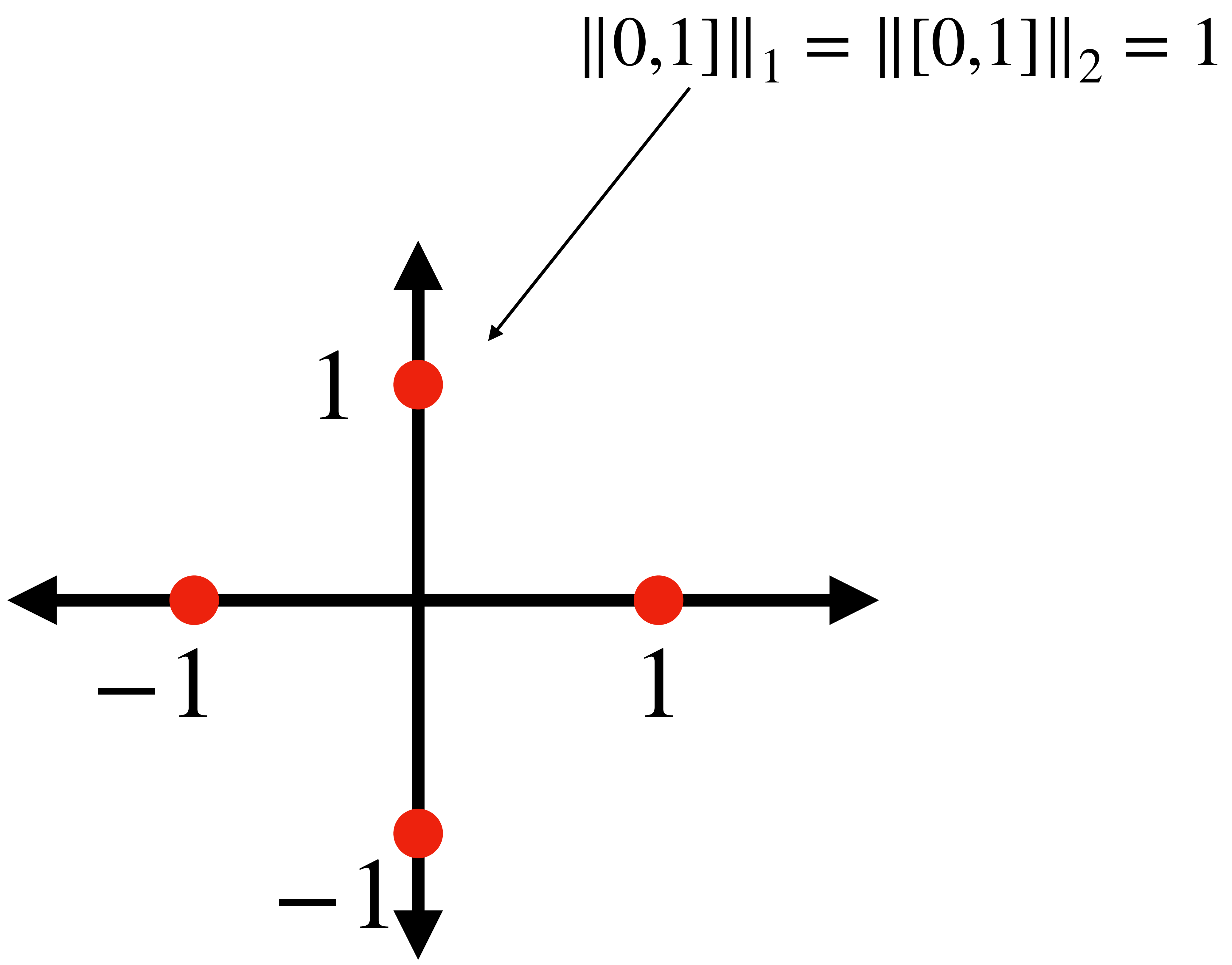
The magnitude/norm of vectors

Understanding

- *All the red points denote vectors with unit norm (i.e. norm = 1)*

- *Thus, called normalised vectors*

- *Trace the normalised vectors connecting the dots, in L1 and then L2*

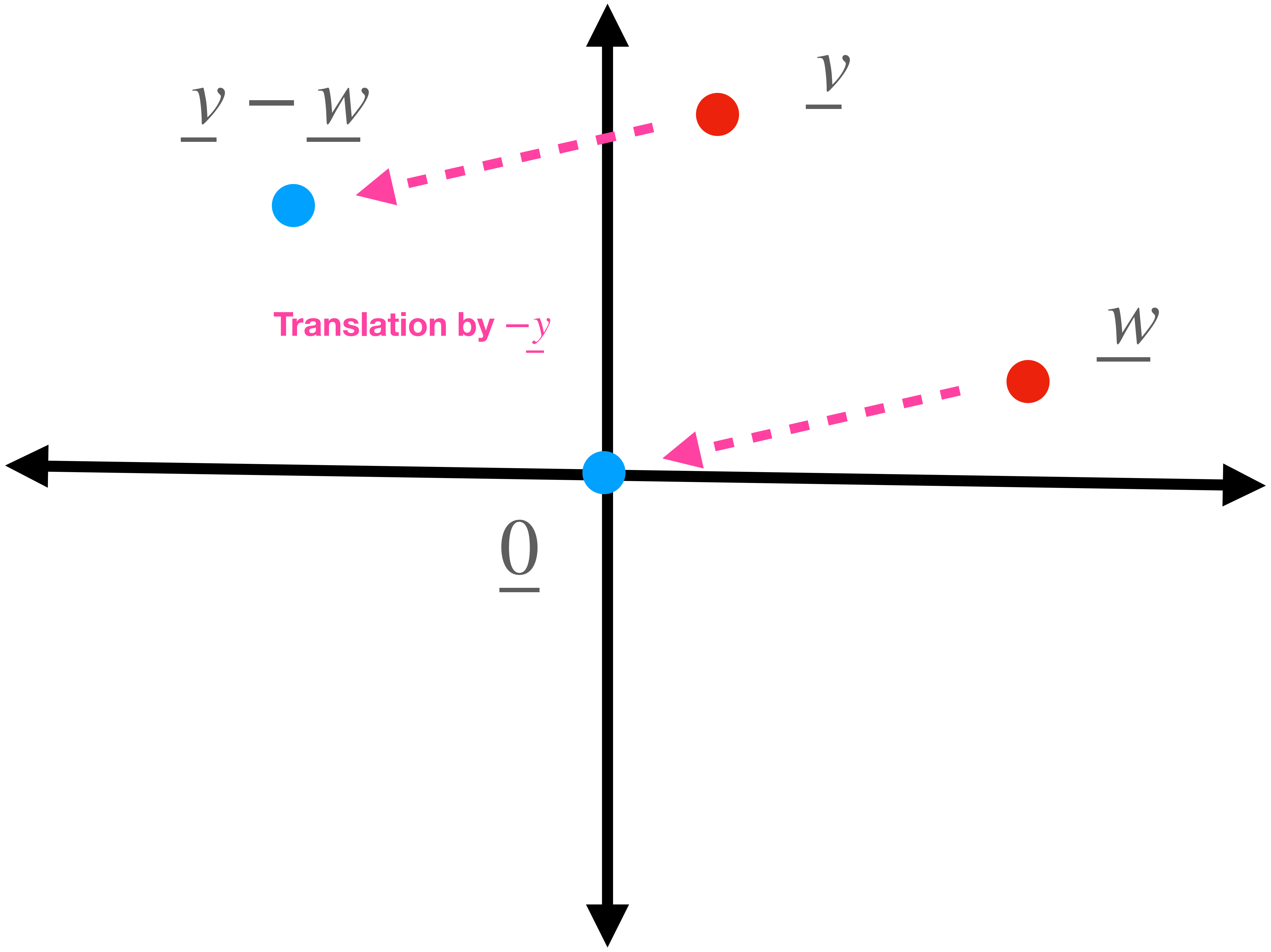


Distance between vectors

- $\|\underline{v}\| =$ *distance of* $\underline{v}$ *from zero*

- *Distance from* $\underline{w}$*?* $\|\underline{v} - \underline{w}\|$

- *Translating vectors equally doesn't change their distance*

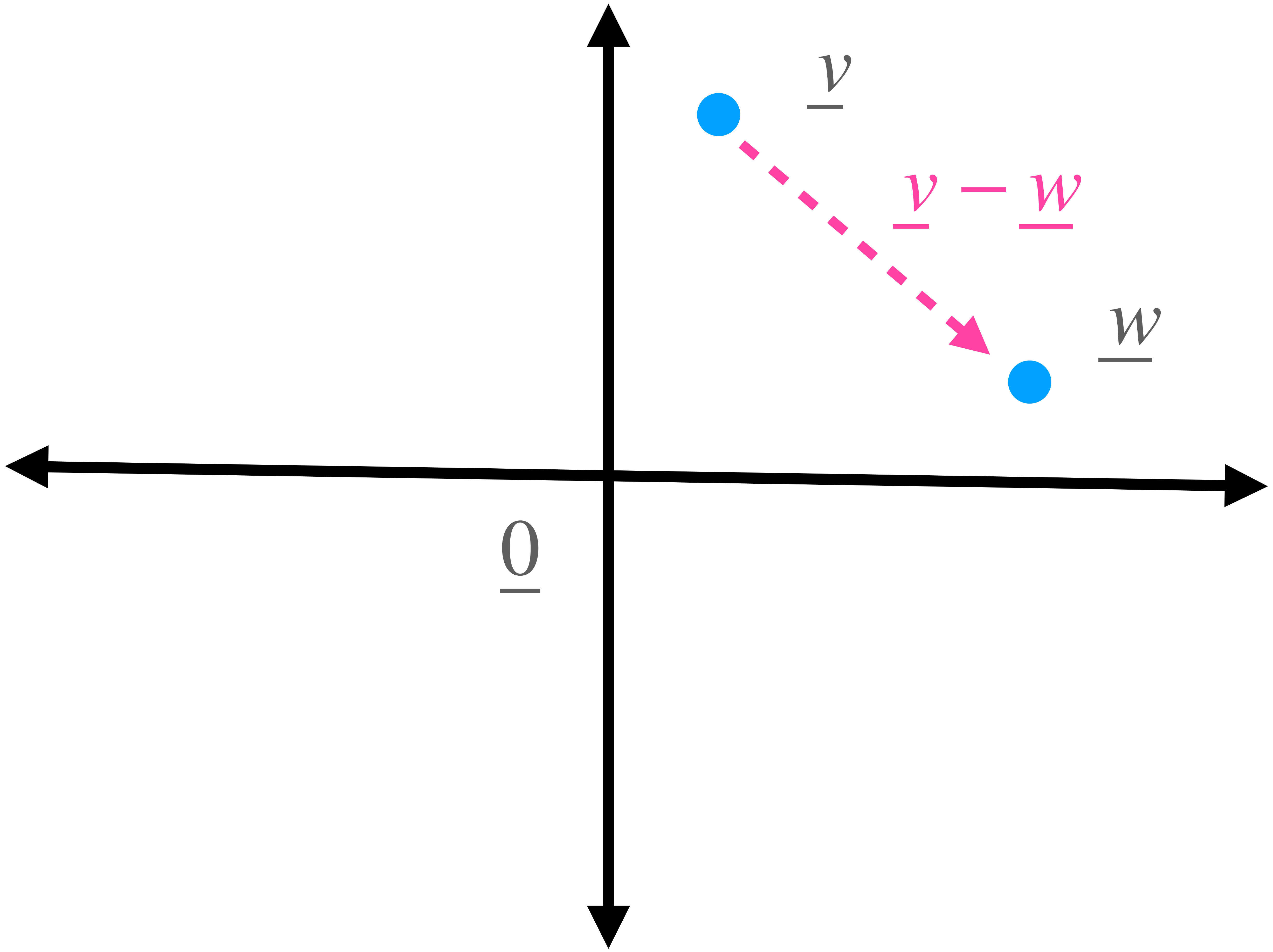


Distance between vectors

Euclidean distance

- *For length-2 vectors: their literal distance*

$$\|\underline{v} - \underline{w}\|_2 = \sqrt{(v_1 - w_1)^2 + (v_2 - w_2)^2}$$



New concept: correlation (similarity)

Maximally dissimilar (anti correlated) images

Pixel is (quite) white in 1 <=> Pixel is (quite) black in 2

New concept: correlation (similarity)

Uncorrelated images

Pixel is (quite) white in 1 <=> No information on 2