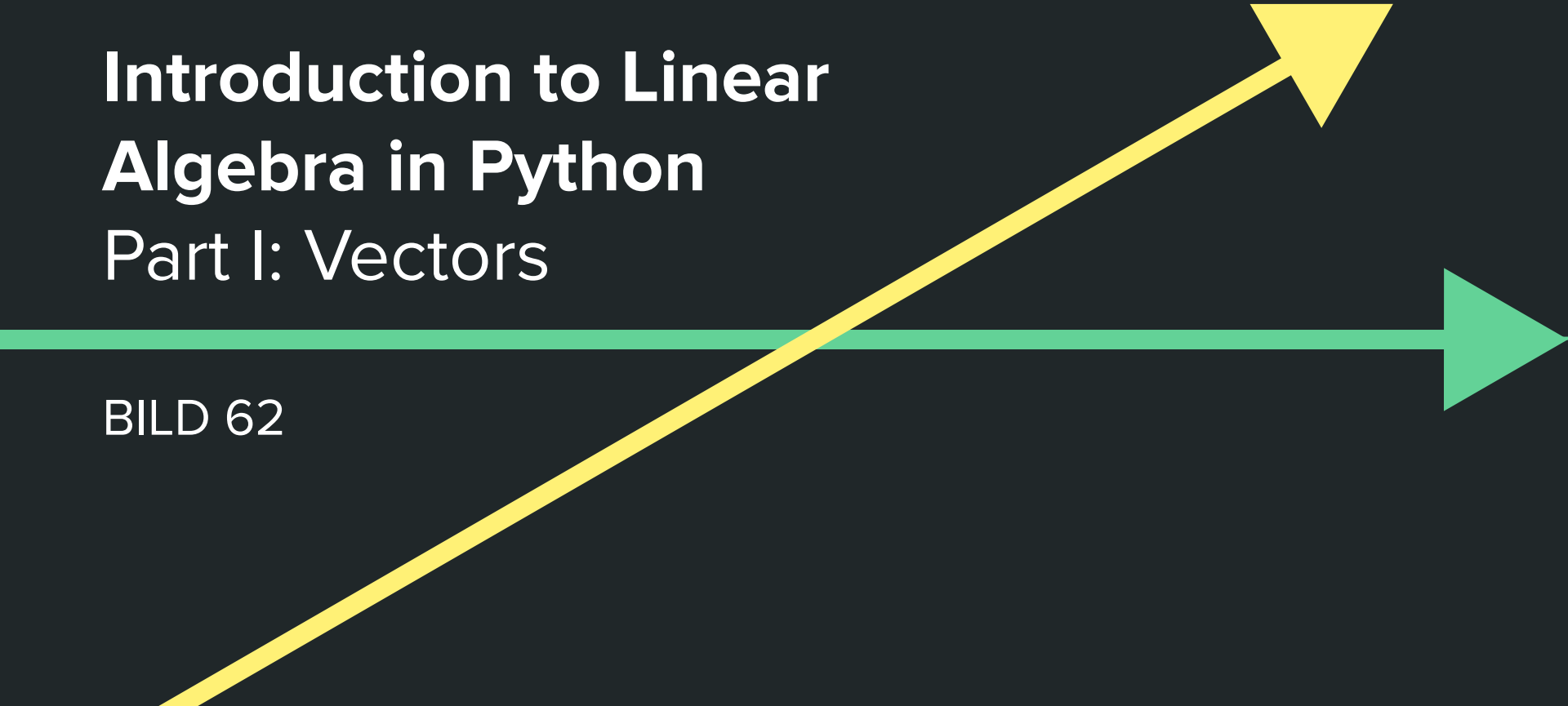


Introduction to Linear Algebra in Python

Part I: Vectors

BILD 62



In order to describe the mutual impact of computer science and mathematics on each other, and their relative roles, I am therefore looking somewhat to the future, to the time when computer science is a bit more mature and sure of itself. Recent trends have made it possible to envision a day when computer science and mathematics will both exist as respected disciplines, serving analogous but different roles in a person's education. To quote George Forsythe again, “The most valuable acquisitions in a scientific or technical education are the general-purpose mental tools which remain serviceable for a lifetime. I rate natural language and mathematics as the most important of these tools, and computer science as a third” [9].

“Computer science and its relation to mathematics”

(Knuth, Computer Scientist & Mathematician, 1974;
George Forsyth coined “computer science” as a term)

3. Educational side-effects. A person well-trained in computer science knows how to deal with algorithms: how to construct them, manipulate them, understand them, analyze them. This knowledge prepares him for much more than writing good computer programs; it is a general-purpose mental tool which will be a definite aid to his understanding of other subjects, whether they be chemistry, linguistics, or music, etc. The reason for this may be understood in the following way: It has often been said that a person does not really understand something until he teaches it to someone else. Actually a person does not *really* understand something until he can teach it to a *computer*, i.e., express it as an algorithm. “The automatic computer really *forces* that precision of thinking which is alleged to be a product of any study of mathematics” [7]. The attempt to formalize things as algorithms leads to a much deeper understanding than if we simply try to comprehend things in the traditional way.

“Computer science and its relation to mathematics”

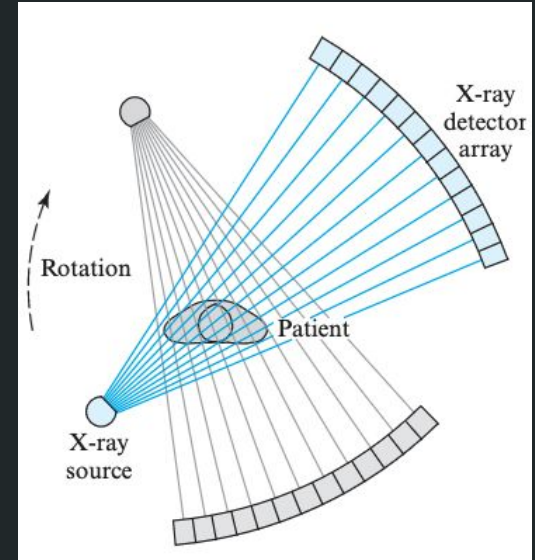
(Knuth, Computer Scientist & Mathematician, 1974)

By the end of
this lecture, you
will be able to:

- Provide an example of how linear algebra is used in biology
 - Describe vectors, their properties (dimensionality/length), and their operations (scalar multiplication, vector addition, dot product)
 - Assess these properties and implement these operations in Python
-

Why linear algebra?

- The “language of data”!
- Useful in a variety of contexts, from simplifying large datasets to modeling populations of animals
- It's how we can find solutions to multiple **linear** equations



Linear algebra can be used to solve for angles in a CAT scan

What do we mean by “linear equations”?

In two dimensions, a line in a rectangular xy-coordinate system can be represented by an equation of the form:

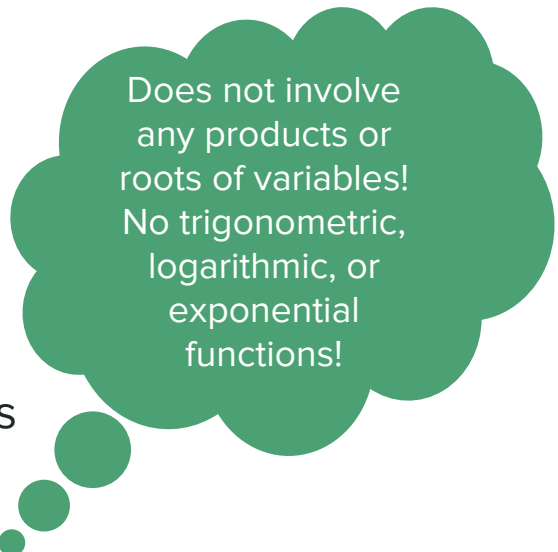
$$ax + by = c \quad (a, b \text{ not both } 0)$$

In three dimensions a plane in a rectangular xyz-coordinate system can be represented by an equation of the form:

$$ax + by + cz = d \quad (a, b, c \text{ not all } 0)$$

More generally, we define a linear equation in the n variables x_1, x_2, \dots, x_n to be one that can be expressed in the form:

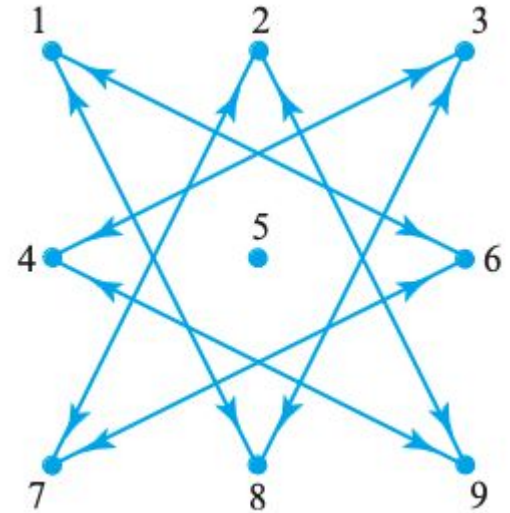
$$a_1x_1 + a_2x_2 + \dots + a_nx_n = b$$



Does not involve
any products or
roots of variables!
No trigonometric,
logarithmic, or
exponential
functions!

Additional use cases of linear algebra in biology

- Modeling of population dynamics
- Analysis of food web dynamics & ecology
- Genetics & DNA sequencing (e.g., sequence alignment)
- Metabolic pathway analysis
- Dimensionality reduction!!!!
 - Protein structure & folding
 - Neurophysiology data
- Image analysis (convolution, matrix transformations)



Suppose that a farmer has a large population of plants consisting of some distribution of all three possible genotypes AA , Aa , and aa . The farmer desires to undertake a breeding program in which **each plant in the population is always fertilized with a plant of genotype AA** and is then replaced by one of its offspring.

We want to derive an expression for the distribution of the three possible genotypes in the population after any number of generations.



Solving punnett squares over generations with linear algebra

Genotype of Offspring	Genotypes of Parents					
	<i>AA-AA</i>	<i>AA-Aa</i>	<i>AA-aa</i>	<i>Aa-Aa</i>	<i>Aa-aa</i>	<i>aa-aa</i>
<i>AA</i>	1	$\frac{1}{2}$	0	$\frac{1}{4}$	0	0
<i>Aa</i>	0	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$	0
<i>aa</i>	0	0	0	$\frac{1}{4}$	$\frac{1}{2}$	1

probability of observing genotype

Formalizing our problem...

a_n = fraction of plants of genotype AA in n th generation

b_n = fraction of plants of genotype Aa in n th generation

c_n = fraction of plants of genotype aa in n th generation

One additional rule...


$$a_n + b_n + c_n = 1 \quad \text{for } n = 0, 1, 2, \dots$$

We can determine the genotype distribution of each generation from the genotype distribution of the *preceding* generation by the following system of linear equations:

$$a_n = a_{n-1} + \frac{1}{2}b_{n-1}$$

$$b_n = c_{n-1} + \frac{1}{2}b_{n-1}$$

$$c_n = 0$$

- 
1. All the offspring of a plant of genotype AA will be AA
 2. Half of the offspring of a plant of Aa will be AA

Putting this in the language of linear algebra

$$\mathbf{x}^{(n)} = M\mathbf{x}^{(n-1)}, \quad n = 1, 2, \dots$$

where

$$\mathbf{x}^{(n)} = \begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix}, \quad \mathbf{x}^{(n-1)} = \begin{bmatrix} a_{n-1} \\ b_{n-1} \\ c_{n-1} \end{bmatrix}, \quad \text{and} \quad M = \begin{bmatrix} 1 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

coefficients matrix

First three columns of our table, AND the coefficients in our three equations!


Example from [Elementary Linear Algebra](#)

Putting this in the language of linear algebra

$$\mathbf{x}^{(n)} = M\mathbf{x}^{(n-1)}, \quad n = 1, 2, \dots$$

where

$$\mathbf{x}^{(n)} = \begin{bmatrix} a_n \\ b_n \\ c_n \end{bmatrix}, \quad \mathbf{x}^{(n-1)} = \begin{bmatrix} a_{n-1} \\ b_{n-1} \\ c_{n-1} \end{bmatrix}, \quad \text{and} \quad M = \begin{bmatrix} 1 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

 **column vectors**

matrix

Example from [Elementary Linear Algebra](#)

Vectors



$$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$\vec{v} \in V$$

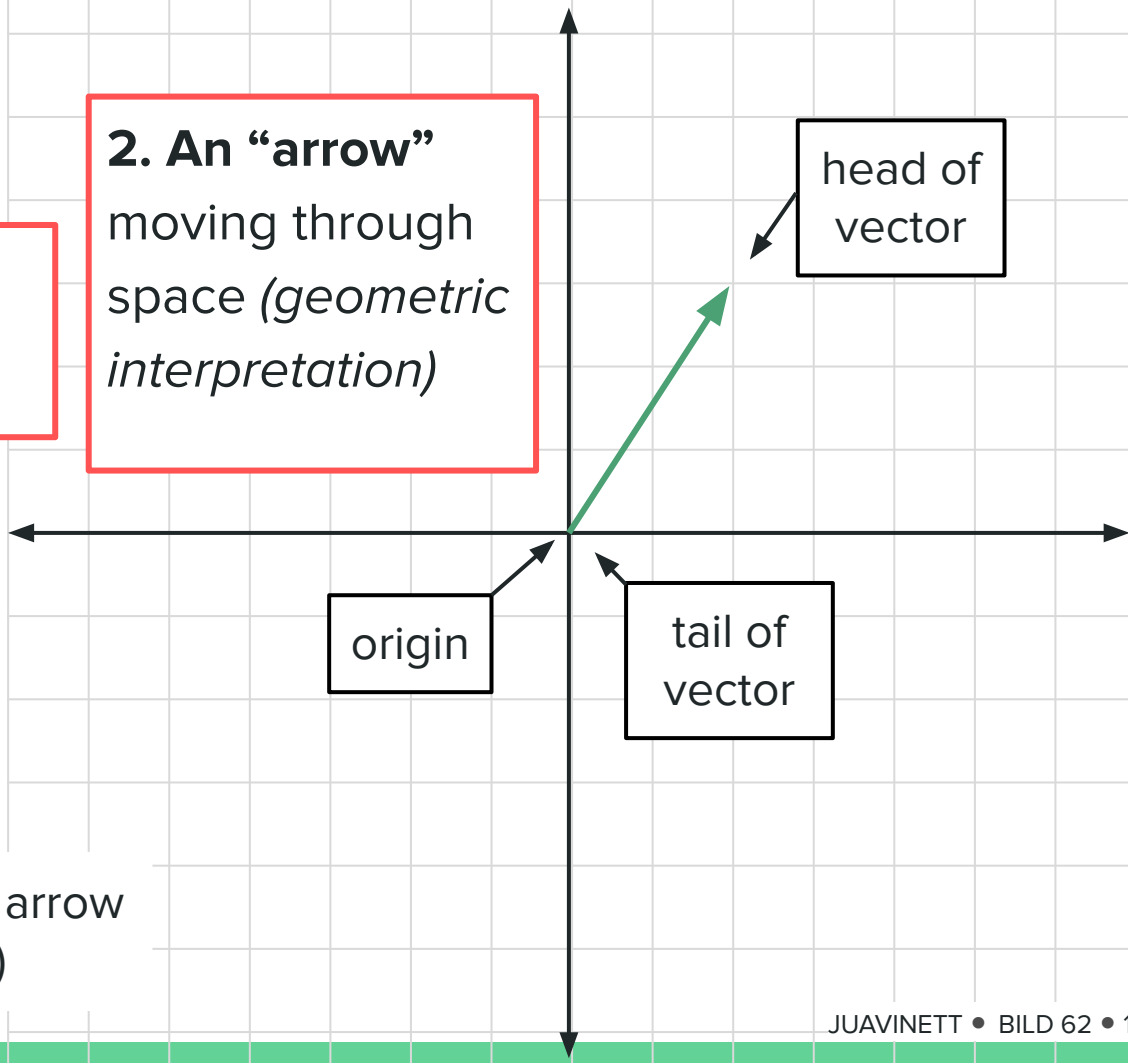
What is a vector? (From [this fantastic YouTube series](#))

Two views of a vector

1. Ordered list of numbers
(algebraic interpretation)

x coordinate → $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$
y coordinate →

2. An “arrow”
moving through
space *(geometric interpretation)*



(We can represent the tip of the arrow
with an ordered list of numbers!)

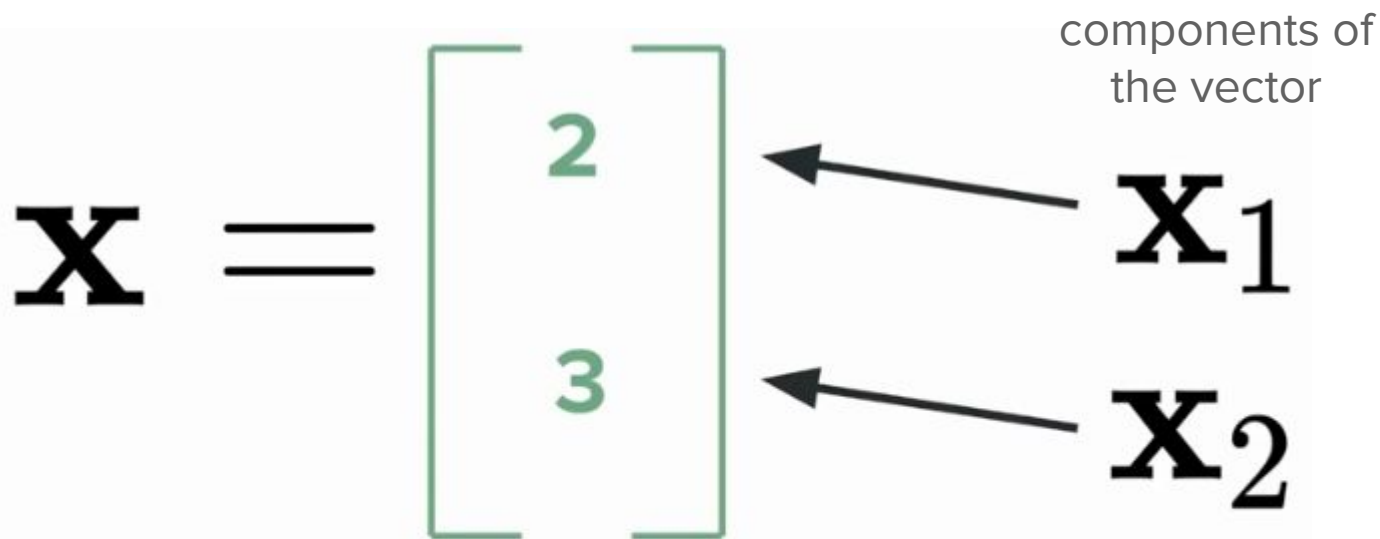
Two views of a vector

1. Ordered list of numbers

(algebraic interpretation)

$$\begin{array}{lcl} \text{x coordinate} & \longrightarrow & \begin{bmatrix} 2 \\ 3 \end{bmatrix} \\ \text{y coordinate} & \longrightarrow & \end{array}$$

Vector notation (there isn't *one* standard!)



The diagram illustrates a vector \mathbf{x} represented as a column matrix with two components, 2 and 3, enclosed in green brackets. To the right of the matrix, the labels \mathbf{x}_1 and \mathbf{x}_2 are shown. Arrows point from \mathbf{x}_1 to the top component (2) and from \mathbf{x}_2 to the bottom component (3). Above these labels, the text "components of the vector" is written.

$$\mathbf{x} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

components of the vector

\mathbf{x}_1

\mathbf{x}_2

Some people use \vec{x}

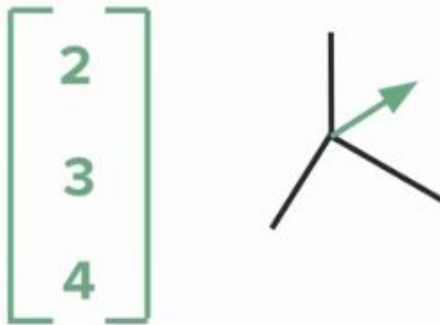
Vector property: **Dimensionality**

(the number of numbers in the vector)

Two dimensional



Three dimensional

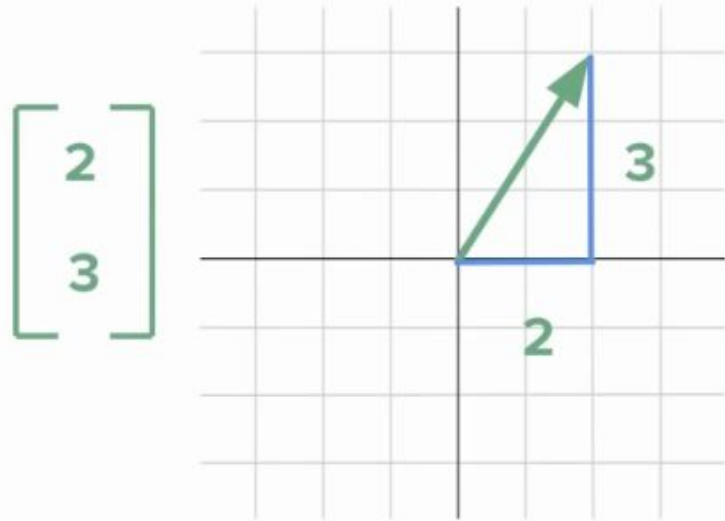


N dimensional



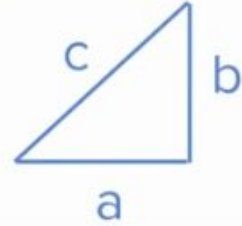
Note: this is different than dimensionality in code!
Mathematical dimensionality is **length** or **shape** in Python.

Vector Property: **Length**



- The length of a vector can also be considered the **magnitude** or **norm**.
- The **norm** is a measure of the size or length of a vector or matrix in linear algebra.
- Norm is a generalization of the concept of the magnitude of a vector in Euclidean space.

Pythagorean Theorem



$$c = \sqrt{a^2 + b^2}$$

$$\text{Length of } \mathbf{x} = \sqrt{2^2 + 3^2}$$

$$\|\mathbf{x}\| = \sqrt{13}$$

//

Vector property: **Orientation**

whether the vector is in **column orientation (standing up tall)** or in **row orientation (flat and wide)**

Are these the same?
Does it matter?
Sometimes yes, sometimes no.

column vector

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

We can move
between these with
transposition

row vector

$$\mathbf{x}^T = \begin{pmatrix} 4 & 1 & 2 \end{pmatrix}$$


Note: The convention in linear algebra is to *assume* vectors are in column orientation, unless otherwise specified.

A brief introduction to matrices

2 x 3 matrix

Matrix A has 2 rows and 3 columns, can be indexed as $A_{i,j}$, where i is the row number and j is the column number.

For example, $A_{1,2} = 4$ and $A_{2,3} = 3$.

$$A = \begin{pmatrix} 2 & 4 & 8 \\ 1 & 7 & 3 \end{pmatrix}$$


Note: Be mindful of indexing differences when translating formulas into Python code!

Special matrices

Square matrices are those with the same number of row and column. $m = n$. For example,

$$A = \begin{pmatrix} 2 & 4 \\ 1 & 5 \end{pmatrix}, b = \begin{pmatrix} 2 & 4 & 8 \\ 1 & 7 & 3 \\ 2 & 5 & 6 \end{pmatrix}$$

Diagonal matrices are square matrices with only the values along the main diagonal are non-zero. For example,

$$C = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 6 \end{pmatrix}$$

Identity matrices are diagonal matrices where all the non-zero values are 1. For example,

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matrix transposition

Transposition flips rows and columns — each row of the original matrix becomes the corresponding column of the new matrix

$$A = \begin{pmatrix} 2 & 4 & 8 \\ 1 & 7 & 3 \end{pmatrix} \longleftrightarrow A^T = \begin{pmatrix} 2 & 1 \\ 4 & 7 \\ 8 & 3 \end{pmatrix}$$

We can move between these with **transposition**

Special vectors

- **Zero vector:** vector with length 0

$$\mathbf{y} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{Each component is zero!})$$

- **Unit vector:** vector with length 1

(or with a “hat” instead of a tilde)

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

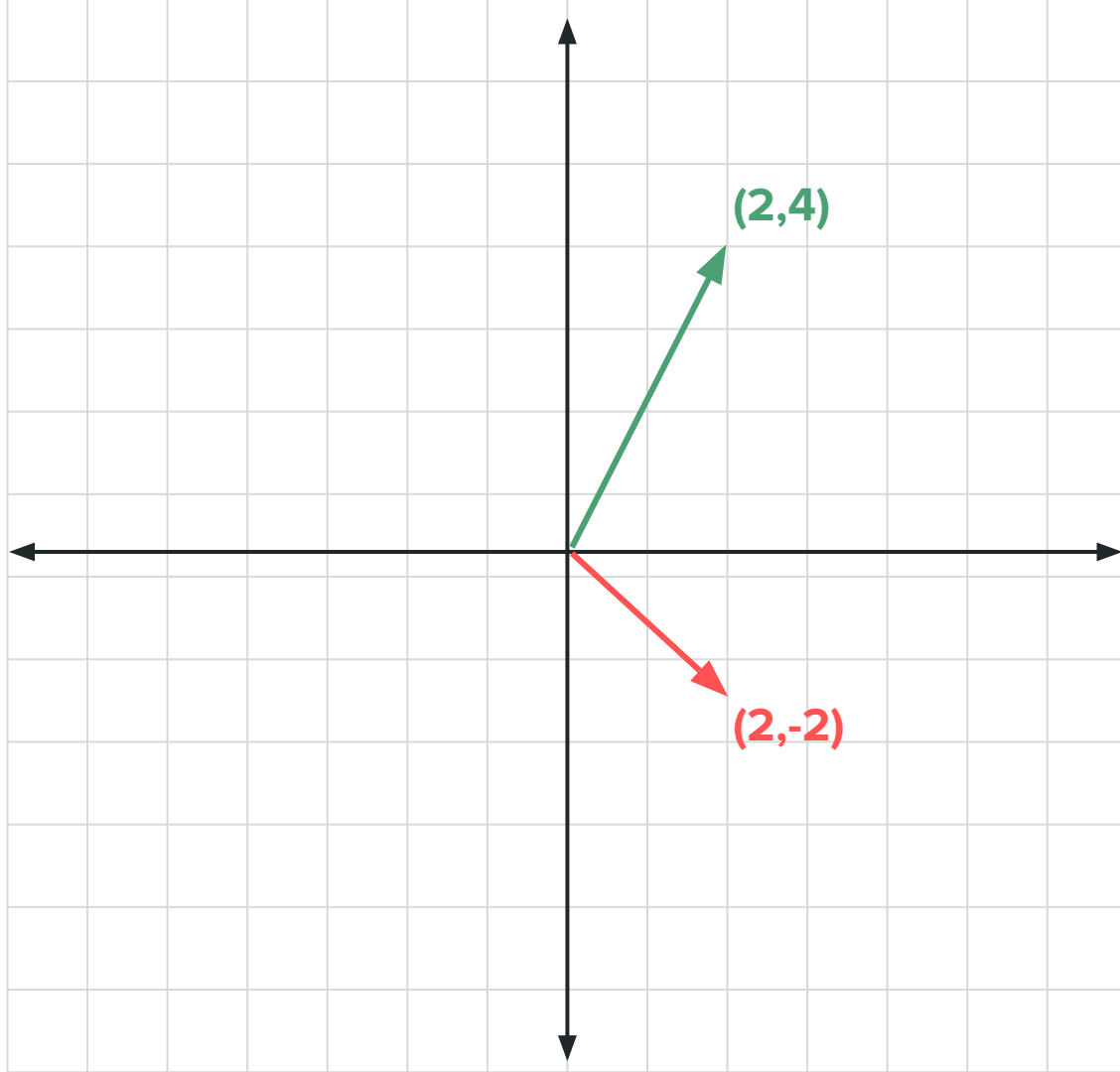
You can turn any vector into a unit vector by dividing **each of its components** by the length of the vector

$$\mathbf{x} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

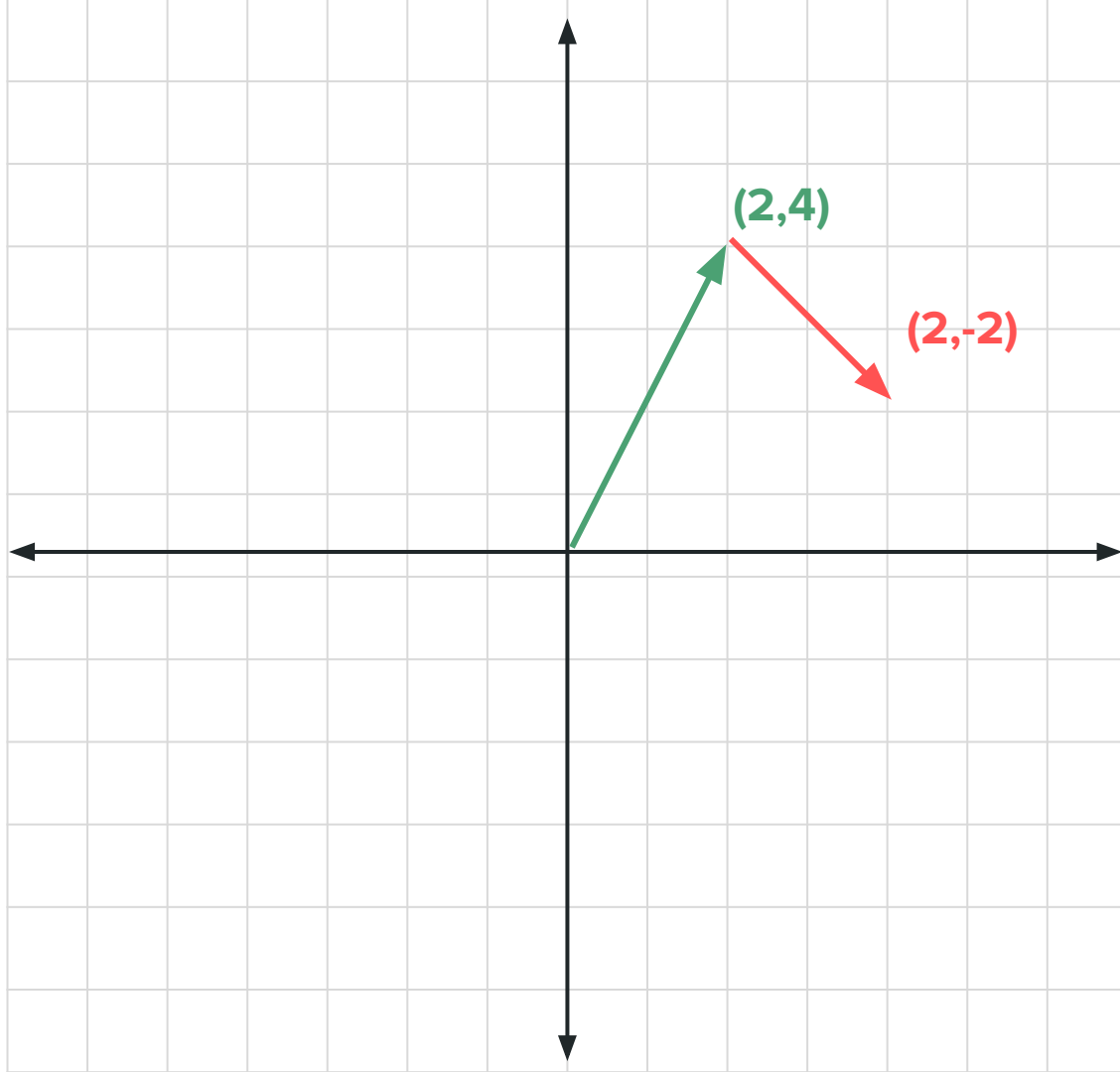
$$\|\mathbf{x}\| = \sqrt{13}$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} \frac{2}{\sqrt{13}} \\ \frac{3}{\sqrt{13}} \end{bmatrix}$$

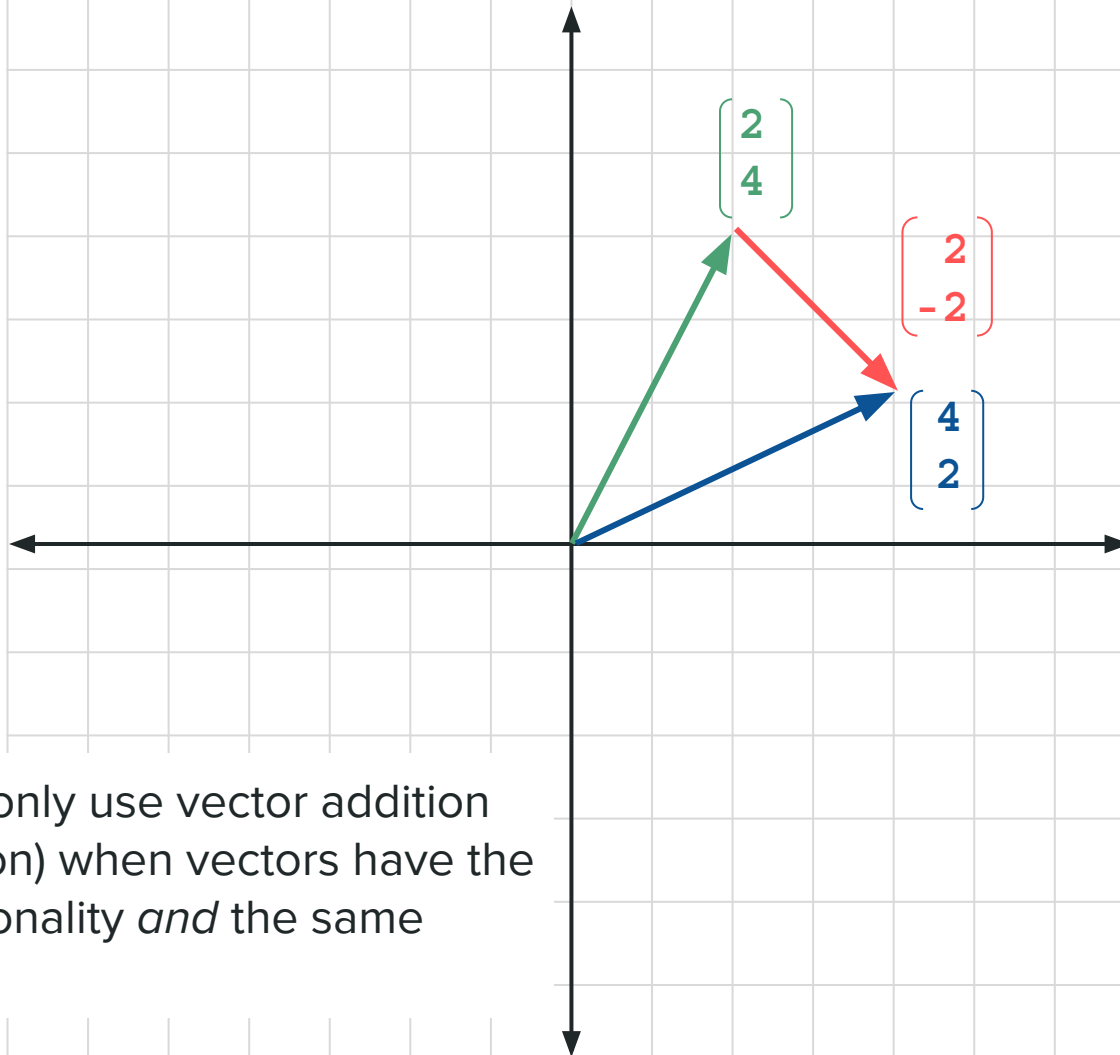
Vector addition



Vector addition

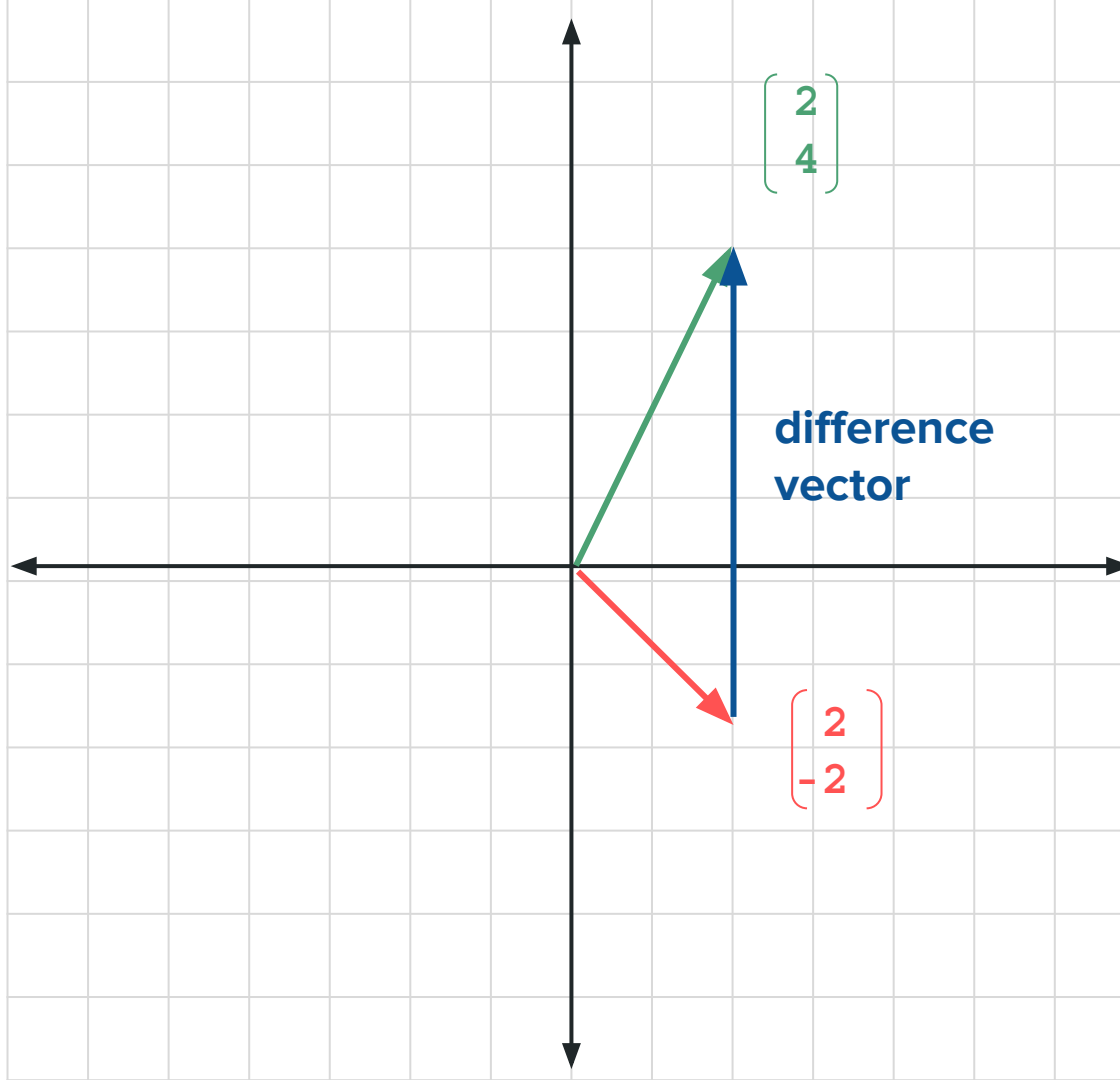


Vector addition



Note: we can only use vector addition (and subtraction) when vectors have the same dimensionality *and* the same orientation.

Vector subtraction



Vector subtraction

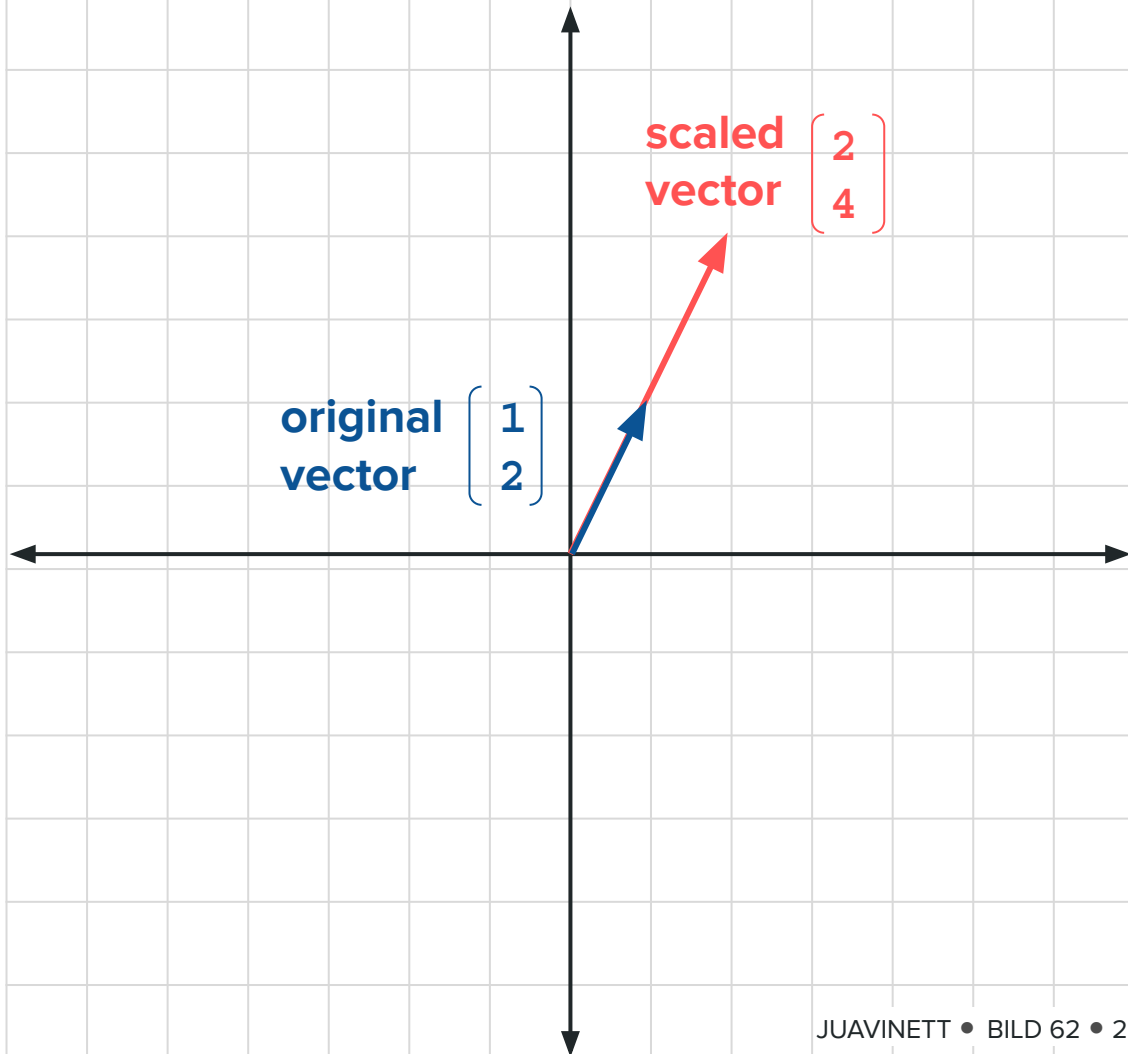
$$\begin{pmatrix} 2 \\ 4 \end{pmatrix} - \begin{pmatrix} 2 \\ -2 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

difference
vector

Vector-Scalar Multiplication

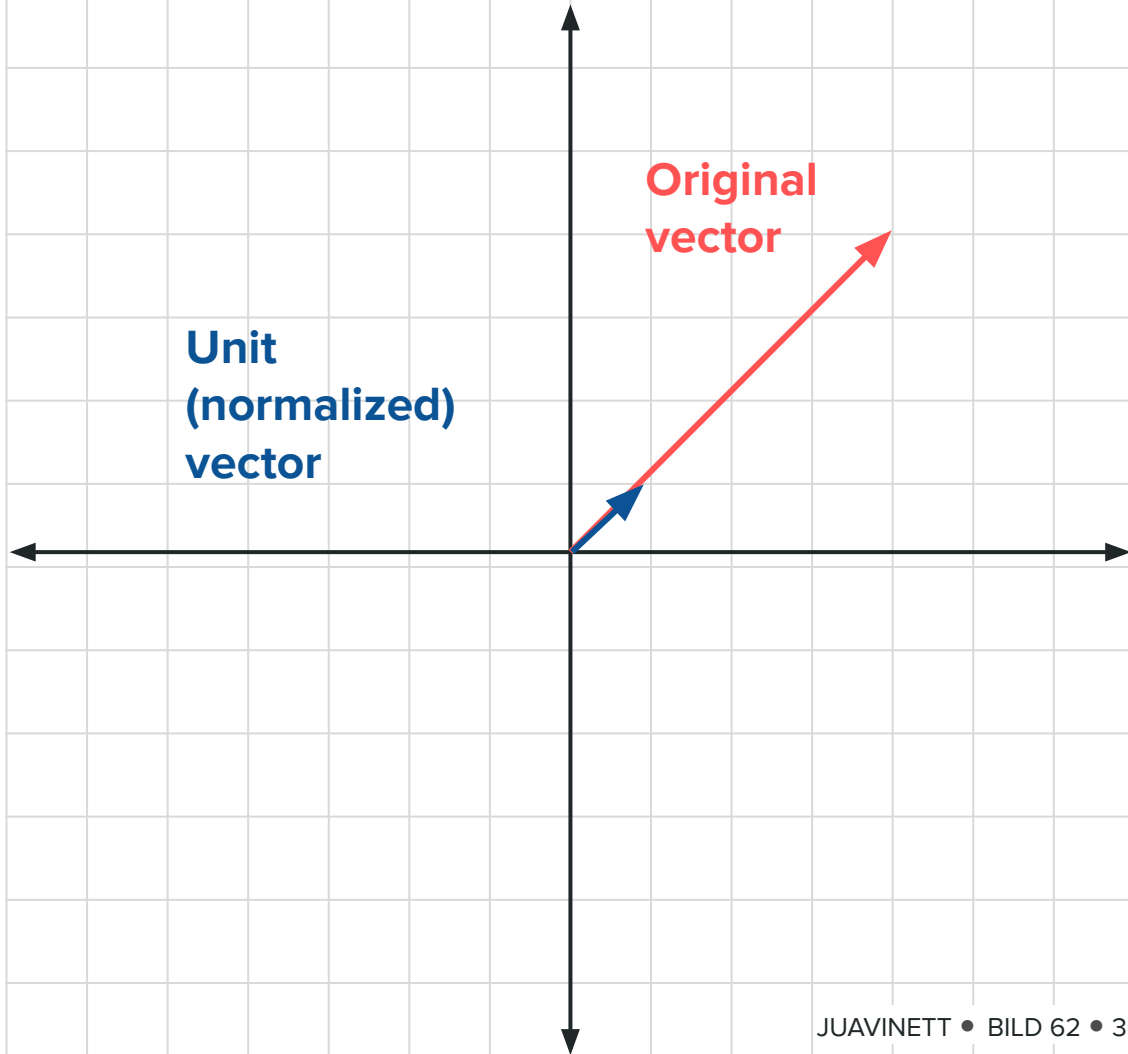
- “Scalars” are non-dimensional quantities
- Typically indicated by lowercase Greek letters such as α or λ
- Multiply each vector element by the scalar.

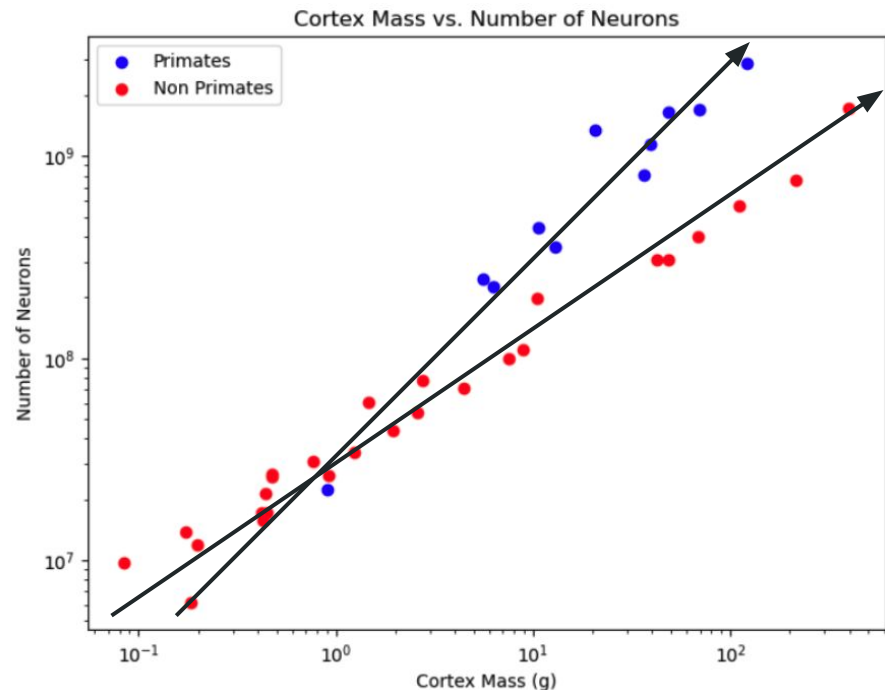


Vector Normalization

- Dividing a vector by its magnitude to generate a unit vector (with magnitude 1)

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{||\mathbf{x}||}$$





What does all of this have to do with *data*?

Well, what if we wanted to know *how correlated* cortex mass and number of neurons were?

We could compute the **Pearson correlation coefficient!**

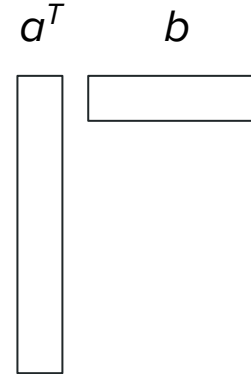
The **dot product**: a single number that provides information about the relationship between two vectors or matrices

Dot operator

You might also see
 $a^T b$ or $\langle a, b \rangle$

transpose

$$a \cdot b = \sum_{i=1}^n a_i b_i$$



a = 1st vector

b = 2nd vector

n = dimension of the vector space

a_i = component of vector a

b_i = component of vector b

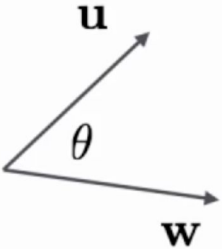
In other words, multiply the corresponding elements of two vectors, and then sum over all of the individual products.

In other *other* words, element-wise multiplication & sum.

The **dot product**: a single number that provides information about the relationship between two vectors or matrices


Another way to write this...

Gives intuition for why dot product is 0 when angle is 90! (because $\cos(90) = 0$!)

$$\mathbf{u} \cdot \mathbf{w} = \|\mathbf{u}\| \|\mathbf{w}\| \cos(\theta)$$


The diagram illustrates the geometric interpretation of the dot product. It shows two vectors, \mathbf{u} and \mathbf{w} , originating from a common point. Vector \mathbf{u} points upwards and to the right, while vector \mathbf{w} points downwards and to the right. The angle between the two vectors is labeled θ .

Example dot product calculation


$$[1 \ 2 \ 3 \ 4] \cdot [5 \ 6 \ 7 \ 8] = 1 \times 5 + 2 \times 6 + 3 \times 7 + 4 \times 8$$

$$= 5 + 12 + 21 + 32$$


1. Multiply corresponding elements of two vectors
2. Sum over all products

$$= 70$$

Okay, but why is the dot product useful?

- It tells us the similarity between vectors!
- Larger dot products mean the vectors are more similar.
- To make meaning of the absolute value of the dot product, we need to normalize it.
- The *normalized* dot product is the **Pearson correlation coefficient**.

Mean
subtracted
vector



$$r_{xy} = \frac{\vec{x}^c \cdot \vec{y}^c}{\|\vec{x}^c\| \|\vec{y}^c\|}$$

Curious about
the proof?

Additional resources

[Essence of linear algebra - YouTube](#)

[Tutorial 1: Vectors — Neuromatch Academy](#) ← much of this lecture was adopted from these materials!

Cohen, *Practical Linear Algebra for Data Science*

@ UCSD: MATH 18