Linear Algebra for Data Science

BIPN 162

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

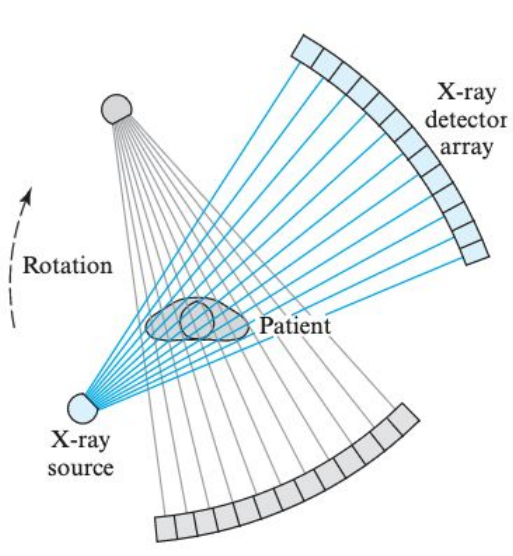
(in addition to the learning objectives for the take home videos!)

- Identify the possible uses of linear algebra in neuroscience
- Compute a dot product and explain its relationship to a correlation
- Construct and multiply matrices in Python (and by hand)
  - Create and manipulate special cases of matrices
- *Define what eigenvalues & eigenvectors are and determine them using Python*

Why linear algebra?

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:

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

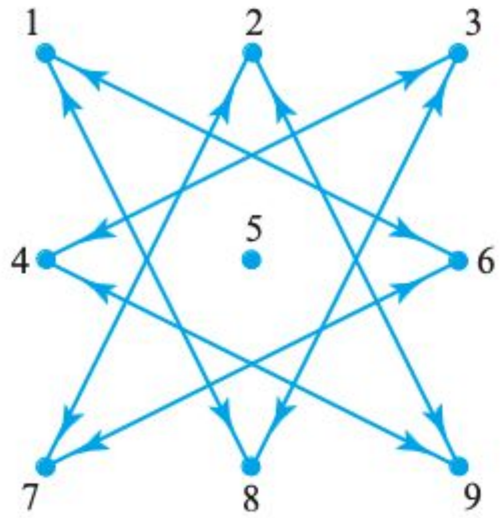
More generally, we define a linear equation in the n variables x1, x2, . . . , xn to be one that can be expressed in the form:

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

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

Additional use cases of linear algebra in biology

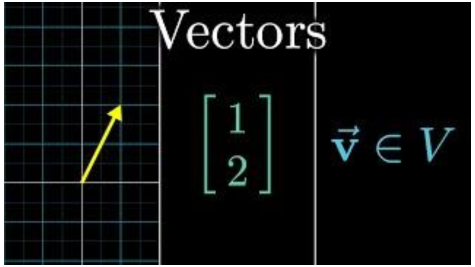
- Modeling of population dynamics
- Analysis of food web dynamics & ecology
- Genetics & DNA sequencing (e.g., sequence alignment)
- Metabolic pathway analysis
- Dimensionality reduction!!!!
    - Protein structure & folding
    - Neurophysiology data
- Image analysis (convolution, matrix transformations)

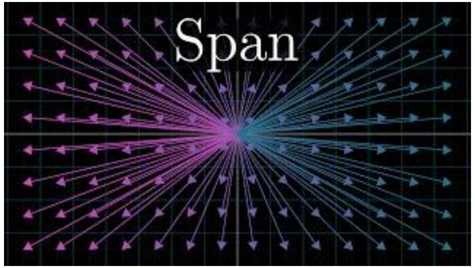


Why is linear algebra useful for (neuroscience) data?

Vectors

Pre-lecture viewing





After watching chapters 1 & 2 you should be able to:

- Describe vectors, their properties (**dimensionality**/length), and two of their operations: **addition & multiplication**
- Define **span** and **basis vectors**
- Determine and explain the number of basis vectors necessary for a given vector space

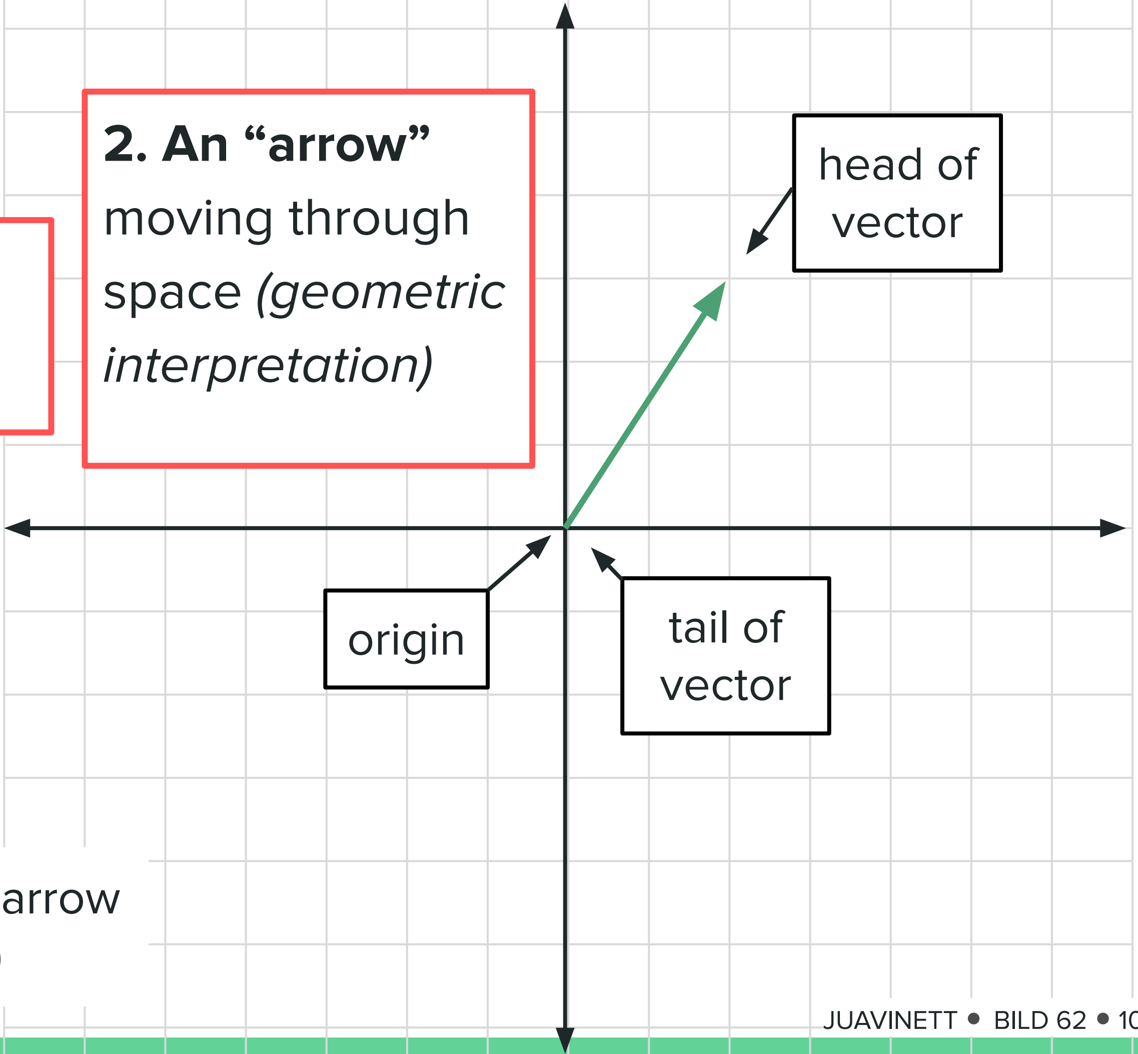
Two views of a vector

1. Ordered list of numbers

(algebraic interpretation)

$$\begin{array}{r l} \text{x coordinate} \rightarrow & \multirow{2}{*}{$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$} \\ \text{y coordinate} \rightarrow & \end{array}$$

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



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

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

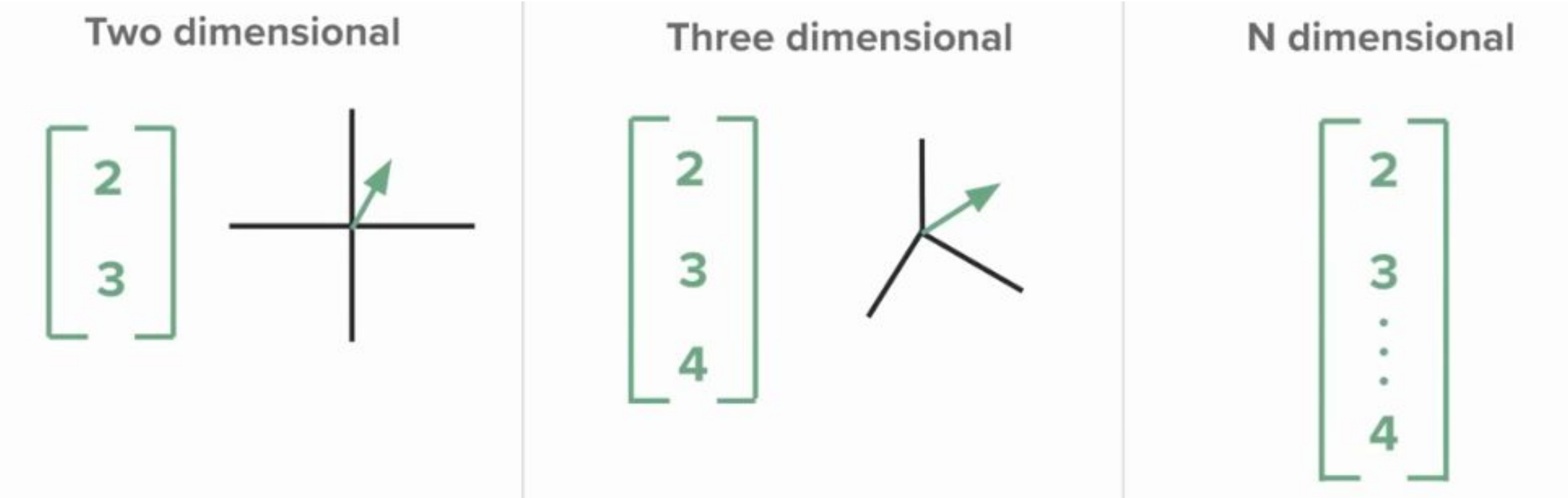
components of the vector

$$\begin{array}{cl} \mathbf{x} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} & \begin{array}{l} \mathbf{x}_1 \\ \mathbf{x}_2 \end{array} \end{array}$$

Some people use $\vec{x}$

Vector property: **Dimensionality**

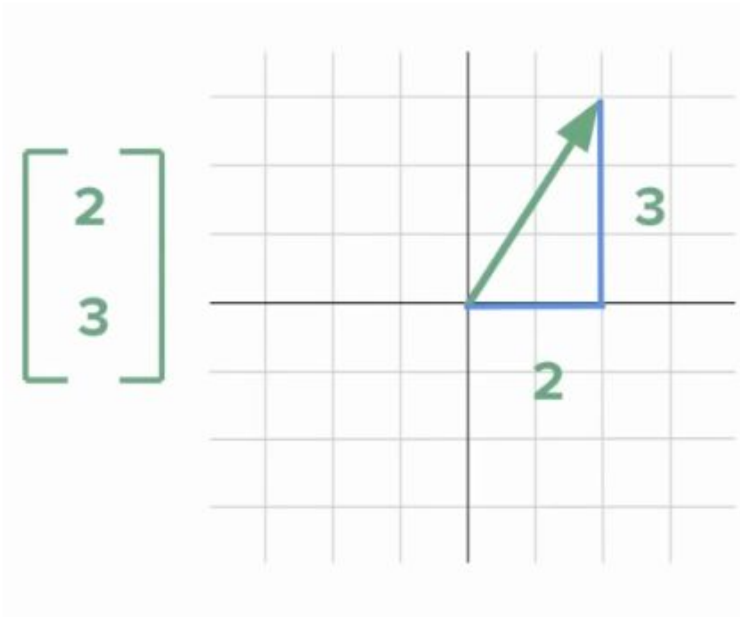
(the number of numbers in the vector)

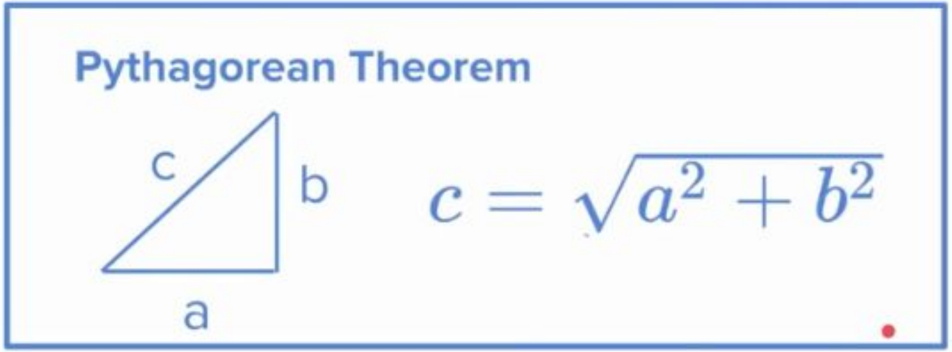


Note: this is different than dimensionality in code!
Mathematical dimensionality is **length** or **shape** in Python.

Vector Property: **Length**

(aka magnitude or <u>norm</u>)





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

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

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

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

We can move between these with **transposition**

row vector

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

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

Special vectors

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

$$\mathbf{y} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

(Each component is zero!)

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

(or with a "hat" instead of a tilde)

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

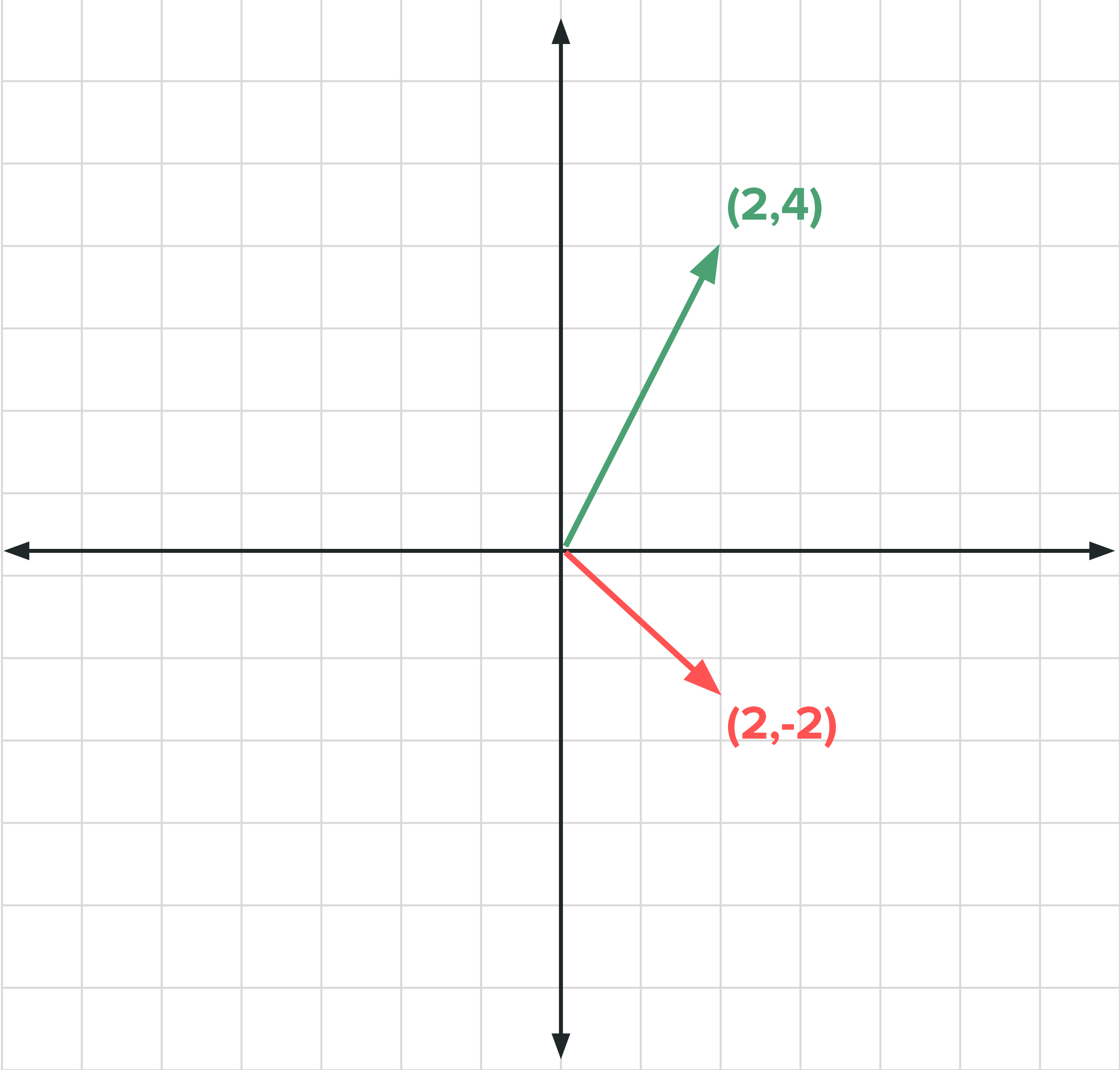
You can turn any vector into a unit vector by dividing **each of its components** by the length of the vector

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

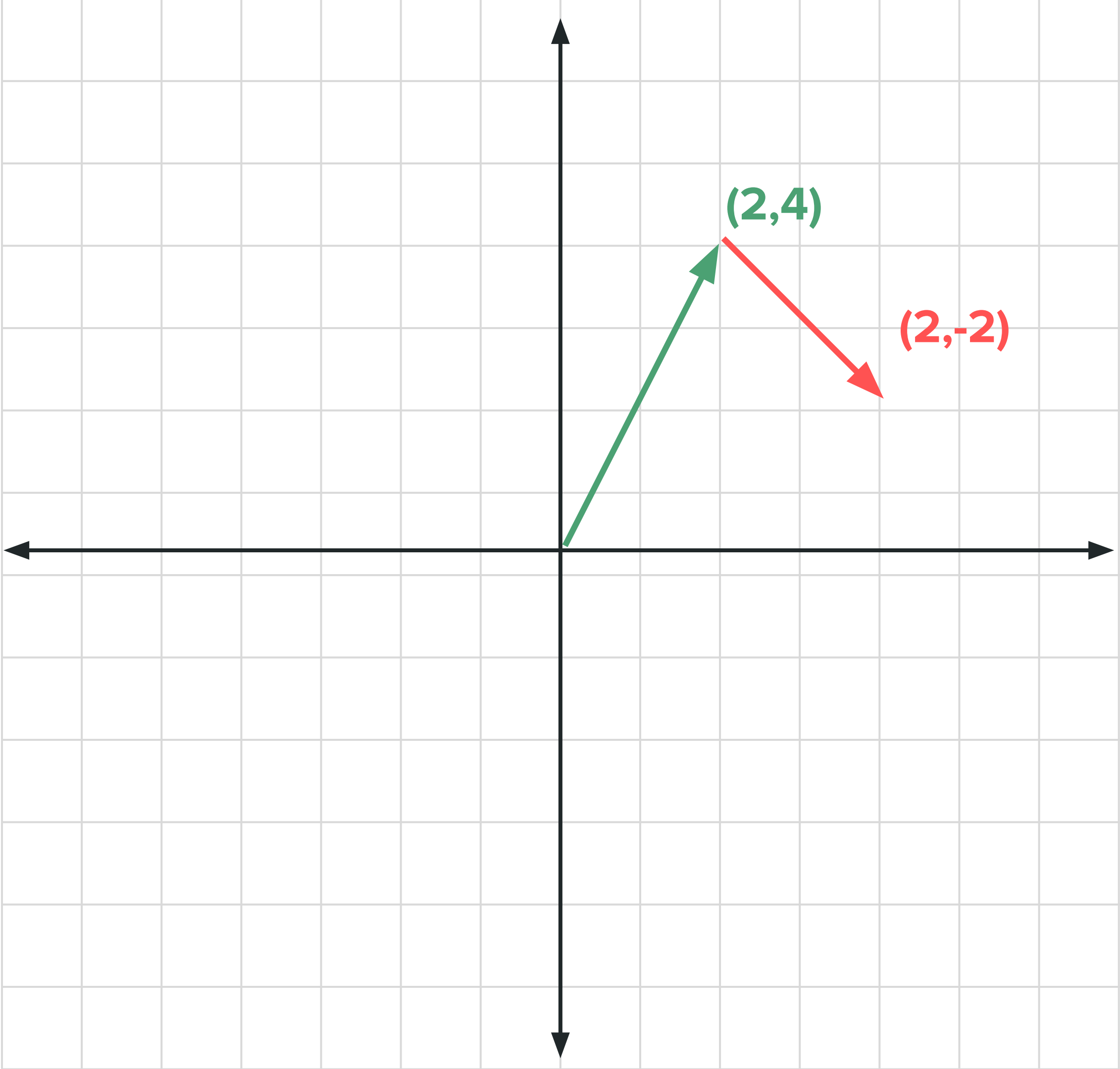
$$||\boldsymbol{x}|| = \sqrt{13}$$

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

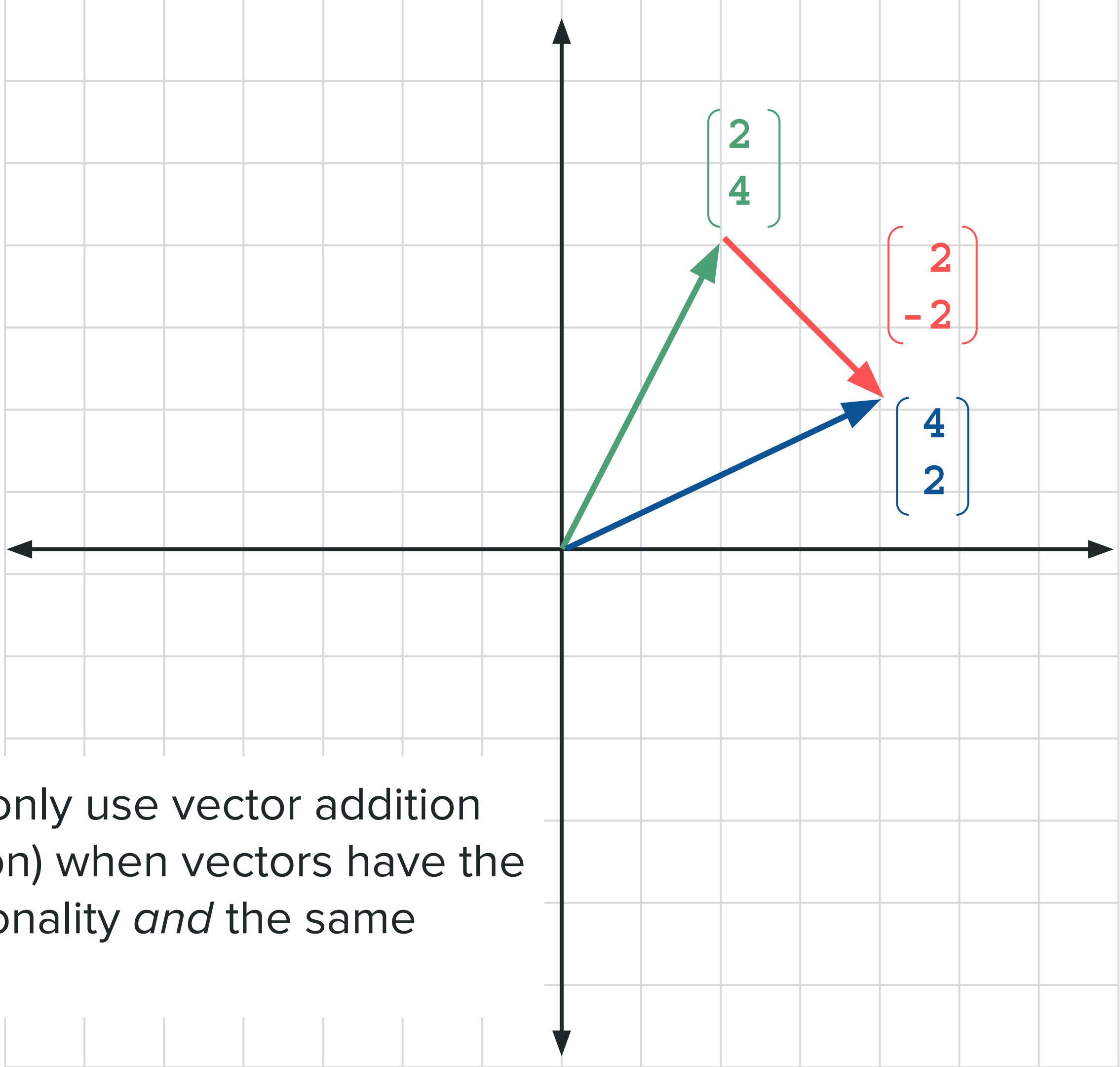
Vector addition



Vector addition

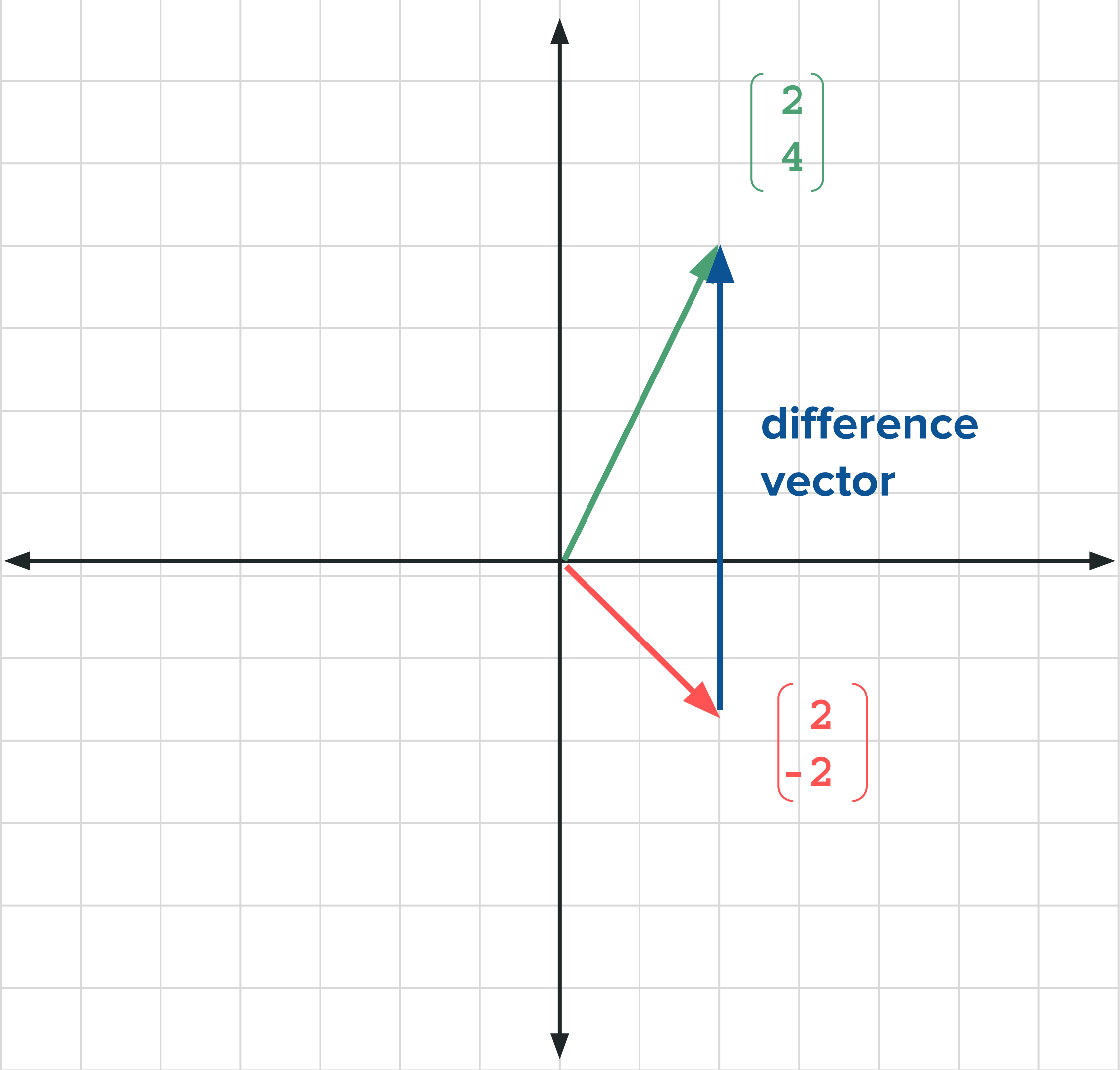


Vector addition



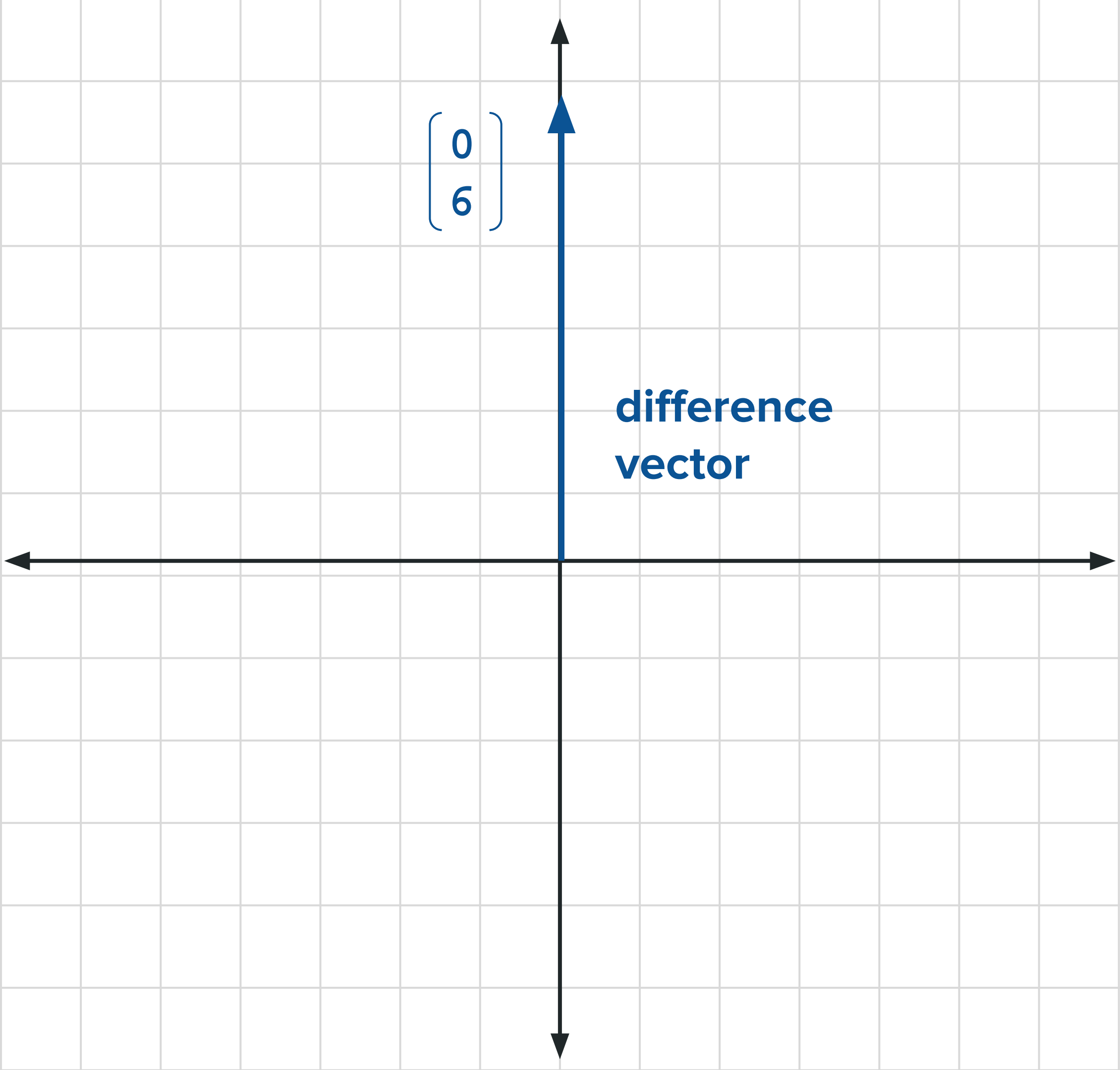
Note: we can only use vector addition (and subtraction) when vectors have the same dimensionality *and* the same orientation.

Vector subtraction



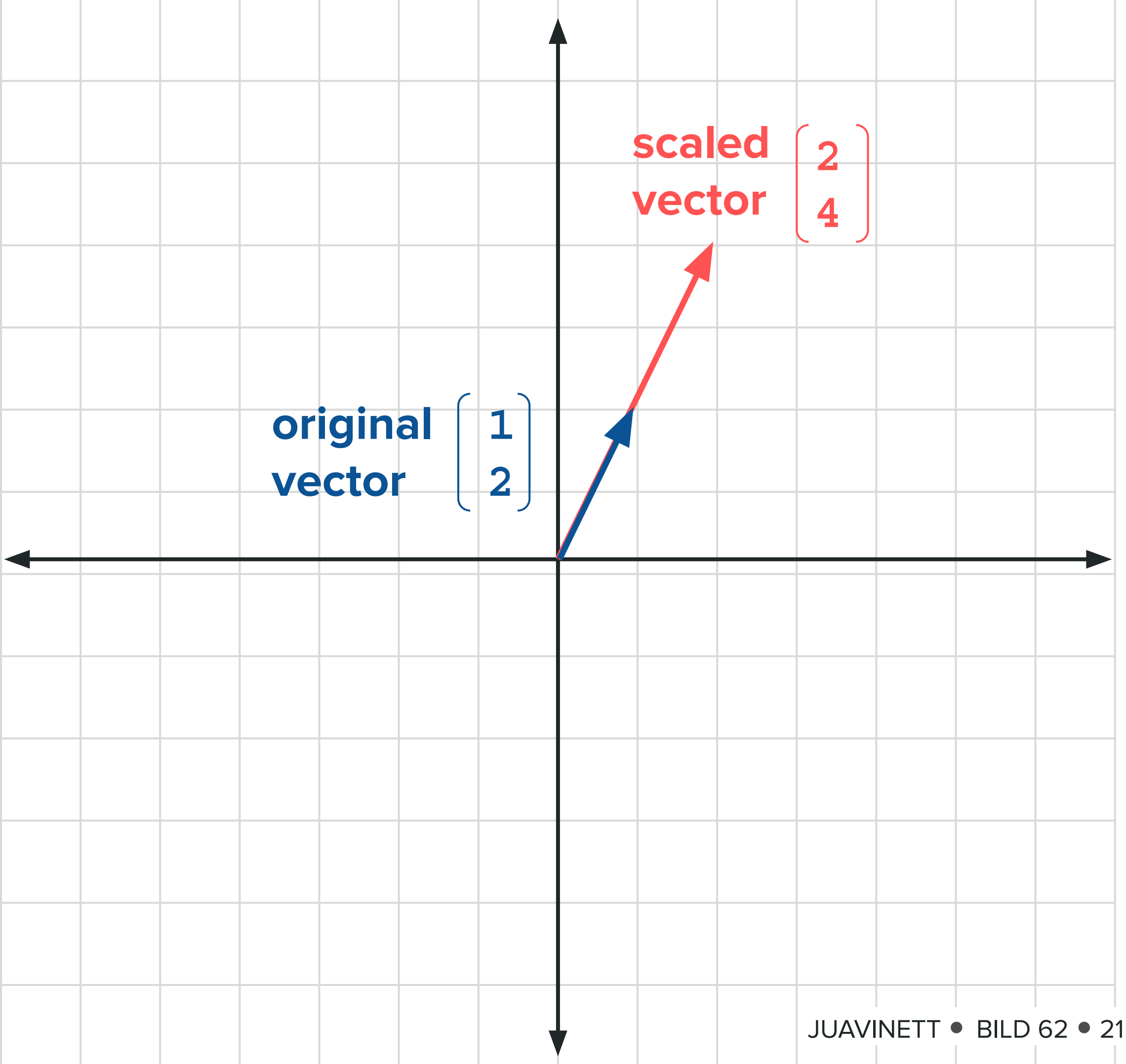
Vector subtraction

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



Vector-Scalar Multiplication

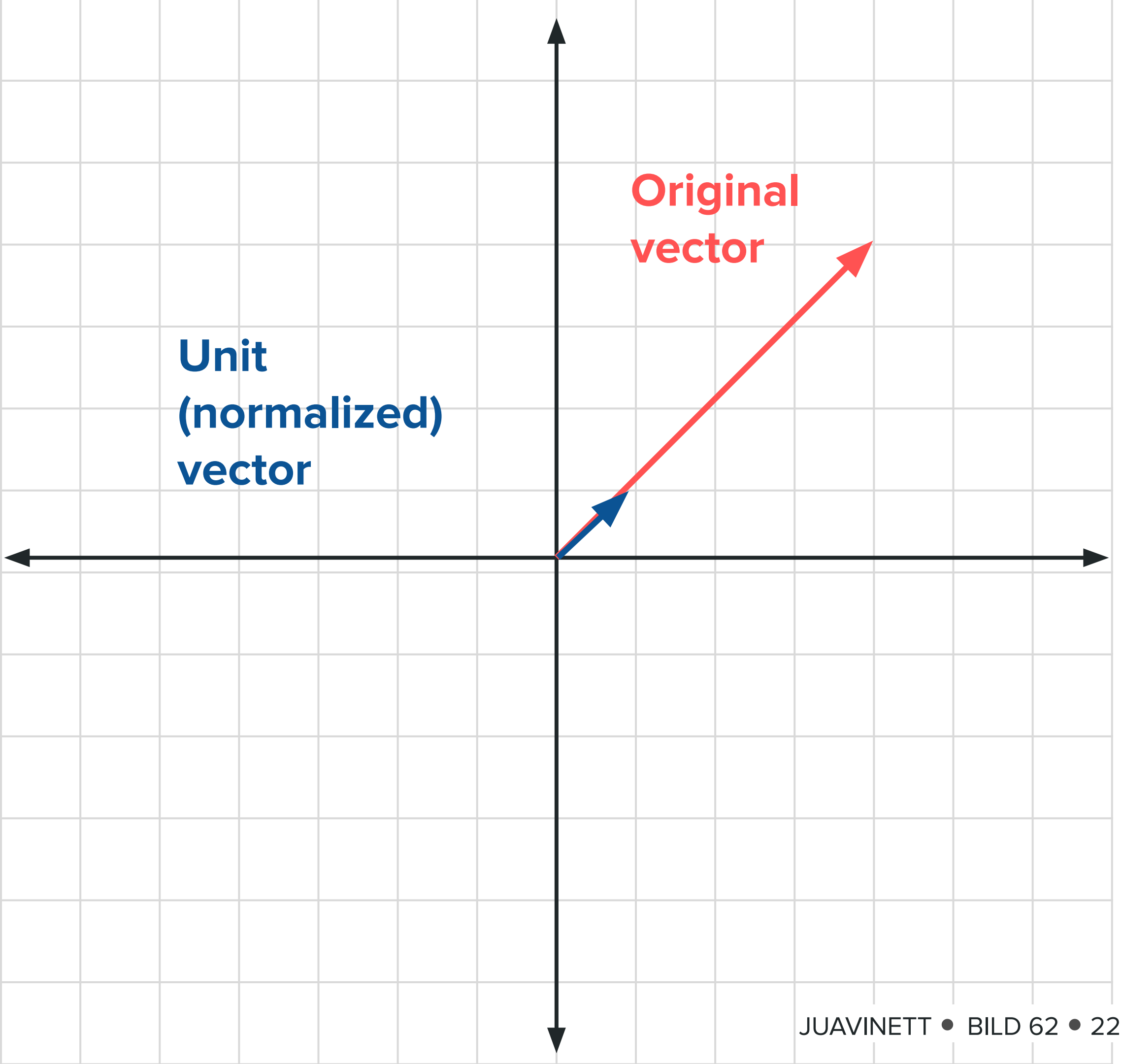
- "Scalars" are non-dimensional quantities
- Typically indicated by lowercase Greek letters such as α or λ
- Multiply each vector element by the scalar.



Vector Normalization

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

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



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

(in addition to the learning objectives for the take home videos!)

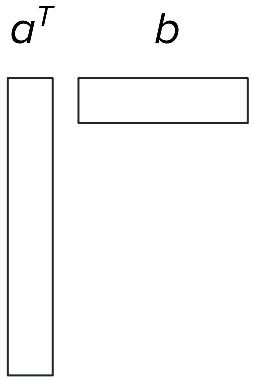
- Identify the possible uses of linear algebra in neuroscience
- **Compute a dot product and explain its relationship to a correlation**
- Construct and multiply matrices in Python (and by hand)
  - Create and manipulate special cases of matrices
- Define what eigenvalues & eigenvectors are and determine them using Python

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.

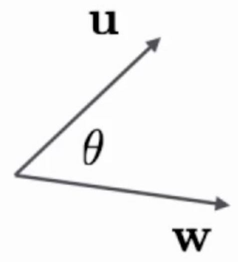
In other *other* words, element-wise multiplication & sum.

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 $\cos(90) = 0!$*)*

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



Example dot product calculation

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

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

$$= 70$$

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

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?

The example for today’s notebook will focus on the visual system — specifically connections from the **retina** to the **LGN**, a visual nucleus in the **thalamus**

The retina *projects* to the LGN

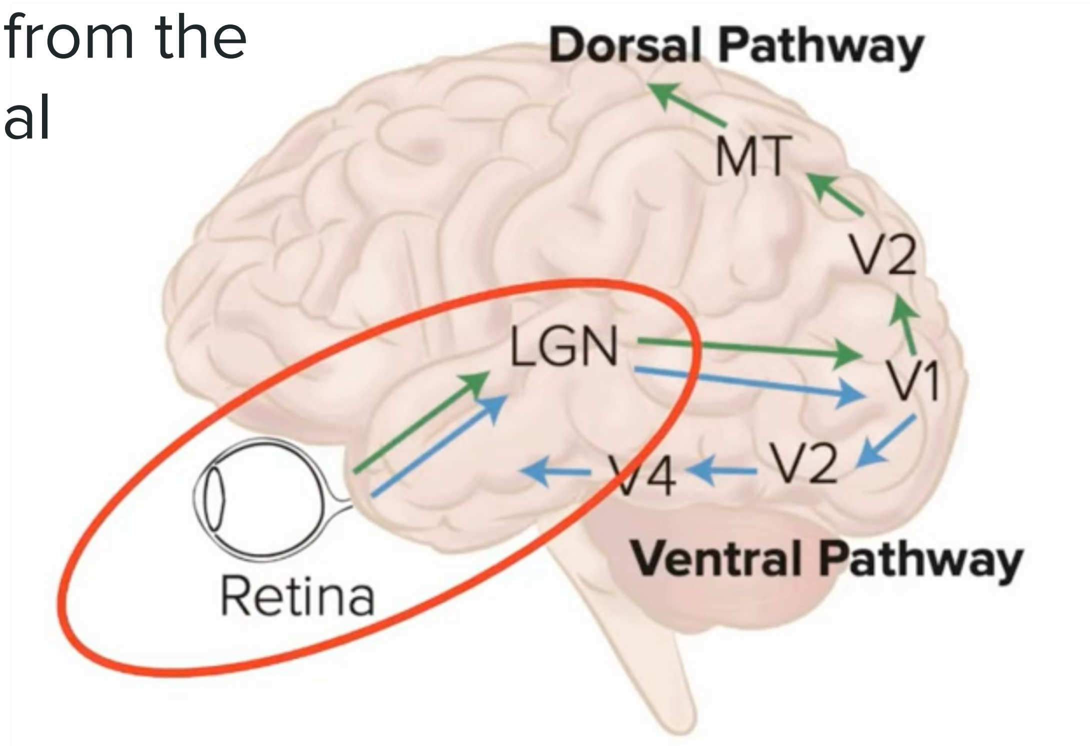
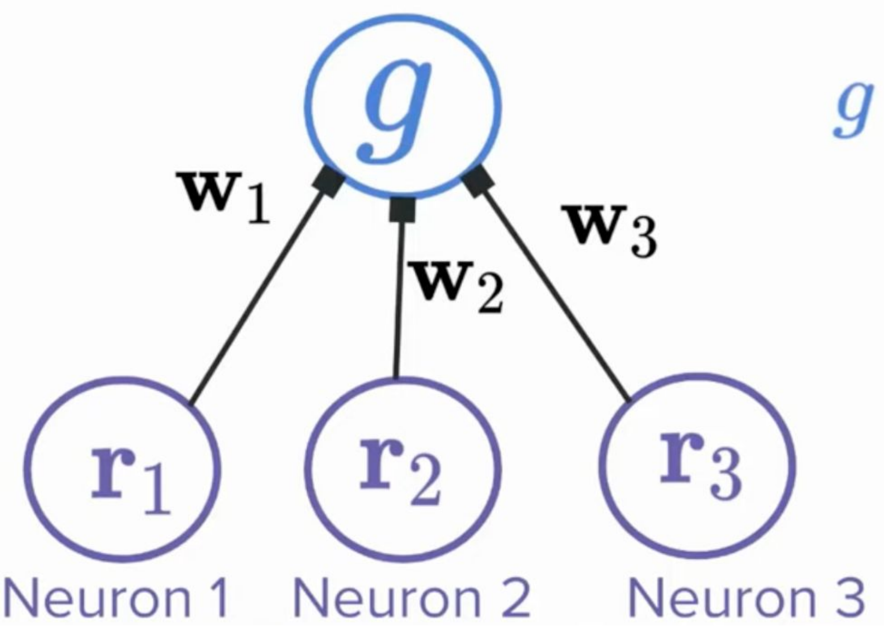


Image: Neuromatch Academy

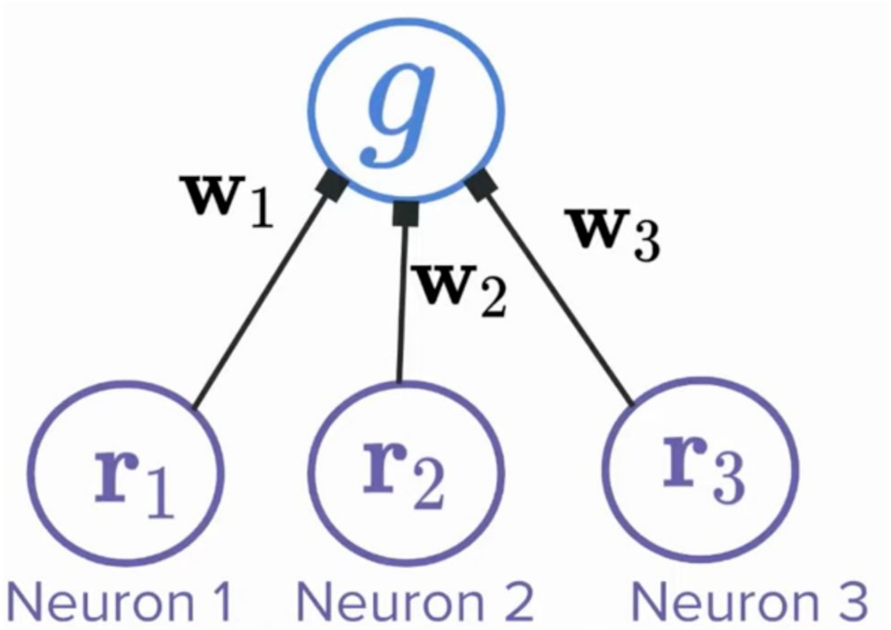
We can describe **LGN** activity as an equation, summing the weights ($\mathbf{w}$) * activities ($\mathbf{r}$) of each **retina** neuron.



$$g = w_1 r_1 + w_2 r_2 + w_3 r_3$$

We can call this a **weighted sum**

We can simplify this by using vectors, and computing the **dot product**.



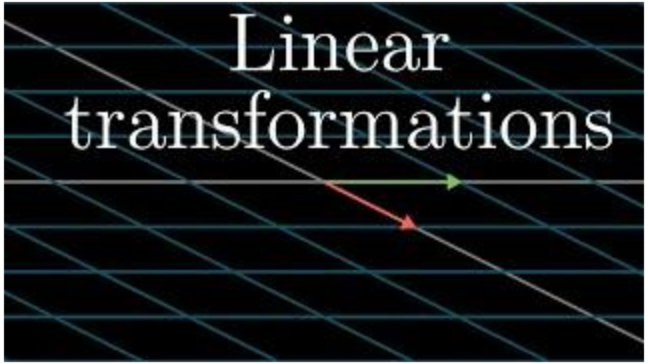
$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{w}_3 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix}$$

$$\begin{aligned} g &= \mathbf{w} \cdot \mathbf{r} \\ &= \mathbf{w}_1\mathbf{r}_1 + \mathbf{w}_2\mathbf{r}_2 + \mathbf{w}_3\mathbf{r}_3 \end{aligned}$$

Building a model of neural connections using linear equations

Matrices

Pre-lecture viewing



After watching chapter 3:

- Explain matrices as a linear transformation
- Relate matrix properties to properties of that linear transformation

A brief introduction to matrices

2 x 3 matrix

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

For example, $\mathbf{A}_{1,2} = \mathbf{4}$ and $\mathbf{A}_{2,3} = \mathbf{3}$.

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

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,

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

Slide: Jing Wang

Matrix transposition

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

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

We can move between these with **transposition**

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

Thinking about the computations as linear transformations of matrices

Additional resources

Essence of linear algebra - YouTube

Tutorial 1: Vectors — Neuromatch Academy ← much of this lecture was adopted from these materials!

https://www.youtube.com/watch?v=Ene_TYyTdNM — modeling neural connections as vectors and dot product review

http://matrixmultiplication.xyz/ — awesome visualization of matrix multiplication!

Cohen, *Practical Linear Algebra for Data Science*

MATH 18