CS-466/566: Math for AI

Module 02: Computational Linear Algebra-1

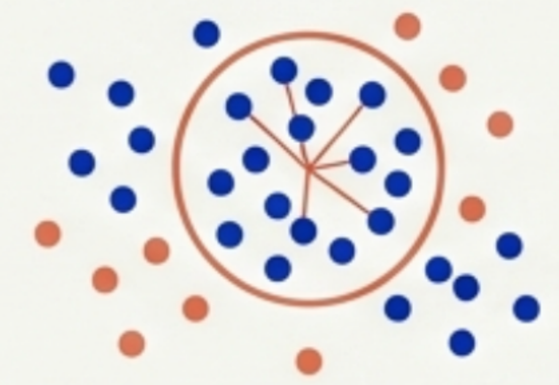
Dr. Mahmoud Mahmoud
The University of Alabama

2026-01-19

TABLE OF CONTENTS

Motivation: Why Geometry Matters in ML

Data lives in vector spaces. Learning depends on **distance**, **similarity**, and **direction**.



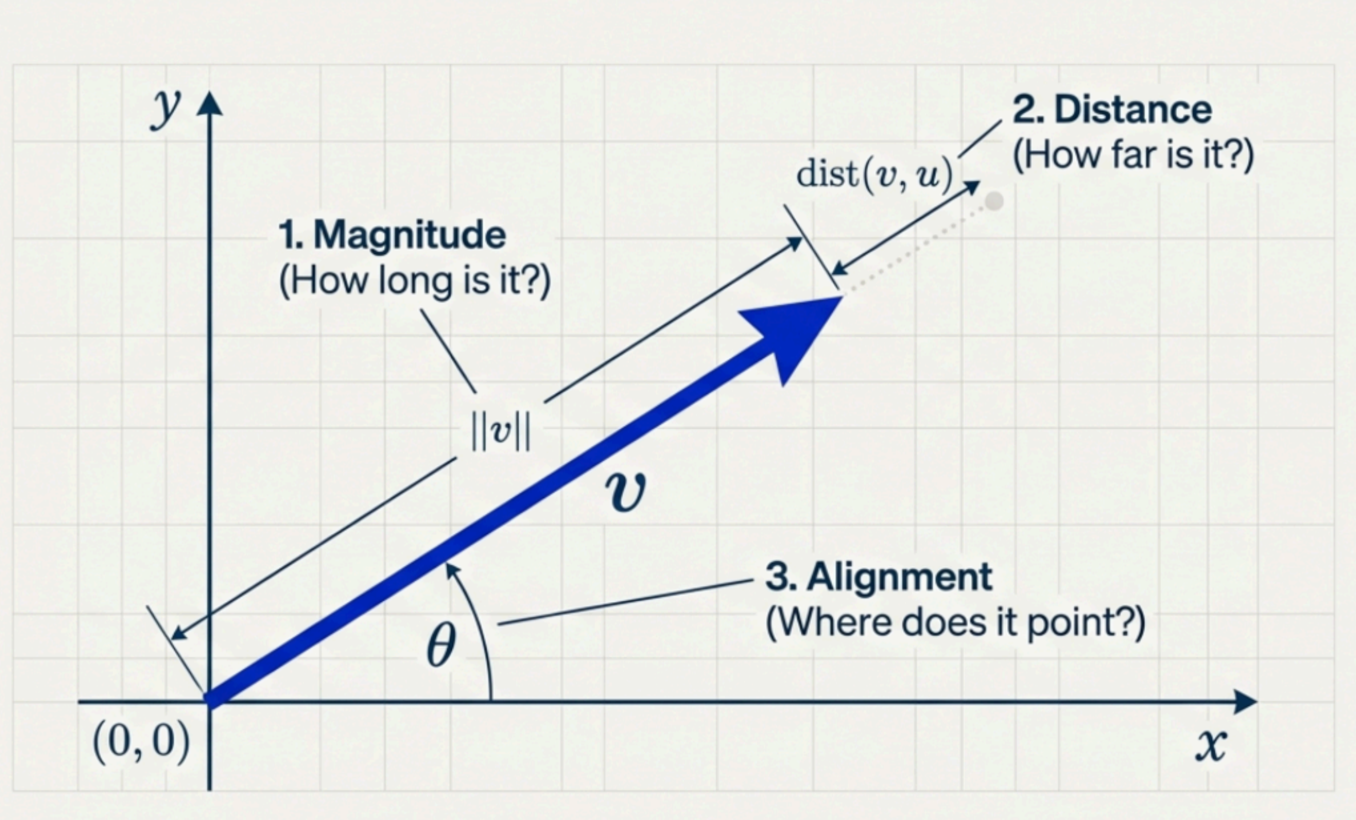
Geometry explains:

- Clustering
- Cosine similarity
- PCA / projections

Geometry gives meaning to the data.

Vectors as Geometry

- A vector is an **arrow** from the origin
- Geometry asks:
  - How long is it?
  - How far apart are two vectors?
  - How aligned are they?



These questions lead to norms and inner products.

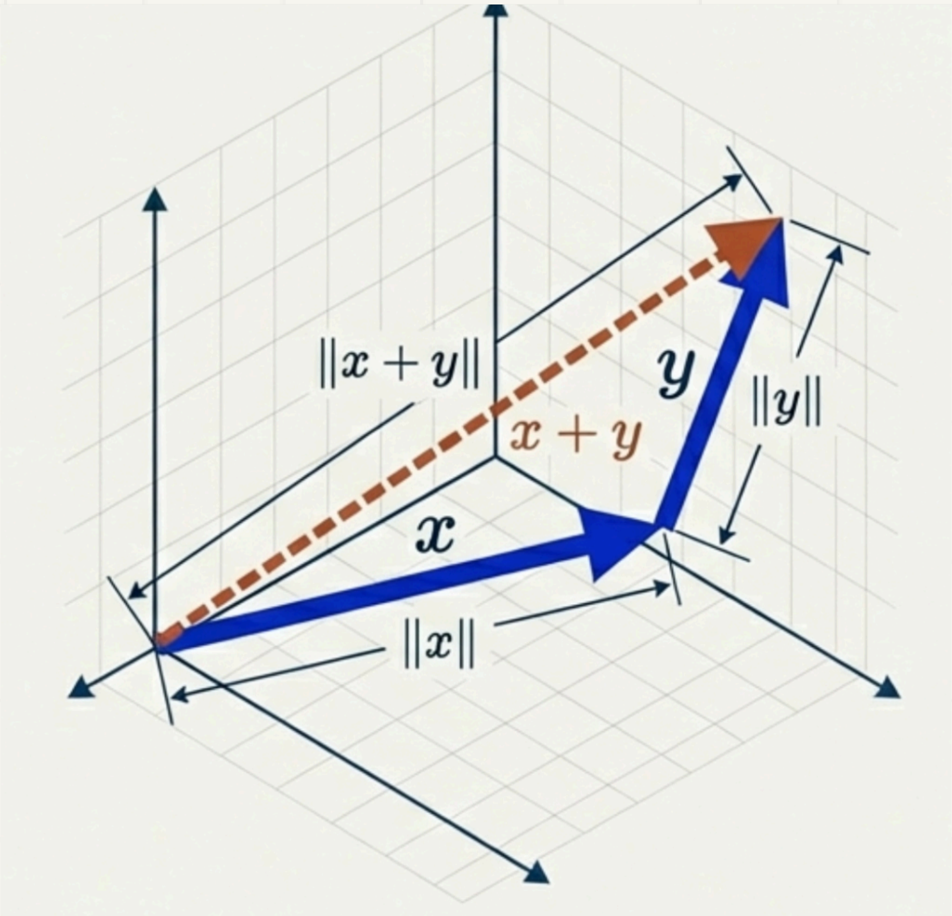
TABLE OF CONTENTS

Norms: Measuring Size

A **norm** is a function that takes a vector and returns a non-negative number: $V \rightarrow [0, \infty)$

$$\|x\|$$

1. **Positivity:** $\|x\| \geq 0$.
   (Length is never negative).
2. **Homogeneity:** $\|\alpha x\| = |\alpha|\|x\|$.
   (Scaling the vector scales the length)
3. **Triangle Inequality:**
   $\|x + y\| \leq \|x\| + \|y\|$.
   (The direct path is shortest).



Triangle inequality

Common Norms in $\mathbb{R}^n$

Euclidean (ℓ_2) – Most common in ML

$$\|\mathbf{v}\|_2 = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

Squaring penalises outliers

Returns to original scale

Manhattan (ℓ_1) – Used in regularization

$$\|\mathbf{v}\|_1 = |v_1| + |v_2| + \cdots + |v_n|$$

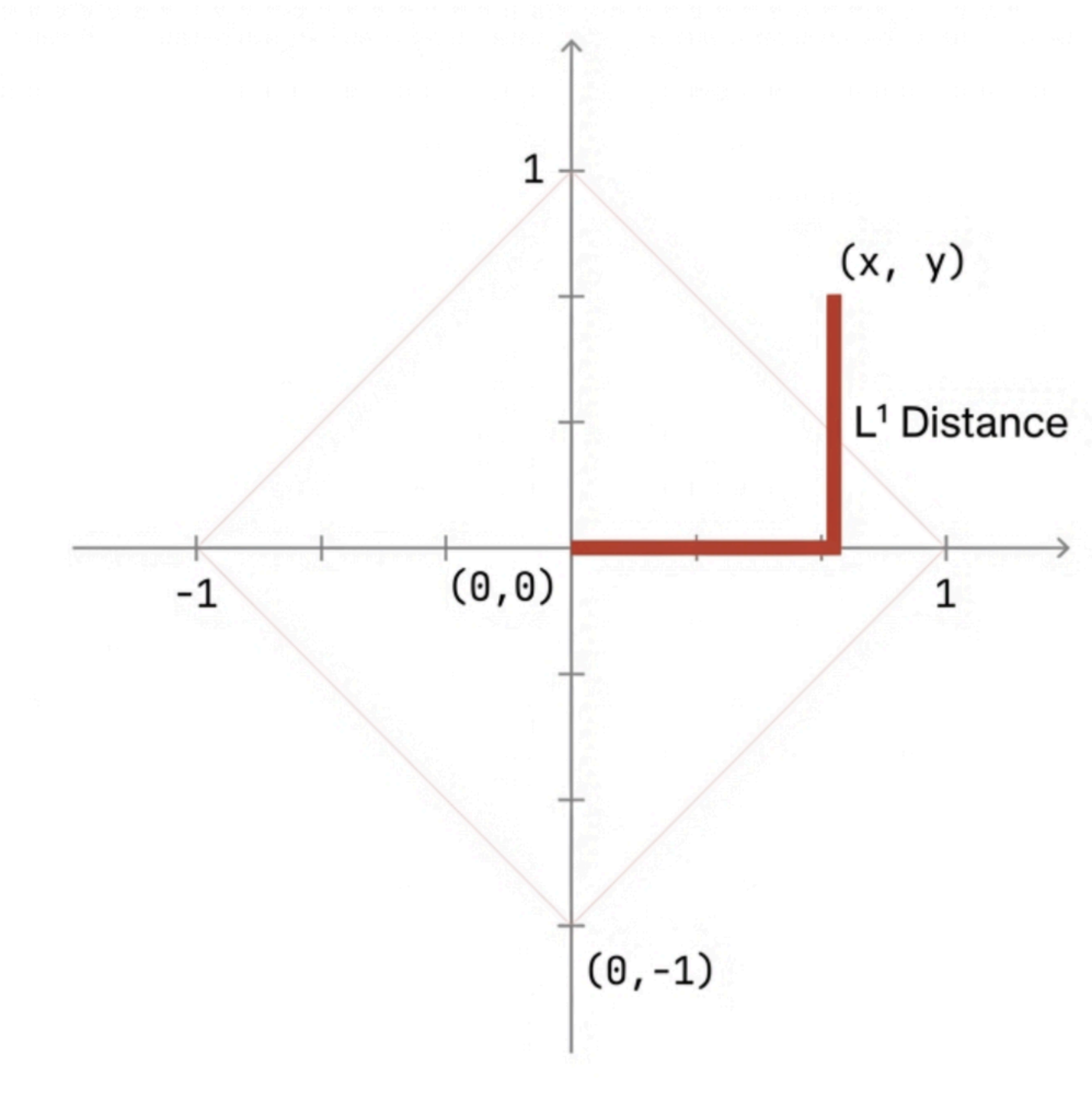
Magnitude regardless of direction

Accumulation of components

Supremum (ℓ_∞) – Largest component

$$\|\mathbf{v}\|_\infty = \max(|v_1|, |v_2|, \ldots, |v_n|)$$

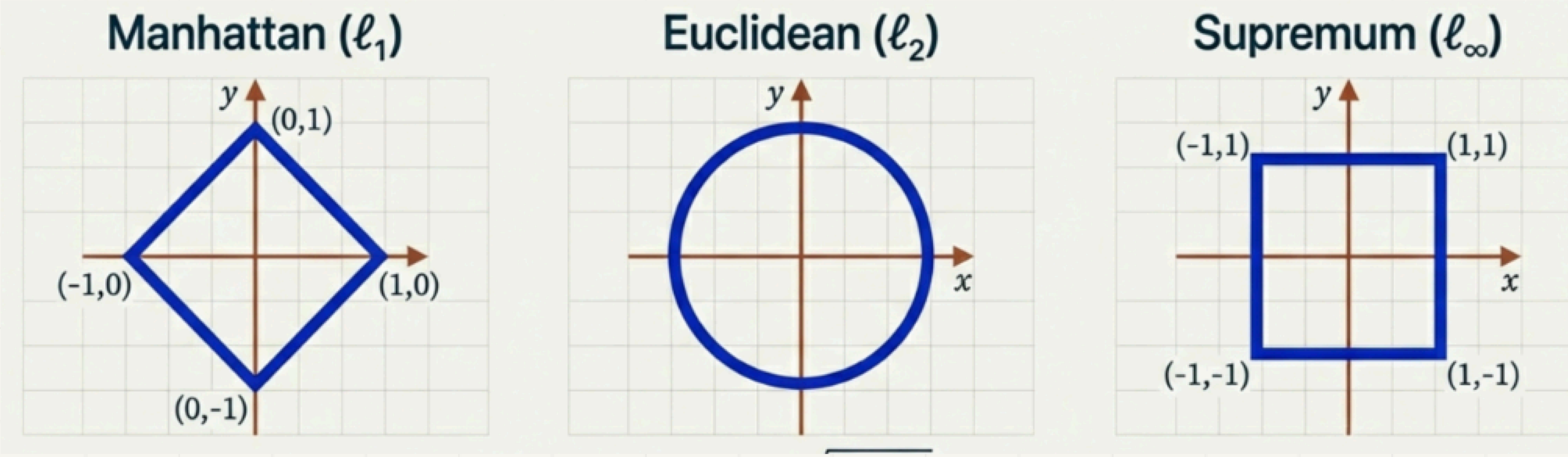
Discards all information except the single greatest dimension



L1 vs L2 distance comparison

Visualizing the Unit Ball

Visualising the 'Unit Ball' (points where distance is 1) for different norms.



- **Manhattan** (ℓ_1): Diamond shape
- **Euclidean** (ℓ_2): Circle (familiar distance)
- **Supremum** (ℓ_∞): Square

Unit Vectors (Normalization)

A **unit vector** has length 1:

$$\|\hat{x}\| = 1$$

Normalization converts any vector to a unit vector:

$$\hat{x} = \frac{x}{\|x\|}$$

Why normalize?

- Focus on **direction**, not magnitude
- Numerical stability
- Cosine similarity (angles only)
- Neural network inputs

Example:

$$x = \begin{bmatrix} 3 \\ 4 \end{bmatrix}, \quad \|x\| = 5$$

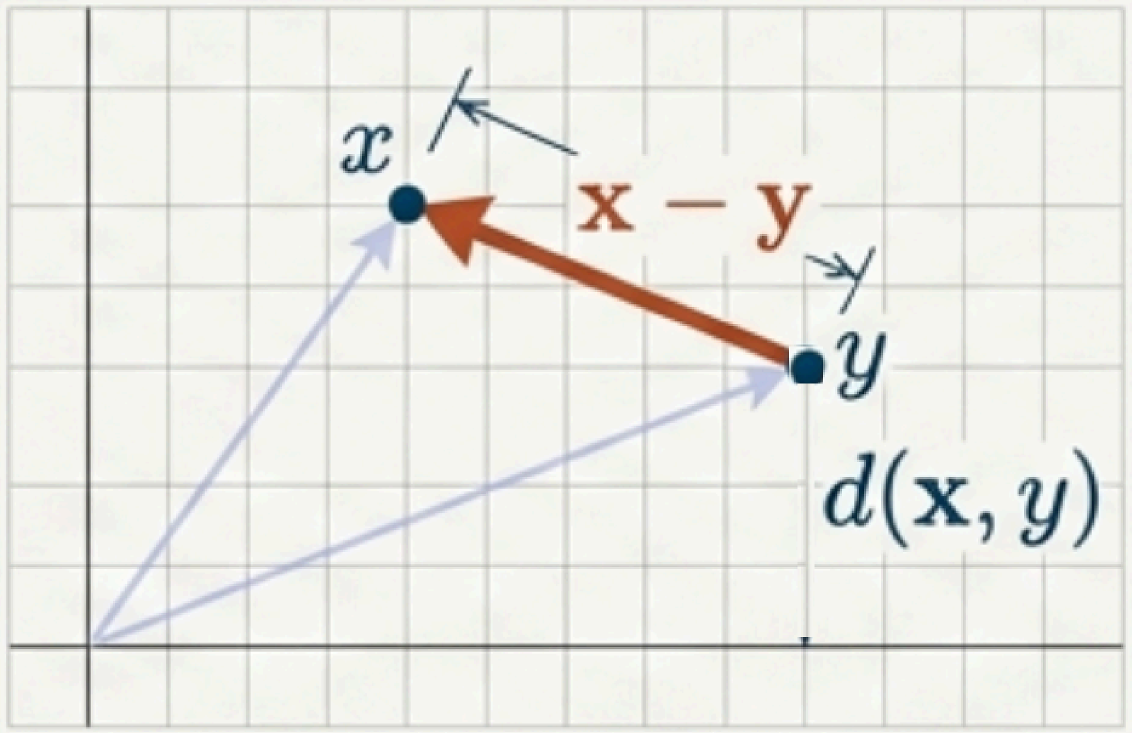
$$\hat{x} = \frac{1}{5}\begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}$$

Check: $\|\hat{x}\| = \sqrt{0.6^2 + 0.8^2} = 1 \checkmark$

Distance from Norms

Given a norm, define distance:

$$d(x, y) = \|x - y\|$$



Properties:

- Nonnegative
- Symmetric
- Triangle inequality:

$d(x, z) \leq d(x, y) + d(y, z)$

Distance = length of displacement

Exercise: Calculating Norms

Let $\mathbf{x} = \begin{bmatrix} 3 \\ -4 \end{bmatrix}$. Calculate:

1. **L1 Norm** (ℓ_1): $||\mathbf{x}||_1$

   Answer: 7

   ($|3| + |-4| = 3 + 4$)

2. **L2 Norm** (ℓ_2): $||\mathbf{x}||_2$

   Answer: 5

   ($\sqrt{3^2 + (-4)^2} = \sqrt{25}$)

3. **L-infinity Norm** (ℓ_∞): $||\mathbf{x}||_\infty$

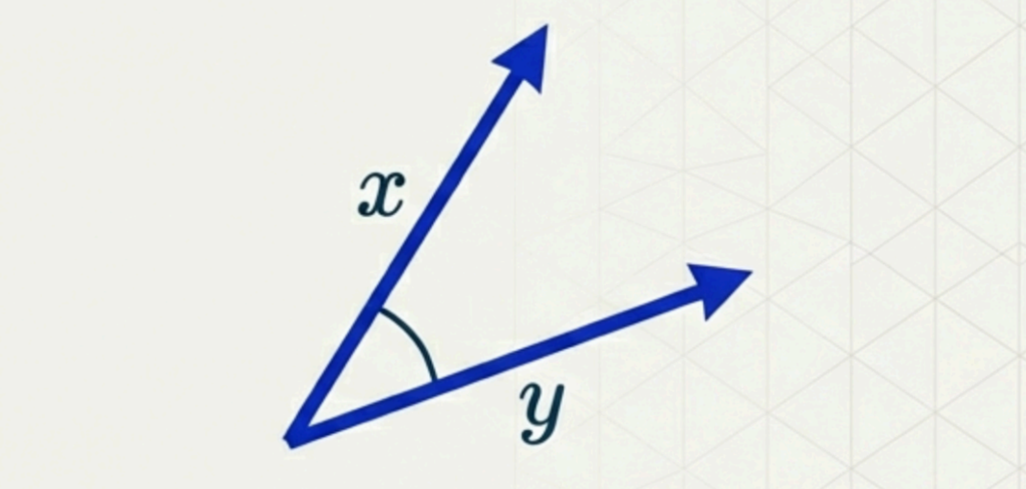
   Answer: 4

   ($\max(|3|, |-4|) = 4$)

TABLE OF CONTENTS

Inner Products: Measuring Similarity

An **inner product** is a function $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ satisfying:



Definition properties:

- **Linearity**
  e.g., $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle = \langle x, \alpha y \rangle$
- **Symmetry**
  e.g., $\langle x, y \rangle = \langle y, x \rangle$
- **Positive Definiteness**
  e.g., $\langle x, x \rangle \geq 0\, ; \langle x, x \rangle = 0 \iff x = 0$

Inner product between vector and itself is the norm squared!

The Standard Dot Product:

$$\langle x, y \rangle = \sum_i x_i y_i = \|x\| \|y\| \cos(\theta)$$

Connecting Inner Products to Norms

Inner products naturally give us a way to measure length. We call this the **Induced Norm**.

$$\|x\| = \sqrt{\langle x, x \rangle}$$

Example: Using the standard dot product $\langle x, y \rangle = \sum x_i y_i$:

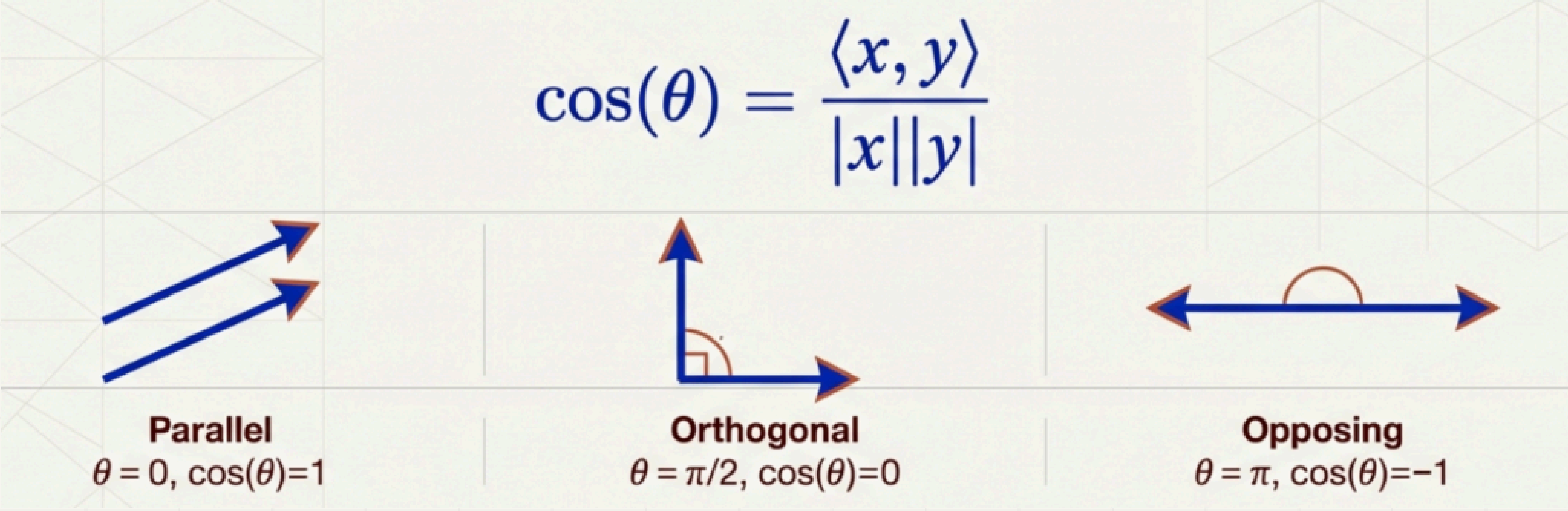
$$\|x\| = \sqrt{x \cdot x} = \sqrt{\sum x_i^2}$$

This recovers the **Euclidean Norm**!

Angles and Similarity

Defined via:

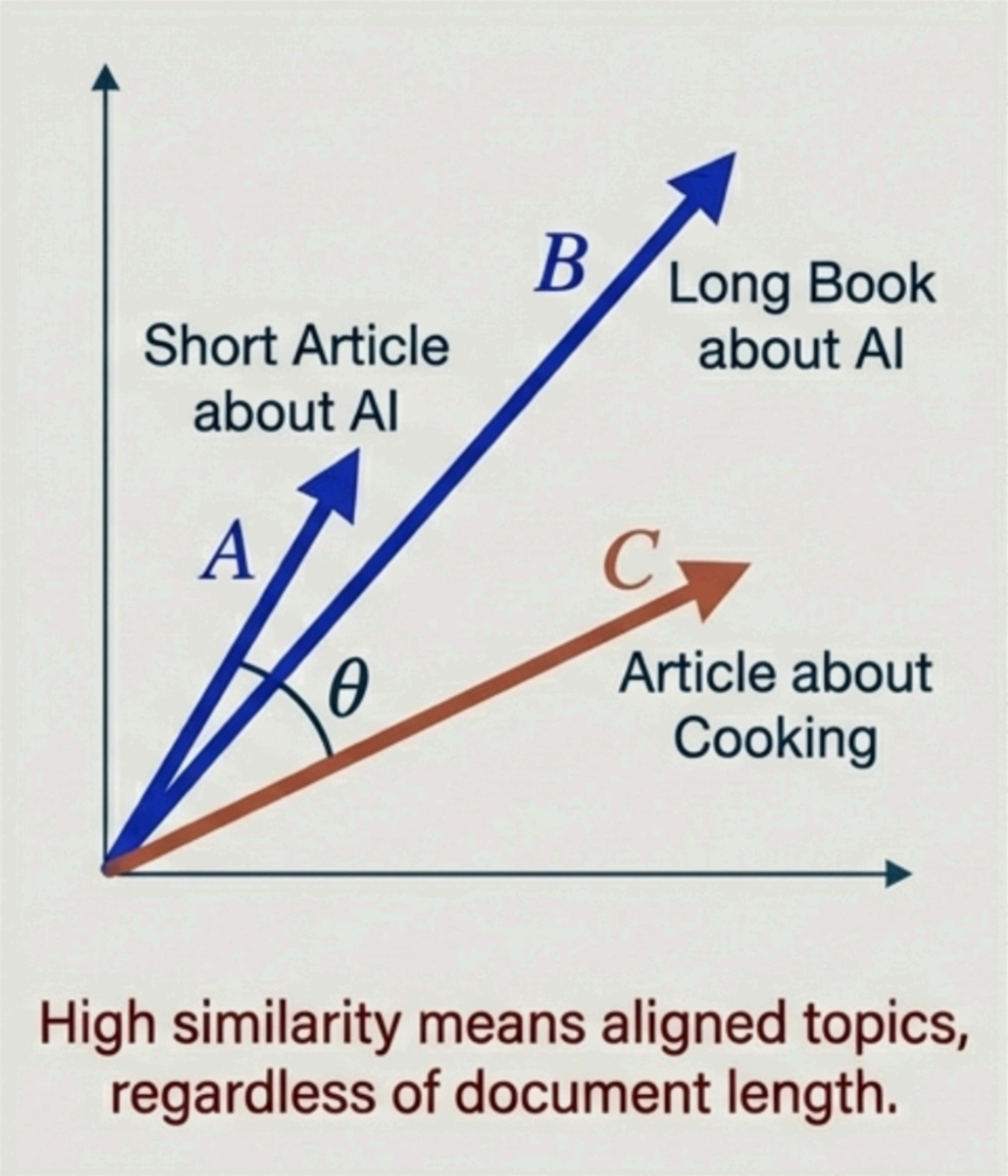
$$\cos(\theta) = \frac{\langle x, y \rangle}{\|x\| \, \|y\|}$$



Cosine Similarity

In high-dimensional spaces (like text analysis), we often care about **direction**, not magnitude.

$$\text{Cosine Similarity} = \frac{\langle x, y \rangle}{\|x\| \|y\|}$$



Exercise: Inner Products & Angles

Consider $\mathbf{u} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 5 \\ -1 \end{bmatrix}$.

1. Compute the dot product $\langle \mathbf{u}, \mathbf{v} \rangle$.

   Answer: 0

   $(1(5) + 5(-1) = 5 - 5)$

2. What is the angle between them?

   Answer: 90° (Orthogonal)

   (Since dot product is 0)

TABLE OF CONTENTS

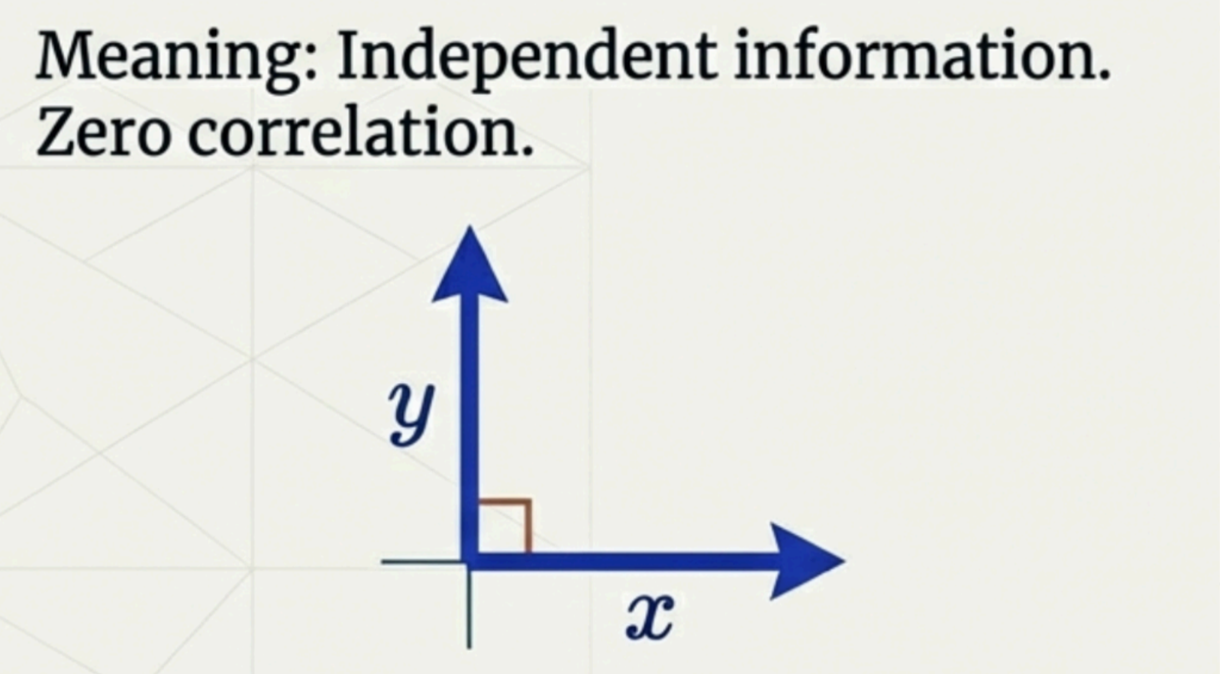
Orthogonality

Two vectors are **orthogonal** if:

$$\langle x, y \rangle = 0$$

Interpretation:

- No shared information
- Independent directions
- Perpendicular in geometric space



Projection of One Vector onto Another

Scalar projection (length of projection):

$$\text{comp}_y(x) = \|x\| \cos(\theta) = \frac{\langle x, y \rangle}{\|y\|}$$

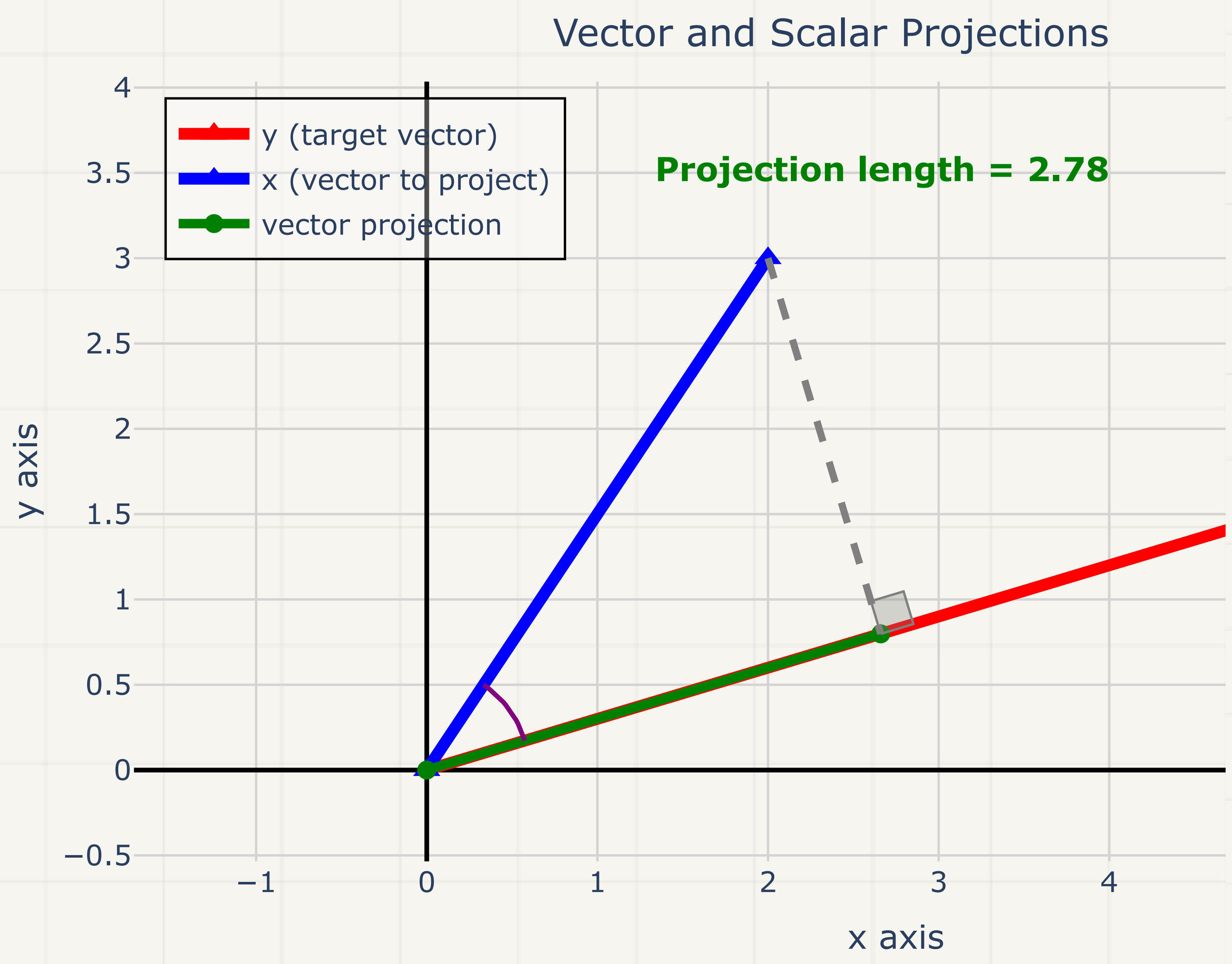
Vector projection (actual vector):

$$\text{proj}_y(x) = \frac{\langle x, y \rangle}{\|y\|^2} y = \frac{\langle x, y \rangle}{\langle y, y \rangle} y$$

Used in:

- Feature extraction
- PCA

Vector and Scalar Projections



Exercise: Projections in $\mathbb{R}^3$

Project vector $\mathbf{b} = \begin{bmatrix} 2 \\ 4 \\ 1 \end{bmatrix}$ onto $\mathbf{a} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$.

1. Scalar Projection ($\frac{\mathbf{b} \cdot \mathbf{a}}{\|\mathbf{a}\|}$):

Answer: $3\sqrt{2} \approx 4.24$ $(\mathbf{b} \cdot \mathbf{a} = 6, \|\mathbf{a}\| = \sqrt{2})$

2. Vector Projection ($\text{proj}_{\mathbf{a}} \mathbf{b}$):

Answer: $\begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix}$

TABLE OF CONTENTS

Orthonormal Bases

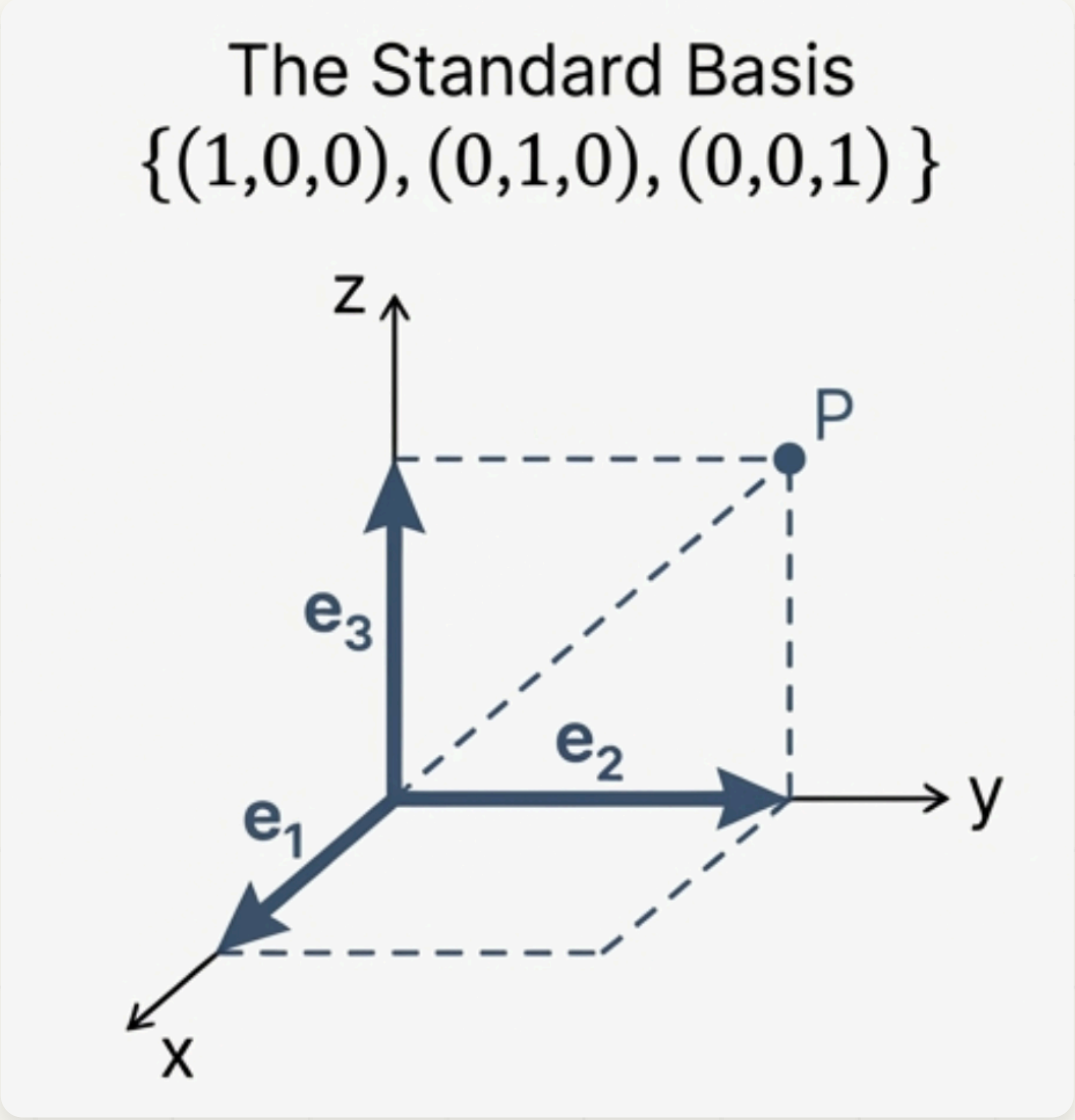
A basis $\{ v_i \}$ is **orthonormal** if every vector has unit length and all are mutually orthogonal.

Conditions:

1. $\|v_i\| = 1$
2. $\langle v_i, v_j \rangle = 0$ for $i \neq j$

Why use orthonormal bases?

- **Coordinates are easy**: Direct projection ($x_i = \langle x, v_i \rangle$)
- **Norms are simple**: Pythagorean theorem holds ($\|x\|^2 = \sum x_i^2$)
- **Computations decouple**: Each dimension is independent



Standard orthonormal basis in $\mathbb{R}^3$

Exercise: Orthonormal Basis

Is the set S an orthonormal basis for $\mathbb{R}^2$?

$$S = \left\{ \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \right\}$$

Check Conditions:

1. **Unit Length? Yes.**

$$(\frac{1}{\sqrt{2}})^2 + (\frac{1}{\sqrt{2}})^2 = \frac{1}{2} + \frac{1}{2} = 1$$

2. **Orthogonal? Yes.**

$$\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}}) + \frac{1}{\sqrt{2}}(-\frac{1}{\sqrt{2}}) = \frac{1}{2} - \frac{1}{2} = 0$$

Conclusion: Yes, it is an orthonormal basis!

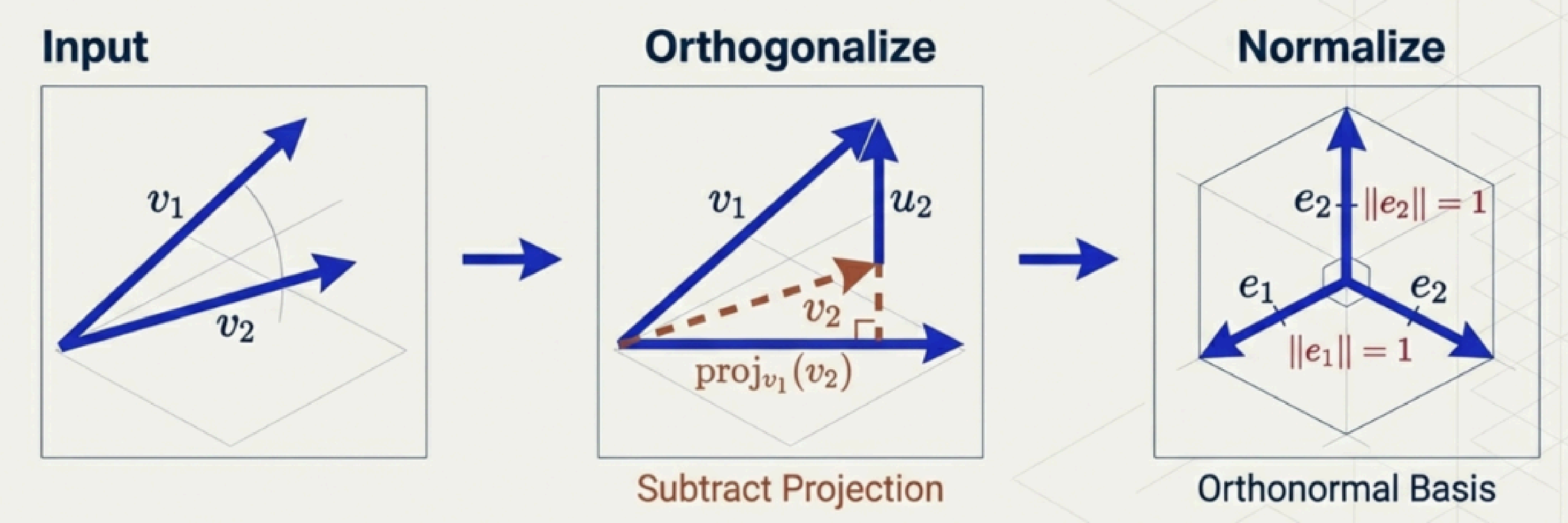
TABLE OF CONTENTS

Gram–Schmidt Orthogonalization

Goal:
Convert any linearly independent set into an orthonormal basis

Algorithm:

1. Start with basis
2. Orthogonalize
3. Normalize



Applications: Why Gram–Schmidt?

Setup: Predicting House Price using two features:

1. x_1 : House Size (m^2)
2. x_2 : Number of Rooms

The Problem: These are highly correlated! (Bigger house → more rooms).

- **Model Confusion:** Which feature actually explains the price?

The Solution (Gram–Schmidt): Create a new feature $x_2^\perp$:

$$x_2^\perp = x_2 - \text{proj}_{x_1}(x_2)$$

"Rooms that CANNOT be explained by Size"

Now x_1 and $x_2^\perp$ are **orthogonal**. The model becomes stable and interpretable.

Gram–Schmidt (Step Form)

Given **input basis** $u_1, \ldots, u_n$:

1. $v_1 = u_1$

2. $v_2 = u_2 - \mathrm{proj}_{v_1}(u_2)$

3. $v_3 = u_3 - \mathrm{proj}_{v_1}(u_3) - \mathrm{proj}_{v_2}(u_3)$

4. Continue…

5. **Normalize** all v_i to get $e_i = v_i / \|v_i\|$

Exercise: Gram–Schmidt Process

Convert $u_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, u_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ into an orthonormal basis.

1. **Step 1** (v_1):

$$v_1 = u_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

2. **Step 2** (v_2):

$$v_2 = u_2 - \frac{u_2 \cdot v_1}{v_1 \cdot v_1} v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}$$

3. **Step 3 (Normalize)**:

$$e_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad e_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Gram–Schmidt in Python (Using Libraries)

In practice, we use **QR Decomposition** (numerically stable):

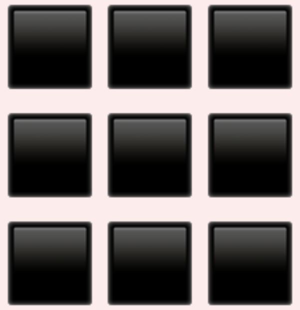
```python
import numpy as np

#> X is our data matrix (features as columns)
#> Q will be the orthonormal basis
#> R will be the upper triangular matrix
Q, R = np.linalg.qr(X)
```

Visualizing Decorrelation (Sparsity of Correlations):

Before ($X^T X$)

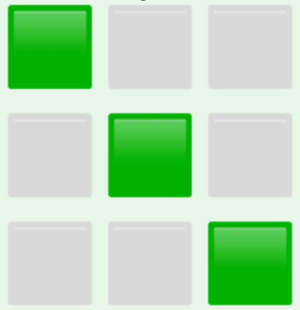
High Values Everywhere
(Correlated)



→

After ($Q^T Q$)

Diagonal / Sparse
(Identity Matrix)



Important Caveat

- If inputs are **linearly dependent**
- Gram–Schmidt produces a **zero vector**

This reveals dependence automatically!

Thank You!