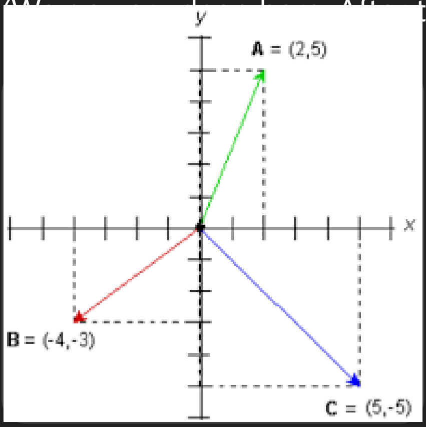
explain these topics one by one try to cover hidden aspects of them

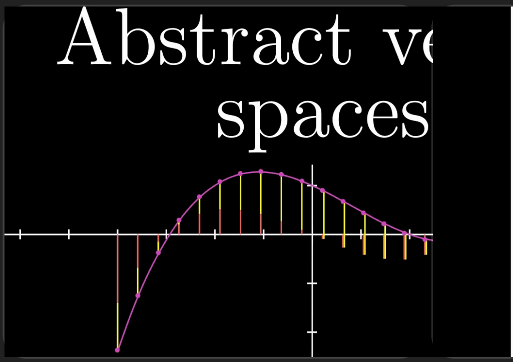
Advanced Linear Algebra
Vector spaces, basis, dimension
Orthogonality & projections
Eigen decomposition, SVD, PCA
Tensors

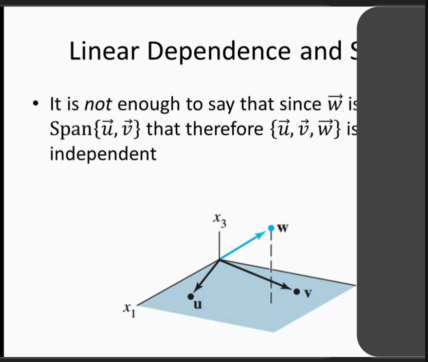
Great — we now enter **Advanced Linear Algebra**, which is the mathematical backbone of **machine learning, deep learning, optimization, computer vision, transformers**, and even **quantum mechanics**.
I will explain **one topic at a time**, and for each topic I'll uncover the *hidden aspects* that textbooks usually skip — the intuitions, geometric pictures, and how these ideas evolve into tools for ML/DL.

⭐ TOPIC 1: VECTOR SPACES, BASIS, DIMENSION

this, we move to Orthogonality & Projections.)






Linear Dependence and S

- It is *not* enough to say that since $\vec{w}$ is Span$\{\vec{u},\vec{v}\}$ that therefore $\{\vec{u},\vec{v},\vec{w}\}$ is independent

---

1.1 What *is* a Vector Space?

ChatGPT 5.1

2. **Scalar multiplication**

But here is the **hidden truth**:

✅ A vector space is *not about arrows*.

Even though we imagine vectors as arrows, mathematically:

- Polynomials can form a vector space
- Functions can form a vector space
- Images can form a vector space
- Probability distributions can form a vector space
- Neural network feature embeddings form a vector space

So a vector space is **any set where "linear combination" works**:

$$a_1 v_1 + a_2 v_2 + \cdots + a_k v_k$$

If linear combinations "stay inside" the set → it's a vector space.

1.2 Linear Combination — the Real Insight

Most students see linear combination as a formula, but **here is the hidden concept**:

🔥 A vector space is completely controlled by what linear combinations can produce.

If enough linear combinations can generate ALL vectors in the space → you have a **spanning set**.

This idea will drive:

- PCA
- Dimensionality reduction
- Neural network representation learning
- Fourier series
- Function approximation

⭐ 1.3 Basis — the DNA of a Vector Space

A **basis** is a **minimal spanning set**, meaning:

1. It spans the space
2. All vectors in the basis are linearly independent
3. Removing any vector destroys the spanning property

💡 Hidden truth:

A basis is like the **alphabet** of a vector space.

Every vector is a unique “word” written using the basis.

Example (R^2)

Standard basis:

$$e_1 = (1, 0), \quad e_2 = (1, 0)$$

But you can choose infinitely many bases:

$$b_1 = (1, 1), \quad b_2 = (1, -2)$$

As long as they’re **not collinear**, they form a basis.

1.4 Why “basis” matters in ML & AI?

Because:

- PCA chooses the “best basis” to express data
- SVD gives an “optimal basis” for matrices
- Neural networks "learn" new bases for representing data
- Fourier transform is a basis of sine/cosine waves
- Transformers learn basis directions called “attention heads”

⭐ 1.5 Dimension — what it *really* means

Dimension = number of vectors in any basis.

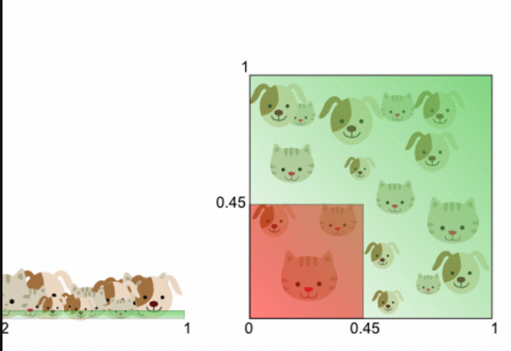
But here is the hidden truth:

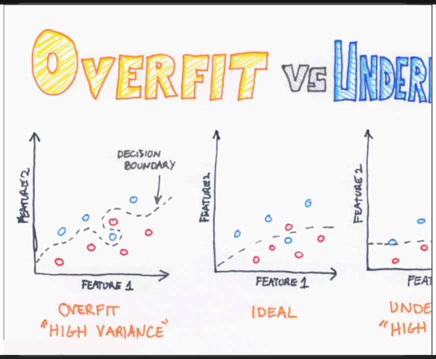
🔥 Dimension = Degrees of freedom

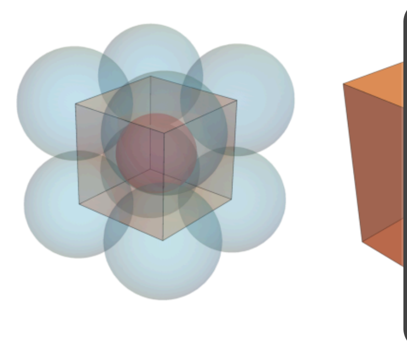
If dimension = 100, you need 100 numbers to describe any vector uniquely.

ML context:

- A 1024-dimensional embedding means 1024 “features”
- PCA reduces dimension by removing redundant features
- High dimensional geometry behaves counterintuitively (curse of dimensionality)







⭐ 1.6 Hidden Properties That Students Often Miss

1. A vector space can have infinite dimension

Examples:

- Space of all polynomials
- Space of continuous functions
- Space of signals
  This is why Fourier transform and neural networks approximate infinite-dimensional spaces.

---

2. A vector space is defined by rules, not by elements

Example: set of all solutions to a differential equation:

$$y'' + y = 0$$

Solutions:

$$y = C_1 \cos x + C_2 \sin x$$

Basis: $\cos x, \sin x$

This is a **function space**, but still a vector space.

---

3. The same vector looks different in different bases

If basis = {e_1, e_2}, vector v = (3, 5)

If basis = {b_1, b_2}, vector becomes something else (different coordinates)

Changing basis = rotating/changing perspective.

This is the heart of:

- PCA
- SVD
- Diagonalization
- Whitening
- Eigenvalues
- Quantum basis transformations

⭐ 1.7 Span, Linear Independence — Hidden Intuition

Span = Everything You Can Reach

Take two vectors in 2D:

- If they point in different directions → they span a plane
- If they point in same direction → they span a line

Linear independence = No vector is “redundant”

You cannot recreate one from the others.

This is why ML reduces dimension — too many features are redundant.

⭐ SUMMARY OF TOPIC 1

You learned:

✔ **Vector space = set where linear combinations make sense**

✔ **Basis = minimal set generating the whole space**

✔ **Dimension = degrees of freedom**

✔ **Real-life vector spaces: functions, polynomials, images, signals**

✔ **Changing basis = changing representation of the same object**

✔ **Hidden insights that drive PCA, SVD, deep learning**

✅ NEXT TOPIC

If you want, we now continue to:

TOPIC 2: Orthogonality & Projections

This topic is extremely important because:

- PCA is just orthogonal projection
- Least squares = projection
- Gradient descent uses projection ideas
- Fourier transform is projection onto sine/cosine basis

Would you like me to continue to **Orthogonality & Projections**?

sure

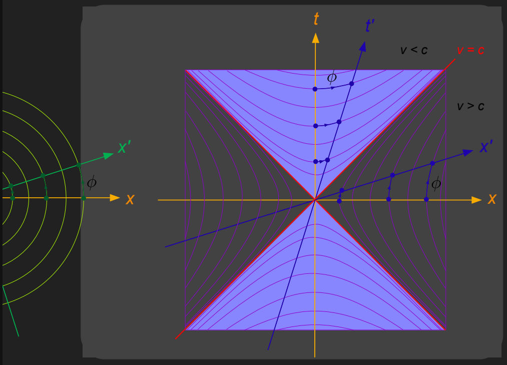
Great — let's proceed to the **NEXT MAJOR TOPIC** in Advanced Linear Algebra:

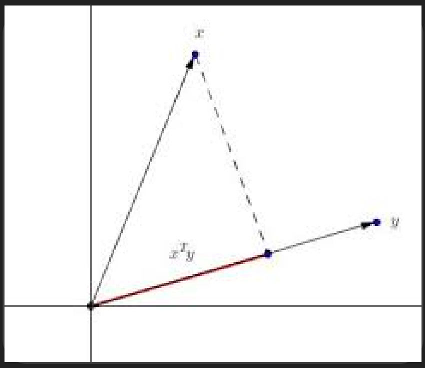
⭐ TOPIC 2: ORTHOGONALITY & PROJECTIONS

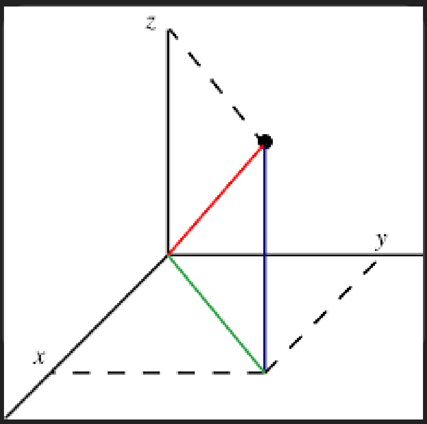
This is one of the most **powerful ideas** in machine learning, statistics, optimization, and deep learning.
Everything from **PCA** → **linear regression** → **SVD** → **Fourier transforms** → **embeddings** is secretly based on projections.

I'll go deep, uncover hidden geometric ideas, and give formulas + intuition.

2.1 What is Orthogonality?







Two vectors **u** and **v** are orthogonal if:

$$u \cdot v = 0$$

Meaning:

- No overlap
- No shared component
- Purely independent directions
- They convey completely different “information”

Hidden truth (students miss this):

Orthogonality means **decorrelation**.
If two features in ML are orthogonal → they are statistically independent in direction.

2.2 Geometric Meaning

Dot product expands as:

$$u \cdot v = ||u|| ||v|| \cos \theta$$

So when $u \cdot v = 0$, then $\theta = 90^\circ$.
That means:

✓ All of u is perpendicular to v

✓ No component of u lies along v

2.3 Orthonormal Vectors

A set of vectors is **orthonormal** if:

- They are orthogonal
- Each has length 1

This is the ideal basis for:

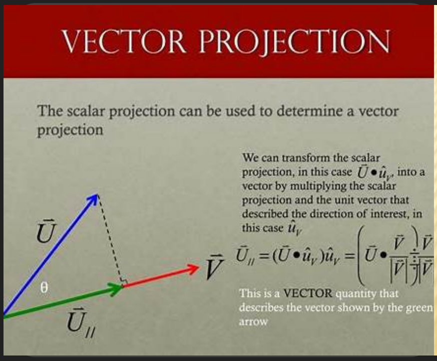
- PCA
- Fourier transform
- SVD
- Gram-Schmidt orthonormalization
- Quantum mechanics

Why? Because orthonormal bases simplify everything:

$$\text{Coordinates} = u \cdot v$$

No matrix inverses, no heavy math — just dot products.

⭐ 2.4 Projection — The Hidden Engine Behind Everything



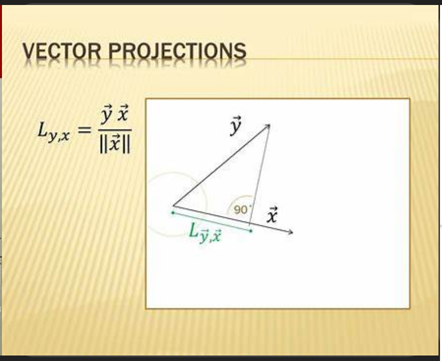
VECTOR PROJECTION

The scalar projection can be used to determine a vector projection

We can transform the scalar projection, in this case $\bar{U} \bullet \hat{u}_V$, into a vector by multiplying the scalar projection and the unit vector that described the direction of interest, in this case $\hat{u}_V$

This is a VECTOR quantity that describes the vector shown by the green arrow

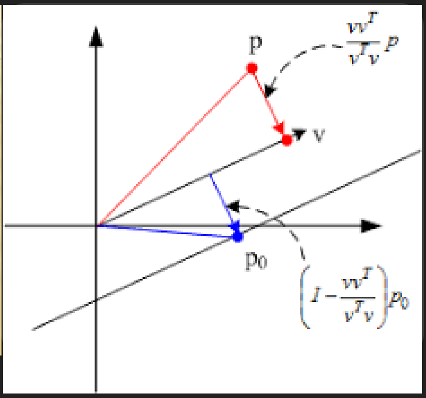
$\bar{U}$
θ
$\bar{U}_{//}$
$\vec{V}$
$\bar{U}_{//} = (\bar{U} \bullet \hat{u}_V)\hat{u}_V =$



VECTOR PROJECTIONS

$$L_{y,x} = \frac{\vec{y}\,\vec{x}}{\|\vec{x}\|}$$

$\vec{y}$
90°
$\vec{x}$
$L_{\vec{y},\vec{x}}$



Projection of vector **v** onto **u**:

$$\mathrm{proj}_u(v) = \frac{v \cdot u}{u \cdot u} u$$

If u is a unit vector (orthonormal basis):

$$\mathrm{proj}_u(v) = (v \cdot u)u$$

Hidden meaning:

Projection extracts the part of **v** that lies *in the direction* of **u**.

2.5 Deep Intuition (Most textbooks don't explain)

⭐ Projection = Best approximation

If you want to approximate vector **v** using direction **u**, projection gives:

- Minimum error
- Minimum squared distance
- Maximum information from v captured by u

This leads to **least squares regression**.

⭐ 2.6 Orthogonal Projection Onto Subspaces

Now instead of a single vector **u**, take a whole subspace **W**.

Projection of v onto W:

- Finds the closest vector to v inside W
- The error vector is orthogonal to W

This generalizes everything.

Formula (matrix form)

If W is spanned by orthonormal vectors $u_1, u_2, \ldots, u_k$:

$$\mathrm{proj}_W(v) = \sum_{i=1}^{k} (v \cdot u_i) u_i$$

This is exactly what PCA does.

2.7 Hidden Insights in ML & AI

Orthogonality + Projections appear everywhere:

✓ PCA = Projection onto orthogonal principal components

PCA finds directions (eigenvectors) where data has maximum variance and projects data onto them.

Projection removes noise and redundancy.

✓ Linear Regression = Projection onto column space

When solving:

$$Ax = b$$

But b is not exactly in the span of A's columns, the best solution is:

$$\hat{b} = \mathrm{proj}_{\mathrm{Col}(A)}(b)$$

That is why normal equations are:

$$A^T Ax = A^T b$$

They enforce the error to be orthogonal to all columns of A.

✓ Gradient Descent uses orthogonality

At optimum point:

- Gradient is orthogonal to the constraint surface
- Movement along tangent directions produces no change

✓ Fourier Transform = Projections onto sine/cosine basis

All signals are decomposed as:

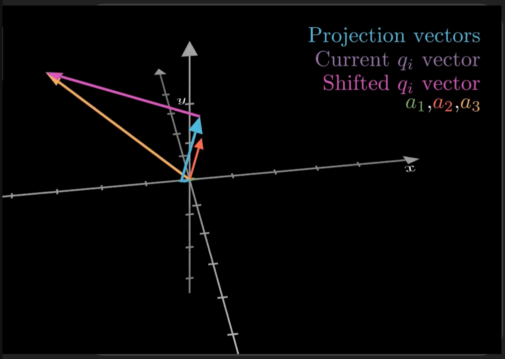
$$f(x) = \sum a_k \sin(kx) + b_k \cos(kx)$$

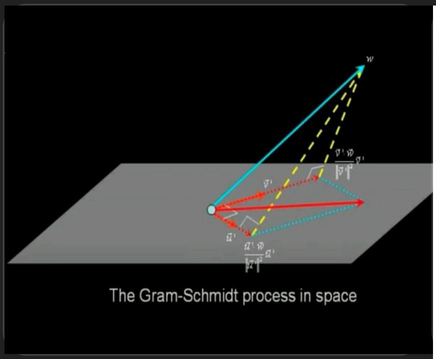
And those coefficients are projections.

✓ Transformers (attention heads) learn orthogonal directions

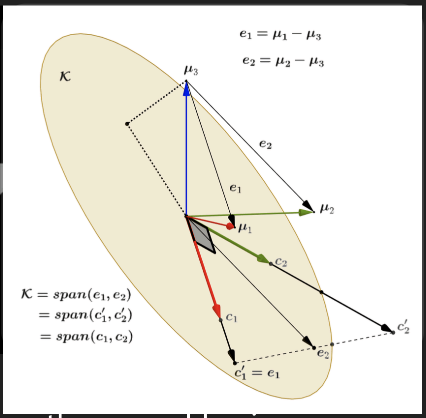
Orthogonal attention heads extract non-overlapping information.

2.8 Gram–Schmidt (Orthogonalizing any basis)





The Gram-Schmidt process in space



Given any basis vectors, Gram-Schmidt converts them into an orthonormal basis.

Process:

1. Take first vector, normalize it
2. Subtract its projection from next vector
3. Normalize the remainder
4. Repeat

This is the core operation behind QR decomposition.

⭐ 2.9 Projection Matrices (Very Important)

If matrix **U** has orthonormal columns:

$$P = UU^T$$

Then **P** is a projection matrix:

- Projects onto the column space of U
- Symmetric
- Idempotent: $P^2 = P$

This matrix appears in:

- Regression
- Variational autoencoders
- Optimization
- Geometry
- SVD

⭐ SUMMARY OF TOPIC 2

You learned:

✓ **Orthogonality = zero-overlap information**

✓ **Orthonormal basis simplifies all linear algebra**

✓ **Projection = best approximation in a subspace**

✓ **Linear regression is projection**

✓ **PCA is projection onto eigenvectors**

✓ **Gram–Schmidt builds orthonormal bases**

✓ **Projection matrices appear everywhere in ML/DL**

NEXT TOPIC

Now we move to a *huge* concept:

⭐ TOPIC 3: EIGEN DECOMPOSITION, SVD, PCA

This is the mathematical engine behind:

- Principal Component Analysis
- Dimensionality reduction
- Recommendation systems
- Image compression
- Latent semantic analysis
- Neural networks

Should I continue to **Eigenvalues, Eigenvectors, and their deeper meaning**?

sure , make detail explanation

Perfect — now we enter the **heart of advanced linear algebra**, the mathematical engine behind **machine learning, deep learning, compression, optimization, and modern AI**:

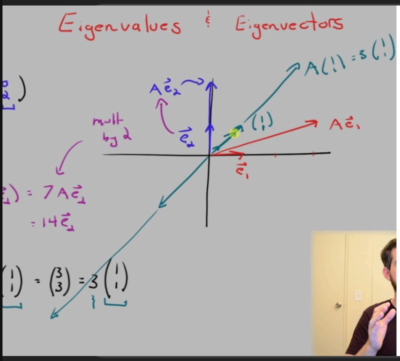
⭐ TOPIC 3: EIGEN DECOMPOSITION, SVD, PCA

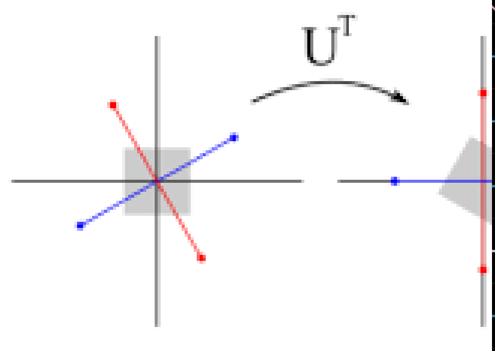
This explanation is **exceptionally detailed**, with hidden geometry, intuitive interpretations, real ML applications, and full mathematical insights.

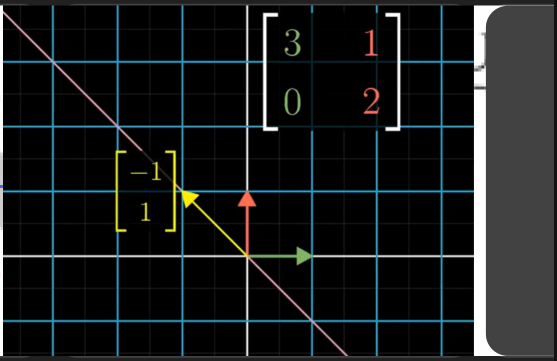
PART A — EIGENVALUES & EIGENVECTORS

Let's start with the foundational idea.

3.1 What are Eigenvectors? (Geometric Truth)






$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$
$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

A **linear transformation** (matrix **A**) acts on vectors by:

- Rotating
- Stretching

- Squashing
- Reflecting
- Shearing

But **eigenvectors are special directions** where A acts *only* by stretching or squashing — **no rotation**.

Mathematically:

$$Av = \lambda v$$

Where

- **v** = eigenvector (direction preserved)
- **λ** = eigenvalue (stretch factor)

Hidden Insight 1

Eigenvectors are the **natural coordinate system** of the transformation.

Hidden Insight 2

Eigenvalues tell you how the space deforms:

- |λ| > 1 → expands
- |λ| < 1 → contracts
- λ = 0 → collapses a direction
- λ negative → flips direction

3.2 Why Only Special Directions Are Eigenvectors?

Because a matrix usually mixes coordinates:

$$A \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

But eigenvectors solve for directions where the output stays **exactly on the same line** as input.

This is deep:
Eigenvectors are **invariant directions** under A.

3.3 Algebraic Equation for Eigenvalues

Starting from:

$$Av = \lambda v$$

Rearrange:

$$(A - \lambda I)v = 0$$

For non-zero v, determinant must be zero:

$$\det(A - \lambda I) = 0$$

This polynomial is the **characteristic equation**.

3.4 Intuition: What Does λ Measure?

Think of eigenvalues as:

- Strength of the transformation
- How much variance a direction captures (in PCA)
- Importance of a feature direction
- Sensitivity of the system

3.5 Eigen Decomposition

If a matrix is diagonalizable:

$$A = V \Lambda V^{-1}$$

Where

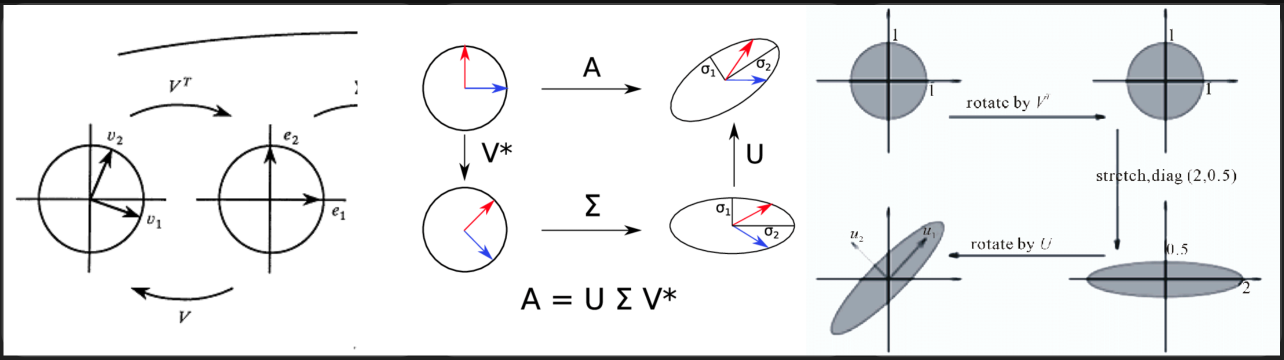
- V = matrix of eigenvectors
- Λ = diagonal matrix of eigenvalues

Hidden meaning

The transformation A =

1. Change basis to eigenvector system (V^{-1})
2. Stretch each coordinate by eigenvalues (Λ)
3. Convert back to original basis (V)

PART B — SVD (Singular Value Decomposition)



Now the **most powerful decomposition in all of linear algebra**.

3.6 What is SVD?

For ANY matrix A (square or rectangular):

$$A = U\Sigma V^T$$

Where

- **V** = orthonormal basis for input space
- **U** = orthonormal basis for output space
- **Σ** = diagonal matrix of singular values ($\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$)

Deep hidden truth

SVD finds **the best coordinate systems** for both domain and range.

- V gives directions in which data should be taken
- U gives directions of output
- Σ tells how strongly each direction is scaled

SVD = eigen decomposition of a rectangular matrix.

Because:

$$A^T A = V\Sigma^2 V^T$$

$$AA^T = U\Sigma^2 U^T$$

3.7 Geometric Interpretation

Any matrix transformation A does the following:

1. **Rotate** — (given by V^T)
2. **Stretch** — (singular values σ_i)
3. **Rotate again** — (U)

This is the most general transformation in linear algebra.

3.8 Why Is SVD More Powerful Than Eigen Decomposition?

Eigen decomposition works only if:

- matrix is square
- matrix is diagonalizable

SVD works for every matrix.

That’s why ML uses SVD everywhere.

3.9 Meaning of Singular Values (σ)

Singular values tell you:

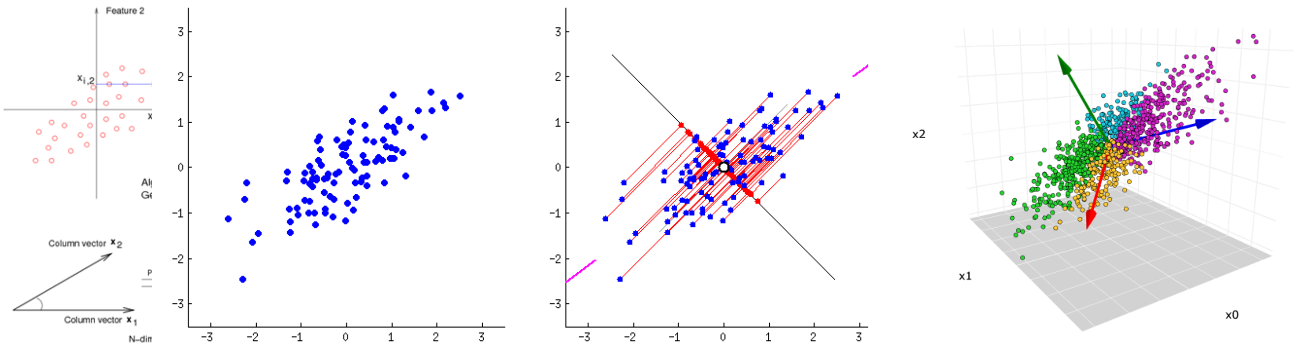
- How important each dimension is
- How much variance is captured
- How sensitive transformation is
- How many independent features the matrix has

If σk = 0 → the matrix loses that dimension.

This is key for **rank**:

$$\mathrm{rank}(A) = \#\text{nonzero singular values}$$

PART C — PCA (Principal Component Analysis)



PCA is essentially **eigen decomposition of the covariance matrix.**

3.10 What PCA Does

Given data matrix X:

1. Center the data
2. Compute covariance matrix $C = \frac{1}{n}X^T X$
3. Compute eigenvectors and eigenvalues of C
4. Sort eigenvectors by descending eigenvalues
5. Project data onto top-k eigenvectors

3.11 Hidden meaning (rarely explained)

✓ **Eigenvectors = directions of maximum variance**

✓ **Eigenvalues = amount of variance in those directions**

✓ **PCA finds new orthogonal basis that best describes the data**

So PCA = **best low-dimensional representation** of data.

3.12 PCA and SVD Connection

Instead of eigen decomposition, PCA is faster using SVD:

$$X = U\Sigma V^T$$

Then:

- Principal components = columns of V
- Variances = squares of singular values (σ^2)
- Reduced data = XV_k

This is why almost all ML libraries use SVD under the hood.

3.13 Hidden Geometric Insight

PCA projection is simply:

✓ **Shadow of data onto important directions**

✓ **Removing noise by ignoring small-variance directions**

✓ **Keeping structure while eliminating redundancy**

3.14 PCA in Machine Learning

Used in:

- Dimensionality reduction
- Noise filtering
- Compression
- Visualization (2D, 3D)
- Preprocessing before clustering
- Covariance analysis
- Autoencoders (nonlinear generalization of PCA)

3.15 Real-world applications

✓ **Face compression (eigenfaces)**

✓ **Recommender systems (matrix factorization)**

✓ **Latent semantic indexing (text)**

✓ **Image similarity**

✓ **Genetic data analysis**

✓ **Portfolio risk analysis**

PART D — Hidden Connections (A-Level Insights)

3.16 Eigenvectors = Stable directions

If you repeatedly apply matrix A:

$$A^k v$$

The direction converges to the eigenvector with largest eigenvalue.

This is how **PageRank** works.

3.17 SVD = Best Low Rank Approximation

Truncate SVD:

$$A_k = U_k \Sigma_k V_k^T$$

This gives **best possible** rank-k approximation.

Used in:

- Image compression
- LLM embedding compression
- Recommender systems
- Spectral clustering

3.18 PCA = SVD applied to data variance

Covariance matrix:

$$C = X^T X$$

Its eigenvectors = PCA directions.

3.19 Relationship Summary

Concept	**Meaning**	**Matrix**
Eigen decomposition	Natural directions of A	$A = V\Lambda V^{-1}$
SVD	Best input/output directions	$A = U\Sigma V^T$
PCA	Best variance directions	Covariance eigenvectors

⭐ FINAL SUMMARY OF TOPIC 3

You now understand:

✔ **What eigenvectors/eigenvalues *really* are**

✔ **How matrices stretch, rotate, and collapse space**

✔ **Why eigen decomposition works**

✔ **What SVD is and why it is universal**

✔️ PCA as projection on eigenvectors of covariance

✔️ Deep ML applications

This is graduate-level understanding.

❓ READY FOR THE NEXT TOPIC?

Next in your list is:

⭐ TOPIC 4 — TENSORS

This will include:

- What a tensor *really* is
- Tensor ranks, modes
- Tensor contractions
- Tensor decomposition (CP, Tucker)
- How tensors generalize matrices
- Why deep learning depends on tensors
- How transformers use tensor operations

Shall I continue with **Tensors** in the same deep, intuitive, mathematical style?

sure

Great — now we enter one of the **most misunderstood but most powerful ideas** in all of modern mathematics and machine learning:

⭐ TOPIC 4 — TENSORS

This explanation goes **far deeper** than the usual “tensors are multi-dimensional arrays” nonsense.

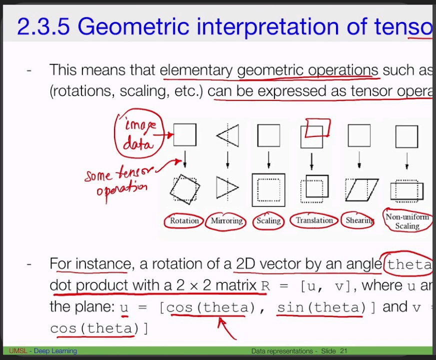
You will learn:

- What tensors *really* are
- How they generalize scalars, vectors, matrices
- Tensor **rank**, **order**, **modes**
- Tensor **operations & contractions**

- Tensor **decompositions** (CP, Tucker, Tensor SVD)
- Why **deep learning** runs entirely on tensor algebra
- How **Transformers** = tensor contraction engines

This is advanced, intuitive, and mathematically grounded.

⭐ 4.1 What IS a Tensor? (The True Meaning)



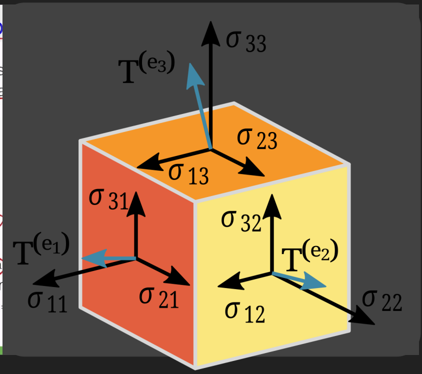
2.3.5 Geometric interpretation of tenso

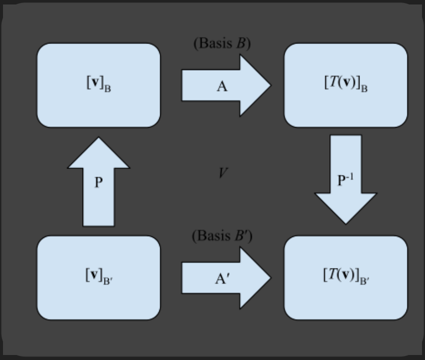
- This means that elementary geometric operations such as (rotations, scaling, etc.) can be expressed as tensor opera

image data
some tensor operation
Rotation
Mirroring
Scaling
Translation
Shearing
Non-uniform Scaling

- For instance, a rotation of a 2D vector by an angle `theta` dot product with a 2 × 2 matrix `R = [u, v]`, where `u` ar the plane: `u = [cos(theta), sin(theta)]` and `v` `cos(theta)]`

UMSL - Deep Learning Data representations - Slide 21





You’ve seen tensors in PyTorch/NumPy as multi-dimensional arrays.
But **that’s just representation**, not definition.

✔ TRUE definition (mathematically)

A tensor is a **multilinear map** between vector spaces.

For example:

- Scalar = 0th-order tensor
- Vector = 1st-order tensor
- Matrix = 2nd-order tensor
- 3D array = 3rd-order tensor
- ... and so on

But the key idea:

⭐ A tensor describes how multiple vectors interact linearly.

It is not the data — it is the rule for combining vectors.

⭐ 4.2 Orders of Tensors (Rank of a Tensor ≠ Rank of a Matrix)

The **order** (or degree) of a tensor is:

- Scalar → order 0
- Vector → order 1
- Matrix → order 2
- 3D array → order 3
- etc.

Hidden fact:

Order is also called **mode**, **way**, or **axis**.

✓ Tensor rank = how many rank-1 tensors are needed to build it

This is different from **matrix rank**, which counts linearly independent columns.

⭐ 4.3 Tensors as Multidimensional Linear Data

Imagine representing:

- Image (height × width × channels) → rank-3 tensor
- Video (frames × height × width × channels) → rank-4 tensor
- Batch of videos → rank-5 tensor
- Transformer attention → rank-4 or rank-5 tensor
- LLM weights → huge rank-4 tensors

Deep learning literally runs tensor contractions billions of times per second.

⭐ 4.4 Tensor Modes

For a tensor T with size:

$$n_1 \times n_2 \times \cdots \times n_k$$

We call dimension n_i the **i-th mode**.

Example: For an image tensor 512 × 512 × 3:

- Mode-1: height
- Mode-2: width
- Mode-3: channels

Changing basis in each mode is possible separately!

This is what makes tensor algebra far more expressive than matrices.

⭐ 4.5 Tensor Operations

Most important: **contraction**.

Contraction = multidimensional generalization of:

- Matrix multiplication
- Dot product
- Summing over shared indices

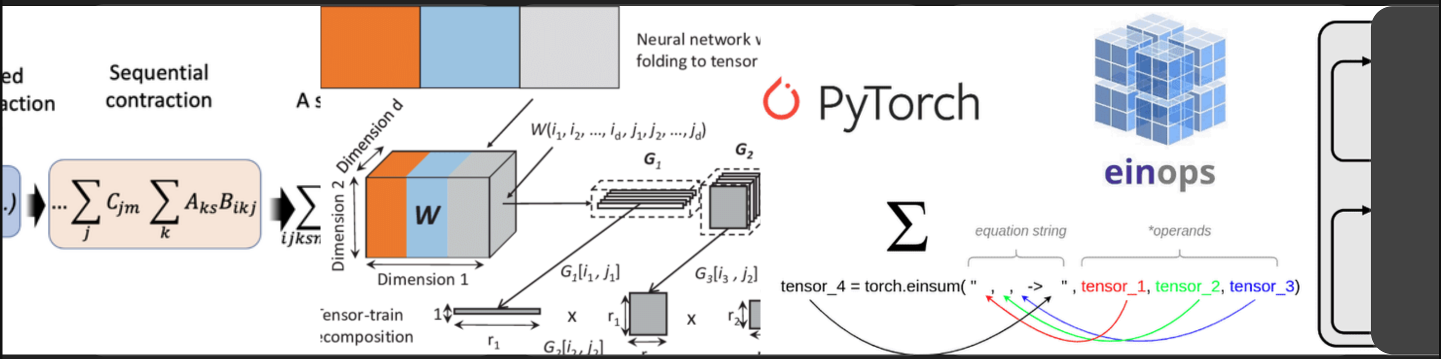
It reduces the order of a tensor by summing over matching indices.

Example:

$$C_{ij} = \sum_k A_{ik} B_{kj}$$

This is matrix multiplication, but also a tensor contraction.

⭐ 4.6 Tensor Contraction Example (Deep Learning)



Transformer attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Here:

- Q = batch × heads × seq × dim
- K = batch × heads × seq × dim
- V = batch × heads × seq × dim

The operations inside involve **tensor contractions** along the "dim" and "seq" axes.

PyTorch implements these using:

`torch.einsum("bhqd, bhkd -> bhqk")`

Einstein summation = elegant tensor contraction notation.

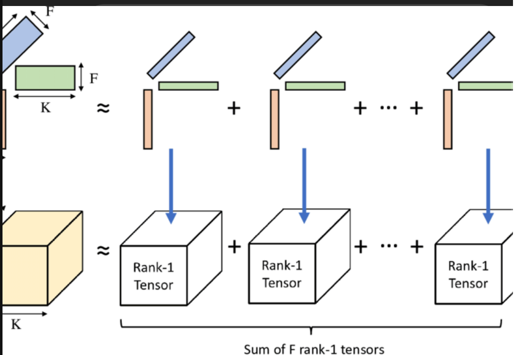
⭐ 4.7 Tensor Decompositions (Advanced and Important)

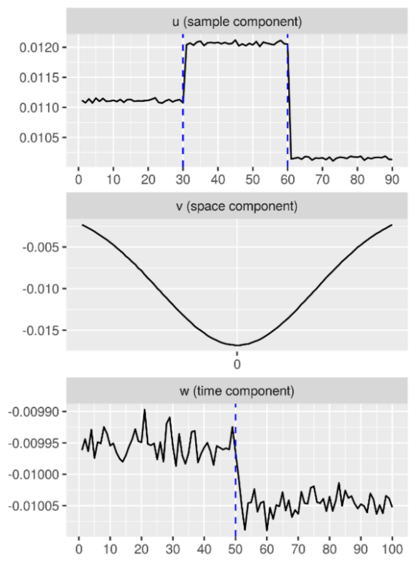
Matrix decompositions (SVD, eigen-decomposition) generalize to tensors.

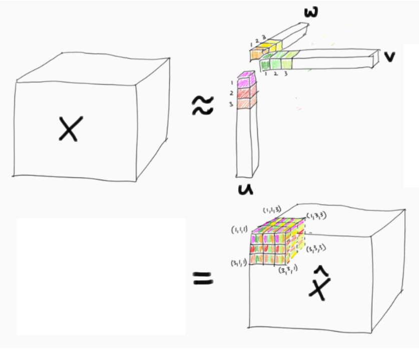
But unlike matrices, **tensors do not have a single universal decomposition**.

Here are the three biggest:

✔️ (1) CP Decompos P/PARAFAC)







Generalizes expressing a matrix as sum of rank-1 matrices.

A 3-way tensor T is decomposed as:

$$T \approx \sum_{r=1}^{R} a_r \otimes b_r \otimes c_r$$

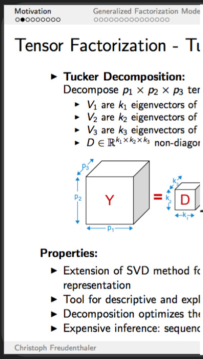
Where ⊗ = outer product.

This expresses T as a sum of rank-1 tensors.

Uses:

- Topic modeling
- Recommender systems
- Multi-relational data
- Neuroscience tensor analysis

✔ (2) Tucker Decomposition


Motivation Generalized Factorization Mode

Tensor Factorization - Tu

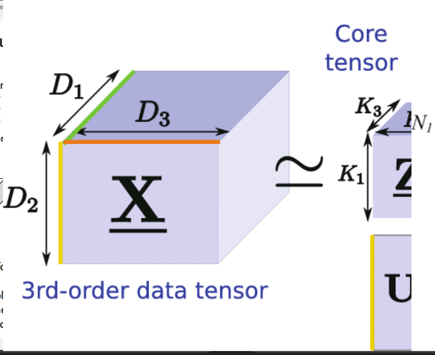
- **Tucker Decomposition:**
  Decompose $p_1 \times p_2 \times p_3$ ter
  - V_1 are k_1 eigenvectors of
  - V_2 are k_2 eigenvectors of
  - V_3 are k_3 eigenvectors of
  - $D \in \mathbb{R}^{k_1 \times k_2 \times k_3}$ non-diagor

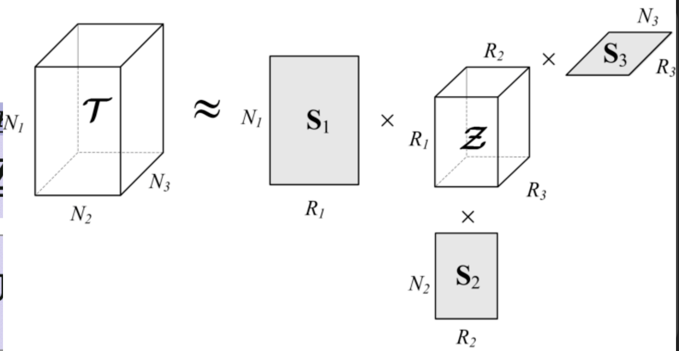
p3
p2
Y
p1
=
k3
k2
D
k1

Properties:

- Extension of SVD method fo representation
- Tool for descriptive and expl
- Decomposition optimizes the
- Expensive inference: sequenc

Christoph Freudenthaler





Generalizes SVD to tensors.

$$T = G \times_1 A \times_2 B \times_3 C$$

Where:

- G = core tensor
- A, B, C = factor matrices
- $\times_n$ = mode-n product

This allows compression and dimensionality reduction along each mode separately.

Uses:

- Multimodal data fusion
- Image compression
- Video compression
- Denoising

✔ (3) Tensor SVD (t-SVD)

Instead of decomposing in real domain, t-SVD uses **Fourier transforms** along one mode.

Allows convolution-like operations.

Used in:

- Hyperspectral image compression
- Video analysis
- 3D medical imaging

⭐ 4.8 Why Deep Learning Uses Tensors Everywhere

Neural networks operate on tensors:

Object	Tensor Order
Scalar loss	0
Weight vector	1
Weight matrix	2
Image batch	4
Transformer attention	4–5
LLM weights	4
Word embeddings	2

Reason:

Neural networks learn **multilinear interactions** across:

- Batch dimension
- Sequence dimension
- Feature dimension
- Head dimension
- Layer dimension

This is fundamentally tensor algebra.

⭐ 4.9 Geometry of Tensors (Hidden Insight)

Matrices act on vectors:

$$A : \mathbb{R}^n \to \mathbb{R}^m$$

Tensors act on **multiple vectors simultaneously**:

$$T : (V_1, V_2, \ldots, V_k) \to \mathbb{R}$$

This allows representing **complex interactions**, not just linear ones.

Examples:

- Diffusion tensors (brain connectivity)

- Elasticity tensors (physics)
- Neural activations (multi-way relationships)
- Multi-modal embeddings

⭐ 4.10 Tensor Rank is Much More Complex Than Matrix Rank

Matrix rank is easy: number of non-zero singular values.

Tensor rank is HARD:

- NP-hard to compute
- Often not unique
- Depends on decomposition type

This is why tensor methods are powerful but computationally expensive.

⭐ 4.11 Examples in Modern AI

✔️ Transformers

Attention is tensor contraction.

✔️ CNNs

Convolutions are multi-dimensional tensor operations.

✔️ LLM Weights

Stored as huge 4D tensors:
(shape: layers × heads × dim × dim)

✔️ Training

Backprop = sequence of tensor contractions + elementwise ops.

✔️ Embeddings

High-dimensional tensors representing semantic meaning.

⭐ 4.12 Summary of Tensor Concepts

✓ **Tensor = multilinear map**

✓ **Order = number of modes**

✓ **Rank = minimum rank-1 tensors needed**

✓ **Contractions = generalized matrix multiplication**

✓ **Decompositions (CP, Tucker, t-SVD) extend SVD to tensors**

✓ **Deep learning is fundamentally tensor algebra**

✓ **Transformers = tensor contraction engines**

This completes the deepest topic in your list.

🎯 NEXT STEPS

We have now covered:

1. Vector spaces
2. Orthogonality & projections
3. Eigenvalues, SVD, PCA
4. Tensors

If you'd like, I can now create:

✅ A **connected map** showing how all these topics unify into ML/DL
OR
✅ Give **advanced practice problems** (theoretical + ML-based)
OR
✅ Move into even more advanced linear algebra (e.g., spectral graph theory, operator theory)

What would you like next?