LINEAR ALGEBRA

The Complete Beginner's Guide

From Vectors to Machine Learning

Step-by-Step • Practice Problems • ML Applications

📚 10 Chapters	✏ 50+ Practice Problems	🤖 ML Connections	💡 Visual Explanations

Prepared for Joel • February 2026

Table of Contents

Chapter 1	**Scalars, Vectors & Their Operations** The building blocks
Chapter 2	**Vector Spaces & Linear Combinations** Span, basis, and independence
Chapter 3	**Matrices: The Core Tool** Creation, types, and basic operations
Chapter 4	**Matrix Multiplication** The most important operation
Chapter 5	**Linear Transformations** What matrices really do
Chapter 6	**Systems of Linear Equations** Solving Ax = b
Chapter 7	**Determinants** Measuring scale and invertibility
Chapter 8	**Eigenvalues & Eigenvectors** The hidden structure
Chapter 9	**Norms, Dot Products & Distances** Measuring in vector spaces
Chapter 10	**Putting It All Together: ML Applications** PCA, regression, and neural nets

How to Use This Guide

This guide is designed to take you from zero knowledge to a solid understanding of linear algebra, with a special focus on the concepts most relevant to machine learning. Here's what each section contains:

■ **Key Insight** - Important concepts and intuitions you should remember

✍ **Practice Problems** - Hands-on exercises to test your understanding

■ **Answers** - Solutions to practice problems - try solving first!

■ **ML Connection** - How this concept is used in machine learning

Difficulty Progression: Each chapter builds on the previous one. Chapters 1-3 are **Easy**, Chapters 4-6 are **Medium**, and Chapters 7-10 are **Advanced**. Take your time with each level before moving on.

Chapter 1

Scalars, Vectors & Their Operations

EASY

1.1 What is a Scalar?

A **scalar** is simply a single number. That's it! Examples: 5, -3.14, 0.001, 100. In math, we usually write scalars as lowercase italic letters like *a*, *b*, *c*.

Definition: Scalar

A scalar is a single numerical value. It represents a magnitude (size) without direction. In programming, think of it as a single variable holding one number: `x = 5`

1.2 What is a Vector?

A **vector** is an ordered list of numbers. Think of it as a container that holds multiple numbers in a specific order. Each number in the vector is called a **component** or **element**.

Physical intuition: A vector represents both a **direction** and a **magnitude** (length). Imagine an arrow pointing from one place to another - the direction it points and how long it is together define the vector.

2D Vector	3D Vector	n-D Vector
`v = [3, -5]` x-component: 3 y-component: -5	`v = [3, -5, 4]` x: 3, y: -5, z: 4	`v = [v1, v2, ..., vn]` n components

■ **Key Insight**

In machine learning, data is stored as vectors. A single data point with 10 features becomes a 10-dimensional vector. An image with 784 pixels becomes a 784-dimensional vector. The math works the same regardless of dimension!

1.3 Vector Addition

To add two vectors, simply add their corresponding components. Both vectors **must have the same dimension** (same number of components).

```
[3, -5] + [2, 1] = [3+2, -5+1] = [5, -4]
[3, -5, 4] + [2, 1, -3] = [3+2, -5+1, 4+(-3)] = [5, -4, 1]
```

Physical intuition: Vector addition is like following directions. Walk along vector A, then walk along vector B. Where you end up is the result of A + B. This is called the "tip-to-tail" method.

Properties of vector addition:

• **Commutative:** a + b = b + a (order doesn't matter)

• **Associative:** (a + b) + c = a + (b + c) (grouping doesn't matter)

• **Zero vector:** a + 0 = a (adding zero changes nothing)

1.4 Scalar Multiplication

To multiply a vector by a scalar, multiply **every component** by that scalar.

```
2 * [3, -5] = [2*3, 2*(-5)] = [6, -10]
-1 * [3, -5] = [-3, 5] (reverses direction!)
0.5 * [3, -5] = [1.5, -2.5] (halves the length)
```

Physical intuition: Scalar multiplication **stretches** or **shrinks** a vector. Multiply by 2 = twice as long. Multiply by 0.5 = half as long. Multiply by -1 = same length but opposite direction.

1.5 Vector Subtraction

Subtraction is just addition with a negated vector:

```
a - b = a + (-1 * b)
[3, -5] - [2, 1] = [3-2, -5-1] = [1, -6]
```

■ Machine Learning Connection

In machine learning, these basic operations are everywhere:

• **Feature vectors:** Each data point is a vector. A house described by [price, sqft, bedrooms] = [250000, 1500, 3]

• **Weight updates:** In gradient descent, weights are updated by: w = w - learning_rate * gradient (scalar multiplication + subtraction!)

• **Batch processing:** Adding vectors lets you combine predictions from multiple models

```
Python: import numpy as np
v1 = np.array([3, -5])
v2 = np.array([2, 1])
print(v1 + v2) # [5, -4]
print(2 * v1) # [6, -10]
```

✍ Practice Problems

P1.1 Calculate: [4, -2, 7] + [1, 5, -3]

P1.2 Calculate: 3 * [2, -1, 4]

P1.3 Calculate: [10, 5] - [3, 8]

P1.4 If v = [1, 2, 3] and w = [4, 5, 6], compute 2v + 3w

P1.5 A robot moves [3, 2] then [-1, 4] then [2, -3]. Where does it end up?

■ Answers

A1.1 [5, 3, 4]

A1.2 [6, -3, 12]

A1.3 [7, -3]

A1.4 2*[1,2,3] + 3*[4,5,6] = [2,4,6] + [12,15,18] = [14, 19, 24]

A1.5 [3+(-1)+2, 2+4+(-3)] = [4, 3]

Chapter 2

Vector Spaces & Linear Combinations

EASY

2.1 Linear Combinations

A **linear combination** takes multiple vectors, scales each by a scalar, and adds them together. This is the most fundamental operation in all of linear algebra.

Definition: Linear Combination

Given vectors $v_1, v_2, ..., v_n$ and scalars $c_1, c_2, ..., c_n$, the linear combination is: $c_1v_1 + c_2v_2 + ... + c_nv_n$

Example: Given v_1 = [1, 0] and v_2 = [0, 1]:

```
3*[1,0] + 2*[0,1] = [3,0] + [0,2] = [3, 2]
```

The scalars 3 and 2 are called **coefficients** or **weights**.

2.2 Span

The **span** of a set of vectors is the collection of ALL possible linear combinations of those vectors. It answers the question: "What points can I reach using these vectors?"

■ **Key Insight**

Think of it like mixing paint colors:

• With just red paint, you can only make shades of red (a line)

• With red AND blue paint, you can make reds, blues, and purples (a plane)

• With red, blue, AND yellow, you can reach almost any color (3D space)

Each new independent vector "opens up" a new dimension of possibility.

2.3 Linear Independence

Vectors are **linearly independent** if none of them can be written as a linear combination of the others. In other words, each vector adds a genuinely new direction.

Independent ■	**Dependent** ■
[1,0] and [0,1] *Neither is a multiple of the other. They point in completely different directions.*	[1,2] and [2,4] *[2,4] = 2*[1,2]. The second vector is just the first stretched by 2. Same direction!*

2.4 Basis

A **basis** is a set of linearly independent vectors that spans the entire space. It's the minimum set of vectors needed to describe every point in the space.

The **standard basis** for 2D is: e_1 = [1, 0] and e_2 = [0, 1]

The **standard basis** for 3D is: e_1 = [1, 0, 0], e_2 = [0, 1, 0], e_3 = [0, 0, 1]

Any 2D vector can be written using the standard basis: [3, -5] = 3*[1,0] + (-5)*[0,1]

2.5 Dimension

The **dimension** of a vector space is the number of vectors in any basis. 2D space has dimension 2 (need 2 basis vectors), 3D space has dimension 3, and so on.

■ Machine Learning Connection

Linear combinations are the foundation of machine learning:

• **Neural network output:** Each neuron computes a linear combination of inputs: $y = w_1x_1 + w_2x_2 + ... + w_nx_n$ + bias
• **Feature engineering:** Creating new features from existing ones is often a linear combination
• **Linear regression:** The prediction is literally a linear combination: y = w_1*feature_1 + w_2*feature_2 + ...
• **Word embeddings:** king - man + woman ≈ queen (vector arithmetic on word vectors!)

✍ Practice Problems

P2.1 Express [7, 11] as a linear combination of [1, 2] and [3, 1]

P2.2 Are [1, 2, 3] and [2, 4, 6] linearly independent? Why or why not?

P2.3 Can you reach the point [5, 5] using only multiples of [1, 1]? What about [3, 7]?

P2.4 How many vectors are in the standard basis for 5D space?

■ Answers

A2.1 Solve: c1*[1,2] + c2*[3,1] = [7,11]. From row 1: c1 + 3c2 = 7. From row 2: 2c1 + c2 = 11. Solving: c1 = 4, c2 = 1. Check: 4*[1,2] + 1*[3,1] = [4,8] + [3,1] = [7,11] ■

A2.2 No! [2,4,6] = 2*[1,2,3]. They are linearly dependent (same direction, different length).

A2.3 [5,5] = 5*[1,1] ■. But [3,7] cannot be reached - any multiple of [1,1] has equal components.

A2.4 5 vectors (one for each dimension).

Chapter 3

Matrices: The Core Tool

EASY

3.1 What is a Matrix?

A **matrix** is a rectangular grid of numbers arranged in rows and columns. A matrix with *m* rows and *n* columns is called an **m x n matrix** (read as "m by n").

> **Definition: Matrix**
>
> An m x n matrix A has m rows and n columns. We write A_{ij} to refer to the element in row i, column j. Matrices are usually written with capital bold letters: **A**, **B**, **C**.

2x3 Matrix	3x2 Matrix	2x2 Matrix
| 1 2 3 | | 4 5 6 | *2 rows, 3 columns*	| 1 2 | | 3 4 | | 5 6 | *3 rows, 2 columns*	| 1 2 | | 3 4 | *Square matrix*

3.2 Special Types of Matrices

Type	Description	Example
Identity (I)	1s on diagonal, 0s elsewhere. Like the number 1 for matrices.	|1 0| |0 1|
Zero Matrix	All elements are 0. Like the number 0.	|0 0| |0 0|
Diagonal	Non-zero only on main diagonal.	|3 0| |0 7|
Symmetric	$A = A^T$ (same when flipped over diagonal).	|1 2| |2 3|

Transpose (A^T)	Rows become columns, columns become rows.	|1 2|^T |3 4| = |1 3| |2 4|

3.3 Matrix Addition & Scalar Multiplication

Just like vectors! Add element-by-element (matrices must be same size), or multiply every element by a scalar.

```
|1 2| + |5 6| = |1+5 2+6| = |6 8 |
|3 4| |7 8| |3+7 4+8| |10 12|

3 * |1 2| = |3 6 |
|3 4| |9 12|
```

■ **Key Insight**

A vector is actually a special case of a matrix! A column vector with n elements is an n x 1 matrix. A row vector is a 1 x n matrix. This is why the same rules apply to both.

■ **Machine Learning Connection**

Matrices are the language of machine learning:

• **Datasets:** Your entire dataset is a matrix! Each row is a data point, each column is a feature.

• **Weight matrices:** Neural network layers store their weights in matrices.

• **Images:** A grayscale image is a matrix of pixel values (0-255). A color image is 3 matrices (R, G, B).

```
Python: dataset = np.array([[5.1, 3.5, 1.4], # iris sample 1
[4.9, 3.0, 1.4], # iris sample 2
[4.7, 3.2, 1.3]]) # iris sample 3
```

✍ Practice Problems

P3.1 What is the size of a matrix with 4 rows and 3 columns?

P3.2 Add: |1 2 3| + |4 5 6|. Write each row.

|4 5 6| |7 8 9|

P3.3 Find the transpose of: |1 2 3|

|4 5 6|

P3.4 Is |1 3| symmetric? Why or why not?

|3 5|

P3.5 If your dataset has 1000 samples and 50 features, what size is the data matrix?

■ Answers

A3.1 4 x 3 (4-by-3)

A3.2 |5 7 9| / |11 13 15|

A3.3 |1 4| / |2 5| / |3 6| (3x2 matrix - rows and columns swap)

A3.4 Yes! Flipping over the diagonal gives the same matrix ($A=A^T$).

A3.5 1000 x 50

Chapter 4

Matrix Multiplication

MEDIUM

4.1 The Key Rule: Dot Product

Before matrix multiplication, you need to understand the **dot product** of two vectors. Multiply corresponding elements and add them up:

```
[1, 2, 3] . [4, 5, 6] = 1*4 + 2*5 + 3*6 = 4 + 10 + 18 = 32
```

The dot product takes two vectors and returns a **single number** (a scalar).

4.2 How Matrix Multiplication Works

Matrix multiplication is NOT element-by-element! Instead, each element of the result is a **dot product** between a row of the first matrix and a column of the second.

■ Key Insight

The Size Rule: To multiply A (m x n) by B (n x p), the number of columns in A MUST equal the number of rows in B. The result will be an m x p matrix.

- (2x**3**) times (**3**x4) = (2x4) ■ The 3s match!
- (2x**3**) times (**2**x4) = ERROR ■ 3 does not equal 2!

Step-by-step example:

```
|1 2| x |5 6| = ?
|3 4|   |7 8|
```

Position	Calculation	Result
Row 1, Col 1	[1,2] . [5,7] = 1*5 + 2*7	**19**
Row 1, Col 2	[1,2] . [6,8] = 1*6 + 2*8	**22**
Row 2, Col 1	[3,4] . [5,7] = 3*5 + 4*7	**43**

Row 2, Col 2	[3,4] . [6,8] = 3*6 + 4*8	**50**

```
Result = |19 22|
         |43 50|
```

4.3 Important Properties

• **NOT commutative:** AB does NOT equal BA in general! Order matters!

• **Associative:** (AB)C = A(BC)

• **Distributive:** A(B + C) = AB + AC

• **Identity:** AI = IA = A (the identity matrix is like multiplying by 1)

4.4 Matrix-Vector Multiplication

A very common operation: multiplying a matrix by a vector. This is just matrix multiplication where the second matrix has only one column.

```
|2 1| x |3| = |2*3 + 1*(-1)| = | 5|
|0 3|   |-1|  |0*3 + 3*(-1)|   |-3|
```

■ **Machine Learning Connection**

Matrix multiplication is THE core computation in ML:

• **Forward pass in neural networks:** output = W * input + bias (matrix-vector multiply!)

• **Batch processing:** Multiply weight matrix by entire batch matrix at once

• **Attention mechanism (Transformers):** Q * K^T (matrix multiplication of query and key matrices)

```
Python: A = np.array([[1,2],[3,4]])
B = np.array([[5,6],[7,8]])
C = A @ B # or np.dot(A, B)
print(C) # [[19 22] [43 50]]
```

✍ Practice Problems

P4.1 Calculate the dot product: [2, 3, 1] . [4, -1, 5]

P4.2 Can you multiply a (3x2) matrix by a (3x2) matrix? Why or why not?

P4.3 Multiply: |1 0| x |5|

|0 1| |3|

P4.4 Multiply: |2 0| x |1 3|

|0 3| |2 4|

P4.5 If A is 5x3 and B is 3x7, what size is AB?

■ Answers

A4.1 2*4 + 3*(-1) + 1*5 = 8 - 3 + 5 = 10

A4.2 No! Columns of first (2) must equal rows of second (3). 2 is not 3.

A4.3 [5, 3] - The identity matrix leaves vectors unchanged!

A4.4 |2*1+0*2 2*3+0*4| = |2 6 |

|0*1+3*2 0*3+3*4| |6 12|

A4.5 5 x 7

Chapter 5

Linear Transformations

MEDIUM

5.1 What is a Linear Transformation?

A **linear transformation** is a function that takes a vector as input and produces a vector as output, while preserving the operations of addition and scalar multiplication. The key insight: **every linear transformation can be represented as matrix multiplication!**

> **Definition: Linear Transformation**
>
> A function T is linear if: (1) T(u + v) = T(u) + T(v), and (2) T(cv) = cT(v) for any scalar c. In simple terms, it doesn't matter whether you add/scale before or after applying the transformation.

5.2 Common 2D Transformations

Transformation	Matrix	What it does
Scaling	$\begin{vmatrix} sx & 0 \\ 0 & sy \end{vmatrix}$	Stretches x by sx, y by sy
Rotation (90 deg)	$\begin{vmatrix} 0 & -1 \\ 1 & 0 \end{vmatrix}$	Rotates counterclockwise 90 degrees
Reflection (y-axis)	$\begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$	Flips across the y-axis
Shear	$\begin{vmatrix} 1 & k \\ 0 & 1 \end{vmatrix}$	Tilts shapes sideways by factor k

■ **Key Insight**

The Big Idea: Every matrix IS a transformation. When you multiply a matrix by a vector, you are transforming that vector. The columns of the matrix tell you where the basis vectors land after the transformation.

For |a b| : the first column [a,c] is where [1,0] goes.

|c d| the second column [b,d] is where [0,1] goes.

5.3 Composition of Transformations

Applying transformation A then transformation B is the same as applying the single transformation BA (note: B comes first in the product because it acts last). This is why matrix multiplication is not commutative - rotating then scaling is different from scaling then rotating!

■ **Machine Learning Connection**

Neural networks are sequences of linear transformations (with non-linear activations in between):

• **Each layer:** applies a linear transformation (matrix multiply) then a non-linear activation

• **Deep learning:** composing many transformations = multiplying many matrices

• **Convolutional layers:** special linear transformations that detect patterns in images

• **Data augmentation:** rotation, scaling, and shearing matrices transform training images

✍ **Practice Problems**

P5.1 Apply the scaling matrix |2 0| to the vector [3, 1]. What happens?

|0 2|

P5.2 Apply the rotation matrix |0 -1| to [1, 0]. Where does it go?

|1 0|

P5.3 What happens when you apply the identity matrix to any vector?

P5.4 If layer 1 has weight matrix W1 (3x5) and layer 2 has W2 (2x3), what is the size of an input vector, and what is the size of the final output?

■ Answers

A5.1 |2 0|*|3| = |6| - The vector doubled in length (scaled by 2).

|0 2| |1| |2|

A5.2 |0 -1|*|1| = |0| - The vector rotated 90 degrees counterclockwise!

|1 0| |0| |1|

A5.3 Nothing! The identity transformation leaves every vector unchanged. **I***v = v.

A5.4 **I**nput: 5-dimensional (5x1). After W1: 3-dimensional. After W2: 2-dimensional.

Chapter 6

Systems of Linear Equations

MEDIUM

6.1 What is a System of Equations?

A system of linear equations is a set of equations that must all be true simultaneously. Linear algebra gives us tools to solve these efficiently.

Example: Find x and y such that:

```
2x + 3y = 8
 x - y = 1
```

This can be written as the matrix equation **Ax = b**:

```
|2 3| * |x| = |8|
|1 -1| |y| |1|
```

6.2 Gaussian Elimination (Row Reduction)

The systematic way to solve systems. Write the **augmented matrix** and use row operations to reduce it to a simple form.

Step-by-step for our example:

Step 1: Write augmented matrix [A|b]:

```
| 2 3 | 8 |
| 1 -1 | 1 |
```

Step 2: R2 = R2 - (1/2)*R1 (eliminate x from row 2):

```
| 2 3 | 8 |
| 0 -5/2 | -3 |
```

Step 3: Back-substitute. From row 2: -5/2 * y = -3, so y = 6/5 = 1.2

From row 1: 2x + 3(1.2) = 8, so 2x = 4.4, x = 2.2

6.3 Three Possible Outcomes

One Solution	No Solution	Infinite Solutions
Lines cross at exactly one point. The system is **consistent** and the matrix is **invertible**.	Lines are parallel (never cross). The equations contradict each other.	Lines overlap completely. There are more unknowns than constraints.

6.4 The Matrix Inverse

If A is an invertible (square) matrix, then A^{-1} exists such that $A * A^{-1}$ = **I**. This lets us solve Ax = b directly:

$$Ax = b \dashrightarrow x = A^{-1} * b$$

2x2 Inverse Formula: For A = |a b| / |c d|:

$$A^{-1} = (1/(ad-bc)) * \begin{vmatrix} d & -b \\ -c & a \end{vmatrix}$$

The value (ad - bc) is the **determinant**. If it equals 0, the inverse does not exist!

■ Machine Learning Connection

Solving systems is central to ML optimization:

- **Linear regression (Normal Equation):** $w = (X^TX)^{-1} X^Ty$ solves for optimal weights directly
- **Least squares:** When no exact solution exists (overdetermined system), we find the best approximate solution
- **Regularization:** Ridge regression modifies the system to $(X^TX + \text{lambda}*I)^{-1} X^Ty$ for stability

```
Python: w = np.linalg.solve(X.T @ X, X.T @ y) # Efficient!
# Better than: w = np.linalg.inv(X.T @ X) @ X.T @ y
```

✍ Practice Problems

P6.1 Write as matrix equation Ax = b: 3x + y = 7, 2x - y = 3

P6.2 Find the inverse of |2 1| using the 2x2 formula.
|1 1|

P6.3 Solve using the inverse you found: 2x + y = 5, x + y = 3

P6.4 Does |1 2| have an inverse? (Hint: check the determinant)
|2 4|

■ Answers

A6.1 |3 1| * |x| = |7|

|2 -1| |y| |3|

A6.2 det = 2*1 - 1*1 = 1. Inverse = | 1 -1|

|-1 2|

A6.3 x = | 1 -1| * |5| = |5-3 | = |2|. So x=2, y=1.

|-1 2| |3| |-5+6| |1|

A6.4 det = 1*4 - 2*2 = 0. No inverse exists! (rows are proportional)

Chapter 7

Determinants

ADVANCED

7.1 What is a Determinant?

The **determinant** is a single number computed from a square matrix that tells you important things about the matrix and the transformation it represents.

> ■ **Key Insight**
>
> **Physical intuition:** The determinant measures how much a transformation **scales area** (in 2D) or **volume** (in 3D).
>
> • det = 2 means areas double
>
> • det = 0.5 means areas halve
>
> • det = 0 means everything collapses to a lower dimension (line or point)
>
> • det < 0 means the orientation flips (like looking in a mirror)

7.2 Computing Determinants

2x2 Determinant:

```
det |a b| = ad - bc
    |c d|

det |3 1| = 3*4 - 1*2 = 10
    |2 4|
```

3x3 Determinant (cofactor expansion along first row):

```
det |a b c|
|d e f| = a(ei-fh) - b(di-fg) + c(dh-eg)
|g h i|
```

7.3 Key Properties

• $\det(A) = 0$ means A is ~~singular~~ (not invertible) - the transformation collapses dimensions

• $\det(AB) = \det(A) * \det(B)$

• $\det(A^T) = \det(A)$

• $\det(cA) = c^n * \det(A)$ for an n x n matrix

• Swapping two rows negates the determinant

■ Machine Learning Connection

Determinants help diagnose problems in ML:

• **Feature redundancy:** If $\det(X^TX)$ is near zero, your features are highly correlated (multicollinearity)

• **Model stability:** A near-zero determinant means small data changes cause huge parameter changes

• **Gaussian distributions:** The multivariate normal distribution formula includes det(Sigma) for the covariance matrix

• **Volume of data:** The determinant of the covariance matrix measures how spread out your data is

✍ Practice Problems

P7.1 Calculate: det |5 3|

|2 4|

P7.2 Calculate: det |1 0|

|0 1| (identity matrix)

P7.3 If det(A) = 3 and det(B) = 4, what is det(AB)?

P7.4 A matrix has det = 0. Can you solve Ax = b for a unique x?

■ Answers

A7.1 5*4 - 3*2 = 20 - 6 = 14

A7.2 1*1 - 0*0 = 1 (the identity always has det = 1)

A7.3 det(AB) = det(A)*det(B) = 3*4 = 12

A7.4 No! det = 0 means the matrix is singular (not invertible). No unique solution.

Chapter 8

Eigenvalues & Eigenvectors

ADVANCED

8.1 The Big Idea

When you apply a matrix transformation to most vectors, both their direction AND length change. But some special vectors only get **stretched or shrunk** - their direction stays the same (or flips 180 degrees). These are **eigenvectors**, and the stretch factor is the **eigenvalue**.

> **Definition: Eigenvalue & Eigenvector**
>
> For a square matrix A, if Av = lambda * v (where v is not zero), then v is an **eigenvector** and lambda is the corresponding **eigenvalue**. The transformation A simply scales v by lambda.

Example: Let A = |2 1| and v = |1|

|0 3| |1|

```
A*v = |2 1| * |1| = |3| = 3 * |1|
      |0 3|   |1|   |3|       |1|
```

So [1, 1] is an eigenvector with eigenvalue 3. The matrix just tripled it!

8.2 How to Find Eigenvalues

Solve the **characteristic equation**: det(A - lambda*I) = 0

Example: A = |4 2|

|1 3|

```
A - lambda*I = |4-lambda 2 |
               | 1 3-lambda|
```

```
det = (4-lambda)(3-lambda) - 2*1 = 0
lambda^2 - 7*lambda + 10 = 0
(lambda - 5)(lambda - 2) = 0
```

```
lambda = 5 or lambda = 2
```

8.3 Key Properties

- An n x n matrix has n eigenvalues (counting repeats, possibly complex)
- The sum of eigenvalues = trace of A (sum of diagonal elements)
- The product of eigenvalues = determinant of A
- If any eigenvalue is 0, the matrix is singular (not invertible)
- Symmetric matrices always have real eigenvalues

■ Key Insight

Why eigenvalues matter: They reveal the "natural axes" of a transformation. Instead of a complex matrix operation, you can understand it as simple stretching along eigenvector directions. This is called **eigendecomposition**: $A = V * D * V^{-1}$, where D is a diagonal matrix of eigenvalues.

■ Machine Learning Connection

Eigenvalues and eigenvectors are critical in ML:

- **PCA (Principal Component Analysis):** Eigenvectors of the covariance matrix are the principal components. Eigenvalues tell you how much variance each component captures.
- **Google PageRank:** Web page importance is computed as the dominant eigenvector of the link matrix.
- **Spectral clustering:** Uses eigenvectors of the graph Laplacian to cluster data.
- **Stability analysis:** Eigenvalues of the Hessian matrix tell you if you're at a local minimum.

```
Python: eigenvalues, eigenvectors = np.linalg.eig(A)
```

✍ Practice Problems

P8.1 Verify: Is [1, 0] an eigenvector of |3 0|? If so, what is the eigenvalue?
|0 5|

P8.2 Find eigenvalues of |2 0|. (Hint: it's a diagonal matrix!)
|0 7|

P8.3 If eigenvalues of A are 3 and 5, what is det(A)?

P8.4 If eigenvalues of A are 3 and 5, what is the trace of A?

■ Answers

A8.1 |3 0|*|1| = |3| = 3*|1|. Yes! Eigenvalue = 3.

|0 5| |0| |0| |0|

A8.2 For diagonal matrices, eigenvalues ARE the diagonal entries: 2 and 7.

A8.3 det(A) = product of eigenvalues = 3 * 5 = 15

A8.4 trace(A) = sum of eigenvalues = 3 + 5 = 8

Chapter 9

Norms, Dot Products & Distances

ADVANCED

9.1 Vector Norms (Length/Magnitude)

A **norm** measures the size (length) of a vector. There are several types:

Norm	Formula	Intuition	Example [3, -4]
L1 (Manhattan)	$\|\|v\|\|_1 = \|v_1\| + \|v_2\| + ...$	Walk along grid lines	3 + 4 = **7**
L2 (Euclidean)	$\|\|v\|\|_2 = \text{sqrt}(v_1^2 + v_2^2 + ...)$	Straight-line distance	sqrt(9+16) = **5**
L-infinity (Max)	$\|\|v\|\|_{inf} = \max(\|v_i\|)$	Largest component	max(3,4) = **4**

9.2 Dot Product (Inner Product) Revisited

The dot product has a geometric interpretation:

```
a . b = ||a|| * ||b|| * cos(theta)
```

where theta is the angle between the two vectors. This gives us:

• If a . b = 0, the vectors are **orthogonal** (perpendicular, 90 degrees apart)

• If a . b > 0, the vectors point in **similar** directions (angle < 90 degrees)

• If a . b < 0, the vectors point in **opposite** directions (angle > 90 degrees)

9.3 Cosine Similarity

Cosine similarity normalizes the dot product to measure only the **angle** between vectors, ignoring their magnitudes:

```
cos_sim(a, b) = (a . b) / (||a|| * ||b||)
```

Result is always between -1 (opposite) and 1 (identical direction). This is one of the most used metrics in ML!

9.4 Orthogonality and Projection

The **projection** of vector a onto vector b gives you the component of a in the direction of b:

```
proj_b(a) = ((a . b) / (b . b)) * b
```

This is fundamental to least-squares regression and many optimization algorithms.

■ Machine Learning Connection

Norms and distances are everywhere in ML:

- **Loss functions:** L2 norm = MSE loss, L1 norm = MAE loss
- **Regularization:** L1 (Lasso) makes weights sparse; L2 (Ridge) makes weights small
- **KNN algorithm:** Uses Euclidean distance to find nearest neighbors
- **Cosine similarity:** Used in recommendation systems and NLP to compare documents/embeddings
- **Unit vectors:** Normalizing vectors (v / ||v||) is used in many algorithms (softmax, normalization layers)

```
Python: norm = np.linalg.norm(v) # L2 norm
cos_sim = np.dot(a, b) / (np.linalg.norm(a) * np.linalg.norm(b))
```

✍ Practice Problems

P9.1 Find the L2 norm of [3, 4]

P9.2 Find the L1 norm of [3, -4, 2]

P9.3 Are [1, 2] and [-2, 1] orthogonal? (Check their dot product)

P9.4 Find the cosine similarity between [1, 0] and [1, 1]

P9.5 Normalize [3, 4] to a unit vector (length 1)

■ Answers

A9.1 sqrt(9 + 16) = sqrt(25) = 5

A9.2 |3| + |-4| + |2| = 3 + 4 + 2 = 9

A9.3 [1,2] . [-2,1] = 1*(-2) + 2*1 = -2 + 2 = 0. Yes, orthogonal!

A9.4 dot = 1*1 + 0*1 = 1. ||a|| = 1, ||b|| = sqrt(2). cos_sim = 1/sqrt(2) = 0.707

A9.5 ||v|| = 5. Unit vector = [3/5, 4/5] = [0.6, 0.8]

Chapter 10

Putting It All Together: ML Applications

ADVANCED

Now let's see how all these concepts come together in real machine learning applications.

10.1 Linear Regression: The Complete Picture

Linear regression finds weights w that minimize the error between predictions and actual values.

Setup: You have m data points with n features. Your data matrix X is (m x n), labels y is (m x 1), and weights w is (n x 1).

$$\text{Predictions: } \hat{y} = X * w$$

$$\text{Error: } e = y - \hat{y}$$

$$\text{Loss: } L = ||e||_2^2 = (y - Xw)^T(y - Xw)$$

Solution (Normal Equation):

$$w = (X^TX)^{-1} X^Ty$$

Linear algebra concepts used: Matrix multiplication, transpose, inverse, systems of equations.

10.2 PCA: Dimensionality Reduction

Principal Component Analysis reduces the number of features while keeping the most important information.

Step	Action	Linear Algebra Used
1	Center the data (subtract mean)	Vector subtraction
2	Compute covariance matrix C = (1/m) X^TX	Matrix multiplication, transpose
3	Find eigenvalues and eigenvectors of C	Eigendecomposition
4	Sort eigenvectors by eigenvalue (largest first)	Eigenvalues = variance captured

5	Keep top k eigenvectors as new basis	Basis, span, dimension
6	Project data onto new basis: X_new = X * V_k	Matrix multiply, projection

10.3 Neural Networks: Linear Algebra in Action

A neural network is essentially a sequence of matrix operations with non-linear activations in between.

Single layer forward pass:

```
z = W * x + b (linear transformation)
a = activation(z) (non-linear function)
```

Where W is the weight matrix, x is the input vector, b is the bias vector.

Full network:

```
Input (784-dim) --> W1 (256x784) --> ReLU --> W2 (128x256) --> ReLU --> W3 (10x128) --> Softmax --> Output (10-dim)
```

Linear algebra concepts in backpropagation:

• Gradient computation uses the **transpose** of weight matrices

• Weight updates use **scalar multiplication** (learning rate) and **vector subtraction**

• Batch processing uses **matrix multiplication** to process many samples at once

10.4 Complete Concept Map

ML Concept	Linear Algebra Foundation
Data representation	Vectors and matrices
Linear regression	Matrix inverse, systems of equations, projection
Neural network layers	Matrix multiplication, linear transformations
PCA / Dimensionality reduction	Eigenvalues, eigenvectors, projection, basis change
Gradient descent	Vector operations, norms, scalar multiplication
Regularization (L1/L2)	Vector norms (L1, L2)

Similarity / distance metrics	Dot product, cosine similarity, norms
SVD / Matrix factorization	Eigendecomposition, orthogonality
Attention (Transformers)	Matrix multiplication, dot product, softmax
Convolutions (CNNs)	Specialized linear transformations

10.5 NumPy Quick Reference

Operation	Python Code
Create vector	`v = np.array([1, 2, 3])`
Create matrix	`A = np.array([[1,2],[3,4]])`
Matrix multiply	`C = A @ B # or np.dot(A, B)`
Transpose	`A.T # or np.transpose(A)`
Inverse	`np.linalg.inv(A)`
Determinant	`np.linalg.det(A)`
Eigenvalues	`vals, vecs = np.linalg.eig(A)`
Solve Ax=b	`x = np.linalg.solve(A, b)`
L2 norm	`np.linalg.norm(v)`
Dot product	`np.dot(a, b) # or a @ b`
Identity matrix	`np.eye(n) # n x n identity`
SVD	`U, S, Vt = np.linalg.svd(A)`

✍ Practice Problems

P10.1 You have 100 data points with 5 features. What size is your data matrix X?

P10.2 In linear regression with the matrix above, what size is the weight vector w?

P10.3 What is X^TX for the above? What size is it?

P10.4 A neural network layer has input size 256 and output size 128. What size is the weight matrix?

P10.5 After PCA, you reduce 50 features to 10 principal components. How much smaller is your data?

■ Answers

A10.1 100 x 5 (100 rows, 5 columns)

A10.2 5 x 1 (one weight per feature)

A10.3 (5x100) * (100x5) = 5 x 5 matrix (the Gram matrix)

A10.4 128 x 256 (output_dim x input_dim)

A10.5 From 50 columns to 10 columns = 5x reduction in feature dimensionality

Your Learning Path Forward

Congratulations on working through this guide! Here's how to continue building your linear algebra skills for machine learning:

Stage	Focus	Resources
1. Practice	Redo all practice problems without looking at answers. Implement each operation in NumPy.	This guide + Python
2. Deepen	3Blue1Brown "Essence of Linear Algebra" (YouTube). Best visual explanations available.	YouTube (free)
3. Apply	Implement linear regression, PCA, and a simple neural network from scratch using only NumPy.	Python + NumPy
4. Expand	Learn SVD (Singular Value Decomposition), matrix factorization, and sparse matrices.	Gilbert Strang textbook
5. Master	Study the math behind Transformers, attention mechanisms, and optimization algorithms.	Research papers

■ **Key Insight**

Remember: Linear algebra is a skill, not just knowledge. You build it by doing, not just reading. Write code, solve problems by hand, and most importantly - when you encounter a new ML concept, ask yourself: "What linear algebra is happening here?"

Every matrix operation you understand makes the next ML paper or algorithm easier to grasp. Keep going!

Linear Algebra + Machine Learning = Your Superpower