Mathematical Foundations for Computer Vision and AI (Draft)

Zenjie Li

8th August 2025

Mathematical Foundations for Computer Vision and AI

Preface

The journey of learning and teaching computer vision and artificial intelligence (AI) has revealed a recurring challenge: the mathematical foundations, so critical to these fields, are often rusty or fragmented for students and practitioners alike. As a postgraduate supervisor and engineer, I have observed that many resources either skip crucial mathematical steps, assuming familiarity, or present concepts in ways that feel disconnected from practical applications. Textbooks, while valuable, can be dauntingly theoretical, scattered in scope, or inconsistent in notation, leaving learners to piece together the puzzle themselves. This book was born from a desire to address these gaps by consolidating the essential mathematical preliminaries for computer vision, machine learning, and related AI disciplines into a single, accessible resource.

My motivation stems from both personal experience and the needs of my students. When I first delved into computer vision and AI, I found myself revisiting concepts from linear algebra, calculus, probability, and more, often struggling to connect them to the algorithms and techniques I was learning. Later, as I taught these subjects, I noticed similar struggles among my students. Many engineering-focused courses prioritize practical implementation over theoretical rigor due to time constraints, which is understandable but leaves a gap for those who crave deeper understanding. This book aims to bridge that gap by not only presenting the "what" and "how" of mathematical concepts but also the "why" and "how we got here". By including historical context, I hope to illuminate the human journey of discovery behind the elegant theorems we rely on today—much like the iterative, often messy process of developing modern technologies.

The scope of this book reflects my exprience in computer vision and language models, with an intentional stretch to encompass machine learning and other AI fields where these mathematical foundations apply. From vector geometry to linear algebra, probability, and numerical methods, each chapter is designed to provide a clear, concise refresher while connecting concepts to their practical applications in AI. I have also included proofs and derivations for key results, many provided in the appendices, not to overwhelm but to satisfy the curiosity of those who, like me, find clarity in understanding the mechanisms beneath the formulas. More advanced topics such as numerical analysis and convex optimization are omitted; this book instead focuses on building the essential intuition and toolkit needed before diving into those areas.

This work is a living document, and I welcome feedback from readers to refine and expand future editions, ensuring it remains relevant and comprehensive.

This book is for you if you have completed foundational math courses—such as calculus, linear algebra, and probability—and are preparing to dive into signal processing, computer vision, or machine learning. It is also for engineers and practitioners seeking a quick reference to refresh concepts or explore how different mathematical pieces fit together in AI applications.

This book is not intended for those without a basic mathematical background, as it assumes familiarity with undergraduate-level math. Nor is it designed for researchers focused on highly theoretical or advanced topics, as its emphasis is on practical foundations rather than cutting-edge mathematical theory.

I hope this book serves as a guide and inspiration, helping you navigate the mathematical landscape of computer vision and AI with confidence and curiosity.

Zenjie Li
July, 2025

Contents

Chapter 1

Linear Algebra Part I - Foundations

In this chapter, we explore the core concepts that underpin all linear methods: vectors, matrices, coordinate systems, and transformations. More advanced ideas like eigenvalues, orthogonality, and matrix decomposition are reserved for the next chapter.

1.1 Matrices

Matrices are fundamental tools in mathematics, with applications ranging from solving systems of linear equations to representing abstract algebraic structures.

The concept of matrices can be traced back to ancient civilizations that sought solutions to systems of linear equations. For instance, the *Nine Chapters on the Mathematical Art*, a Chinese text from around 200 BCE, contains systematic methods for solving linear systems. These methods involved manipulating coefficients in ways that are equivalent to modern row operations on matrices. These early developments laid the groundwork for organizing coefficients into arrays, which would later evolve into the concept of matrices.

The formal definition of matrices as rectangular arrays of numbers was introduced in the mid-19th century by the British mathematician **Arthur Cayley**.

Matrices were later recognized as elements of broader algebraic structures, enabling mathematicians to study their key properties and apply them across fields like engineering and computer science.

1.1.1 Example: From Linear Systems to Matrices

To illustrate how matrices represent systems of linear equations, consider the following simple system of two equations with two variables:

$$\begin{aligned} 3x + 2y &= 7, \\ x - 4y &= -5. \end{aligned}$$

This system can be written in matrix form as:

$$\begin{bmatrix} 3 & 2 \\ 1 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 7 \\ -5 \end{bmatrix}.$$

Here:

- The 2×2 matrix $\begin{bmatrix} 3 & 2 \\ 1 & -4 \end{bmatrix}$ contains the coefficients of x and y from the two equations.
- The column vector $\begin{bmatrix} x \\ y \end{bmatrix}$ represents the variables.
- The column vector $\begin{bmatrix} 7 \\ -5 \end{bmatrix}$ represents the constants on the right-hand side of the equations.

This compact representation allows us to manipulate the entire system using matrix operations, as we will see in the following sections.

1.1.2 Preliminary Knowledge – Summation Operations

In mathematics, particularly in linear algebra and discrete mathematics, summations often arise in expressions involving multiple indices. Let x_{ij} denote a scalar quantity associated with row i and column j of a matrix. A double sum has the general form:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij},$$

which means we first fix an index i and sum over all values of j, then sum the resulting values over all i. This process is equivalent to summing over all pairs (i, j) such that $1 \leq i \leq m$ and $1 \leq j \leq n$.

When the range of summation is clear from context, for simplicity, the double sum can be written as:

$$\sum_{i}\sum_{j} x_{ij}.$$

Switching the Order of Summation

A key property of finite sums is that the order of summation can be interchanged without changing the result:

$$\sum_{i}\sum_{j} x_{ij} = \sum_{j}\sum_{i} x_{ij}.$$

This follows directly from the commutative and associative properties of addition.

We can extend this rule to cases where the terms involve products of scalars. For example, suppose $x_{ij} = a_i b_{ij} c_j$, where a_i, b_{ij}, and c_j are scalar quantities (such as entries of matrices). Then we have:

$$\sum_{i}\sum_{j} a_i b_{ij} c_j = \sum_{j}\sum_{i} a_i b_{ij} c_j.$$

Even though the terms are products, they remain scalar values, and the entire expression is still a finite sum of such values. Therefore, the order of summation can be safely interchanged. This principle is crucial in many areas of linear algebra.

1.1.3 Basic Matrix Operations

Matrix Addition

Matrix addition is defined as the element-wise sum of two matrices of the same dimensions. For two matrices $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$, their sum $\mathbf{C} = \mathbf{A} + \mathbf{B}$ is given by:

$$c_{ij} = a_{ij} + b_{ij}.$$

This operation is intuitive because it corresponds to adding corresponding coefficients of two systems of linear equations.

Matrix addition is:

- *Commutative*: $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$,
- *Associative*: $(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$,
- Each matrix has an *additive inverse*: $-\mathbf{A}$ such that $\mathbf{A} + (-\mathbf{A}) = \mathbf{0}$.

These properties follow directly from the analogous properties of scalar addition.

Scalar Multiplication

For a scalar α and a matrix $\mathbf{A} = [a_{ij}]$, the product $\alpha\mathbf{A}$ is a matrix $\mathbf{C} = [\alpha a_{ij}]$. That is:

$$c_{ij} = \alpha a_{ij}.$$

Scalar multiplication satisfies:

- *Distributive over matrix addition*: $\alpha(\mathbf{A} + \mathbf{B}) = \alpha\mathbf{A} + \alpha\mathbf{B}$,
- *Distributive over scalar addition*: $(\alpha + \beta)\mathbf{A} = \alpha\mathbf{A} + \beta\mathbf{A}$,
- *Associative with scalars*: $\alpha(\beta\mathbf{A}) = (\alpha\beta)\mathbf{A}$.

Again, these properties are inherited from scalar arithmetic.

Matrix Multiplication

Matrix multiplication is defined as follows: for two matrices $\mathbf{A} = [a_{ik}]$ (of size $m \times n$) and $\mathbf{B} = [b_{kj}]$ (of size $n \times p$), their product $\mathbf{C} = \mathbf{AB}$ is a matrix of size $m \times p$, where each entry c_{ij} is computed as:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Matrix multiplication was defined to reflect the composition of linear transformations or systems of linear equations. Consider two systems:

$$\begin{aligned} &\text{System 1: } \mathbf{Ay} = \mathbf{b}, \quad \text{where } \mathbf{A} \text{ is an } m \times n \text{ matrix,} \\ &\text{System 2: } \mathbf{Bx} = \mathbf{y}, \quad \text{where } \mathbf{B} \text{ is an } n \times p \text{ matrix.} \end{aligned}$$

Substituting $\mathbf{y} = \mathbf{Bx}$ into the first equation gives:

$$\mathbf{A}(\mathbf{Bx}) = \mathbf{b},$$

which motivates defining the matrix product $\mathbf{C} = \mathbf{AB}$ so that $\mathbf{Cy} = \mathbf{b}$. The entries of $\mathbf{C}$ must satisfy:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Properties of matrix multiplication from its definition:

- Distributive over addition: $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$,
- Distributive over addition: $(\mathbf{A} + \mathbf{B})\mathbf{C} = \mathbf{AC} + \mathbf{BC}$,
- Scalar multiplication Distributivity: $\alpha(\mathbf{AB}) = (\alpha\mathbf{A})\mathbf{B} = \mathbf{A}(\alpha\mathbf{B})$,
- Associativity of Matrix Multiplication: $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$.

Since other properties are obivous, below we only prove the Associativity of Matrix Multiplication.

Associativity of Matrix Multiplication

The key property we now prove is:

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC}).$$

Let:

- $\mathbf{A}$ be an $m \times n$ matrix with entries a_{ik},
- $\mathbf{B}$ be an $n \times p$ matrix with entries b_{kl},
- $\mathbf{C}$ be a $p \times q$ matrix with entries c_{lj}.

We compute both sides of the equality using index notation.

Left-hand side:

$$((\mathbf{AB})\mathbf{C})_{ij} = \sum_l (\mathbf{AB})_{il} c_{lj} = \sum_l \left(\sum_k a_{ik} b_{kl} \right) c_{lj} = \sum_l \sum_k a_{ik} b_{kl} c_{lj}.$$

Note that $\sum_k a_{ik} b_{kl}$ is a scalar when l is fixed.

Switching the order of summation (as shown in 1.1.2), the left-hand side becomes:

$$\sum_k \sum_l a_{ik} b_{kl} c_{lj}.$$

Right-hand side:

$$(\mathbf{A}(\mathbf{BC}))_{ij} = \sum_k a_{ik} (\mathbf{BC})_{kj} = \sum_k a_{ik} \left(\sum_l b_{kl} c_{lj} \right) = \sum_k \sum_l a_{ik} b_{kl} c_{lj}.$$

Similarly, note that $\sum_l b_{kl} c_{lj}$ is a scalar when k is fixed.

Since both expressions reduce to the same value, we have proved the associativity of matrix multiplication.

Block Representation of Matrices

Matrices can often be viewed as being composed of smaller matrices, or **blocks**, which can simplify operations like addition and multiplication.

For example, consider an $m \times n$ matrix $\mathbf{A}$. We can represent it in terms of its row vectors:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1^\top \\ \mathbf{A}_2^\top \\ \vdots \\ \mathbf{A}_m^\top \end{bmatrix},$$

where each $\mathbf{A}_i^\top$ is the i-th row of $\mathbf{A}$, and thus a $1 \times n$ row vector.

Similarly, an $n \times p$ matrix $\mathbf{B}$ can be represented in terms of its column vectors:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_p \end{bmatrix},$$

where each $\mathbf{B}_j$ is the j-th column of $\mathbf{B}$, and thus an $n \times 1$ column vector.

The entry c_{ij} of $\mathbf{C} = \mathbf{AB}$ is simply:

$$c_{ij} = \mathbf{A}_i^\top \mathbf{B}_j.$$

This aligns with the standard definition of matrix multiplication:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj},$$

where $\mathbf{A}_i^\top = [a_{i1}, a_{i2}, \ldots, a_{in}]$ and $\mathbf{B}_j = [b_{1j}, b_{2j}, \ldots, b_{nj}]^\top$.

Thus, the block-based interpretation provides both a conceptual and computational framework for understanding matrix multiplication — and its correctness follows directly from the summation properties we previously established.

1.1.4 Matrix Rank: Understanding the Essence of a Matrix

Having explored the basics of matrices and their operations, we now turn our attention to a fundamental concept that reveals crucial information about a matrix: its **rank**. The rank of a matrix provides insight into the "size" of the space spanned by its rows or columns and plays a key role in solving systems of linear equations.

Definition of Matrix Rank

The **rank** of a matrix $\mathbf{A}$ is defined as the maximum number of linearly independent rows (or equivalently, columns) in the matrix. Linear independence means that no row (or column) can be expressed as a linear combination of the others.

For example, consider the following 3×3 matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 7 & 9 \end{bmatrix}.$$

Upon closer inspection, we notice that the third row is the sum of the first two rows:

$$\text{Row } 3 = \text{Row } 1 + \text{Row } 2.$$

Thus, the rows of $\mathbf{A}$ are not all linearly independent. In fact, the rank of this matrix is 2, as there are only two linearly independent rows.

Geometric Interpretation of Rank

The rank of a matrix has a geometric interpretation. For an $m \times n$ matrix $\mathbf{A}$, the rank represents the dimension of the vector space spanned by its rows or columns. This space is often referred to as the **row space** or **column space** of the matrix.

For instance, if $\mathbf{A}$ has rank r, then:

- The row space of $\mathbf{A}$ is a subspace of $\mathbb{R}^n$ with dimension r.
- The column space of $\mathbf{A}$ is a subspace of $\mathbb{R}^m$ with dimension r.

This geometric perspective helps us understand why the rank is so important when solving systems of linear equations.

Connection to Systems of Linear Equations

The rank of a matrix is closely tied to the solvability of a system of linear equations. Recall the general form of a system:

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{x}$ is the vector of unknowns, and $\mathbf{b}$ is the vector of constants.

To determine whether this system has a solution, we examine the ranks of two matrices:

1. The **coefficient matrix** $\mathbf{A}$.
2. The **augmented matrix** $[\mathbf{A}|\mathbf{b}]$, which is formed by appending the column vector $\mathbf{b}$ to $\mathbf{A}$.

The system has a solution if and only if:

$$\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A}|\mathbf{b}]).$$

If these ranks are equal, the system is consistent (i.e., it has at least one solution). Otherwise, the system is inconsistent and has no solutions.

Row Reduction is a systematic method of simplifying a matrix using elementary row operations. These operations mirror the algebraic manipulations used when solving systems of equations:

- Swapping two equations (rows),
- Multiplying an equation (row) by a nonzero constant,
- Adding a multiple of one equation (row) to another.

By applying these operations, we can transform the matrix into a simpler form (like reduced row echelon form), making it easier to analyze the system's structure and determine its solution set.

Example: Consider the following system of equations:

$$\begin{aligned} x + y &= 2 \\ 2x + 2y &= 4 \\ x + y &= 3 \end{aligned}$$

Here, the coefficient matrix $\mathbf{A}$ and augmented matrix $[\mathbf{A}|\mathbf{b}]$ are:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix}, \quad [\mathbf{A}|\mathbf{b}] = \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 2 & 2 & 4 \\ 1 & 1 & 3 \end{array}\right].$$

Row-reducing $\mathbf{A}$, we find:

$$\text{rref}(\mathbf{A}) = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

so $\text{rank}(\mathbf{A}) = 1$.

Now row-reduce the augmented matrix:

$$\text{rref}([\mathbf{A}|\mathbf{b}]) = \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{array}\right].$$

This final matrix corresponds to the system:

$$\begin{aligned} x + y &= 2 \\ 0x + 0y &= 0 \\ 0x + 0y &= 1, \end{aligned}$$

which is clearly inconsistent due to the last equation $0 = 1$. Hence, $\text{rank}([\mathbf{A}|\mathbf{b}]) = 2$, which is greater than $\text{rank}(\mathbf{A}) = 1$, confirming that the system has no solution.

This illustrates that when the ranks differ, the system is inconsistent and has no solution.

1.2 Vector Geometry

The origins of vector geometry can be traced back to **René Descartes**, who pioneered analytic geometry in the 17th century by unifying algebra and geometry through the use of coordinate systems. Vector geometry has become a cornerstone of mathematics, physics, and engineering, with the *dot product*, *cross product*, and *scalar triple product* standing out as its most essential tools.

1.2.1 Definition of a Vector

In the 17th century, the study of physical quantities such as displacement, velocity, force, and electric fields revealed the need for a mathematical construct that could represent both magnitude and direction. This led to the formalization of **vectors**.

In mathematics and physics, a **vector** is defined as a quantity possessing both magnitude and direction. Graphically, vectors are depicted as arrows, where the length of the arrow represents the magnitude, and the orientation of the arrowhead indicates the direction.

1.2.2 Vector Addition: The Parallelogram Rule

Having defined vectors, the next challenge was determining how to combine two such quantities. This led to the development of the **parallelogram rule** for vector addition.

To add two vectors $\mathbf{a}$ and $\mathbf{b}$, place the tail of one vector at the tip of the other (or vice versa). The sum $\mathbf{a} + \mathbf{b}$ is then represented by the diagonal of the parallelogram formed by $\mathbf{a}$ and $\mathbf{b}$, starting from their common tail. See Figure 1.1 for an illustration.

The parallelogram rule was first experimentally demonstrated by **Simon Stevinus** (1548–1620) in his studies on static equilibrium. He showed that when two forces act on a point object, the resultant

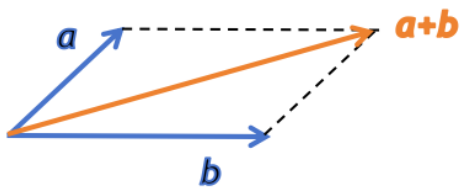

Fig. 1.1: Illustration of vector addition using the parallelogram rule.

force aligns with the diagonal of the parallelogram formed by the two force vectors. This principle was later formalized in Isaac Newton's *Principia Mathematica* (1687) and has since been rigorously validated across various domains of physics, including classical mechanics, particle physics, and electromagnetism.

From this geometric interpretation, it follows that vector addition satisfies the following properties:

- **Commutativity**: $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$,
- **Associativity**: $(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c})$.

1.2.3 The Dot Product

The dot product, formally introduced in the late 19th century by **Josiah Willard Gibbs** and **Oliver Heaviside**, serves as a fundamental tool in vector geometry. It quantifies the degree of alignment between two vectors and arises naturally from both geometric intuition and physical applications.

Vector Projection

To study the relationship between two vectors, it is often necessary to determine the component of one vector along the direction of another. This concept is formalized through **vector projection**. The projection of a vector $\mathbf{a}$ onto another vector $\mathbf{u}$ is defined as:

$$\mathbf{a_u} = |\mathbf{a}| \cos \alpha,$$

where α is the angle between $\mathbf{a}$ and $\mathbf{u}$. Geometrically, this represents the magnitude of $\mathbf{a}$ along the direction of $\mathbf{u}$.

From the principles of vector addition and projection, it follows that projection is distributive over vector sums:

$$(\mathbf{a} + \mathbf{b})_\mathbf{u} = \mathbf{a_u} + \mathbf{b_u}. \tag{1.1}$$

Definition of the Dot Product

Consider a force $\mathbf{F}$ acting on an object that undergoes a displacement $\mathbf{s}$. The work done W is given by:

$$W = |\mathbf{F}||\mathbf{s}| \cos \theta,$$

where θ is the angle between $\mathbf{F}$ and $\mathbf{s}$. This expression motivates the definition of the dot product:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos \theta. \tag{1.2}$$

Using this definition, the work done can be expressed compactly as:

$$W = \mathbf{F} \cdot \mathbf{s}.$$

Alternatively, the dot product can be expressed in terms of projections:

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}|\mathbf{b}_\mathbf{a} = |\mathbf{b}|\mathbf{a}_\mathbf{b},$$

where $\mathbf{b}_\mathbf{a}$ and $\mathbf{a}_\mathbf{b}$ represent the projections of $\mathbf{b}$ onto $\mathbf{a}$ and $\mathbf{a}$ onto $\mathbf{b}$, respectively.

Properties of the Dot Product

The dot product exhibits several key properties:

- **Commutativity**: $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$.
- **Distributivity**: $\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$.
- **Scalar Multiplication**: For any scalar k, the dot product satisfies:

$$(k\mathbf{a}) \cdot \mathbf{b} = k(\mathbf{a} \cdot \mathbf{b}) = \mathbf{a} \cdot (k\mathbf{b}).$$

The commutativity and scalar multiplications are obvious from the dot product definition. The distributivity can be rigorously derived using the distributive nature of vector projections (Equation 1.1). Geometrically, when projecting $\mathbf{b} + \mathbf{c}$ onto $\mathbf{a}$, the resulting projection is equivalent to the sum of the individual projections of $\mathbf{b}$ and $\mathbf{c}$ onto $\mathbf{a}$.

1.2.4 The Cross Product

The **cross product**, like the dot product, emerged as a fundamental tool in vector geometry to formalize spatial relationships. However, its development and applications differ significantly from those of the dot product. While the dot product quantifies alignment between vectors, the cross product captures perpendicularity and orientation in three-dimensional space, making it indispensable in fields such as mechanics, electromagnetism, and computer graphics.

Motivation for the Cross Product

The cross product was introduced in the late 19th century by **Josiah Willard Gibbs** and **Oliver Heaviside** as part of their pioneering work in reformulating vector analysis. Its invention was driven by practical problems in physics and engineering where the interaction between two vectors produces a third vector orthogonal to both. A classic example is torque in mechanics: the rotational effect of a force depends on both the position vector $\mathbf{r}$ and the applied force $\mathbf{F}$. The resulting torque vector $\boldsymbol{\tau}$ lies along the axis of rotation (perpendicular to the plane containing $\mathbf{r}$ and $\mathbf{F}$), and its magnitude reflects the strength of the rotational effect (Figure 1.2).

Such applications highlight the need for a mathematical operation that generates a vector orthogonal to two given vectors while encoding information about their relative orientation and magnitudes.

Definition of the Cross Product

Given two vectors $\mathbf{a}$ and $\mathbf{b}$ in three-dimensional space, their cross product $\mathbf{a} \times \mathbf{b}$ is defined as:

$$\mathbf{a} \times \mathbf{b} = |\mathbf{a}||\mathbf{b}| \sin\theta\, \hat{\mathbf{n}},$$

where:

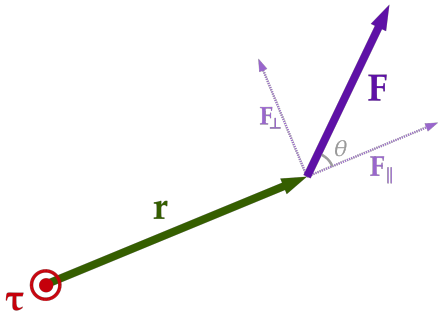

Fig. 1.2: Illustration of torque. Attribution: StradivariusTV. License: CC BY-SA 3.0.

- $|\mathbf{a}|$ and $|\mathbf{b}|$ are the magnitudes of $\mathbf{a}$ and $\mathbf{b}$,
- θ is the angle between $\mathbf{a}$ and $\mathbf{b}$,
- $\hat{\mathbf{n}}$ is the unit vector perpendicular to the plane containing $\mathbf{a}$ and $\mathbf{b}$, determined by the right-hand rule.

The right-hand rule specifies the direction of $\hat{\mathbf{n}}$: if the fingers of the right hand curl from $\mathbf{a}$ toward $\mathbf{b}$ through the smaller angle between them, the thumb points in the direction of $\hat{\mathbf{n}}$.

The cross product exhibits several key properties:

- **Anticommutativity**: $\mathbf{a} \times \mathbf{b} = -(\mathbf{b} \times \mathbf{a})$.
- **Distributivity**: $\mathbf{a} \times (\mathbf{b} + \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) + (\mathbf{a} \times \mathbf{c})$.
- **Scalar Multiplication**: For any scalar k, $(k\mathbf{a}) \times \mathbf{b} = k(\mathbf{a} \times \mathbf{b}) = \mathbf{a} \times (k\mathbf{b})$.
- **Orthogonality**: $\mathbf{a} \times \mathbf{b}$ is orthogonal to both $\mathbf{a}$ and $\mathbf{b}$.
- **Magnitude**: The magnitude of $\mathbf{a} \times \mathbf{b}$ equals the area of the parallelogram formed by $\mathbf{a}$ and $\mathbf{b}$.

We will derive the distributive property in Section 1.2.5. Other properties follow directly from the definition.

1.2.5 The Scalar Triple Product

The **scalar triple product** combines the dot and cross products to address problems where both alignment and perpendicularity of vectors are relevant. This hybrid construction arises naturally in diverse fields, including fluid dynamics, where vorticity involves the curl of a velocity field (a cross product) and its projection onto a direction (a dot product).

Definition

The scalar triple product is defined as:

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}),$$

and represents the volume of the parallelepiped formed by the vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ (Figure 1.3). Geometrically, this expression measures how much the three vectors "span" three-dimensional space.

In computer graphics, the scalar triple product is used to compute surface normals and lighting effects, demonstrating its versatility in practical applications.

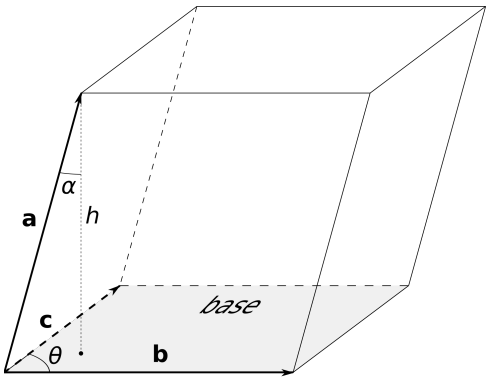

Fig. 1.3: Illustration of the parallelepiped formed by **a**, **b**, and **c**. Attribution: Jitse Niesen. License: Public Domain.

Properties of the Scalar Triple Product

The scalar triple product exhibits several key properties, all of which can be intuitively understood from its geometric interpretation as the volume of a parallelepiped:

- **Cyclic Permutation**:
$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \mathbf{b} \cdot (\mathbf{c} \times \mathbf{a}) = \mathbf{c} \cdot (\mathbf{a} \times \mathbf{b}).$$
- **Zero Volume**: If **a**, **b**, and **c** are coplanar (i.e., they lie in the same plane), the scalar triple product evaluates to zero.

Derivation of Cross Product Distributivity

To derive the distributive property of the cross product, consider an arbitrary vector **k**. Using the scalar triple product, we compute:

$$\begin{aligned}\mathbf{k} \cdot [(\mathbf{a} + \mathbf{b}) \times \mathbf{c}] &= (\mathbf{a} + \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{k}) \\ &= \mathbf{a} \cdot (\mathbf{c} \times \mathbf{k}) + \mathbf{b} \cdot (\mathbf{c} \times \mathbf{k}) \\ &= \mathbf{k} \cdot (\mathbf{a} \times \mathbf{c}) + \mathbf{k} \cdot (\mathbf{b} \times \mathbf{c}) \\ &= \mathbf{k} \cdot [(\mathbf{a} \times \mathbf{c}) + (\mathbf{b} \times \mathbf{c})].\end{aligned}$$

Since **k** is arbitrary, the equality holds for all vectors, proving the distributive property:

$$(\mathbf{a} + \mathbf{b}) \times \mathbf{c} = (\mathbf{a} \times \mathbf{c}) + (\mathbf{b} \times \mathbf{c}).$$

1.2.6 Vector Operations in the XYZ Coordinate System

In the Cartesian coordinate system, vectors are often expressed in terms of their components along the x, y, and z axes using the standard unit vectors **i**, **j**, and **k**. These unit vectors are orthogonal to each other and have a magnitude of 1. A vector **a** can be written as:

$$\mathbf{a} = a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k},$$

where a_x, a_y, and a_z are the scalar components of **a** along the x, y, and z axes, respectively.

Dot Product in the XYZ Coordinate System

The dot product of two vectors $\mathbf{a}$ and $\mathbf{b}$ in the Cartesian coordinate system is computed as:

$$\mathbf{a} \cdot \mathbf{b} = (a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}) \cdot (b_x\mathbf{i} + b_y\mathbf{j} + b_z\mathbf{k}).$$

Using the distributive property of the dot product and the fact that $\mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j} = \mathbf{k} \cdot \mathbf{k} = 1$ (since they are unit vectors) and $\mathbf{i} \cdot \mathbf{j} = \mathbf{j} \cdot \mathbf{k} = \mathbf{k} \cdot \mathbf{i} = 0$ (since they are orthogonal), the dot product simplifies to:

$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z. \tag{1.3}$$

Cross Product in the XYZ Coordinate System

The cross product of two vectors $\mathbf{a}$ and $\mathbf{b}$ in the Cartesian coordinate system is computed as:

$$\mathbf{a} \times \mathbf{b} = (a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}) \times (b_x\mathbf{i} + b_y\mathbf{j} + b_z\mathbf{k}).$$

Using the distributive property of the cross product and the relationships between the unit vectors ($\mathbf{i} \times \mathbf{j} = \mathbf{k}$, $\mathbf{j} \times \mathbf{k} = \mathbf{i}$, $\mathbf{k} \times \mathbf{i} = \mathbf{j}$, and $\mathbf{i} \times \mathbf{i} = \mathbf{j} \times \mathbf{j} = \mathbf{k} \times \mathbf{k} = \mathbf{0}$), the cross product can be expressed as:

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}.$$

Expanding this determinant yields:

$$\mathbf{a} \times \mathbf{b} = (a_y b_z - a_z b_y)\mathbf{i} - (a_x b_z - a_z b_x)\mathbf{j} + (a_x b_y - a_y b_x)\mathbf{k}.$$

Scalar Triple Product in the XYZ Coordinate System

The scalar triple product $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$ in the Cartesian coordinate system can be computed by first finding the cross product $\mathbf{b} \times \mathbf{c}$ and then taking the dot product of $\mathbf{a}$ with the result. Let:

$$\mathbf{b} = b_x\mathbf{i} + b_y\mathbf{j} + b_z\mathbf{k}, \quad \mathbf{c} = c_x\mathbf{i} + c_y\mathbf{j} + c_z\mathbf{k}.$$

The cross product $\mathbf{b} \times \mathbf{c}$ is given by:

$$\mathbf{b} \times \mathbf{c} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{vmatrix}.$$

Expanding this determinant gives:

$$\mathbf{b} \times \mathbf{c} = (b_y c_z - b_z c_y)\mathbf{i} - (b_x c_z - b_z c_x)\mathbf{j} + (b_x c_y - b_y c_x)\mathbf{k}.$$

Now, taking the dot product of $\mathbf{a} = a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k}$ with $\mathbf{b} \times \mathbf{c}$, we get:

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = a_x(b_y c_z - b_z c_y) - a_y(b_x c_z - b_z c_x) + a_z(b_x c_y - b_y c_x).$$

This expression can also be written compactly as the determinant of a 3×3 matrix:

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \begin{vmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{vmatrix}.$$

1.3 Vector Transformations Using Matrices

In this section, we delve into the concept of using matrices to transform vectors and explore their applications in geometry and physics.

1.3.1 Matrix-Vector Multiplication as a Transformation

The multiplication of a matrix $\mathbf{A}$ with a vector $\mathbf{v}$ results in a new vector $\mathbf{w} = \mathbf{A}\mathbf{v}$. This operation can be interpreted geometrically as a transformation of the vector $\mathbf{v}$ into a new position or orientation in space. The matrix $\mathbf{A}$ encodes the rules governing this transformation, which may include scaling, rotation, reflection, shearing, or a combination of these effects.

Such transformations are fundamental in various fields, including computer graphics, robotics, and physics, where they are used to model changes in position, orientation, or shape.

1.3.2 Example: 2D Rotation

A classic example of a matrix transformation is the rotation of a vector in two-dimensional space. Consider a vector $\mathbf{v} = \begin{bmatrix} x \\ y \end{bmatrix}$ in the plane. To rotate this vector counterclockwise by an angle θ, we use the rotation matrix:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Multiplying $\mathbf{R}$ by $\mathbf{v}$ yields the rotated vector:

$$\mathbf{w} = \mathbf{R}\mathbf{v} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}.$$

For instance, consider rotating the vector $\mathbf{v} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ (a unit vector along the x-axis) by 90° ($\theta = \pi/2$):

$$\mathbf{R} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

As expected, the result is a unit vector along the y-axis.

1.3.3 3D Cross Product and Skew-Symmetric Matrices

The cross product of two vectors in three-dimensional space is a fundamental operation in vector algebra. Given two vectors $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$, their cross product $\mathbf{u} \times \mathbf{v}$ is defined as:

$$\mathbf{u} \times \mathbf{v} = \begin{bmatrix} u_2v_3 - u_3v_2 \\ u_3v_1 - u_1v_3 \\ u_1v_2 - u_2v_1 \end{bmatrix}.$$

Interestingly, the cross product can also be represented using a skew-symmetric matrix. For a vector $\mathbf{u}$, we define its corresponding skew-symmetric matrix $\mathbf{U}_\times$ as:

$$\mathbf{U}_\times = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}.$$

Using this matrix, the cross product $\mathbf{u} \times \mathbf{v}$ can be expressed as a matrix-vector multiplication:

$$\mathbf{u} \times \mathbf{v} = \mathbf{U}_{\times}\mathbf{v}.$$

This representation is particularly useful in computational settings, as it allows the cross product to be computed efficiently using matrix operations.

1.4 Projective Geometry

Projective geometry extends Euclidean geometry by incorporating points at infinity, allowing parallel lines to intersect and providing a unified framework for studying geometric transformations.

1.4.1 Homogeneous Coordinates

At the heart of projective geometry lies a powerful algebraic tool: **homogeneous coordinates**. These provide an elegant way to represent both finite and infinite elements—such as regular points and directions—in a consistent mathematical structure.

The concept of homogeneous coordinates traces its roots back to the 19th century. In the mid-1800s, **Julius Plücker** developed homogeneous coordinates explicitly to study lines and conic sections in projective geometry. **Felix Klein** further refined these ideas in 1872, advocating for the unification of geometries through transformation groups. In the 20th century, homogeneous coordinates became a practical tool for computer graphics, robotics, and vision.

In 2D projective geometry, a point is often initially expressed in **inhomogeneous coordinates** , such as (x, y), which correspond to standard Cartesian coordinates. However, these coordinates are limited in their ability to represent points at infinity or support projective transformations in a unified way.

To overcome this limitation, people use **homogeneous coordinates**, which extend the representation by introducing an additional scalar parameter $w \neq 0$. In this form, a point is represented as a 3-vector:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix}.$$

The original Cartesian (inhomogeneous) coordinates can be recovered by dividing through by w:

$$\left(\frac{x}{w}, \frac{y}{w}\right).$$

Similarly, in three-dimensional space, a point (x, y, z) in inhomogeneous coordinates becomes a 4-vector $\begin{bmatrix} x & y & z & w \end{bmatrix}^{\top}$, with the corresponding Cartesian coordinates given by $\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$.

Points for which the last coordinate is zero (e.g., $(x, y, 0)$ in 2D) represent directions or **ideal points**—also known as **points at infinity**. These ideal points allow us to model behaviors like parallel lines meeting at a common point in projective space.

1.4.2 Lines in Projective Space

A line in 2D projective geometry can be represented by a 3-vector:

$$\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix},$$

where the equation of the line is $ax + by + cw = 0$. A point $\mathbf{x} = \begin{bmatrix} x & y & w \end{bmatrix}^\top$ lies on the line if:

$$\mathbf{x}^\top \mathbf{l} = 0.$$

The **line at infinity** is defined as:

$$\mathbf{l}_\infty = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

All points at infinity satisfy $w = 0$ and lie on this special line.

Given two points $\mathbf{x}_1$ and $\mathbf{x}_2$ in homogeneous coordinates, the line $\mathbf{l}$ that passes through both points can be computed using the cross product:

$$\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2.$$

This operation is valid because the cross product of $\mathbf{x}_1$ and $\mathbf{x}_2$ yields a third vector l orthogonal to both. That is,

$$x_1 l^\top = 0 \quad x_2 l^\top = 0$$

Thus, the cross product method correctly computes the unique line in projective space that contains both points $\mathbf{x}_1$ and $\mathbf{x}_2$.

Similarly, the intersection of two lines $\mathbf{l}_1$ and $\mathbf{l}_2$ is the point:

$$\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2.$$

This duality between points and lines—where the cross product computes either a line from two points or a point from two lines—is a fundamental property of projective geometry.

1.4.3 Projective Transformations

A **projective transformation**, also called a **homography**, is an invertible mapping from the projective plane to itself that preserves incidence relations: if three points are collinear before the transformation, they remain so afterward.

A planar projective transformation acts on homogeneous 3-vectors and is represented by a non-singular 3×3 matrix H:

$$\mathbf{x}' = H\mathbf{x}.$$

This transformation is linear in homogeneous coordinates and invertible due to the non-singularity of H. The inverse transformation is given by $\mathbf{x} = H^{-1}\mathbf{x}'$.

Transformation of Lines. If a point $\mathbf{x}$ lies on a line $\mathbf{l}$, then the transformed point $\mathbf{x}' = H\mathbf{x}$ lies on the transformed line:

$$\mathbf{l}' = H^{-T}\mathbf{l},$$

where H^{-T} denotes the inverse transpose of H.

Let $\mathbf{x}$ lie on the line $\mathbf{l}$, i.e., $\mathbf{x}^\top\mathbf{l} = 0$. Under the transformation H, the point becomes $\mathbf{x}' = H\mathbf{x}$. We seek the transformed line $\mathbf{l}'$ such that:

$$(\mathbf{x}')^\top\mathbf{l}' = 0.$$

Substituting $\mathbf{x}' = H\mathbf{x}$, we have:

$$(H\mathbf{x})^\top\mathbf{l}' = 0 \quad \Rightarrow \quad \mathbf{x}^\top H^\top\mathbf{l}' = 0.$$

But we know $\mathbf{x}^\top\mathbf{l} = 0$, so this implies:

$$H^\top\mathbf{l}' = \mathbf{l} \quad \Rightarrow \quad \mathbf{l}' = H^{-T}\mathbf{l}.$$

Thus, the transformation rule for lines under a homography is $\mathbf{l}' = H^{-T}\mathbf{l}$, as claimed.

These transformations form the basis for modeling how points and lines behave under changes of view, such as in stereo vision, image rectification, and augmented reality applications.

1.4.4 Application Example: Camera Model

In computer vision, projective geometry plays a central role in modeling the imaging process of a pinhole camera. The *pinhole camera model* is a simplified yet powerful geometric representation of how light rays from a 3D scene are projected onto a 2D image plane. In this idealized model, a single small aperture (the "pinhole") allows only one ray of light from each point in the scene to reach the image plane, resulting in a sharp, undistorted image.

As shown in Figure 1.4, this model assumes that the image plane is located *in front* of the camera center (i.e., the pinhole), which may seem counterintuitive at first. This convention simplifies the mathematics of projection by avoiding the inversion that would occur if the image plane were behind the pinhole, as in a real camera setup.

The line perpendicular to the image plane and passing through the camera center is called the *principal axis* . This axis defines the viewing direction of the camera and intersects the image plane at a specific point known as the *principal point* . In an ideal setup, the principal point corresponds to the center of the image, but in practice, it may be offset due to sensor misalignment or digital cropping.

The distance between the camera center (pinhole) and the image plane is known as the *focal length* , typically denoted by f.

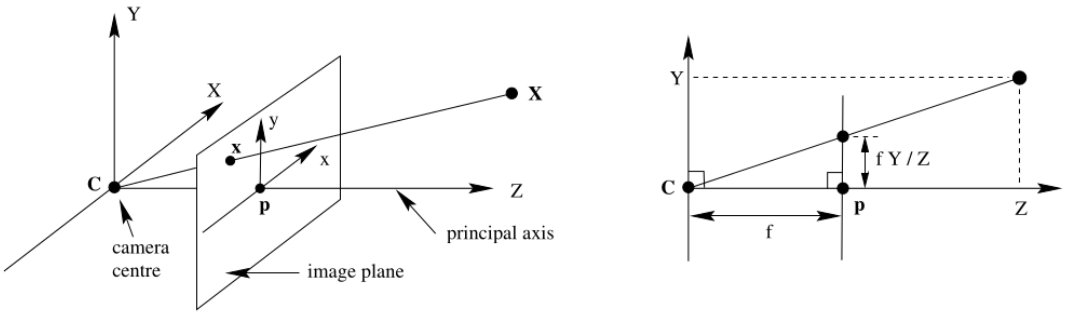

Fig. 1.4: Illustration of the pinhole camera model. *Figure adapted from: Multiple View Geometry in Computer Vision (Second Edition) by Hartley and Zisserman.*

Intrinsic Parameters

The intrinsic camera model maps 3D world points (expressed in camera coordinates) to 2D image coordinates. As shown in Figure 1.4, according to similar triangle relations, the mapping between a spatial point $(X, Y, Z)^\top$ in the camera coordinate system and its projection on the image plane $(x, y)^\top$ is given by:

$$x = f \cdot \frac{X}{Z},$$
$$y = f \cdot \frac{Y}{Z}.$$

In the ideal case, the principal axis passes through the camera center, and the principal point coincides with the center of the image plane. However, in practice, there may be offsets due to sensor misalignment or digital cropping. Let the principal point be (c_x, c_y) in the image coordinate system. Then, the equations become:

$$x = f \cdot \frac{X}{Z} + c_x,$$
$$y = f \cdot \frac{Y}{Z} + c_y.$$

To express these equations in homogeneous coordinates, we write:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

The *camera intrinsic matrix* is written as:

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.4}$$

Using this, the mapping from camera to image coordinates can be expressed as:

$$\mathbf{x}_{\text{image}} = K[I|\mathbf{0}]\mathbf{X}_{\text{cam}}, \tag{1.5}$$

where $\mathbf{x}_{\text{image}} \in \mathbb{R}^3$ is a point in homogeneous image coordinates and $\mathbf{X}_{\text{cam}} \in \mathbb{R}^4$ is a 3D point in homogeneous camera coordinates. Here, $\mathbf{0}$ is a 1×3 zero vector.

Actually, the use of homogeneous coordinates for the 3D point $\mathbf{X}_{\text{cam}}$ is not strictly necessary for this mapping. However, it is useful for consistency, as both $\mathbf{X}_{\text{cam}}$ and $\mathbf{x}_{\text{image}}$ are then represented in homogeneous coordinates. Furthermore, this representation allows for easy combination with extrinsic parameters, as will be shown in the next section.

Extrinsic Parameters

To express the position and orientation of the camera in the world coordinate system, we use extrinsic parameters. If $\mathbf{X}_{\text{world}}$ is a 3D point in world coordinates, its representation in camera coordinates is:

$$\tilde{\mathbf{X}}_{\text{cam}} = R(\tilde{\mathbf{X}}_{\text{world}} - \tilde{\mathbf{t}}),$$

where $R \in \mathbb{R}^{3\times3}$ is a rotation matrix and $\tilde{\mathbf{t}} \in \mathbb{R}^3$ is the translation vector representing the camera center in world coordinates.

In homogeneous coordinates,

$$\mathbf{X}_{\text{cam}} = \begin{bmatrix} \tilde{\mathbf{X}}_{\text{cam}} \\ 1 \end{bmatrix}, \qquad \mathbf{X}_{\text{world}} = \begin{bmatrix} \tilde{\mathbf{X}}_{\text{world}} \\ 1 \end{bmatrix}.$$

We then have:

$$\mathbf{X}_{\text{cam}} = \begin{bmatrix} R & -R\mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_{\text{world}}.$$

Combining intrinsic parameters (Equation 1.5) and extrinsic parameters, the full projection from 3D world coordinates to 2D image coordinates is:

$$\mathbf{x}_{\text{image}} = K[R \mid -R\mathbf{t}] \cdot \mathbf{X}_{\text{world}}, \tag{1.6}$$

which defines the standard **pinhole camera model** in homogeneous coordinates.

It is noted that this formulation has **9 degrees of freedom**: 3 from the intrinsic matrix K, 3 from the rotation R, and 3 from the translation $\mathbf{t}$.

Chapter 2

Linear Algebra Part II - Advanced Topics

In this part, we move beyond basic operations and geometric interpretations to uncover powerful tools that reveal the hidden structure within data and systems.

2.1 Eigenvalues and Eigenvectors

Eigenvalues and **eigenvectors** arise when studying how a square matrix $\mathbf{A}$ acts on vectors. Specifically, an eigenvector $\mathbf{v}$ of a matrix $\mathbf{A}$ is a nonzero vector that satisfies the equation:

$$\mathbf{Av} = \lambda\mathbf{v},$$

where λ (a scalar) is the corresponding eigenvalue. This equation implies that applying the transformation represented by $\mathbf{A}$ to $\mathbf{v}$ results in a scaled version of $\mathbf{v}$, rather than changing its direction.

The study of eigenvalues and eigenvectors can be traced back to the 18th century, with contributions from mathematicians such as **Leonhard Euler**, who studied principal axes of rotation in rigid body dynamics, and **Joseph-Louis Lagrange**, who analyzed quadratic forms. However, the formalization of eigenvalues and eigenvectors as we know them today emerged in the 19th century through the work of **Augustin-Louis Cauchy** and **Charles Hermite**.

The term "eigenvalue" itself comes from the German word *eigen*, meaning "own" or "characteristic" reflecting the idea that eigenvalues and eigenvectors capture intrinsic properties of a matrix.

2.1.1 Computation of Eigenvalues and Eigenvectors: A Simple Example

Rewriting the equation defining eigenvalues and eigenvectors to bring all terms to one side:

$$A\mathbf{x} - \lambda\mathbf{x} = 0$$

This can be factored as:

$$(A - \lambda I)\mathbf{x} = 0$$

where I is the identity matrix of the same size as A. The key point here is that we are looking for *non-zero solutions* to this equation because the zero vector $\mathbf{x} = 0$ is trivial and uninteresting. For a non-trivial solution to exist, the matrix $(A - \lambda I)$ must be singular, meaning it does not have an inverse.

Recall from linear algebra that a matrix is singular if and only if its determinant is zero. Therefore, we impose the condition:

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

This equation is known as the **characteristic equation** of the matrix A, and solving it yields the eigenvalues λ of A. Once the eigenvalues are found, the corresponding eigenvectors can be determined by substituting each λ back into the equation $(A - \lambda I)\mathbf{x} = 0$ and solving for $\mathbf{x}$.

Let us consider a simple 2×2 matrix:

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

We form the matrix $\mathbf{A} - \lambda\mathbf{I}$:

$$\mathbf{A} - \lambda\mathbf{I} = \begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}.$$

The determinant of this matrix gives the characteristic polynomial:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3.$$

Solving the quadratic equation:

$$\lambda^2 - 4\lambda + 3 = 0 \quad \Rightarrow \quad \lambda_1 = 3, \quad \lambda_2 = 1.$$

Now that we have the eigenvalues, we compute the corresponding eigenvectors by solving $(\mathbf{A}-\lambda\mathbf{I})\mathbf{v} = \mathbf{0}$ for each λ.

For $\lambda_1 = 3$, we solve:

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This system implies $x = y$, so an eigenvector is:

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

For $\lambda_2 = 1$, we solve:

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This system implies $x = -y$, so an eigenvector is:

$$\mathbf{v}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

These eigenvectors are orthogonal, which is not coincidental—they arise from a symmetric matrix, as discussed next.

2.1.2 Symmetric Matrices and Their Eigenvalues/Eigenvectors

A real square matrix $\mathbf{A}$ is called **symmetric** if it satisfies:

$$\mathbf{A} = \mathbf{A}^\top.$$

One of the most fundamental results in linear algebra — the **Spectral Theorem** — applies specifically to real symmetric matrices. This theorem guarantees that every real symmetric matrix can be diagonalized by an orthogonal matrix. In other words, its eigenvectors form a complete orthonormal basis for $\mathbb{R}^n$, and all of its eigenvalues are real.

More precisely, if $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric (i.e., $\mathbf{A} = \mathbf{A}^\top$), then the following properties hold:

1. All eigenvalues of $\mathbf{A}$ are real numbers.
2. There exists an orthonormal basis for $\mathbb{R}^n$ consisting entirely of eigenvectors of $\mathbf{A}$.
3. Consequently, $\mathbf{A}$ is orthogonally diagonalizable:

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^\top,$$

where:

- $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix whose columns are orthonormal eigenvectors of $\mathbf{A}$,

- $\Lambda \in \mathbb{R}^{n \times n}$ is a diagonal matrix containing the corresponding real eigenvalues.

This decomposition not only simplifies computations involving $\mathbf{A}$, but also ensures that even when eigenvalues are repeated, we can still construct a full set of orthonormal eigenvectors.

The study of symmetric matrices and their spectral properties has deep historical roots. One of the earliest contributors was **Augustin-Louis Cauchy**, who investigated quadratic forms and laid the groundwork for what would later become the modern formulation of the spectral theorem.

In the next section, we prove that eigenvectors corresponding to distinct eigenvalues of a symmetric matrix are orthogonal. A full proof of the spectral theorem is provided in Appendix B.1.

Orthogonality of Eigenvectors for Distinct Eigenvalues

Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be eigenvectors of a symmetric matrix $\mathbf{A}$ corresponding to distinct eigenvalues λ_1 and λ_2. Then:

$$\mathbf{A}\mathbf{v}_1 = \lambda_1 \mathbf{v}_1, \quad \mathbf{A}\mathbf{v}_2 = \lambda_2 \mathbf{v}_2.$$

Taking the dot product of both sides of the first equation with $\mathbf{v}_2$, and using the symmetry of $\mathbf{A}$, we get:

$$\mathbf{v}_2^\top \mathbf{A}\mathbf{v}_1 = \lambda_1 \mathbf{v}_2^\top \mathbf{v}_1.$$

On the other hand, taking the transpose of the second equation and multiplying by $\mathbf{v}_1$, we obtain:

$$\mathbf{v}_2^\top \mathbf{A}\mathbf{v}_1 = \lambda_2 \mathbf{v}_2^\top \mathbf{v}_1.$$

Equating both expressions:

$$\lambda_1 \mathbf{v}_2^\top \mathbf{v}_1 = \lambda_2 \mathbf{v}_2^\top \mathbf{v}_1 \quad \Rightarrow \quad (\lambda_1 - \lambda_2)\mathbf{v}_2^\top \mathbf{v}_1 = 0.$$

Since $\lambda_1 \neq \lambda_2$, we must have:

$$\mathbf{v}_2^\top \mathbf{v}_1 = 0,$$

which means the eigenvectors are orthogonal.

Repeated Eigenvalues and Orthonormal Bases

Even when eigenvalues are repeated, we can still choose an orthonormal set of eigenvectors spanning the corresponding eigenspace. This is again ensured by the spectral theorem. Therefore, symmetric matrices always admit a full set of orthonormal eigenvectors.

2.1.3 Positive Semi-Definite Matrices

A special subclass of symmetric matrices is the set of **positive semi-definite** matrices. These matrices play a crucial role in optimization, statistics, and machine learning, particularly in applications involving quadratic forms, covariance matrices, and convex optimization problems.

A real symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is said to be **positive semi-definite**, written as $\mathbf{A} \succeq 0$, if for all non-zero vectors $\mathbf{x} \in \mathbb{R}^n$, it satisfies:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0.$$

This expression, $\mathbf{x}^\top \mathbf{A}\mathbf{x}$, is known as the *quadratic form* associated with $\mathbf{A}$. Intuitively, this means that for any vector $\mathbf{x}$, the result of applying the transformation $\mathbf{A}$ to $\mathbf{x}$ maintains a non-negative inner product with $\mathbf{x}$ itself, indicating a kind of "non-opposing" directional behavior.

An equivalent and highly useful definition of positive semi-definiteness is based on eigenvalues: A symmetric matrix $\mathbf{A}$ is positive semi-definite if and only if all of its eigenvalues are non-negative:

$$\lambda_i \geq 0 \quad \text{for all } i = 1, 2, \ldots, n.$$

This characterization is powerful because it allows us to determine whether a matrix is positive semi-definite simply by computing its eigenvalues.

Why Are Eigenvalues Non-Negative for Positive Semi-Definite Matrices? To understand why all eigenvalues of a positive semi-definite matrix must be non-negative, recall the following facts from linear algebra:

Every real symmetric matrix is diagonalizable and has real eigenvalues. Let $(\lambda, \mathbf{v})$ be an eigenpair of $\mathbf{A}$, so that $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$, where $\mathbf{v} \neq \mathbf{0}$. Multiplying both sides by $\mathbf{v}^\top$ gives:

$$\mathbf{v}^\top \mathbf{A}\mathbf{v} = \lambda \mathbf{v}^\top \mathbf{v}.$$

Since $\mathbf{A}$ is positive semi-definite, we have $\mathbf{v}^\top \mathbf{A}\mathbf{v} \geq 0$, and since $\mathbf{v}^\top \mathbf{v} = \|\mathbf{v}\|^2 > 0$, it follows that:

$$\lambda = \frac{\mathbf{v}^\top \mathbf{A}\mathbf{v}}{\mathbf{v}^\top \mathbf{v}} \geq 0.$$

Thus, every eigenvalue λ of a positive semi-definite matrix must be non-negative.

Why Eigenvalues Being Non-Negative Implies Positive Semi-Definiteness We have already established that if a symmetric matrix $\mathbf{A}$ is positive semi-definite, then all of its eigenvalues are non-negative. We now prove the converse: if all eigenvalues of a symmetric matrix $\mathbf{A}$ are non-negative, then $\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$, which means $\mathbf{A}$ is positive semi-definite.

Since $\mathbf{A}$ is symmetric, it has an orthonormal basis of eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ corresponding to real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, with $\lambda_i \geq 0$ for all i.

Now consider any vector $\mathbf{x} \in \mathbb{R}^n$. Because the eigenvectors $\mathbf{v}_i$ form a basis for $\mathbb{R}^n$, we can write:

$$\mathbf{x} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n,$$

for some real coefficients $c_1, c_2, \ldots, c_n$.

Using the linearity of $\mathbf{A}$, we compute the quadratic form:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = (\mathbf{x})^\top \mathbf{A}\mathbf{x} = \left(\sum_{i=1}^{n} c_i\mathbf{v}_i\right)^\top \mathbf{A} \left(\sum_{j=1}^{n} c_j\mathbf{v}_j\right).$$

Because $\mathbf{A}\mathbf{v}_j = \lambda_j\mathbf{v}_j$, this becomes:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = \left(\sum_{i=1}^{n} c_i\mathbf{v}_i\right)^\top \left(\sum_{j=1}^{n} c_j\lambda_j\mathbf{v}_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \lambda_j \mathbf{v}_i^\top \mathbf{v}_j.$$

Now, since the eigenvectors are orthonormal, $\mathbf{v}_i^\top \mathbf{v}_j = 0$ when $i \neq j$, and $\mathbf{v}_i^\top \mathbf{v}_i = 1$. Therefore, only the diagonal terms survive:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = \sum_{i=1}^{n} \lambda_i c_i^2.$$

Since $\lambda_i \geq 0$ and $c_i^2 \geq 0$, each term in the sum is non-negative, so:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0.$$

This proves that if all eigenvalues of a symmetric matrix $\mathbf{A}$ are non-negative, then $\mathbf{A}$ is positive semi-definite. Combined with the earlier result, we conclude that a symmetric matrix is positive semi-definite if and only if all of its eigenvalues are non-negative.

These properties make positive semi-definite matrices particularly well-behaved in many applications, including least-squares problems, principal component analysis (PCA), and kernel methods in machine learning.

2.2 Matrix Orthogonality

In this section, we explore two key concepts in linear algebra: the covariance matrix and orthogonality. These concepts are foundational for many applications, particularly in dimensionality reduction techniques like Principal Component Analysis (PCA).

2.2.1 Covariance Matrix

The covariance matrix is a fundamental tool in statistics and linear algebra, used to summarize the relationships between variables in multivariate data.

Definition

Given a dataset with n observations of p variables, let $\mathbf{X}$ be the $n \times p$ data matrix where each row represents an observation and each column represents a variable. The covariance matrix $\mathbf{\Sigma}$ of $\mathbf{X}$ is defined as:

$$\mathbf{\Sigma} = \frac{1}{n-1}(\mathbf{X} - \bar{\mathbf{X}})^T(\mathbf{X} - \bar{\mathbf{X}}) \tag{2.1}$$

where $\bar{\mathbf{X}}$ is the mean-centered data matrix obtained by subtracting the column means from each column of $\mathbf{X}$.

Properties

- The covariance matrix $\mathbf{\Sigma}$ is a $p \times p$ symmetric matrix.

- The diagonal entries of $\mathbf{\Sigma}$ represent the variances of the individual variables.

- The off-diagonal entries represent the covariances between pairs of variables, indicating how two variables vary together.

- $\mathbf{\Sigma}$ is positive semi-definite, meaning all its eigenvalues are non-negative.

2.2.2 Orthogonality

Orthogonality is a key concept in linear algebra that has significant implications in many applications, including matrix decompositions and dimensionality reduction techniques like PCA.

Definition of Orthogonal Vectors

Two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^n$ are said to be orthogonal if their dot product is zero:

$$\mathbf{u} \cdot \mathbf{v} = 0$$

Geometrically, this means that $\mathbf{u}$ and $\mathbf{v}$ are perpendicular to each other.

Orthogonal Basis

A set of vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ forms an orthogonal basis for a subspace if:

$$\mathbf{v}_i \cdot \mathbf{v}_j = 0 \quad \text{for all } i \neq j$$

If the vectors are also normalized to have unit length ($\|\mathbf{v}_i\| = 1$), the basis is called orthonormal.

2.3 Singular Value Decomposition

The **Singular Value Decomposition (SVD)** is one of the most powerful matrix factorizations in linear algebra.

The concept of the Singular Value Decomposition (SVD) has its origins in the 19th century, emerging from the study of bilinear forms and differential geometry. The Italian mathematician **Eugenio Beltrami** first laid the foundation in 1873 by demonstrating that any real $m \times n$ matrix can be diagonalized through orthogonal transformations. The French mathematician **Camille Jordan** further developed these ideas in 1874, formalizing much of the theoretical framework.

However, it wasn't until the mid-20th century—with the rise of numerical linear algebra and the advent of computers—that SVD became widely recognized as an essential computational tool. Today, SVD has broad numerous practical applications in engineering and Computer Vision.

2.3.1 Overview

Given any real $m \times n$ matrix A with $m \geq n$, the SVD expresses A as:

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

where:

- U is an $m \times m$ orthogonal matrix whose columns are the **left singular vectors** of A.
- Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal, known as the **singular values** of A.
- V is an $n \times n$ orthogonal matrix whose columns are the **right singular vectors** of A.

This decomposition always exists for any real or complex matrix A, even if it is not square or invertible.

2.3.2 Relation to Eigenvalues and Eigenvectors

While eigenvalue decomposition applies only to square matrices, the **Singular Value Decomposition (SVD)** generalizes this concept to any real or complex rectangular matrix. The key insight behind SVD lies in its deep connection to the symmetric matrices A^TA and AA^T.

Let A be an $m \times n$ real matrix with $m \geq n$. Consider the symmetric matrices:

- A^TA: $n \times n$, symmetric, positive semi-definite
- AA^T: $m \times m$, symmetric, positive semi-definite

These matrices have:

- Real, non-negative eigenvalues
- Orthonormal sets of eigenvectors

The singular values of A arise naturally as the square roots of the eigenvalues of these matrices.

Connection Between SVD and Eigenvalues Given the SVD of A:

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

Then,

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^T\Sigma V^T$$

Since $\Sigma^T\Sigma$ is diagonal with entries σ_i^2, this becomes:

$$A^TA = V(\Sigma^T\Sigma)V^T$$

which is precisely the *eigenvalue decomposition* of A^TA:

- V contains the eigenvectors
- $\Sigma^T\Sigma$ contains the eigenvalues σ_i^2

Similarly,

$$AA^T = U\Sigma V^TV\Sigma^TU^T = U\Sigma\Sigma^TU^T$$

So,

$$AA^T = U(\Sigma\Sigma^T)U^T$$

This is the eigenvalue decomposition of AA^T:

- U contains the eigenvectors
- $\Sigma\Sigma^T$ contains the eigenvalues σ_i^2

Thus, we can summarize:

- The columns of V are the eigenvectors of A^TA.
- The columns of U are the eigenvectors of AA^T.
- The diagonal entries of Σ, i.e., the singular values σ_i, are the square roots of the eigenvalues of A^TA (or equivalently AA^T):

$$\sigma_i = \sqrt{\lambda_i(A^TA)} = \sqrt{\lambda_i(AA^T)}$$

Example: Connecting SVD to Eigenvalues Let's take a simple matrix:

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

Compute:

$$A^T A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad AA^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Eigenvalues of $A^T A$:

$$\det(A^T A - \lambda I) = (2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = 0 \Rightarrow \lambda_1 = 3,\ \lambda_2 = 1 \Rightarrow \sigma_1 = \sqrt{3},\ \sigma_2 = 1$$

Eigenvectors of $A^T A$ form matrix V, and left singular vectors (from AA^T) form matrix U. These, along with the singular values, yield the full SVD:

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

This confirms how the eigenvalues of $A^T A$ and AA^T directly determine the singular values and vectors.

2.3.3 Geometric Interpretation of SVD

Singular Value Decomposition (SVD) offers a powerful geometric interpretation of how a linear transformation reshapes space. Consider any real 2×2 matrix M, such as the following shearing matrix:

$$M = \begin{bmatrix} 1.0 & 0.8 \\ 0.0 & 1.0 \end{bmatrix}$$

This matrix transforms the unit circle in $\mathbb{R}^2$ into an ellipse, revealing how the transformation stretches and rotates the input space. The SVD decomposes this transformation into three interpretable steps:

1. An initial rotation or reflection represented by V^T,
2. A scaling along the coordinate axes determined by the diagonal matrix Σ,
3. A final rotation or reflection given by U.

The singular values σ_1 and σ_2, which are the diagonal entries of Σ, represent the lengths of the semi-major and semi-minor axes of the resulting ellipse. These values quantify the amount of stretching applied in orthogonal directions.

Figure 2.1 provides a view of how the original shape is rotated, scaled, and rotated again to produce the final result.

In the next section, we will demonstrate how to compute the SVD explicitly using both theoretical and numerical methods.

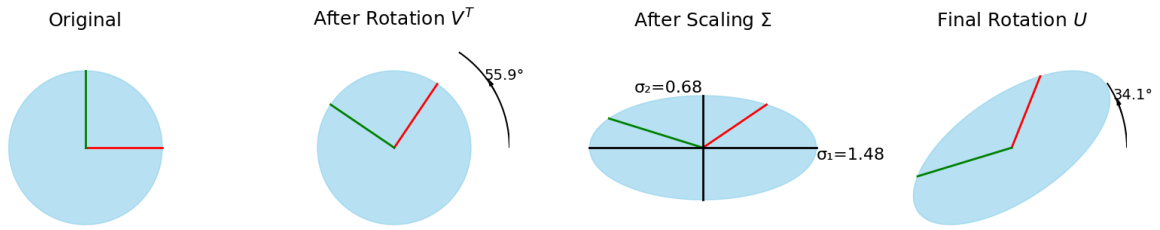

Fig. 2.1: Geometric interpretation of SVD applied to a 2D shearing matrix $\left(\begin{smallmatrix}1.0 & 0.8\\ 0.0 & 1.0\end{smallmatrix}\right)$. From left to right: (1) original unit circle, (2) after rotation V^T, (3) after scaling Σ with singular values σ_1 and σ_2 shown, and (4) final rotation U resulting in the transformed ellipse.

2.3.4 Computing the Singular Value Decomposition

The computation of the **Singular Value Decomposition (SVD)** can be approached from two perspectives: a theoretical derivation based on eigenvalue decomposition, and a practical numerical method that avoids explicit eigencomputations for stability and efficiency.

Theoretical Computation via Eigenvalue Decomposition

From a mathematical standpoint, SVD can be derived using the symmetric matrices A^TA and AA^T. Given an $m \times n$ matrix A with $m \geq n$, the singular values and vectors are computed as follows:

- The columns of V are the eigenvectors of A^TA.
- The diagonal entries of Σ are the square roots of the eigenvalues of A^TA:

$$\sigma_i = \sqrt{\lambda_i(A^TA)}$$

- The columns of U are the eigenvectors of AA^T.

This method provides a clean theoretical framework and is useful for small-scale problems or classroom examples. Let's illustrate this with a simple example.

Example: Let:

$$A = \begin{bmatrix}1 & 0.8\\ 0 & 1\end{bmatrix}$$

Compute A^TA:

$$A^TA = \begin{bmatrix}1 & 0.8\\ 0.8 & 1.64\end{bmatrix}$$

Now compute the eigenvalues of A^TA by solving the characteristic equation:

$$\det(A^TA - \lambda I) = \begin{vmatrix}1-\lambda & 0.8\\ 0.8 & 1.64-\lambda\end{vmatrix} = (1-\lambda)(1.64-\lambda) - 0.64 = 0$$

Expanding:

$$\lambda^2 - 2.64\lambda + 1 = 0$$

Solving this quadratic yields:

$$\lambda_1 = \frac{2.64 + \sqrt{(2.64)^2 - 4}}{2} \approx 2.096, \quad \lambda_2 = \frac{2.64 - \sqrt{(2.64)^2 - 4}}{2} \approx 0.544$$

Thus, the singular values are:

$$\sigma_1 = \sqrt{\lambda_1} \approx \sqrt{2.096} \approx 1.448, \quad \sigma_2 = \sqrt{\lambda_2} \approx \sqrt{0.544} \approx 0.738$$

Find corresponding eigenvectors of $A^T A$, normalize them, and use them to form the orthogonal matrix V. Then compute the left singular vectors:

$$u_i = \frac{1}{\sigma_i} A v_i$$

to construct the orthogonal matrix U. Finally, assemble:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}, \quad U = [\mathbf{u}_1 \ \mathbf{u}_2], \quad V = [\mathbf{v}_1 \ \mathbf{v}_2]$$

This confirms the factorization:

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

While conceptually clear, this method becomes numerically unstable and computationally expensive for large or ill-conditioned matrices.

Practical Numerical Computation

In practice, especially in software libraries like *LAPACK* and *NumPy*, the SVD is not computed using eigenvalue decomposition. This is because forming $A^T A$ can square the condition number, leading to a loss of numerical precision. Solving the characteristic equation for eigenvalues becomes numerically unstable for large matrices, and explicit eigenvalue methods are generally too slow for high-dimensional data.

Instead, most modern linear algebra libraries, such as *LAPACK*, use efficient numerical methods like bidiagonal reduction and iterative solvers (e.g., QR algorithm or divide-and-conquer) to compute the SVD. Tools like *NumPy* and *SciPy* in Python build on these optimized routines, with some offering randomized SVD for handling large datasets in machine learning.

2.4 Linear Least Squares Regression

Linear Least Squares Regression is a fundamental technique used to model the relationship between a dependent variable and one or more independent variables by minimizing the sum of the squares of the residuals (i.e., the differences between observed and predicted values). It is particularly useful when solving overdetermined systems, i.e., systems with more equations than unknowns.

The method was first introduced by the French mathematician **Adrien-Marie Legendre** in 1806 and later formalized by the German mathematician **Carl Friedrich Gauss**, who used it in his work on celestial mechanics.

2.4.1 Mathematical Formulation

Suppose we are given a set of n observations $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^p$ represents the input features and $y_i \in \mathbb{R}$ is the corresponding output. We aim to find a linear function that best fits the data:

$$y_i = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i,$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$ is the vector of coefficients (parameters) to be estimated, and ε_i represents the residual error.

Stacking all observations into matrices, we can write the system as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where:

- $\mathbf{y} \in \mathbb{R}^n$ is the vector of observed outputs,
- $\mathbf{X} \in \mathbb{R}^{n \times p}$ is the design matrix whose rows are the input vectors $\mathbf{x}_i^\top$,
- $\boldsymbol{\beta} \in \mathbb{R}^p$ is the coefficient vector,
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is the vector of residuals.

The goal is to minimize the sum of squared residuals:

$$J(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

Expanding this expression gives:

$$J(\boldsymbol{\beta}) = \mathbf{y}^\top \mathbf{y} - 2\boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{y} + \boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta}$$

To find the minimum, we take the derivative of $J(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ and set it equal to zero:

$$\frac{\partial J}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^\top \mathbf{y} + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} = 0$$

Solving this leads to the *normal equation*:

$$\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} = \mathbf{X}^\top \mathbf{y}$$

If $\mathbf{X}^\top \mathbf{X}$ is invertible, the unique solution is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

This provides the best linear unbiased estimator (BLUE) under the assumptions of the classical linear regression model.

2.4.2 Geometric Interpretation

From a geometric perspective, linear least squares finds the projection of the response vector $\mathbf{y}$ onto the column space of the design matrix $\mathbf{X}$. The residual vector $\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$ is orthogonal to the column space of $\mathbf{X}$, ensuring that the fitted values $\mathbf{X}\boldsymbol{\beta}$ are the closest point in that subspace to $\mathbf{y}$ in terms of Euclidean distance.

Linear least squares is widely used in signal processing, computer vision, and machine learning—particularly in linear regression and ridge regression—due to its simplicity, interpretability, and strong theoretical foundation.

2.5 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical procedure that transforms a set of possibly correlated variables into a smaller set of uncorrelated variables called *principal components*. These components capture the most significant patterns of variation in the data, enabling dimensionality reduction while preserving as much variance as possible.

PCA was first introduced by the British statistician and geneticist **Karl Pearson** in 1901, who formulated it as a method for fitting linear subspaces to multivariate data. It was later extended by the American mathematician **Harold Hotelling** in the 1930s. The technique gained widespread popularity with the advent of modern computing, especially in fields such as pattern recognition and machine learning.

PCA leverages core concepts from linear algebra—particularly eigenvalues and eigenvectors—to identify directions in the data that explain the maximum amount of variance. These directions form a new coordinate system where the first principal component corresponds to the direction of greatest variance, the second captures the next highest variance orthogonal to the first, and so on.

2.5.1 Mathematical Foundations of PCA

The steps involved in performing PCA are rooted in linear algebra and can be summarized as follows:

1. **Standardize the Data**: Since PCA is sensitive to the scale of the features, it is important to standardize the dataset so that each variable has zero mean and unit variance.
2. **Compute the Covariance Matrix**: The covariance matrix $\boldsymbol{\Sigma}$ captures how the features vary together. For a standardized data matrix $\mathbf{X}$ of size $n \times p$, where n is the number of data points (observations) and p is the number of features (dimensions), we have:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix},$$

where each $\mathbf{x}_i$ is a p-dimensional column vector representing a single observation.

According to the definition of sample covariance (see also Equation 2.1), the covariance matrix is computed as:

$$\boldsymbol{\Sigma} = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X} = \frac{1}{n-1}[\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_n] \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix},$$

This ensures that each entry Σ_{ij} reflects the covariance between feature i and feature j. The normalization factor $n-1$ provides an unbiased estimate of the population covariance matrix based on a sample.

3. **Calculate Eigenvalues and Eigenvectors**: The eigenvectors of the covariance matrix represent the principal components, while the corresponding eigenvalues indicate the amount of variance explained by each component. Sorting the eigenvectors by descending eigenvalues allows us to rank the principal components by their importance in explaining the variance.

4. **Select Top k Principal Components**: Choose the top k eigenvectors (where $k < p$) to form a projection matrix $\mathbf{W} \in \mathbb{R}^{p \times k}$, reducing the original data to a lower-dimensional space:

$$\mathbf{Y} = \mathbf{X}\mathbf{W}$$

where $\mathbf{Y} \in \mathbb{R}^{n \times k}$ contains the transformed data in the reduced space.

2.5.2 Projection Maximization and Line Fitting

An alternative way to understand PCA is through the concept of *line fitting*. In this interpretation, PCA aims to find a line (or hyperplane in higher dimensions) that best fits the data by maximizing the variance of the projections of the data points onto that line.

Let a direction vector be represented by $\mathbf{w} \in \mathbb{R}^p$, constrained to unit length: $\|\mathbf{w}\| = 1$. The projection of a data point $\mathbf{x}_i \in \mathbb{R}^p$ onto this direction is given by $\mathbf{w}^\top \mathbf{x}_i$. The goal is to maximize the sum of squared projections:

$$J(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{w}^\top \mathbf{x}_i)^2$$

This can be rewritten as:

$$J(\mathbf{w}) = \sum_{i=1}^{n} \mathbf{w}^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{w} = \mathbf{w}^\top \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top \right) \mathbf{w} = \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}$$

Since $\mathbf{X}^\top \mathbf{X} = (n-1)\mathbf{\Sigma}$, we can express the objective function using the covariance matrix:

$$J(\mathbf{w}) = (n-1)\mathbf{w}^\top \mathbf{\Sigma} \mathbf{w}$$

To ensure the constraint $\|\mathbf{w}\| = 1$, we apply the method of Lagrange multipliers. The optimization problem becomes:

$$\mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^\top \mathbf{\Sigma} \mathbf{w} - \lambda(\mathbf{w}^\top \mathbf{w} - 1)$$

Taking the derivative with respect to $\mathbf{w}$ (see Equation 3.9) and setting it to zero yields:

$$\mathbf{\Sigma}\mathbf{w} = \lambda \mathbf{w}$$

Thus, $\mathbf{w}$ is an eigenvector of the covariance matrix $\mathbf{\Sigma}$, and the corresponding eigenvalue λ represents the variance captured along that direction. Therefore, the first principal component corresponds to the eigenvector associated with the largest eigenvalue.

Chapter 3

Calculus

3.1 Preliminary: Essential Trigonometry

This document provides a concise review of essential trigonometric concepts and properties that are foundational for studying calculus. It assumes familiarity with the basic definitions of sine, cosine, and tangent but focuses on their advanced properties, identities, and applications.

3.1.1 Unit Circle and Trigonometric Functions

The unit circle is a critical tool in trigonometry. For any angle θ measured counterclockwise from the positive x-axis:

- The coordinates of the point where the terminal side intersects the unit circle are $(\cos\theta, \sin\theta)$.
- $\tan\theta = \frac{\sin\theta}{\cos\theta}$, provided $\cos\theta \neq 0$.

The **Pythagorean Identity** is derived from the Pythagorean theorem applied to the unit circle:

$$\sin^2\theta + \cos^2\theta = 1$$

3.1.2 Angle Sum and Difference Formulas

The angle sum and difference formulas are fundamental tools for simplifying trigonometric expressions involving sums or differences of angles. These are expressed as follows:

$$\begin{aligned}
\cos(a \pm b) &= \cos a \cos b \mp \sin a \sin b, \\
\sin(a \pm b) &= \sin a \cos b \pm \cos a \sin b, \\
\tan(a \pm b) &= \frac{\tan a \pm \tan b}{1 \mp \tan a \tan b}.
\end{aligned}$$

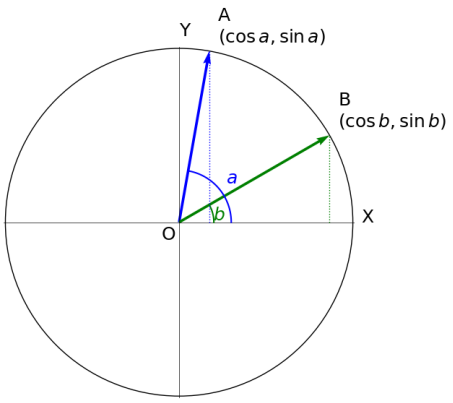

Fig. 3.1: Geometric illustration of points $A = (\cos a, \sin a)$ and $B = (\cos b, \sin b)$ on the unit circle. The vectors $\vec{OA}$ and $\vec{OB}$ form an angle of $a - b$, used in the proof of the identity $\cos(a - b) = \cos a \cos b + \sin a \sin b$.

Proof

To prove the formula for $\cos(a-b)$, consider two points on the unit circle: $A = (\cos a, \sin a)$ and $B = (\cos b, \sin b)$ (see Figure 3.1). Let the origin of the circle be denoted as O.

Considering the two vectors $\overrightarrow{OB}$ and $\overrightarrow{OA}$, according to the definiton of dot product (Equation 1.2), we have:

$$\overrightarrow{OA} \cdot \overrightarrow{OB} = |\overrightarrow{OA}||\overrightarrow{OB}| \cos(a-b) = \cos(a-b)$$

According to Equation 1.3, the dot product of the vectors $\overrightarrow{OA} = (\cos a, \sin a)$ and $\overrightarrow{OB} = (\cos b, \sin b)$ is given by:

$$\overrightarrow{OA} \cdot \overrightarrow{OB} = \cos a \cos b + \sin a \sin b.$$

Therefore, we conclude that:

$$\cos(a-b) = \cos a \cos b + \sin a \sin b.$$

The remaining formulas can be derived from the formula for $\cos(a-b)$; their proofs are omitted here.

3.1.3 Double-Angle and Half-Angle Formulas

These formulas are useful for simplifying trigonometric expressions that involve multiples or fractions of angles. They can be directly derived from the sum and difference identities in Section 3.1.2.

- **Double-Angle Formulas**:

$$\begin{aligned}
&\sin(2\theta) = 2\sin\theta\cos\theta, \\
&\cos(2\theta) = \cos^2\theta - \sin^2\theta = 2\cos^2\theta - 1 = 1 - 2\sin^2\theta, \\
&\tan(2\theta) = \frac{2\tan\theta}{1-\tan^2\theta}.
\end{aligned}$$

- **Half-Angle Formulas**:

$$\begin{aligned}
&\sin\left(\frac{\theta}{2}\right) = \pm\sqrt{\frac{1-\cos\theta}{2}}, \\
&\cos\left(\frac{\theta}{2}\right) = \pm\sqrt{\frac{1+\cos\theta}{2}}, \\
&\tan\left(\frac{\theta}{2}\right) = \frac{1-\cos\theta}{\sin\theta} = \frac{\sin\theta}{1+\cos\theta}.
\end{aligned}$$

3.2 Preliminary: Limit

The concept of a limit is central to calculus and mathematical analysis. Limits allow us to study the behavior of functions as their inputs approach specific values.

Definition of a Limit

Let $f(x)$ be a function defined on an open interval containing c, except possibly at c itself. We say that the **limit of** $f(x)$ **as** x **approaches** c is L, written as:

$$\lim_{x \to c} f(x) = L,$$

if for every $\epsilon > 0$, there exists a $\delta > 0$ such that whenever $0 < |x - c| < \delta$, it follows that $|f(x) - L| < \epsilon$.

Intuitively, this means that as x gets arbitrarily close to c, the value of $f(x)$ gets arbitrarily close to L.

The Squeeze Theorem

The **squeeze theorem** (also known as the sandwich theorem or pinching theorem) states the following:

Suppose three functions $f(x)$, $g(x)$, and $h(x)$ satisfy the inequality:

$$f(x) \leq g(x) \leq h(x),$$

for all x near c (except possibly at c). If:

$$\lim_{x \to c} f(x) = \lim_{x \to c} h(x) = L,$$

then:

$$\lim_{x \to c} g(x) = L.$$

The squeeze theorem is particularly useful when direct evaluation of a limit is difficult, but bounding the function between two simpler functions allows us to deduce the desired result.

Common Limit Formulas

$$\lim_{x \to 0} \frac{\sin(x)}{x} = 1, \tag{3.1}$$

$$\lim_{x \to 0} \frac{1 - \cos(x)}{x} = 0. \tag{3.2}$$

See proof in A.2

3.3 The Natural Constant *e*

The natural constant e, also known as Euler's number, is a cornerstone of mathematics. Below is an overview of its historical development.

1. Logarithms, introduced by John Napier (1550–1617) in the early 17th century, provided a powerful tool for simplifying calculations involving multiplication and exponentiation. Later, the natural logarithm, defined as the logarithm with base e, emerged as a fundamental concept in calculus and analysis.

2. The Swiss mathematician Jacob Bernoulli (1655-1705) first encountered e around 1683 while investigating compound interest. He posed the following question: If an investment yields 100% annual interest compounded n times per year, what happens to the total amount as n approaches infinity? Mathematically, this problem can be expressed as:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n.$$

Bernoulli demonstrated numerically that this limit converges to a specific value, which we now recognize as e. However, he did not provide a formal proof or compute the exact value of the limit.

3. Later, Leonhard Euler (1707–1783) and other mathematicians generalized Bernoulli's result. Instead of 100% as annual interest, they considered the interest rate as x, so the limit becomes more general:

$$S(x)=\lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n.$$

They proved that $S(x)=e^x$ (see proof in A.3.1).

Using the binomial theorem, $S(x)$ can be expanded into an infinite series (see proof in A.3.2):

$$S(x)=e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}. \tag{3.3}$$

Euler and his contemporaries further explored solutions to the fundamental differential equation:

$$\frac{dy}{dx}=y,\quad \text{subject to the initial condition: } y(0)=1.$$

They demonstrated that $S(x)$ satisfies these conditions, thereby identifying it as the exponential function e^x. When $x=1$, the value of e itself can be approximated using the series expansion:

$$e=\sum_{n=0}^{\infty}\frac{1}{n!}.$$

4. The derivative of the natural logarithm, $\frac{d}{dx}\ln(x)=\frac{1}{x}$, was later derived during the development of calculus.

3.4 Derivatives

This section provides a concise list of commonly used derivatives, starting from single-variable functions and extending to multi-variable functions. This progression naturally leads into concepts such as gradients, directional derivatives, and the Hessian matrix — all of which are essential in fields like machine learning, physics, and optimization.

3.4.1 Derivatives of Polynomials

These are the most basic derivatives in single-variable calculus:

- **Constant Rule:** $\frac{d}{dx}[c]=0$
- **Power Rule:** For any real number n, $\frac{d}{dx}[x^n]=nx^{n-1}$

- **Sum/Difference Rule:** If $f(x)$ and $g(x)$ are differentiable, then:

$$\frac{d}{dx}[f(x) \pm g(x)] = f'(x) \pm g'(x)$$

- **Constant Multiple Rule:** For any constant c,

$$\frac{d}{dx}[cf(x)] = cf'(x)$$

3.4.2 Trigonometric Functions

The derivatives of trigonometric functions rely on geometric reasoning and the limit $\lim_{x\to 0} \frac{\sin x}{x} = 1$.

$$\begin{aligned}
&\frac{d}{dx}[\sin x] = \cos x \\
&\frac{d}{dx}[\cos x] = -\sin x \\
&\frac{d}{dx}[\tan x] = \sec^2 x \quad \text{where} \sec x = \frac{1}{\cos x}, \quad \cos x \neq 0.
\end{aligned}$$

3.4.3 Exponential Functions

The derivative of the natural exponential function e^x is fundamental:

- $\frac{d}{dx}[e^x] = e^x$
- $\frac{d}{dx}[a^x] = a^x \ln a$ for $a > 0$.

3.4.4 Logarithmic Functions

The derivatives of logarithmic functions are derived using the inverse relationship between $\ln x$ and e^x:

$$\begin{aligned}
\frac{d}{dx}[\ln x] &= \frac{1}{x}, \quad x > 0 \\
\frac{d}{dx}[\log_a x] &= \frac{1}{x \ln a}, \quad \text{where } x > 0,\ a > 0,\ a \neq 1.
\end{aligned}$$

3.4.5 Inverse Trigonometric Functions

The derivatives of inverse trigonometric functions are derived using implicit differentiation and the chain rule:

$$\begin{aligned}
&\frac{d}{dx}[\arcsin x] = \frac{1}{\sqrt{1-x^2}}, \quad -1 < x < 1 \\
&\frac{d}{dx}[\arccos x] = -\frac{1}{\sqrt{1-x^2}}, \quad -1 < x < 1 \\
&\frac{d}{dx}[\arctan x] = \frac{1}{1+x^2}
\end{aligned}$$

3.4.6 The Chain Rule

The chain rule is one of the most important tools in differential calculus, especially in machine learning where functions are often composed of multiple layers of transformations (e.g., neural networks). It allows us to compute the derivative of composite functions.

If $y = f(u)$ and $u = g(x)$, then y is a composite function of x: $y = f(g(x))$. The chain rule states:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Alternatively, if $h(x) = f(g(x))$, then:

$$h'(x) = f'(g(x)) \cdot g'(x)$$

This means that to differentiate a composite function, we first differentiate the outer function evaluated at the inner function, and then multiply it by the derivative of the inner function.

Examples

- Let $y = e^{\sin x}$. Let $u = \sin x$, so $y = e^u$. Then:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} = e^u \cdot \cos x = e^{\sin x} \cdot \cos x$$

- Let $z = \ln(3x^2 + 4)$. Let $u = 3x^2 + 4$, so $z = \ln u$. Then:

$$\frac{dz}{dx} = \frac{1}{u} \cdot \frac{du}{dx} = \frac{1}{3x^2 + 4} \cdot 6x = \frac{6x}{3x^2 + 4}$$

Generalization to Multiple Layers

The chain rule can be extended to more than two functions. For example, if $y = f(g(h(x)))$, then:

$$\frac{dy}{dx} = f'(g(h(x))) \cdot g'(h(x)) \cdot h'(x)$$

This generalizes naturally to deep compositions, making it foundational for computing gradients in multilayered models such as deep neural networks.

3.4.7 From Single to Multi-Variable Derivatives

So far, we've explored derivatives of functions of a single variable — $f(x)$. However, in many real-world applications, especially in machine learning and optimization, we deal with functions of multiple variables — such as $f(x_1, x_2, \ldots, x_n)$. In this context, the concept of a derivative must be extended to account for how the function changes with respect to each input variable individually, as well as collectively in any direction in the input space.

3.4.8 Partial Derivatives

For a function $f(x_1, x_2, \ldots, x_n)$, the **partial derivative** of f with respect to x_i is defined as:

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_i + h, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots, x_n)}{h}$$

It measures how f changes as we vary only x_i, holding all other variables constant.

3.4.9 Gradient

The **gradient** of a scalar-valued function $f(x_1, x_2, \ldots, x_n)$ is a vector containing all its first-order partial derivatives:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

The gradient points in the direction of maximum rate of increase of f, and its magnitude gives the slope in that direction.

3.4.10 Directional Derivatives

Let $f(x_1, x_2, \ldots, x_n)$ be a function of n variables with continuous partial derivatives, and let $\vec{u} = \langle u_1, u_2, \ldots, u_n \rangle$ be a unit vector representing a direction. The directional derivative of f at a point in the direction of $\vec{u}$ is defined as:

$$D_{\vec{u}} f = \nabla f \cdot \vec{u}$$

This generalizes the idea of a derivative to arbitrary directions in the input space.

3.4.11 The Hessian Matrix

The second-order behavior of a multi-variable function is captured by the **Hessian matrix**, a square matrix of second-order partial derivatives.

For a function $f(x_1, x_2, \ldots, x_n)$ with continuous second partial derivatives, the Hessian matrix $\mathbf{H}$ is defined as:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Properties and Usefulness of the Hessian

- The Hessian is symmetric if the second partial derivatives are continuous (Clairaut's theorem).
- It is used in optimization to determine whether a critical point is a minimum, maximum, or saddle point via the definiteness of the matrix.
- In Newton's method for optimization, the Hessian is used to approximate the curvature of the function.

Example

Let $f(x, y) = x^2 + xy + y^3$. Compute the Hessian at $(1, 2)$.

Step 1: Compute second partial derivatives.

$$\frac{\partial^2 f}{\partial x^2} = 2, \quad \frac{\partial^2 f}{\partial x \partial y} = 1, \quad \frac{\partial^2 f}{\partial y \partial x} = 1, \quad \frac{\partial^2 f}{\partial y^2} = 6y$$

Step 2: Evaluate at $(1, 2)$**.**

$$\mathbf{H} = \begin{bmatrix} 2 & 1 \\ 1 & 12 \end{bmatrix}$$

This matrix can be used to analyze the local curvature of f at that point.

3.5 Integration

Integration provides a way to compute the total accumulation of a quantity over a range—for example, computing total distance from velocity, or total probability in statistics.

3.5.1 Indefinite Integrals

An **indefinite integral** represents the set of all antiderivatives of a function. Since the derivative of a constant is zero, we add an arbitrary constant C to account for all possible solutions:

$$\int f(x)\,dx = F(x) + C \quad \text{where} \quad F'(x) = f(x)$$

This means integration "undoes" differentiation, just as subtraction undoes addition.

Example:

$$\int x^n\,dx = \frac{x^{n+1}}{n+1} + C \quad (\text{for } n \neq -1)$$

3.5.2 Definite Integrals

A **definite integral** computes the net area under the curve of a function $f(x)$ between two points a and b. It results in a specific number:

$$\int_a^b f(x)\,dx = F(b) - F(a)$$

Here, $F(x)$ is any antiderivative of $f(x)$.

Geometric interpretation: If $f(x) \geq 0$, then $\int_a^b f(x)dx$ gives the area between the curve and the x-axis from $x = a$ to $x = b$.

Key Properties:

- $\int_a^a f(x)\,dx = 0$: No area when limits are equal.
- $\int_a^b f(x)\,dx = -\int_b^a f(x)\,dx$: Reversing limits flips the sign.
- $\int_a^b [f(x) \pm g(x)]\,dx = \int_a^b f(x)\,dx \pm \int_a^b g(x)\,dx$: Integration respects linearity.
- $\int_a^b cf(x)\,dx = c\int_a^b f(x)\,dx$: Constants can be factored out.

3.5.3 Fundamental Theorem of Calculus

The Fundamental Theorem of Calculus bridges the gap between derivatives and integrals. One key form says:

If f is continuous on $[a, b]$, and we define:

$$F(x) = \int_a^x f(t)\,dt$$

Then:

$$F'(x) = \frac{d}{dx}\left[\int_a^x f(t)\,dt\right] = f(x)$$

This tells us that the derivative of the area under the curve up to point x is simply the height of the curve at x. This is powerful—it allows us to evaluate complex derivatives involving integrals.

3.5.4 Derivative of an Integral (Leibniz Rule)

Now consider a more general case where both bounds of the integral depend on x:

$$\frac{d}{dx}\left[\int_{u(x)}^{v(x)} f(t)\,dt\right] = f(v(x))v'(x) - f(u(x))u'(x)$$

This is known as Leibniz's Rule.

Special Case:

$$\frac{d}{dx}\left[\int_{-\infty}^x f(t)\,dt\right] = f(x) \quad \text{if } \int_{-\infty}^x f(t)\,dt \text{ converges.}$$

This rule is especially important in machine learning and probability theory. For example, if $f(t)$ is a probability density function (PDF), then $\int_{-\infty}^{x} f(t)dt$ is the cumulative distribution function (CDF), and its derivative is again the PDF.

To build intuition, Figure 3.2 visualize the area represented by a definite integral and how its rate of change relates to the function value.

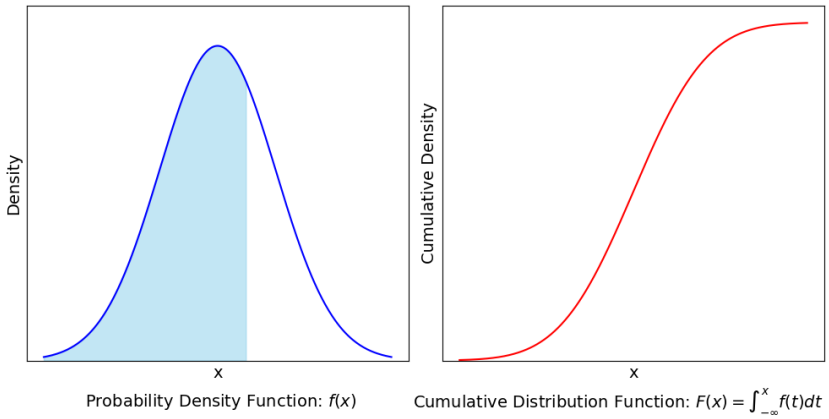

Fig. 3.2: Plot showing the function $f(x)$ and the shaded area under the curve from $-\infty$ to x, representing $F(x) = \int_{-\infty}^{x} f(t)dt$. The slope of $F(x)$ at any point equals $f(x)$.

3.5.5 From 1D to 2D: Intuition Behind Double Integrals

In single-variable calculus, the definite integral $\int_a^b f(x)\,dx$ represents the *area under the curve* $y = f(x)$ between $x = a$ and $x = b$. This is a one-dimensional concept: integrating over a line segment on the x-axis.

Now imagine extending this idea into two dimensions. Instead of a function of one variable, $f(x)$, consider a function of two variables, $z = f(x, y)$, which defines a surface above the xy-plane. The double integral allows us to calculate the *volume under this surface* over a given region R in the plane.

Mathematically, this is written as:

$$\iint_R f(x, y)\,dA$$

This expression tells us: "sum up the values of $f(x, y)$ over all points (x, y) in the region R, weighted by an infinitesimal area element dA."

3.5.6 Evaluating Double Integrals Using Iteration

Depending on the shape of the region R, we can write the double integral as:

$$\iint_R f(x, y)\,dA = \int_{y=c}^{d} \int_{x=a(y)}^{b(y)} f(x, y)\,dx\,dy \quad \text{or} \quad \int_{x=a}^{b} \int_{y=c(x)}^{d(x)} f(x, y)\,dy\,dx$$

The choice of order depends on the geometry of the region and the simplicity of the limits.

Here's how it works:

1. Fix one variable (say y) and integrate over the other (x), which gives you a "slice" of the volume.
2. Then integrate those slices over the second variable to obtain the total volume.

Example

Compute the double integral of $f(x, y) = xy$ over the rectangle $R = [0, 2] \times [1, 3]$:

$$\int_1^3 \int_0^2 xy \, dx \, dy = \int_1^3 \left[\frac{x^2}{2} y \right]_0^2 dy = \int_1^3 \left(\frac{4}{2} y \right) dy = \int_1^3 2y \, dy = \left[y^2 \right]_1^3 = 9 - 1 = 8$$

This result means that the volume under the surface $z = xy$ over the rectangle $[0, 2] \times [1, 3]$ is 8 cubic units.

3.5.7 Integration in Polar Coordinates

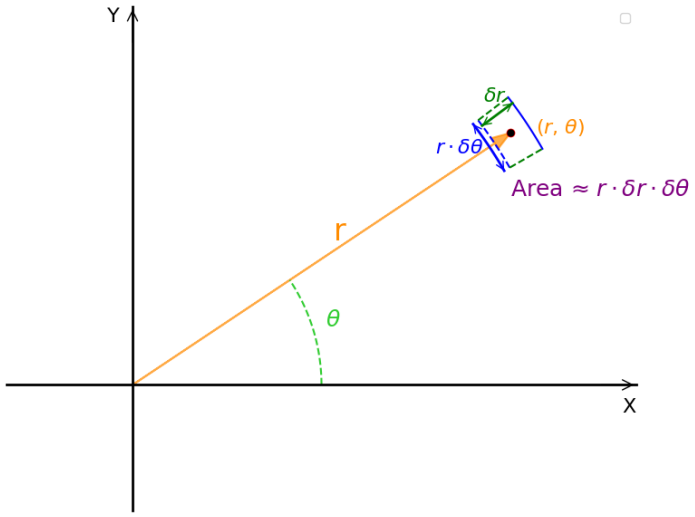

Fig. 3.3: Illustration of a differential area element in polar coordinates for integration. The point is shown with its polar coordinates (r, θ), where r is the radial distance and θ is the angle from the positive x-axis. The region represents the differential area element defined by increments δr and $\delta\theta$, with the area approximated as $r \cdot \delta r \cdot \delta\theta$.

Certain regions and functions are more naturally expressed using radius and angle rather than Cartesian coordinates. In such cases, we use **polar coordinates**, defined as:

$$x = r \cos\theta, \quad y = r \sin\theta$$

Consider an infinitesimal area element around a point (x, y), which corresponds to the point (r, θ) in polar coordinates (see Figure 3.3). Because $\delta\theta$ is very small, one side of the area element can be approximated as $r \cdot \delta\theta$, while the other side is simply δr.

Thus, the differential area element dA in polar coordinates becomes:

$$dA = r \, dr \, d\theta$$

This leads to the transformation of a double integral from Cartesian to polar coordinates:

$$\iint_R f(x,y)\,dA = \iint_{R'} f(r\cos\theta, r\sin\theta)\,r\,dr\,d\theta$$

where R' is the region R described in terms of polar coordinates.

Example

Evaluate the double integral

$$\iint_{\mathbb{R}^2} e^{-(x^2+y^2)}\,dA$$

over the entire plane.

In polar coordinates, this becomes:

$$\int_0^{2\pi}\int_0^{\infty} e^{-r^2}\,r\,dr\,d\theta$$

We first evaluate the inner integral, and then integrate over θ. The value of the double integral becomes:

$$\iint_{\mathbb{R}^2} e^{-(x^2+y^2)}\,dA = \pi$$

This result plays a crucial role in probability theory and statistics, particularly in evaluating Gaussian integrals.

3.6 Euler's Formula

In 1748, Leonhard Euler introduced in his publication a result which is now celebrated as **Euler's Formula**. This formula elegantly unifies exponential functions, trigonometric functions, and complex number theory into a single expression.

The formula is given by:

$$e^{ix} = \cos(x) + i\sin(x), \tag{3.4}$$

where:

- i is the imaginary unit,
- x is a real number, typically interpreted as an angle measured in radians.

One common method to derive Euler's formula involves using **Taylor series expansions** of e^x, $\cos(x)$, and $\sin(x)$. Recall that the series expansion in Equation (3.3) for e^x is given by:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}. \tag{3.5}$$

The Taylor series expansions for $\cos(x)$ and $\sin(x)$ are:

$$\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}, \tag{3.6}$$

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}. \tag{3.7}$$

By substituting ix into the Taylor series for e^x, and grouping the real and imaginary terms, the Euler's formula can be derived.

A particularly famous special case of Euler's formula arises when $x = \pi$. Substituting $x = \pi$ yields:

$$e^{i\pi} + 1 = 0.$$

This equation, known as **Euler's identity**, is often hailed as the "most beautiful equation in mathematics" due to its concise combination of five fundamental mathematical constants: e, i, π, 1, and 0.

3.7 Matrix Calculus

Matrix calculus extends the principles of scalar calculus to functions involving vectors and matrices. It provides a systematic way to compute derivatives of multivariate functions, which is essential in fields such as machine learning, optimization, and statistics.

3.7.1 Basic Definitions

Let $\mathbf{x} \in \mathbb{R}^p$ be a **column** vector, and let $f : \mathbb{R}^p \to \mathbb{R}$ be a scalar-valued function of $\mathbf{x}$. The derivative of f with respect to $\mathbf{x}$ is defined as the gradient vector:

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_p} \end{bmatrix} \in \mathbb{R}^p$$

When dealing with derivatives of vector-valued or matrix-valued functions, it is important to specify the layout convention being used, as there are multiple ways to arrange the resulting derivative.

In this document, we adopt the **numerator layout convention**, where the derivative of a **column** vector $\mathbf{y} \in \mathbb{R}^m$ with respect to another **column** vector $\mathbf{x} \in \mathbb{R}^p$ is defined as the $p \times m$ matrix:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_p} & \frac{\partial y_2}{\partial x_p} & \cdots & \frac{\partial y_m}{\partial x_p} \end{bmatrix} \in \mathbb{R}^{p \times m}$$

In this convention, each column corresponds to the derivative of one component of $\mathbf{y}$ with respect to the entire vector $\mathbf{x}$.

To understand this better, consider a linear transformation $f(\mathbf{x}) = A\mathbf{x}$, where $A \in \mathbb{R}^{m\times p}$. For a small perturbation $\delta\mathbf{x} \in \mathbb{R}^p$, we have:

$$f(\mathbf{x} + \delta\mathbf{x}) - f(\mathbf{x}) = A\delta\mathbf{x}$$

We can also express this using the derivative:

$$f(\mathbf{x} + \delta\mathbf{x}) - f(\mathbf{x}) = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^\top \delta\mathbf{x}$$

Comparing both expressions gives:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = A^\top$$

Thus, under the numerator layout convention, the derivative of a linear transformation $A\mathbf{x}$ is $A^\top$.

3.7.2 Common Expressions

We now present several standard results in matrix calculus that are widely used in applied mathematics and data science.

- If $f(\mathbf{x}) = A\mathbf{x}$, where $A \in \mathbb{R}^{m\times p}$, then:

$$\frac{\partial f}{\partial \mathbf{x}} = \frac{\partial (A\mathbf{x})}{\partial \mathbf{x}} = A^\top \tag{3.8}$$

- If $f(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x}$, where $A \in \mathbb{R}^{p\times p}$ is symmetric, then:

$$\frac{\partial f}{\partial \mathbf{x}} = \frac{\partial (\mathbf{x}^\top A\mathbf{x})}{\partial \mathbf{x}} = 2A\mathbf{x} \tag{3.9}$$

The first derivative was shown earlier. Let us now derive the result for the derivative of the quadratic form $f(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x}$, assuming A is symmetric.

Start with a small perturbation $\delta\mathbf{x} \in \mathbb{R}^p$, and expand:

$$f(\mathbf{x} + \delta\mathbf{x}) = (\mathbf{x} + \delta\mathbf{x})^\top A(\mathbf{x} + \delta\mathbf{x}) = \mathbf{x}^\top A\mathbf{x} + \mathbf{x}^\top A\delta\mathbf{x} + \delta\mathbf{x}^\top A\mathbf{x} + \delta\mathbf{x}^\top A\delta\mathbf{x}$$

Ignoring the second-order term $\delta\mathbf{x}^\top A\delta\mathbf{x}$, we get:

$$f(\mathbf{x} + \delta\mathbf{x}) - f(\mathbf{x}) \approx \mathbf{x}^\top A\delta\mathbf{x} + \delta\mathbf{x}^\top A\mathbf{x}$$

Since A is symmetric ($A = A^\top$), by expanding the matrix multiplication by indices, it is not hard to see that:

$$\mathbf{x}^\top A\delta\mathbf{x} = \delta\mathbf{x}^\top A\mathbf{x}$$

Thus:

$$f(\mathbf{x} + \delta\mathbf{x}) - f(\mathbf{x}) \approx 2(A\mathbf{x})^\top \delta\mathbf{x}$$

This implies:

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top A\mathbf{x}) = 2A\mathbf{x}$$

3.8 Convolution and Cross-Correlation

3.8.1 Convolution

The convolution of two real- or complex-valued functions f and g is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau, \tag{3.10}$$

provided the integral converges. Convolution is a fundamental operation in signal processing and mathematical analysis, often used to describe linear time-invariant systems.

3.8.2 Algebraic Properties of Convolution

Convolution satisfies several important algebraic properties:

- **Commutativity:** $f * g = g * f$
- **Associativity:** $(f * g) * h = f * (g * h)$
- **Distributivity over addition:** $f * (g + h) = f * g + f * h$

For detailed proofs of these properties, see Appendix A.4.

3.8.3 Cross-Correlation

The cross-correlation of two functions f and g, denoted $f \star g$, is defined as:

$$(f \star g)(t) = \int_{-\infty}^{\infty} f^*(\tau)g(t+\tau)\,d\tau, \tag{3.11}$$

where f^* denotes the complex conjugate of f. Cross-correlation measures the similarity between two signals as a function of the displacement of one relative to the other.

Unlike convolution, cross-correlation is generally neither commutative nor associative. However, there exists a useful relationship between convolution and cross-correlation:

$$(f \star g)(t) = (f^* * g)(-t),$$

which expresses cross-correlation as a time-reversed convolution involving the complex conjugate of f.

3.8.4 Note on Convolution vs. Cross-Correlation in Deep Learning

In the context of deep learning—particularly in **Convolutional Neural Networks (CNNs)**—the term "convolution" is commonly used, but what is actually implemented is more accurately described as **cross-correlation**. In standard mathematical convolution, the filter/kernel is flipped both spatially and temporally before being slid over the input. That is:

$$(f * g)(t) = \int f(\tau)g(t-\tau)\,d\tau.$$

However, in CNNs, this flipping is omitted. Instead, the operation performed is:

$$(f \circledast g)(t) = \int f(\tau)g(t+\tau)\,d\tau,$$

which is exactly the definition of cross-correlation when f is not conjugated.

This design choice simplifies implementation and training, as the kernel weights are learned directly without needing to account for any implicit flipping. Thus, while the operation is called "convolution", it behaves like cross-correlation.

Special Case: Convolution with Odd Functions

If h is an odd function, i.e., $h(-t) = -h(t)$, then the convolution of f with h yields:

$$(f * h)(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)\,d\tau.$$

This structure is commonly used in edge detection and gradient estimation in signal and image processing, where h approximates a derivative operator and f represents the input signal.

Chapter 4

Fourier Series and Fourier Transform

The Fourier series and Fourier transform are mathematical tools used to analyze functions in terms of sinusoidal components. These tools are widely applied in signal processing and image analysis. This document explores their relationship and their role in image processing, assuming familiarity with calculus, series expansion, and Euler's formula.

The development of Fourier analysis traces back to the early 19th century, primarily through the work of *Jean-Baptiste Joseph Fourier* (1768–1830), a French mathematician and physicist. Fourier introduced the concept of representing arbitrary periodic functions as sums of sine and cosine waves while studying heat conduction in solid bodies. In his seminal work *Théorie analytique de la chaleur* (1822), he demonstrated that even discontinuous functions could be expressed as infinite trigonometric series, a revolutionary idea that laid the foundation for modern harmonic analysis. The Fourier transform, a generalization of the Fourier series, later emerged as a powerful tool for analyzing non-periodic signals. The development of the Fast Fourier Transform (FFT) algorithm by **Cooley** and **Tukey** in 1965 made frequency-domain computations feasible on digital systems, enabling widespread use in science and engineering.

Fig. 4.1: Jean-Baptiste Joseph Fourier (1768–1830), a French mathematician and physicist. His idea that any function, even discontinuous ones, could be expanded into a trigonometric series revolutionized mathematics and physics.

These foundational ideas paved the way for both the Fourier series and its continuous counterpart, the Fourier transform, which we will explore in detail.

4.1 Fourier Series

A *Fourier series* is a powerful mathematical tool used to represent periodic functions as an infinite sum of sine and cosine functions. This method allows us to decompose complex waveforms into simpler oscillatory components, which can then be analyzed or synthesized individually.

4.1.1 Key Idea

A periodic function $f(x)$ with period T can be expressed as a Fourier series:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{2\pi n}{T} x\right) + b_n \sin\left(\frac{2\pi n}{T} x\right) \right]. \tag{4.1}$$

Here's the breakdown:

- a_0: The average value (or DC component) of $f(x)$, representing the baseline around which the function oscillates.
- $a_n \cos$ and $b_n \sin$: These terms capture the contributions of cosine and sine waves at specific frequencies:
  - $n = 1$: Fundamental frequency (matches the period T).
  - $n = 2$: Second harmonic (twice the fundamental frequency).
  - $n = 3$: Third harmonic (three times the fundamental frequency), and so on.
- Coefficients (a_n, b_n): These determine the amplitude of each sine or cosine wave.

This representation reveals how any sufficiently well-behaved periodic signal can be built from simple harmonic building blocks — a concept foundational to signal processing, acoustics, and image analysis.

4.1.2 Orthogonality and Completeness of Sine and Cosine Functions

One of the key mathematical principles behind the success of Fourier series is the **orthogonality and completeness** of sine and cosine functions over a given interval.

Two functions $f(x)$ and $g(x)$ are said to be orthogonal on the interval $[0, T]$ if their inner product is zero:

$$\int_0^T f(x)g(x)\,dx = 0.$$

For the set of functions $\left\{1, \cos\left(\frac{2\pi nx}{T}\right), \sin\left(\frac{2\pi nx}{T}\right)\right\}$, the following orthogonality relations hold:

$$\int_0^T \cos\left(\frac{2\pi nx}{T}\right)\cos\left(\frac{2\pi mx}{T}\right)dx = \begin{cases} T/2 & \text{if } n = m \neq 0 \\ T & \text{if } n = m = 0 \\ 0 & \text{if } n \neq m \end{cases} \tag{4.2}$$

$$\int_0^T \sin\left(\frac{2\pi nx}{T}\right)\sin\left(\frac{2\pi mx}{T}\right)dx = \begin{cases} T/2 & \text{if } n = m \\ 0 & \text{if } n \neq m \end{cases} \tag{4.3}$$

$$\int_0^T \cos\left(\frac{2\pi nx}{T}\right)\sin\left(\frac{2\pi mx}{T}\right)dx = 0 \quad \text{for all } n, m. \tag{4.4}$$

These relationships ensure that when computing the Fourier coefficients, each term in the series contributes independently.

Furthermore, the set of sine and cosine functions forms a **complete basis** for the space of square-integrable periodic functions. This means:

1. Any periodic function satisfying mild conditions (e.g., piecewise smoothness) can be represented exactly by a Fourier series.
2. The sine and cosine terms do not interfere with each other—they are independent, so we can calculate their contributions separately.

This combination of orthogonality and completeness is what makes the Fourier series both mathematically rigorous and practically useful in signal processing, physics, engineering, and many other fields.

4.1.3 Derivation of Fourier Series Coefficients

The coefficients a_0, a_n, and b_n in the Fourier series representation of a periodic function $f(x)$ are determined by exploiting the orthogonality of sine and cosine functions over one period $[0, T]$. Starting from the general form in Equation 4.1, we derive the expressions for each coefficient using inner products (integrals).

Finding a_0: DC or Average Value

To isolate a_0, we integrate both sides of the equation over one period $[0, T]$:

$$\int_0^T f(x)\,dx = \int_0^T a_0\,dx + \sum_{n=1}^{\infty}\left[a_n \int_0^T \cos\left(\frac{2\pi n}{T}x\right)\,dx + b_n \int_0^T \sin\left(\frac{2\pi n}{T}x\right)\,dx\right].$$

Since:

$$\int_0^T \cos\left(\frac{2\pi n}{T}x\right)\,dx = 0 \quad \text{and} \quad \int_0^T \sin\left(\frac{2\pi n}{T}x\right)\,dx = 0,$$

only the constant term remains:

$$\int_0^T f(x)\,dx = a_0 T \quad \Rightarrow \quad a_0 = \frac{1}{T}\int_0^T f(x)\,dx.$$

This gives the average value of $f(x)$ over one period.

Finding Cosine and Sine Coefficients

To find a_n, we multiply both sides of the Fourier series in Equation 4.1 by $\cos\left(\frac{2\pi m}{T}x\right)$ and integrate over $[0, T]$:

$$\int_0^T f(x)\cos\left(\frac{2\pi m}{T}x\right)\,dx = \int_0^T a_0\cos\left(\frac{2\pi m}{T}x\right)\,dx$$
$$+\sum_{n=1}^{\infty}\left[a_n \int_0^T \cos\left(\frac{2\pi n}{T}x\right)\cos\left(\frac{2\pi m}{T}x\right)\,dx + b_n \int_0^T \sin\left(\frac{2\pi n}{T}x\right)\cos\left(\frac{2\pi m}{T}x\right)\,dx\right].$$

According to the orthogonality relations in Equation 4.2, only the term where $n = m$ is non-zero:

$$\int_0^T f(x)\cos\left(\frac{2\pi m}{T}x\right)\,dx = a_m \cdot \frac{T}{2}.$$

So,

$$a_m = \frac{2}{T}\int_0^T f(x)\cos\left(\frac{2\pi m}{T}x\right)\,dx.$$

Similarly, multiplying both sides of the Fourier series by $\sin\left(\frac{2\pi m}{T}x\right)$ and integrating leads to:

$$b_m = \frac{2}{T}\int_0^T f(x)\sin\left(\frac{2\pi m}{T}x\right)\,dx.$$

Thus, the full set of formulas for the Fourier coefficients is:

$$a_0 = \frac{1}{T}\int_0^T f(x)\,dx \tag{4.5}$$

$$a_n = \frac{2}{T}\int_0^T f(x)\cos\left(\frac{2\pi n}{T}x\right)\,dx \tag{4.6}$$

$$b_n = \frac{2}{T}\int_0^T f(x)\sin\left(\frac{2\pi n}{T}x\right)\,dx \tag{4.7}$$

These integrals project the function $f(x)$ onto the orthogonal basis functions $\cos(2\pi nx/T)$ and $\sin(2\pi nx/T)$, effectively extracting the amplitude of each harmonic component.

4.1.4 Finite Summation of Sinusoidal Sequences

While Fourier series deal with continuous, infinite sums, many practical applications — such as digital signal processing and image compression — operate on **finite, discrete sequences**. In these settings, we often encounter sums of the form:

$$\sum_{k=0}^{n-1}\cos(a+kd) \quad \text{and} \quad \sum_{k=0}^{n-1}\sin(a+kd),$$

which represent uniformly sampled cosine and sine waves with phase offset a and angular step d.

Using Euler's formula (see Section 3.6) $e^{i\theta} = \cos\theta + i\sin\theta$, these sums can be evaluated exactly using complex exponentials and the geometric series formula. The results are:

$$\sum_{k=0}^{n-1}\cos(a+kd) = \frac{\sin\left(\frac{nd}{2}\right)\cos\left(a+\frac{n-1}{2}d\right)}{\sin\left(\frac{d}{2}\right)}, \tag{4.8}$$

$$\sum_{k=0}^{n-1}\sin(a+kd) = \frac{\sin\left(\frac{nd}{2}\right)\sin\left(a+\frac{n-1}{2}d\right)}{\sin\left(\frac{d}{2}\right)}. \tag{4.9}$$

The full derivation of these formulas relies on summing the geometric series $\sum_{k=0}^{n-1} z^k$ with $z = e^{id}$, then extracting real and imaginary parts. This derivation is provided in Appendix C.1.2 for completeness.

This transition from continuous integrals to finite sums paves the way for the discrete transforms that underpin modern digital media — particularly the Discrete Cosine Transform, which we now explore.

4.2 Fourier Transform

The Fourier Transform builds on ideas introduced by Joseph Fourier in his 1822 work *Théorie analytique de la chaleur*, where he proposed that any periodic function can be expressed as an infinite sum of sine

and cosine functions — now known as the Fourier Series. Though initially met with skepticism from mathematicians like Lagrange and Laplace, Fourier's insight became foundational to modern harmonic analysis.

His work inspired further exploration into how such frequency-based representations might apply not only to periodic signals, but also to general non-periodic functions.

While the Fourier Series represents periodic signals using discrete sine and cosine terms, the Fourier Transform extends this idea by expressing arbitrary signals — including non-periodic ones — as integrals over a continuous range of frequencies.

This extension arises naturally when considering the limit of a Fourier Series as the period $T \to \infty$. Intuitively, stretching the period indefinitely transforms a repeating signal into a non-repeating one. In this limit, the discrete frequency components merge into a continuum, and the summation becomes an integral.

4.2.1 Preliminary Knowledge

To fully appreciate the Fourier Transform, we first need to review some key mathematical concepts.

The Dirac Delta Function

The **Dirac delta function**, denoted as $\delta(t)$, is not a true function but rather a distribution (or generalized function). It is defined by:

- $\delta(t) = 0$ for all $t \neq 0$,
- $\int_{-\infty}^{\infty} \delta(t)dt = 1$.

A shifted version of the delta function, $\delta(t - t_0)$, is nonzero only at $t = t_0$. Its **sifting property** states:

$$\int_{-\infty}^{\infty} f(t)\delta(t - t_0)dt = f(t_0).$$

This property allows the delta function to "sift" the value of a function $f(t)$ at a specific point t_0.

Integral Representation of the Dirac Delta Function

An important result in Fourier analysis is that the integral of a complex exponential over all frequencies evaluates to a Dirac delta function:

$$\int_{-\infty}^{\infty} e^{i\omega(t-t')}d\omega = 2\pi\delta(t - t').$$

To understand why this is true, consider the behavior of the integral:

- When $t = t'$, the exponential becomes $e^{i\omega \cdot 0} = 1$, and the integral diverges, reflecting the infinite spike of the delta function at $t = t'$.
- When $t \neq t'$, the oscillatory nature of $e^{i\omega(t-t')}$ causes the integral to cancel out to zero due to rapid oscillations.

Formally, this result can be derived using a limiting process with a Gaussian damping factor or other regularization techniques (see C.2.1).

4.2.2 The Fourier Transform: Idea and Intuition

The **Fourier Transform** converts a signal from the time domain $f(t)$ into the frequency domain $F(\omega)$. The key idea is to decompose $f(t)$ into a superposition of complex exponentials with different frequencies.

The Fourier Transform is given by:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt.$$

Intuitively:

- The complex exponential $e^{-i\omega t}$ acts as a "probe" that isolates the contribution of each frequency ω in $f(t)$.
- The result $F(\omega)$ tells us how much of each frequency is present in the signal.

This formulation can be understood as a natural extension of the Fourier Series when we consider non-periodic functions. The Euler's formula:

$$e^{i\theta} = \cos\theta + i\sin\theta,$$

allows sine and cosine terms to be expressed in terms of complex exponentials. The Fourier Series equation 4.1 can be written in a more compact form as:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{2\pi n}{T}x}, \tag{4.10}$$

where the complex coefficients c_n are accordingly:

$$c_n = \frac{1}{T}\int_0^T f(t)e^{-i\frac{2\pi n}{T}t}dt. \tag{4.11}$$

The derivation is straightforward and omitted here.

Now, imagine taking the limit as the period $T \to \infty$. In this case, the function becomes non-periodic, and the discrete index n corresponds to a continuous variable ω (angular frequency). The spacing between successive frequency components shrinks to zero, and the sum becomes an integral.

This limiting process transforms the discrete Fourier Series into the continuous Fourier Transform. Specifically:

- The discrete frequencies $\omega_n = \frac{2\pi n}{T}$ become a continuous variable ω.
- The discrete coefficients c_n become a continuous function $F(\omega)$.
- The summation over discrete harmonics becomes an integral over all frequencies.

Thus, the Fourier Transform unifies the three distinct coefficient formulas of the Fourier Series into a single, continuous function $F(\omega)$, which describes the amplitude and phase of every frequency component in the signal — whether it is periodic or not.

The Inverse Fourier Transform: Reconstructing the Signal

Once a signal has been transformed into the frequency domain via the Fourier Transform $F(\omega)$, we often want to recover the original signal $f(t)$ in the time (or spatial) domain.

The Inverse Fourier Transform is defined as:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} d\omega.$$

This expression reconstructs the original function $f(t)$ by summing up all the frequency components $F(\omega)$, each scaled by its corresponding complex exponential $e^{i\omega t}$. Using Euler's formula:

$$e^{i\omega t} = \cos(\omega t) + i \sin(\omega t),$$

we see that this summation over all frequencies effectively recombines the cosine and sine contributions encoded in $F(\omega)$, recovering the original waveform.

It is worth noting that this reconstruction works precisely under certain conditions on $f(t)$, such as absolute integrability ($f \in L^1(\mathbb{R})$) and piecewise smoothness.

Together, the Fourier Transform and its inverse form a complete mathematical framework for analyzing and reconstructing signals in both the time and frequency domains.

The Fourier Transform is reversible because it is based on a one-to-one mapping between the time domain and the frequency domain. The complex exponentials $e^{i\omega t}$ form a complete and orthogonal basis, ensuring that no information is lost during the transformation.

To see this rigorously, we can derive the inverse Fourier Transform from the forward transform using properties of the Dirac delta function (see Appendix C.2.2).

This reversibility makes the Fourier Transform a cornerstone of modern signal processing, quantum mechanics, image processing, and many other fields where frequency analysis is essential.

4.3 Fourier Transform in Image Processing

In image processing, the 2D Fourier transform converts an image from the spatial domain to the frequency domain, revealing the contribution of different spatial frequencies.

4.3.1 2D Fourier Transform

For a 2D image $f(x, y)$, the Fourier transform is:

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i2\pi(ux+vy)} dx dy.$$

The inverse transform is:

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{i2\pi(ux+vy)} du dv.$$

For discrete images, the Discrete Fourier Transform (DFT) is used:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}.$$

4.3.2 Frequency Domain Interpretation

- **Low frequencies**: Smooth variations (e.g., large regions of uniform color).
- **High frequencies**: Rapid changes (e.g., edges, fine details, noise).

The magnitude spectrum $|F(u, v)|$ visualizes the strength of each frequency component.

Applications in Image Processing Frequency domain techniques are used in image processing for filtering, compression, edge detection, restoration, and pattern recognition. Low-pass filtering smooths images by removing high-frequency components, while high-pass filtering enhances edges. Compression methods like JPEG reduce file size by discarding less important high-frequency data. These techniques are also widely used in areas such as edge detection and image restoration.

4.4 Discrete Cosine Transform

The **Discrete Cosine Transform (DCT)** is a close relative of the Fourier Transform, widely used in signal processing, data compression, and image coding — most notably in JPEG image compression and MP3 audio compression. Unlike the Fourier Transform, which represents signals using complex exponentials (sines and cosines), the DCT expresses a finite sequence of real-valued data points as a sum of cosine functions oscillating at different frequencies. This makes it particularly effective for real-world signals such as images and audio, where neighboring samples are often highly correlated.

The DCT comes in several variants; the most commonly used one, especially in image compression standards like JPEG, is the **DCT-II**, defined as:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right], \quad k = 0, 1, \ldots, N-1,$$

where x_n is the input signal (e.g., pixel values) and X_k are the resulting transform coefficients.

Each coefficient X_k corresponds to a specific cosine basis function of frequency k. The first coefficient, X_0, is proportional to the average value of the input and is often referred to as the *DC component*; the remaining coefficients represent higher-frequency variations and are known as *AC components*. One key advantage of the DCT is its excellent **energy compaction property**: for most natural signals, especially images, the majority of the signal energy is concentrated in a few low-frequency coefficients.

This property is exploited in lossy compression schemes: high-frequency AC coefficients, which contribute less to perceived quality (due to limitations of human visual perception), can be quantized more coarsely or even discarded without significant visual degradation.

The inverse DCT (IDCT) reconstructs the original signal from its DCT coefficients. For DCT-II, the corresponding inverse transform is:

$$x_n = \sum_{k=0}^{N-1} C_k X_k \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right],$$

where C_k is a normalization factor:

$$C_k = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } k = 0 \\ \sqrt{\frac{2}{N}} & \text{otherwise} \end{cases}.$$

This reconstruction formula ensures perfect recovery of the original signal when no quantization or rounding errors are introduced.

The DCT can also be derived from the Discrete Fourier Transform (DFT) by imposing even symmetry on the input sequence. Specifically, if the input x_n is extended symmetrically about its endpoints, the resulting DFT becomes purely real, leading to a cosine-only representation—this yields the DCT.

Because of its computational efficiency and superior energy compaction for real-world data—especially signals with strong local correlations such as images—the DCT has become a cornerstone of modern digital media systems.

4.5 Application: The Mathematics Behind JPEG Compression

The *Joint Photographic Experts Group* (JPEG) format is one of the most widely used standards for digital image compression. At its core, JPEG leverages key mathematical ideas from harmonic analysis—particularly the **DCT**—to achieve high compression ratios while preserving perceptual quality.

4.5.1 Color Space Conversion and Subsampling

JPEG typically operates on color images represented in the RGB format. First, the image is converted to the **YCbCr color space**, where:

- Y: Luminance (brightness), capturing overall intensity.

- Cb, Cr: Chrominance (color difference) components.

Human vision is more sensitive to changes in brightness than in color. Therefore, after conversion, the chrominance channels (Cb and Cr) are often *downsampled* (e.g., by a factor of 2 in each dimension), reducing their spatial resolution without noticeable degradation. This step alone reduces data size significantly.

4.5.2 Block Division

Each channel (now possibly subsampled) is divided into 8×8 pixel blocks. If the image dimensions are not multiples of 8, edge blocks are padded. The DCT is applied independently to each block, making the process highly parallelizable and memory-efficient.

4.5.3 Forward 2D DCT

For each 8×8 block of pixel values $f(x, y)$, the **2D DCT** is computed:

$$F(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\left[\frac{\pi}{8}\left(x + \frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y + \frac{1}{2}\right)v\right],$$

where the normalization factors are:

$$\alpha(u) = \begin{cases} \frac{1}{\sqrt{8}} & \text{if } u = 0 \\ \sqrt{\frac{2}{8}} & \text{otherwise} \end{cases}, \quad \alpha(v) = \begin{cases} \frac{1}{\sqrt{8}} & \text{if } v = 0 \\ \sqrt{\frac{2}{8}} & \text{otherwise} \end{cases}.$$

The basis function for $u = 0$ is constant:

$$\cos\left[\frac{\pi}{8}\left(x + \frac{1}{2}\right) \cdot 0\right] = 1,$$

so,

$$\sum_{x=0}^{7} 1^2 = 8.$$

For $u > 0$, according to the general sinusoidal sum in Equation (4.8), we find:

$$\sum_{x=0}^{7} \cos^2\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right] = 4, \quad \text{for } u = 1, 2, \ldots, 7.$$

To ensure that all basis vectors have unit norm (i.e., the DCT is **orthonormal**), we must scale them appropriately: - For $u = 0$: scale by $\frac{1}{\sqrt{8}}$ so that $\left(\frac{1}{\sqrt{8}}\right)^2 \cdot 8 = 1$, - For $u > 0$: scale by $\sqrt{\frac{2}{8}} = \frac{1}{2}$ so that $\left(\sqrt{\frac{2}{8}}\right)^2 \cdot 4 = 1$.

Thus, the factors $\alpha(u)$ and $\alpha(v)$ are chosen to normalize the basis functions and preserve energy in the transform domain. This orthonormality guarantees that the inverse DCT perfectly reconstructs the original signal when no quantization is applied, and it simplifies the form of the IDCT by making it the transpose of the forward transform.

4.5.4 Quantization — The Lossy Step

This is where compression becomes *lossy*. Each DCT coefficient $F(u, v)$ is divided by a corresponding value $Q(u, v)$ from a **quantization matrix**, and rounded to the nearest integer:

$$F_Q(u, v) = \text{round}\left(\frac{F(u, v)}{Q(u, v)}\right).$$

Quantization matrices are designed based on human visual perception:

- Smaller values in the top-left (preserve low frequencies).

- Larger values in the bottom-right (discard high frequencies).

A typical luminance quantization matrix looks like:

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Note: This quantization matrix corresponds to a medium-quality setting (approximately 50% on common JPEG quality scales), as defined by the Independent JPEG Group (IJG)[1]. Higher-quality settings scale down these values to preserve more detail.

High-frequency coefficients often become zero after quantization, enabling efficient encoding.

4.5.5 Entropy Coding

Before final storage or transmission, the quantized coefficients undergo **entropy coding**:

[1] https://www.ijg.org/files/Wallace.JPEG.pdf

1. *Zigzag scanning*: Coefficients are read in a zigzag pattern starting from $(0, 0)$, grouping non-zero values early and creating long runs of zeros at the end.
2. *Run-Length Encoding (RLE)* (see Section 5.6.5): Sequences of zeros are encoded as (run_length, value) pairs.
3. *Huffman coding* (see Section 5.6.5): A variable-length prefix code assigns shorter codes to common patterns, minimizing bit usage.

The result is a compact bitstream that constitutes the compressed JPEG file.

4.5.6 Decoding: Reconstructing the Image

To decode a JPEG image:

1. Parse the bitstream using Huffman tables.
2. Undo RLE and reconstruct the quantized DCT coefficients.
3. Multiply by the quantization matrix: $\hat{F}(u, v) = F_Q(u, v) \times Q(u, v)$.
4. Apply the **Inverse 2D DCT (IDCT)** to recover each 8×8 block:

$$f(x, y) = \sum_{u=0}^{7} \sum_{v=0}^{7} \alpha(u)\alpha(v)\hat{F}(u, v) \cos\left[\frac{\pi}{8}\left(x + \frac{1}{2}\right)u\right] \cos\left[\frac{\pi}{8}\left(y + \frac{1}{2}\right)v\right].$$

5. Upsample chrominance channels and convert back to RGB.

The Discrete Cosine Transform (DCT) is highly effective for image compression due to its strong energy compaction, which concentrates image information into few low-frequency coefficients. It uses real-valued arithmetic for computational efficiency, is separable for row and column computations, and aligns with human visual perception by discarding less noticeable high-frequency details.

Chapter 5

Probability and Statistics

5.1 Probability

Probability theory is the mathematical framework for reasoning about uncertainty. In machine learning, probability plays a central role in modeling uncertainty in data, predictions, and decision-making processes. This chapter introduces key concepts in probability that are essential for understanding probabilistic models and algorithms used in machine learning.

The formal study of probability began in the 17th century with the work of mathematicians such as **Blaise Pascal** and **Pierre de Fermat**, who corresponded about problems related to gambling.

Later, **Jacob Bernoulli**'s *Ars Conjectandi* (1713) introduced the law of large numbers, a fundamental concept in probability and statistics. In the 19th century, **Pierre-Simon Laplace** further developed the classical definition of probability and applied it to scientific inference.

In the early 20th century, **Andrey Kolmogorov** provided a rigorous axiomatic formulation of probability theory using measure theory, which remains the standard foundation today. This mathematical formalism enabled the application of probability to diverse fields, including physics, finance, and, more recently, artificial intelligence and machine learning.

In 17th-century Europe, gambling was a widespread pastime among all social classes, including noblemen and scholars. Out of this fascination with games of chance arose some of the earliest mathematical inquiries into probability.

5.1.1 Sample Space and Events

An **experiment** is any process whose outcome cannot be predicted with certainty. The set of all possible outcomes of an experiment is called the **sample space**, denoted by Ω.

An **event** is a subset of the sample space, representing a collection of outcomes we are interested in. If the outcome of the experiment belongs to a given event, we say that the event has occurred.

For example, if we roll a six-sided die, the sample space is:

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

The event "rolling an even number" would be the subset $A = \{2, 4, 6\}$.

5.1.2 Basic Probability Axioms

Probability assigns a value between 0 and 1 to quantify the likelihood of an event occurring. Formally, a probability measure P satisfies the following axioms:

1. For any event A, $0 \leq P(A) \leq 1$
2. $P(\Omega) = 1$
3. For any countable sequence of disjoint events $A_1, A_2, \ldots$, the probability of their union is the sum of their probabilities:
$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$$

These axioms form the foundation of modern probability theory.

5.1.3 Conditional Probability

In many applications, we are interested in the probability of one event given that another has occurred. The conditional probability of B given A (with $P(A) > 0$) is defined as:

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

This formula allows us to update our beliefs about the likelihood of an event based on new information.

From the definition of conditional probability, we can derive the multiplication rule:

$$P(A \cap B) = P(A)P(B|A)$$

This rule is particularly useful when calculating the joint probability of two events.

If A and B are independent events, meaning that the occurrence of one does not affect the probability of the other, then:

$$P(B|A) = P(B)$$

and hence,

$$P(A \cap B) = P(A)P(B)$$

5.1.4 Combinatorics Preliminary

Combinatorics plays a crucial role in computing probabilities, especially in finite sample spaces where counting outcomes is necessary.

Basic Counting Principles

- **Multiplication Principle**: If there are m ways to do something and n ways to do another thing independently, then there are $m \times n$ ways to do both.

- **Addition Principle**: If there are m ways to do one thing and n ways to do another, and the two cannot happen simultaneously, then there are $m + n$ total ways.

Permutations and Combinations

Let n be a positive integer. Then:

$$n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1$$

is the number of ways to arrange n distinct items — this is called a **permutation**.

The number of ways to choose k elements from n distinct elements without regard to order is given by the **combination**:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Example: Drawing Cards

Suppose we draw 5 cards at random from a standard 52-card deck. What is the probability that all five cards are hearts?

There are $\binom{52}{5}$ total possible 5-card hands.

There are 13 hearts in a deck, so the number of favorable outcomes (all hearts) is $\binom{13}{5}$.

Thus, the probability is:

$$P(\text{All Hearts}) = \frac{\binom{13}{5}}{\binom{52}{5}} \approx 0.000495$$

This illustrates how combinatorial methods help compute probabilities in finite sample spaces.

5.1.5 Law of Total Probability

Let $A_1, A_2, \ldots, A_n$ be a partition of the sample space Ω, meaning the events are mutually exclusive and collectively exhaustive. Then, for any event B, the **Law of Total Probability** states:

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i) P(A_i)$$

This result is particularly useful when the direct computation of $P(B)$ is difficult, but it is easier to compute the conditional probabilities $P(B \mid A_i)$ for each partitioning event A_i. The law serves as a foundational tool in probabilistic modeling and Bayesian inference.

Pierre-Simon Laplace (1749–1827) was instrumental in developing the mathematical foundations of probability and formalized Bayesian reasoning.

5.1.6 Bayes' Theorem

Building upon the Law of Total Probability, **Bayes' Theorem** provides a principled way to reverse the conditioning in a conditional probability. Given a set of model parameters θ (or hypotheses A_i) and observed data $\mathcal{D}$, with $P(\mathcal{D}) > 0$, Bayes' Theorem allows us to compute the posterior probability:

$$P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta) P(\theta)}{P(\mathcal{D})}$$

Here:

- $P(\theta)$: The *prior* distribution — our belief about the parameters before observing any data.
- $P(\mathcal{D} \mid \theta)$: The *likelihood* — the probability of observing the data given the parameters.
- $P(\theta \mid \mathcal{D})$: The *posterior* distribution — our updated belief about the parameters after incorporating the evidence from the data.
- $P(\mathcal{D})$: The *evidence* or marginal likelihood — computed using the Law of Total Probability:

$$P(\mathcal{D}) = \sum_{i=1}^{n} P(\mathcal{D} \mid \theta_i) P(\theta_i)$$

  assuming a discrete set of parameter values or models $\theta_1, \theta_2, \ldots, \theta_n$.

In machine learning, Bayes' Theorem is central to probabilistic inference, especially in classification tasks such as Naive Bayes classifiers and in full Bayesian learning frameworks. It enables principled uncertainty quantification by combining prior knowledge with observed data through the likelihood function.

5.1.7 Independence

Two events A and B are said to be **independent** if:

$$P(A \cap B) = P(A)P(B)$$

This implies that knowing whether A has occurred gives no information about the likelihood of B, and vice versa.

Note that independence should not be confused with *disjointness*. Two disjoint events cannot occur simultaneously, but they are not necessarily independent unless at least one has zero probability.

5.1.8 Chain Rule of Probability

The chain rule generalizes the multiplication rule to more than two events. For a sequence of events $A_1, A_2, \ldots, A_n$, the joint probability can be written as:

$$P(A_1 \cap A_2 \cap \cdots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2) \ldots P(A_n|A_1 \cap \cdots \cap A_{n-1})$$

This decomposition is especially useful in probabilistic graphical models and sequential prediction problems.

5.2 Expectation and Variance

The variance of a random variable X is defined as:

$$\mathrm{Var}(X) = E[(X - \mu)^2] = E[X^2] - \mu^2$$

Important properties:

- $\mathrm{Var}(aX + b) = a^2 \mathrm{Var}(X)$, where a and b are constants.
- If X and Y are independent, $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$.

5.2.1 Chebyshev Inequality

Chebyshev's inequality provides a bound on the probability that a random variable deviates from its mean by more than a certain amount. It is independent of the underlying distribution.

Let X be a random variable with finite mean $\mu = \mathbb{E}[X]$ and finite variance $\sigma^2 = \text{Var}(X)$. For any $k > 0$,

$$P(|X - \mu| \geq \epsilon) \leq \frac{\sigma^2}{\epsilon^2}. \tag{5.1}$$

Proof for a continuous random variable:

$$\begin{aligned} P(|X - \mu| \geq \epsilon) &= \int_{|x-\mu|\geq\epsilon} f(x)\,dx \leq \int_{|x-\mu|\geq\epsilon} \frac{|x-\mu|^2}{\epsilon^2} f(x)\,dx \\ &\leq \frac{1}{\epsilon^2}\int_{-\infty}^{\infty} (x-\mu)^2 f(x)\,dx = \frac{\sigma^2}{\epsilon^2} \end{aligned}$$

This inequality can also be written as

$$P(|X - \mu| < \epsilon) \geq 1 - \frac{\sigma^2}{\epsilon^2}.$$

Chebyshev inequality is useful in estimating the bound of $P(|X - \mu|) < \epsilon$ in case the expectation and variance are known while the distribution is unknown.

inputStatistics/covariance

5.3 Distributions

Binomial Distribution

A Bernoulli trial has only two possible outcomes, success (1) and failure (0), with success probability p. Its distribution follows:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

The Binomial distribution is defined as:

$$X \sim \text{Binomial}(n, p)$$

According to the Central Limit Theorem, a binomial distribution Binomial(n, p) can be approximated by $N(np, np(1-p))$.

5.4 Normal distribution

Normal distribution has been briefly introduced in the previous chapter. It is the most important distribution in Computer vision and deep learning. Here we describe the history, properties and relevant proofs in more detail.

Fig. 5.1: Carl Friedrich Gauss (1777–1855), German mathematician and physicist, whose work on modeling measurement errors led to the formulation of the normal distribution, now often called the Gaussian distribution. He independently developed the method of least squares. His contributions underpin many areas of mathematics, statistics, physics, and engineering.

5.4.1 History

The development of the normal distribution is deeply tied to the study of probability, statistics, and error analysis.

The concept of randomness and probability has been studied since antiquity, but formal mathematical investigations began in the 16th and 17th centuries:

- Blaise Pascal and Pierre de Fermat laid the foundations of probability theory in the mid-1600s while solving problems related to games of chance.

- Later, Jacob Bernoulli introduced the *Law of Large Numbers* (published posthumously in 1713), which states that the average of repeated independent trials converges to the expected value as the number of trials increases.

Abraham de Moivre, a French mathematician, made a key contribution by studying the binomial distribution. He observed that for large numbers of trials, the binomial probabilities could be approximated using a smooth curve. In 1733, de Moivre derived a formula for this approximation, which we now recognize as the first appearance of the normal distribution. His work showed that the binomial distribution approaches a symmetric bell-shaped curve as the number of trials becomes large. Mathematically:

$$P(a \leq X \leq b) \approx \int_a^b \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx,$$

where X follows the standard normal distribution.

In 1774, Pierre-Simon Laplace introduced the concept of the *Central Limit Theorem* (CLT). The CLT explains why sums of independent random variables tend to follow a normal distribution, regardless of the original distributions of those variables.

Carl Friedrich Gauss played a pivotal role in popularizing the normal distribution:

- While working on astronomical data, Gauss developed the method of least squares to minimize errors in measurements.

- In 1809, Carl Friedrich Gauss demonstrated that if measurement errors followed a normal distribution, the least-squares method would yield the most probable estimates of unknown parameters.

The *Central Limit Theorem*, rigorously proven in the 19th and 20th centuries, cemented the normal distribution's importance. It explains why so many natural and social phenomena approximate the normal distribution when aggregated over many independent factors.

5.4.2 Preliminary Knowledge

The **Moment Generating Function (MGF)** is a powerful tool in probability theory that helps characterize the distribution of a random variable. The MGF of a random variable X is defined as:

$$M_X(t) = \mathbb{E}[e^{tX}],$$

where t is a real parameter, and $\mathbb{E}[\cdot]$ denotes the expectation. The MGF is particularly useful because it uniquely determines the distribution of X. That is, if two random variables have the same MGF, they must follow the same probability distribution.

Properties of MGFs

Moment generating functions are powerful tools in probability theory because they encode all the moments of a random variable in a compact, algebraic form. Here are some key properties:

- **Linearity of Expectation and Moment Generation:**
  The MGF of a random variable X, defined as

  $$M_X(t) = \mathbb{E}[e^{tX}],$$

  encodes all the moments of X — provided the expectation exists for values of t in an open interval around zero. This is one of the most important features of MGFs:
  - The **first derivative** of $M_X(t)$ evaluated at $t = 0$ gives the **mean** (or expected value) of X:

    $$\mathbb{E}[X] = M_X'(0).$$

  - The **second derivative** evaluated at $t = 0$ gives the **second raw moment**, which is $\mathbb{E}[X^2]$:

    $$\mathbb{E}[X^2] = M_X''(0).$$

  - In general, the n-th derivative of $M_X(t)$ evaluated at $t = 0$ gives the n-th raw moment:

    $$\mathbb{E}[X^n] = M_X^{(n)}(0).$$

  This means that if we can compute or recognize the MGF of a distribution, we can easily extract its moments through differentiation.

- **Independence and Sums of Random Variables:**
  If X and Y are independent, then the MGF of their sum $Z = X + Y$ is simply the product of their individual MGFs:

  $$M_Z(t) = M_X(t) \cdot M_Y(t). \tag{5.2}$$

MGF of a Normal Distribution

The normal distribution is one of the most widely used probability distributions. A random variable X is said to follow a normal distribution with mean μ and variance σ^2, written as $X \sim \mathcal{N}(\mu, \sigma^2)$, if its probability density function (PDF) is:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

The MGF of $X \sim \mathcal{N}(\mu, \sigma^2)$ is given by:

$$M_X(t) = e^{\mu t + \frac{1}{2}\sigma^2 t^2}.$$

The derivation of the MGF for a Normal Distribution is shown in Appendix D.2.2.

5.4.3 Sum of Two Independent Normal Distributions

Let $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$ be independent normal random variables. We aim to show that their sum $Z = X + Y$ is also normally distributed.

Using the independence of X and Y (Equation 5.2), the MGF of $Z = X + Y$ is:

$$\begin{aligned} M_Z(t) &= e^{\mu_1 t + \frac{1}{2}\sigma_1^2 t^2} \cdot e^{\mu_2 t + \frac{1}{2}\sigma_2^2 t^2} \\ &= e^{(\mu_1+\mu_2)t + \frac{1}{2}(\sigma_1^2+\sigma_2^2)t^2} \end{aligned}$$

Thus, $Z \sim \mathcal{N}(\mu_1 + \mu_2,\ \sigma_1^2 + \sigma_2^2)$.

This result generalizes to any finite sum of independent normal random variables. If $Z = X_1 + X_2 + \cdots + X_n$, where each $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, then:

$$Z \sim \mathcal{N}\left(\sum_{i=1}^{n} \mu_i, \sum_{i=1}^{n} \sigma_i^2\right).$$

5.5 Central Limit Theorem

The Central Limit Theorem (CLT) is one of the most fundamental results in probability theory. It asserts that the sum (or average) of a large number of independent and identically distributed (i.i.d.) random variables, when appropriately normalized, converges in distribution to a standard normal random variable. Below, we state the theorem and provide a proof using moment-generating functions (MGFs).

5.5.1 Statement of the Central Limit Theorem

Let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables with:

- Mean: $\mu = \mathbb{E}[X_i]$,

- Variance: $\sigma^2 = \mathrm{Var}(X_i) < \infty$.

Define the standardized sum:

$$Z_n = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}}.$$

The Central Limit Theorem states that as $n \to \infty$, Z_n converges in distribution to a standard normal random variable $Z \sim N(0, 1)$. Symbolically:

$$Z_n \xrightarrow{d} N(0, 1).$$

See Appendix D.3 for a proof.

5.6 Information Theory

In machine learning, measuring uncertainty and comparing probability distributions are essential tasks. Information theory offers a rigorous framework for these purposes, with fundamental concepts such as *entropy* and *Kullback–Leibler (KL) divergence* playing key roles in probabilistic modeling, Bayesian inference, and optimization.

5.6.1 Entropy: Measuring Uncertainty

Entropy quantifies the uncertainty associated with a random variable. It was introduced by **Claude Shannon** in 1948 as a cornerstone of his mathematical theory of communication. The concept was inspired by thermodynamics, where entropy measures disorder or randomness in physical systems.

Intuitively, the information content of an event increases as its probability decreases. If an event is almost certain, it conveys little new information; if it is rare, it conveys more. For independent events, the joint probability is the product of individual probabilities, but the total information should be additive. This motivates the use of a logarithmic measure of information:

$$h(x) = -\log p(x)$$

where the base of the logarithm determines the unit of information:

- Base 2 $\rightarrow$ measured in *bits* (binary digits), commonly used in computer science.
- Base e $\rightarrow$ measured in *nats*, often preferred in theoretical mathematics due to its analytical properties.

The expected value of this information over all outcomes defines the **Shannon entropy** of a discrete random variable X:

$$H(X) = -\sum_x p(x) \log p(x) \tag{5.3}$$

For continuous random variables, we define the **differential entropy** using a probability density function $p(x)$:

$$H(X) = -\int_{-\infty}^{\infty} p(x) \log p(x)\, dx \tag{5.4}$$

Note that differential entropy does not share all properties of discrete entropy (e.g., it can be negative), but remains a useful extension in many applications.

Key insights about entropy:

- Higher entropy corresponds to greater uncertainty or randomness.
- Entropy is maximized when all outcomes are equally probable — that is, under a uniform distribution.

5.6.2 Cross-Entropy

When comparing two probability distributions p and q, cross-entropy measures how well q models p. Specifically, it represents the average number of bits needed to encode data from distribution p using a code optimized for distribution q.

The cross-entropy between two discrete distributions is defined as:

$$H(p, q) = -\sum_x p(x) \log q(x) \quad \text{(discrete)} \tag{5.5}$$

$$H(p, q) = -\int_{-\infty}^{\infty} p(x) \log q(x)\, dx \quad \text{(continuous)} \tag{5.6}$$

If $p = q$, the cross-entropy reduces to the entropy of p, i.e., $H(p, p) = H(p)$. Cross-entropy is widely used in machine learning as a loss function in classification tasks.

5.6.3 Kullback–Leibler (KL) Divergence

The **Kullback–Leibler (KL) divergence** quantifies the information lost when one distribution q is used to approximate another distribution p. Introduced by **Solomon Kullback** and **Richard Leibler** in 1951, it provides a directional measure of dissimilarity between probability distributions.

To derive KL divergence, consider the difference in expected log-probabilities under p and q:

$$D_{\mathrm{KL}}(p \parallel q) = -\int_{-\infty}^{\infty} p(x) \log q(x)\, dx + \int_{-\infty}^{\infty} p(x) \log p(x)\, dx = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx$$

This leads to the general definitions:

$$D_{\mathrm{KL}}(p \parallel q) = \sum_{x} p(x) \log \frac{p(x)}{q(x)} \quad \text{(discrete)} \tag{5.7}$$

$$D_{\mathrm{KL}}(p \parallel q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \quad \text{(continuous)} \tag{5.8}$$

Alternatively, KL divergence can be expressed in terms of entropy and cross-entropy:

$$D_{\mathrm{KL}}(p \parallel q) = H(p, q) - H(p)$$

This formulation shows that KL divergence captures the additional information required to represent p using q instead of p itself.

Important properties of KL divergence:

- **Non-negative**: $D_{\mathrm{KL}}(p \parallel q) \geq 0$, with equality if and only if $p = q$ almost everywhere. This follows from Jensen's inequality (see Appendix D.4.1) and reflects the fact that no coding scheme can be more efficient than one matched to the true distribution.
- **Not symmetric**: $D_{\mathrm{KL}}(p \parallel q) \neq D_{\mathrm{KL}}(q \parallel p)$. This asymmetry makes KL divergence sensitive to the role each distribution plays — whether it is the "true" one or the "approximate" one.
- **Does not satisfy the triangle inequality**: For three distributions p, q, and r, it is not guaranteed that
  $$D_{\mathrm{KL}}(p \parallel r) \leq D_{\mathrm{KL}}(p \parallel q) + D_{\mathrm{KL}}(q \parallel r).$$
  In fact, the left-hand side can exceed the right-hand side, violating a fundamental requirement for any function to be considered a mathematical *metric*.

Because KL divergence lacks symmetry and fails the triangle inequality, it is not a true distance in the geometric sense — hence the use of the term **divergence** rather than *distance*. A true metric (such as Euclidean distance) must satisfy all three conditions: non-negativity, symmetry and triangle inequality. KL divergence satisfies only the first.

To understand this intuitively, consider an analogy with one-way toll roads: the "cost" of going from distribution p to q might be low (e.g., encoding a slightly noisy version of a clean signal), but returning

from q to p could require much more information. Similarly, taking a detour via another distribution might surprisingly be cheaper than going directly — something impossible under standard geometry.

Despite its asymmetry and failure to form a metric space, KL divergence is exceptionally valuable for capturing directional differences between distributions. For example, approximating a unimodal Gaussian with a bimodal distribution leads to very different consequences (in terms of missed modes or misplaced probability mass) compared to the reverse scenario. This directionality is crucial in applications such as variational inference, where q is optimized to approximate p, and the cost of missing tails or splitting mass has real implications.

See the proof of the non-negativity of KL divergence in Appendix D.4.2.

5.6.4 Example: KL Divergence Between Two Gaussians

Let $p(x) = \mathcal{N}(\mu_1, \sigma_1^2)$ and $q(x) = \mathcal{N}(\mu_2, \sigma_2^2)$. The Kullback–Leibler (KL) divergence from p to q is given by the closed-form expression:

$$D_{\mathrm{KL}}(p \parallel q) = \frac{1}{2}\left(\frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{\sigma_2^2} - 1 + \log\left(\frac{\sigma_2^2}{\sigma_1^2}\right)\right)$$

See the derivation in Appendix D.4.3

This result is one of the few analytically tractable cases of KL divergence, making it invaluable for both theoretical analysis and practical applications. But beyond its computational convenience, this formula offers deep insight into how KL divergence behaves as a measure of dissimilarity between probability distributions.

Breaking down the components: The expression can be interpreted as a sum of three intuitive contributions:

- **Mean mismatch penalty:** $\dfrac{(\mu_1 - \mu_2)^2}{2\sigma_2^2}$ — this term quantifies how far the mean of p is from the mean of q, scaled by the variance of q. If q is sharply peaked (small σ_2^2), even a small shift in the mean leads to a large divergence, since p places significant probability mass where q assigns very low density.
- **Variance ratio penalty:** $\dfrac{\sigma_1^2}{2\sigma_2^2} - \dfrac{1}{2}$ — this captures the mismatch in spread. If $\sigma_1 > \sigma_2$, then p is more diffuse than q, meaning p assigns non-negligible probability to regions where q is nearly zero — a costly error under KL divergence.

Together, these terms reveal that KL divergence is particularly sensitive to how well q covers the regions where p has high probability — a key idea we'll explore further below.

Non-negativity in action: It can be shown (and numerically verified) that $D_{\mathrm{KL}}(p \parallel q) \geq 0$ for all parameter values, with equality *if and only if* $\mu_1 = \mu_2$ and $\sigma_1^2 = \sigma_2^2$. This aligns with the general property of KL divergence: it vanishes precisely when the two distributions are identical.

For example:

- If $\mu_1 = \mu_2$, $\sigma_1^2 = \sigma_2^2$, then $D_{\mathrm{KL}} = 0$.

- If $\mu_1 \neq \mu_2$ or $\sigma_1^2 \neq \sigma_2^2$, the divergence is strictly positive — reflecting information loss when using q to approximate p.

This reinforces the interpretation of KL divergence as a measure of *information inefficiency* or *model misspecification cost*.

Asymmetry: direction matters. KL divergence is not symmetric: $D_{\mathrm{KL}}(p \parallel q) \neq D_{\mathrm{KL}}(q \parallel p)$. The Gaussian case makes this strikingly clear.

Consider two scenarios:

1. p: wide Gaussian ($\sigma_1 = 2$), q: narrow Gaussian centered at the same mean ($\sigma_2 = 1$).
2. Reverse: p: narrow ($\sigma_1 = 1$), q: wide ($\sigma_2 = 2$).

In case (1), $D_{\mathrm{KL}}(p \parallel q)$ becomes *very large*, because q assigns extremely low probability to the tails of p. Encoding samples from the wide p using a code optimized for the narrow q would require many extra bits.

In case (2), $D_{\mathrm{KL}}(p \parallel q)$ is much smaller: the wide q covers the narrow p well, so encoding is efficient, even if q spreads probability too thinly.

This illustrates that KL divergence is highly sensitive to *underestimation of uncertainty*. Approximating a broad, uncertain distribution with an overconfident one incurs a high cost. Conversely, being overly conservative (wide q) is less penalized.

5.6.5 Source Coding and the Role of Redundancy

One of the foundational results in information theory is **Shannon's source coding theorem**, which establishes that the entropy $H(X)$ of a discrete memoryless source sets a fundamental lower bound on the average number of bits per symbol required to represent the source without loss.

Formally, no lossless compression scheme can compress data below $H(X)$ bits per symbol on average. Conversely, there exist codes that can approach this limit arbitrarily closely as block length increases.

However, real-world data often contains *redundancy* — predictable structure or repeated patterns — that allows practical compression schemes to reduce bit usage even when operating far from asymptotic limits. Such redundancies manifest in two primary forms:

- **Statistical redundancy**: Some symbols occur more frequently than others.
- **Structural redundancy**: Symbols are correlated across time or space (e.g., long runs of identical values).

Exploiting these redundancies lies at the heart of efficient encoding strategies. We now examine two classic methods—Run-Length Encoding and Huffman coding—that operationalize information-theoretic ideas in complementary ways.

Run-Length Encoding (RLE)

Run-Length Encoding is a simple yet effective technique designed to exploit structural redundancy. It replaces sequences of repeated symbols (called "runs") with pairs consisting of the symbol and the number of repetitions.

For example, the sequence:

$$a, a, a, b, b, c, c, c, c$$

is encoded as:

$$(a, 3),\ (b, 2),\ (c, 4).$$

RLE achieves significant compression when the input contains long runs of identical elements. Its efficiency depends directly on the degree of correlation or predictability in the data stream — a property indirectly captured by entropy: low-entropy sources tend to produce longer runs.

While RLE does not require knowledge of the full probability distribution, it implicitly benefits from high skew in local symbol frequencies. Though suboptimal compared to entropy-limited codes, its simplicity makes it suitable for fast preprocessing stages or hardware-constrained environments.

Huffman Coding: Optimal Prefix Codes

Huffman coding is a method for constructing minimum-redundancy prefix codes based on symbol probabilities. A *prefix code* ensures that no codeword is a prefix of another, enabling unambiguous decoding without delimiters.

Given a discrete source with known symbol probabilities $p_1, p_2, \ldots, p_n$, Huffman's algorithm builds a binary tree through successive merging of the least probable symbols. The resulting codewords correspond to paths from root to leaf, assigning shorter codes to more probable symbols.

Let l_i denote the length of the codeword assigned to symbol i. Then the expected code length is:

$$L = \sum_i p_i l_i.$$

It can be shown that Huffman coding satisfies:

$$H(X) \leq L < H(X) + 1,$$

meaning the average number of bits per symbol falls within one bit of the entropy lower bound.

Thus, Huffman coding realizes a constructive solution to the problem posed by Shannon's source coding theorem: it approaches the theoretical limit using finite-length, uniquely decodable codes.

Moreover, the gap between L and $H(X)$ reflects the inefficiency incurred due to integer constraint on codeword lengths. When symbol probabilities are inverse powers of two (i.e., $p_i = 2^{-k_i}$), Huffman coding achieves $L = H(X)$ exactly.

This connection underscores a deep principle: *efficient representation arises from aligning code design with the underlying probability structure of the data.*

Connection to KL Divergence Suppose we use a code optimized for distribution q to encode data drawn from p. The excess expected code length (over the entropy $H(p)$) is precisely the KL divergence:

$$\mathbb{E}[l] - H(p) = D_{\mathrm{KL}}(p \parallel q).$$

This elegant identity links coding efficiency to probabilistic modeling: misestimating the true distribution incurs a measurable cost in bits.

5.7 Markov Chains and Kalman Filters: Mathematical Foundations

The two probabilistic models, *Markov chains* and the *Kalman filter*, are particularly useful when dealing with sequences of random variables and dynamic systems evolving over time.

5.7.1 Markov Chains: A Probabilistic Sequence Model

A Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. This property is known as the *Markov property*, or memoryless property. The concept was introduced by Russian mathematician **Andrey Markov** in the early 20th century.

Let $\{X_t\}_{t=0}^{\infty}$ be a sequence of discrete random variables representing the state of a system at time t. The process is said to satisfy the first-order Markov property if:

$$P(X_{t+1} = x_{t+1} \mid X_0 = x_0, X_1 = x_1, \ldots, X_t = x_t) = P(X_{t+1} = x_{t+1} \mid X_t = x_t)$$

This conditional independence greatly simplifies modeling and computation. The transition probabilities between states can be represented using a *transition matrix* P, where each entry P_{ij} denotes the probability of transitioning from state i to state j:

$$P_{ij} = P(X_{t+1} = j \mid X_t = i)$$

For time-homogeneous Markov chains, the transition probabilities do not change over time. If the chain has a finite number of states, the entire dynamics can be compactly modeled using the initial distribution vector π_0 and the transition matrix P.

An important concept in Markov chains is the *stationary distribution*, denoted by π, which satisfies:

$$\pi = \pi P$$

If such a distribution exists and the chain is ergodic (irreducible and aperiodic), then regardless of the starting state, the distribution over states converges to π as time progresses.

Markov chains are foundational to more complex models like Hidden Markov Models (HMMs) and are widely used in natural language processing, reinforcement learning, and Bayesian sampling techniques like Markov Chain Monte Carlo (MCMC).

5.7.2 Kalman Filter: Optimal Estimation in Linear Gaussian Systems

The Kalman filter is a recursive algorithm used to estimate the state of a dynamic system from a series of noisy observations. It was developed by **Rudolf E. Kalman** in the early 1960s. In computer vision, Kalman filters are commonly employed in object tracking systems to predict and update the position and velocity of moving objects across video frames.

We assume the system evolves according to the following linear Gaussian model:

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{w}_t$$
$$\mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t$$

where:

- $\mathbf{x}_t$ is the hidden state vector at time t,
- $\mathbf{z}_t$ is the observation vector,
- $\mathbf{u}_t$ is the control input (optional),
- $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q}_t)$ is the process noise,
- $\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{R}_t)$ is the observation noise,
- $\mathbf{F}_t$, $\mathbf{B}_t$, and $\mathbf{H}_t$ are known matrices.

The Kalman filter maintains a posterior belief about the state $\mathbf{x}_t$ given all observations up to time t, represented by the mean $\hat{\mathbf{x}}_{t|t}$ and covariance $\mathbf{P}_{t|t}$. The algorithm consists of two main steps:

Prediction Step: Predict the next state based on the previous estimate.

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t \mathbf{u}_t$$
$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^\top + \mathbf{Q}_t$$

Update Step: Incorporate the new observation to refine the estimate.

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t^\top (\mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^\top + \mathbf{R}_t)^{-1}$$
$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1})$$
$$\mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_{t|t-1}$$

Here, $\mathbf{K}_t$ is called the *Kalman gain*, which determines how much weight to give to the new measurement versus the prediction.

While the standard Kalman filter assumes linearity and Gaussian noise, extensions such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) handle nonlinear systems. These are commonly used in robotics, navigation, and sensor fusion applications.

Although detailed implementation and tuning of Kalman filters fall under the domain of signal processing and control systems, understanding their probabilistic foundation helps machine learning practitioners apply them effectively in real-world scenarios involving sequential data.

Chapter 6

Numerical Analysis in Foundations

Machine learning and computer vision systems rely heavily on numerical computation. From training deep neural networks to processing images and videos, nearly all operations are performed using finite-precision arithmetic on digital computers. While high-level frameworks like PyTorch and TensorFlow abstract away many low-level details, understanding the underlying numerical behavior is crucial for building robust, stable, and efficient models.

This chapter introduces key concepts from numerical analysis that are especially relevant to machine learning and computer vision. We begin by examining how real numbers are represented in computers and explore the limitations and implications of floating-point arithmetic. Next, we cover essential topics such as numerical differentiation, error propagation, matrix conditioning, and interpolation—each with practical applications in ML and CV tasks.

By grounding our understanding in numerical methods, we equip ourselves to:

- Diagnose and prevent numerical instability in model training
- Optimize performance-critical code
- Understand the behavior of optimization algorithms
- Improve the robustness of image processing pipelines

This chapter assumes familiarity with basic linear algebra, calculus, and Python programming. Throughout, we emphasize practical examples and connections to real-world ML and CV problems.

6.1 Floating Point Arithmetic

The floating-point representation most commonly follow the IEEE 754 standard[1] .

A floating-point number is typically represented in the form:

$$x = (-1)^s \times M \times 2^{E-\text{bias}}$$

where:

- s is the sign bit (0 or 1)
- M is the significand (or mantissa), which encodes the precision
- E is the exponent, which determines the scale
- bias is an offset used to allow negative exponents

In the widely-used 32-bit single-precision (float32) format:

- 1 bit for sign

[1] https://en.wikipedia.org/wiki/IEEE_754

- 8 bits for the exponent
- 23 bits for the significand

6.1.1 Precision and Rounding Errors

Because only a finite number of bits are available, not all real numbers can be represented exactly. For example, the decimal value 0.1 has a repeating binary fraction and cannot be stored precisely in binary floating point.

This leads to **rounding errors**, which can accumulate during computations. Common pitfalls include:

- Loss of associativity: $(a + b) + c \neq a + (b + c)$
- Catastrophic cancellation: subtracting two nearly equal large numbers yields a small result with very low relative accuracy

Floating-point limitations can lead to problems such as overflow or underflow in functions like softmax, instability in gradient calculations, reduced performance due to poor numerical conditioning, and unexpected results during image normalization or thresholding. To address these issues, common strategies include adding small constants to denominators, using log-space computations such as the log-sum-exp trick, clipping gradients during optimization, and selecting appropriate floating-point types like float32 or float64 based on the required precision.

Example: Softmax Function

The softmax function is commonly used in classification tasks:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Naively computing it can lead to overflow when z_i is large. To stabilize it, we subtract the maximum value before exponentiation:

$$\text{softmax}(z_i) = \frac{e^{z_i - c}}{\sum_j e^{z_j - c}}, \quad \text{where } c = \max(z)$$

This ensures numerical stability while preserving correctness.

6.1.2 Common Floating Point Formats

The three most common floating point formats are `float32` (IEEE 754 single-precision), `float16` (IEEE 754 half-precision) and `bfloat16` (Brain Floating Point).

Each format offers a unique trade-off between precision and range.

Property	float32	float16	bfloat16
Total Bits	32	16	16
Sign Bits	1	1	1
Exponent Bits	8	5	8
Mantissa Bits	23	10	7
Approx. Decimal Digits	7–8	3–4	2–3
Max Value	3.4×10^{38}	6.5×10^{4}	3.4×10^{38}
Min Positive	1.2×10^{-38}	6.1×10^{-5}	1.2×10^{-38}

float32: The default in many general-purpose numerical libraries (e.g., NumPy), scientific simulations, and machine learning frameworks.

float16: Ideal for applications where speed and efficiency are prioritized over high precision.

bfloat16: Originally developed for AI accelerators, particularly useful in iterative numerical algorithms (e.g., gradient descent) and hardware-constrained environments.

6.2 Numerical Differentiation

In machine learning and computer vision, gradients are essential for optimization, model interpretation, and sensitivity analysis. While automatic differentiation (AD) is widely used in modern deep learning frameworks like PyTorch and TensorFlow, understanding **numerical differentiation**—the approximation of derivatives using finite differences—is still valuable for debugging, testing, and implementing custom operations.

The derivative of a function $f(x)$ at a point x is defined as:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

However, on digital computers, we cannot take the limit as $h \to 0$ due to the limitations of floating-point arithmetic. Instead, we approximate the derivative using small but finite values of h.

6.2.1 Finite Difference Methods

There are three commonly used finite difference approximations:

Forward Difference:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

This method has an error of $O(h)$, meaning the error decreases linearly with h.

Backward Difference:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

Similar to forward difference, it also has an error of $O(h)$.

Central Difference:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

This method is more accurate, with an error of $O(h^2)$, making it preferable when higher precision is needed.

6.2.2 Choosing the Step Size h

Selecting an appropriate step size h is critical. If h is too large, the approximation may be inaccurate due to truncation error. If h is too small, rounding errors dominate because $f(x+h)$ and $f(x)$ become numerically indistinguishable.

A commonly used heuristic is:

$$h = \epsilon^{1/2} \cdot \max(|x|, 1)$$

where ϵ is the machine epsilon for the floating-point type being used (e.g., approximately 1.2×10^{-7} for float32).

6.2.3 Higher-Order Derivatives and Multivariate Gradients

For second-order derivatives or multivariate functions, finite differences can be extended accordingly.

For example, the second derivative can be approximated by:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

For a multivariate function $f(\mathbf{x})$, the gradient vector ∇f can be estimated by perturbing each input dimension independently:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(\mathbf{x} + h \cdot \mathbf{e}_i) - f(\mathbf{x} - h \cdot \mathbf{e}_i)}{2h}$$

where $\mathbf{e}_i$ is the unit vector in the i-th direction.

6.3 Numerical Linear Algebra

This section focuses on two key topics—**matrix conditioning** and the **singular value decomposition (SVD)**.

6.3.1 Matrix Conditioning and Stability

The **condition number** of a matrix quantifies how sensitive the solution of a system of linear equations is to small changes in the input data. It plays a central role in determining whether a problem is *well-conditioned* or *ill-conditioned*, which directly affects the reliability and accuracy of numerical computations.

For a square invertible matrix A, the condition number with respect to inversion is defined as:

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$

where $\|\cdot\|$ is a matrix norm (e.g., the spectral norm or Frobenius norm). When using the spectral norm, the condition number becomes:

$$\kappa(A) = \frac{\sigma_{\max}}{\sigma_{\min}}$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest singular values of A, respectively.

- A **well-conditioned** matrix has a low condition number ($\kappa(A) \approx 1$), meaning small perturbations in the input result in small changes in the output.

- An **ill-conditioned** matrix has a large condition number, indicating that small errors in the input can lead to large errors in the solution.

In machine learning, ill-conditioning often arises in tasks such as:

- Solving linear regression problems via normal equations
- Inverting covariance matrices
- Training deep networks with poorly scaled inputs

An ill-conditioned system may lead to numerical instability, poor convergence, or unreliable results. Regularization techniques like Tikhonov regularization (ridge regression) are commonly used to improve conditioning by adding a small positive constant to the diagonal of $A^T A$, thereby increasing $\sigma_{\min}$.

6.3.2 Singular Value Decomposition (SVD)

The **singular value decomposition (SVD)** is one of the most powerful tools in linear algebra, particularly useful in machine learning and computer vision for tasks such as dimensionality reduction, noise filtering, and feature extraction.

Any real $m \times n$ matrix A can be decomposed as:

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

where:

- U is an $m \times m$ orthogonal matrix whose columns are the left singular vectors
- Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal—the *singular values*
- V is an $n \times n$ orthogonal matrix whose columns are the right singular vectors

The singular values σ_i represent the magnitude of each component in the direction of the corresponding singular vectors. They are related to the eigenvalues λ_i of $A^T A$ via $\sigma_i = \sqrt{\lambda_i}$.

Numerical Considerations

Computing the SVD of large matrices can be computationally expensive, but modern implementations (e.g., in NumPy, SciPy, and PyTorch) use optimized algorithms like divide-and-conquer or iterative methods to handle large-scale problems efficiently.

It's also important to monitor the singular values to detect near-rank deficiency or numerical instability. A large disparity between the largest and smallest singular values indicates high sensitivity to perturbations, which can affect generalization performance or lead to unstable solutions.

6.3.3 Example: PCA Using SVD

Principal Component Analysis (PCA) can be implemented using SVD as follows:

Given a zero-centered data matrix $X \in \mathbb{R}^{n \times d}$, where each row is a sample:

1. Compute the SVD: $X = U\Sigma V^T$
2. The principal components are the columns of V
3. Project the data onto the first k components: $X_{\text{reduced}} = XV_k$, where $V_k \in \mathbb{R}^{d \times k}$ contains the first k right singular vectors

This approach avoids explicitly computing the covariance matrix $X^T X$, making it more numerically stable than eigendecomposition-based PCA.

Appendix **A**

Calculus Derivations

A.1 Derivations for Common Derivatives

This section provides detailed derivations for fundamental rules in calculus.

A.1.1 The Product Rule

The product rule states that if $f(x)$ and $g(x)$ are differentiable functions, then the derivative of their product is given by:

$$(fg)'(x) = f'(x)g(x) + f(x)g'(x).$$

Proof:

$$\begin{aligned}
(fg)'(x) &= \lim_{h\to 0} \frac{f(x+h)g(x+h) - f(x)g(x)}{h} \\
&= \lim_{h\to 0} \frac{f(x+h)g(x+h) - f(x)g(x+h) + f(x)g(x+h) - f(x)g(x)}{h} \\
&= \lim_{h\to 0} \left[\frac{f(x+h) - f(x)}{h} g(x+h) + f(x)\frac{g(x+h) - g(x)}{h}\right] \\
&= f'(x)g(x) + f(x)g'(x)
\end{aligned}$$

A.1.2 The Chain Rule

The chain rule states that if $f(u)$ is differentiable at $u = g(x)$ and $g(x)$ is differentiable at x, then the composite function $f(g(x))$ is differentiable at x:

$$\frac{d}{dx}[f(g(x))] = f'(g(x))g'(x).$$

To prove this, we define an error function $E(k)$:

$$E(k) = \begin{cases} 0, & \text{if } k = 0, \\ \frac{f(u+k)-f(u)}{k} - f'(u), & \text{if } k \neq 0. \end{cases}$$

Using this definition, we have:

$$f(u+k) - f(u) = k(E(k) + f'(u)).$$

Now, let $u = g(x)$ and $k = g(x+h) - g(x)$.

$$f(g(x+h)) - f(g(x)) = (g(x+h) - g(x))(E(k) + f'(g(x))).$$

Dividing through by h:

$$\frac{f(g(x+h)) - f(g(x))}{h} = \frac{g(x+h) - g(x)}{h} \cdot (E(k) + f'(g(x))).$$

As $h \to 0$, $E(k) \to 0$ by definition of the derivative, we obtain

$$\frac{d}{dx}[f(g(x))] = f'(g(x))g'(x),$$

completing the proof of the chain rule.

A.2 Derivations for Trigonometric Limits

Consider the unit circle (shown in the accompanying figure). The points A, K, and L are defined as follows:

- $A = (1, 0)$,
- $K = (\cos x, \sin x)$,
- $L = (1, \tan x)$.

From the unit circle, we can compare the areas of three regions:

1. The area of the sector subtended by the angle x: $\frac{1}{2}x$.
2. The area of the triangle $\triangle OAK$: $\frac{1}{2}\sin x$
3. The area of the larger triangle $\triangle OAL$: $\frac{1}{2}\tan x$.

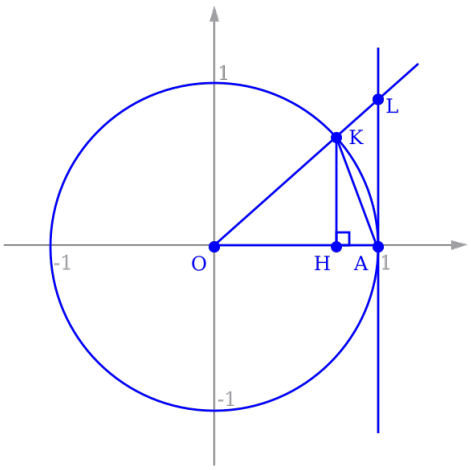

Fig. A.1: Unit circle

These areas satisfy the inequality:

$$\text{Area of } \triangle OAK \leq \text{Area of sector} \leq \text{Area of } \triangle OAL.$$

Substituting the expressions for the areas, we have:

$$\frac{1}{2}\sin x \leq \frac{1}{2}x \leq \frac{1}{2}\tan x.$$

Dividing through by $\sin x$ (valid for $x > 0$), we get:

$$1 \leq \frac{x}{\sin x} \leq \frac{1}{\cos x}.$$

Taking reciprocals (and reversing the inequalities), we find:

$$\cos x \leq \frac{\sin x}{x} \leq 1.$$

As $x \to 0$, the cosine function satisfies $\cos x \to 1$. Thus, by the squeeze theorem:

$$\lim_{x \to 0} \frac{\sin x}{x} = 1.$$

This result forms the foundation of calculus, and other trigonometric limits can be derived from it.

A.3 Formal Derivations for e

A.3.1 Limit Representation of e^x

The exponential function e^x can be represented by the following limit:

$$S(x) = e^x = \lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n. \tag{A.1}$$

We now prove this result based on the definition of e.

Taking the natural logarithm of both sides:

$$\ln[S(x)] = \ln\left[\lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n\right].$$

Since the natural logarithm is a continuous function, we can interchange the limit and the logarithm:

$$\ln[S(x)] = \lim_{n \to \infty} n \ln\left(1 + \frac{x}{n}\right).$$

Now substitute $u = \frac{x}{n}$, so that $n = \frac{x}{u}$. As $n \to \infty$, we have $u \to 0$. Thus:

$$\ln(S(x)) = \lim_{u \to 0} \frac{x}{u} \ln(1 + u) = x \lim_{u \to 0} \frac{1}{u} \ln(1 + u). \tag{A.2}$$

From Bernoulli's definition of e, we know:

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n.$$

Using a similar process as above, we find:

$$\ln(e) = \lim_{v \to 0} \frac{1}{v} \ln(1 + v).$$

By the definition of the natural logarithm, $\ln(e) = 1$. Therefore:

$$\lim_{v \to 0} \frac{1}{v} \ln(1 + v) = 1. \tag{A.3}$$

Combining Equations (A.2) and (A.3), we obtain:

$$\ln(S(x)) = x \cdot 1 = x.$$

Therefore,

$$S(x) = e^x.$$

A.3.2 Series Expansion of e^x

To derive the series expansion for $\exp(x)$, we begin with the limit definition given in Equation A.1:

$$e^x = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n.$$

Using the binomial theorem, for any positive integer n, we have:

$$\left(1 + \frac{x}{n}\right)^n = \sum_{k=0}^{n} \binom{n}{k} \frac{x^k}{n^k}.$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the binomial coefficient.

Expanding the binomial coefficient gives:

$$\left(1 + \frac{x}{n}\right)^n = \sum_{k=0}^{n} \frac{n(n-1)(n-2)\cdots(n-k+1)}{k!} \cdot \frac{x^k}{n^k}.$$

This expression can be rewritten using a product form:

$$\frac{n(n-1)(n-2)\cdots(n-k+1)}{n^k} = \prod_{j=0}^{k-1} \left(1 - \frac{j}{n}\right).$$

Define:

$$a_{n,k} = \prod_{j=0}^{k-1} \left(1 - \frac{j}{n}\right),$$

so that:

$$\left(1 + \frac{x}{n}\right)^n = \sum_{k=0}^{n} \frac{x^k}{k!} a_{n,k}.$$

Now consider the behavior of $a_{n,k}$ as $n \to \infty$:

- For fixed k, each factor $(1 - j/n) \to 1$, so $a_{n,k} \to 1$.
- If k grows with n (e.g., $k = n$), then $a_{n,k} \to 0$. In particular:

$$a_{n,n} = \frac{n!}{n^n} \to 0 \quad \text{as } n \to \infty.$$

Thus, while $a_{n,k} \not\to 1$ uniformly over all $k \leq n$, we can still show:

$$\lim_{n\to\infty} \sum_{k=0}^{n} \frac{x^k}{k!} a_{n,k} = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

by splitting the sum into two parts:

$$\sum_{k=0}^{n} \frac{x^k}{k!} a_{n,k} = \sum_{k=0}^{N} \frac{x^k}{k!} a_{n,k} + \sum_{k=N+1}^{n} \frac{x^k}{k!} a_{n,k},$$

for some large but fixed N.

Step 1: For fixed N, as $n \to \infty$, the first sum becomes:

$$\sum_{k=0}^{N} \frac{x^k}{k!} a_{n,k} \to \sum_{k=0}^{N} \frac{x^k}{k!}.$$

Step 2: Bound the tail.

Note that $0 < a_{n,k} \leq 1$ for all $k \leq n$, so we have the bound:

$$\left| \sum_{k=N+1}^{n} \frac{x^k}{k!} a_{n,k} \right| \leq \sum_{k=N+1}^{n} \frac{|x|^k}{k!} \leq \sum_{k=N+1}^{\infty} \frac{|x|^k}{k!}.$$

Since the exponential series converges absolutely, its tail tends to zero as $N \to \infty$. That is:

$$\lim_{N \to \infty} \sum_{k=N+1}^{\infty} \frac{|x|^k}{k!} = 0.$$

Therefore, for any $\varepsilon > 0$, there exists an N such that:

$$\sum_{k=N+1}^{\infty} \frac{|x|^k}{k!} < \frac{\varepsilon}{2}.$$

For this fixed N, there exists an n_0 such that for all $n \geq n_0$,

$$\left| \sum_{k=0}^{N} \frac{x^k}{k!} a_{n,k} - \sum_{k=0}^{N} \frac{x^k}{k!} \right| < \frac{\varepsilon}{2}.$$

Combining both bounds, we get:

$$\left| \sum_{k=0}^{n} \frac{x^k}{k!} a_{n,k} - \sum_{k=0}^{\infty} \frac{x^k}{k!} \right| < \varepsilon.$$

Hence:

$$\lim_{n \to \infty} \sum_{k=0}^{n} \frac{x^k}{k!} a_{n,k} = \sum_{k=0}^{\infty} \frac{x^k}{k!}.$$

From the above, we conclude:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}. \tag{A.4}$$

A.3.3 Derivative of e^x

We prove that $\frac{d}{dx} e^x = e^x$ using the series expansion Equation A.4 and term-by-term differentiation.

Differentiate the series representation:

$$\begin{aligned} \frac{d}{dx} e^x &= \frac{d}{dx} \left(\sum_{k=0}^{\infty} \frac{x^k}{k!} \right) = \frac{d}{dx} \left(1 + \sum_{k=1}^{\infty} \frac{x^k}{k!} \right) \\ &= \sum_{k=1}^{\infty} \frac{d}{dx} \left(\frac{x^k}{k!} \right) = \sum_{k=1}^{\infty} \frac{x^{k-1}}{(k-1)!} \\ &= \sum_{k=0}^{\infty} \frac{x^k}{k!} \end{aligned}$$

Thus, we conclude:

$$\frac{d}{dx}e^x = e^x.$$

A.3.4 Derivative of $ln(x)$

We aim to compute the derivative of the natural logarithm function, $\ln(x)$, using implicit differentiation. Specifically, we will show that:

$$\frac{d}{dx}\ln(x) = \frac{1}{x}.$$

Let $y = \ln(x)$. By the definition of the natural logarithm, this implies $e^y = x$.

Differentiate both sides of $e^y = x$ with respect to x. Using the chain rule on the left-hand side, we have:

$$\frac{d}{dx}e^y = e^y \frac{dy}{dx}.$$

The right-hand side is simply $\frac{d}{dx}x = 1$.

Thus, $e^y \frac{dy}{dx} = 1$.

Since $e^y = x$, we have $x\frac{dy}{dx} = 1$. Since $x > 0$, it follows that $\frac{dy}{dx} = \frac{1}{x}$.

This completes the proof.

A.4 Proofs of Convolution Properties

1. Commutativity: $f * g = g * f$

Proof: Let us perform a change of variable $u = t - \tau$, so that $\tau = t - u$ and $d\tau = -du$. Substituting into the definition:

$$\begin{aligned}(f * g)(t) &= \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau \\ &= \int_{\infty}^{-\infty} f(t-u)g(u)(-du) \\ &= \int_{-\infty}^{\infty} g(u)f(t-u)\,du = (g * f)(t).\end{aligned}$$

2. Associativity: $(f * g) * h = f * (g * h)$

Proof: Starting from the left-hand side:

$$\begin{aligned}((f * g) * h)(t) &= \int_{-\infty}^{\infty} (f * g)(\tau)h(t-\tau)\,d\tau \\ &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} f(u)g(\tau-u)\,du\right) h(t-\tau)\,d\tau.\end{aligned}$$

By interchanging the order of integration (Fubini's theorem), we obtain:

$$\begin{aligned} &= \int_{-\infty}^{\infty} f(u) \left(\int_{-\infty}^{\infty} g(\tau - u) h(t - \tau)\, d\tau \right) du \\ &= \int_{-\infty}^{\infty} f(u)(g * h)(t - u)\, du = (f * (g * h))(t). \end{aligned}$$

3. Distributivity over addition: $f * (g + h) = f * g + f * h$

Proof: This follows directly from the linearity of the integral:

$$\begin{aligned} f * (g + h) &= \int_{-\infty}^{\infty} f(\tau)(g + h)(t - \tau)\, d\tau \\ &= \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau + \int_{-\infty}^{\infty} f(\tau) h(t - \tau)\, d\tau \\ &= f * g + f * h. \end{aligned}$$

Appendix B

Matrix Property Derivations

B.1 Proof of the Spectral Theorem

In this section, we prove one of the most important results in linear algebra: the **Spectral Theorem** for real symmetric matrices. This theorem guarantees that every real symmetric matrix can be diagonalized by an orthogonal matrix — that is, its eigenvectors form an orthonormal basis for $\mathbb{R}^n$, and all its eigenvalues are real.

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a symmetric matrix, i.e., $\mathbf{A} = \mathbf{A}^\top$. Then:

1. All eigenvalues of $\mathbf{A}$ are real.
2. There exists an orthonormal basis of $\mathbb{R}^n$ consisting of eigenvectors of $\mathbf{A}$.
3. Consequently, $\mathbf{A}$ is orthogonally diagonalizable:

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^\top,$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix (i.e., $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$), and Λ is a diagonal matrix whose entries are the eigenvalues of $\mathbf{A}$.

Step 1: Eigenvalues of a Symmetric Matrix Are Real

Let $\lambda \in \mathbb{C}$ be an eigenvalue of $\mathbf{A}$ with corresponding eigenvector $\mathbf{v} \in \mathbb{C}^n$, so:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}.$$

Take the conjugate transpose of both sides:

$$\mathbf{v}^*\mathbf{A}^* = \overline{\lambda}\mathbf{v}^*.$$

Since $\mathbf{A}$ is real and symmetric, $\mathbf{A}^* = \mathbf{A}^\top = \mathbf{A}$, so:

$$\mathbf{v}^*\mathbf{A} = \overline{\lambda}\mathbf{v}^*.$$

Now multiply the original equation on the left by $\mathbf{v}^*$:

$$\mathbf{v}^*\mathbf{A}\mathbf{v} = \lambda\mathbf{v}^*\mathbf{v}.$$

From the conjugate-transposed equation, we also have:

$$\mathbf{v}^*\mathbf{A}\mathbf{v} = \overline{\lambda}\mathbf{v}^*\mathbf{v}.$$

Equating both expressions:

$$\lambda\mathbf{v}^*\mathbf{v} = \overline{\lambda}\mathbf{v}^*\mathbf{v}.$$

Since $\mathbf{v} \neq 0$, $\mathbf{v}^*\mathbf{v} > 0$, so we can divide both sides to get:

$$\lambda = \overline{\lambda},$$

which implies that λ is real.

Step 2: Eigenvectors Corresponding to Distinct Eigenvalues Are Orthogonal

See Section 2.1.2.

Step 3: Existence of a Full Set of Orthonormal Eigenvectors

We now show that there exists an orthonormal basis of $\mathbb{R}^n$ consisting of eigenvectors of $\mathbf{A}$. We proceed by induction on n.

- **Base case ($n = 1$)**: A 1×1 symmetric matrix is just a scalar, which is trivially diagonalizable.
- **Inductive step**: Assume that any $(n-1) \times (n-1)$ symmetric matrix has an orthonormal set of $n-1$ eigenvectors. Now consider an $n \times n$ symmetric matrix $\mathbf{A}$.

By the Fundamental Theorem of Algebra, the characteristic polynomial of $\mathbf{A}$ has at least one root, say λ_1, which is real (from Step 1). Let $\mathbf{v}_1$ be a corresponding unit eigenvector.

Let $V = \text{span}\{\mathbf{v}_1\}$, and let $V^\perp$ be its orthogonal complement. Since $\mathbf{A}$ is symmetric, it maps $V^\perp$ into itself:

$$\mathbf{x} \in V^\perp \Rightarrow \mathbf{v}_1^\top \mathbf{x} = 0 \Rightarrow \mathbf{v}_1^\top \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{v}_1^\top \mathbf{x} = \lambda_1 \mathbf{v}_1^\top \mathbf{x} = 0 \Rightarrow \mathbf{A}\mathbf{x} \in V^\perp.$$

Therefore, the restriction of $\mathbf{A}$ to $V^\perp$ is a symmetric operator on an $(n-1)$-dimensional space. By the inductive hypothesis, it has an orthonormal basis of eigenvectors $\{\mathbf{v}_2, \ldots, \mathbf{v}_n\}$. Adding $\mathbf{v}_1$, we obtain an orthonormal basis of $\mathbb{R}^n$ consisting of eigenvectors of $\mathbf{A}$.

Step 4: Orthogonal Diagonalization

Let $\mathbf{Q}$ be the matrix whose columns are the orthonormal eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, and let Λ be the diagonal matrix with the corresponding eigenvalues on the diagonal. Then:

$$\mathbf{A}\mathbf{Q} = \mathbf{Q}\Lambda.$$

Multiplying both sides on the right by $\mathbf{Q}^\top$ and using the fact that $\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}$, we get:

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^\top.$$

This completes the proof of the spectral theorem.

Appendix C

Derivations for Fourier Series and Fourier Transforms

C.1 Fourier Series

C.1.1 Derivation of a_n

1. Start with the Fourier series representation of $f(x)$:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{2\pi n}{T} x\right) + b_n \sin\left(\frac{2\pi n}{T} x\right) \right].$$

2. To isolate a_n, multiply both sides by $\cos\left(\frac{2\pi n}{T} x\right)$ and integrate over one period $[0, T]$:

$$\int_0^T f(x) \cos\left(\frac{2\pi n}{T} x\right) dx = \int_0^T \left[a_0 \cos\left(\frac{2\pi n}{T} x\right) + \sum_{m=1}^{\infty} \left(a_m \cos\left(\frac{2\pi m}{T} x\right) + b_m \sin\left(\frac{2\pi m}{T} x\right) \right) \cos\left(\frac{2\pi n}{T} x\right) \right] dx.$$

3. Use the orthogonality property: - The integral of $\cos\left(\frac{2\pi n}{T} x\right)$ with any other cosine term $\cos\left(\frac{2\pi m}{T} x\right)$ (where $m \neq n$) is zero. - The integral of $\cos\left(\frac{2\pi n}{T} x\right)$ with any sine term $\sin\left(\frac{2\pi m}{T} x\right)$ is also zero. - The only nonzero term comes from $\cos\left(\frac{2\pi n}{T} x\right) \cdot \cos\left(\frac{2\pi n}{T} x\right)$, which integrates to $\frac{T}{2}$.

4. After simplifying, you get:

$$\int_0^T f(x) \cos\left(\frac{2\pi n}{T} x\right) dx = a_n \cdot \frac{T}{2}.$$

5. Solve for a_n:

$$a_n = \frac{2}{T} \int_0^T f(x) \cos\left(\frac{2\pi n}{T} x\right) dx.$$

The factor $\frac{2}{T}$ arises because the orthogonality integral of $\cos^2$ or $\sin^2$ over one period gives $\frac{T}{2}$, not T. To compensate for this, we multiply by $\frac{2}{T}$ to correctly extract a_n.

For a_0, the calculation is slightly different. The formula for a_0 is:

$$a_0 = \frac{1}{T} \int_0^T f(x) dx.$$

Here, we're finding the average value of $f(x)$ over one period. Since there's no cosine or sine term involved, there's no orthogonality adjustment needed, and the factor is simply $\frac{1}{T}$.

The "2" in $\frac{2}{T}$ comes from the orthogonality property of sine and cosine functions. Specifically: - The integral of $\cos^2$ or $\sin^2$ over one period is $\frac{T}{2}$, not T. - To correctly extract the coefficients a_n and b_n, we multiply by $\frac{2}{T}$ to account for this.

C.1.2 Appendix: Complex Exponentials and Finite Geometric Sums

This appendix provides the mathematical foundation for the summation formulas of sinusoidal sequences used in Section 4.1.4. We begin with a key result from series theory — the finite geometric sum — and then apply it to derive the closed-form expressions for sums of cosine and sine functions in arithmetic progression.

Preliminary: Finite Geometric Series

The sum of a finite geometric sequence is given by:

$$\sum_{k=0}^{n-1} z^k = \begin{cases} \dfrac{1-z^n}{1-z}, & \text{if } z \neq 1 \\ n, & \text{if } z = 1 \end{cases} \tag{C.1}$$

This result can be derived by considering the partial sum $S = 1 + z + z^2 + \cdots + z^{n-1}$. Multiplying both sides by z yields:

$$zS = z + z^2 + \cdots + z^{n-1} + z^n$$

Subtracting this from S:

$$S - zS = 1 - z^n$$

Thus, for $z \neq 1$:

$$S = \frac{1-z^n}{1-z}$$

This formula holds for any complex number $z \neq 1$, which is essential for our application to oscillatory functions.

Application to Sinusoidal Sums via Euler's Formula

We now use this result to compute the sum:

$$\sum_{k=0}^{n-1} \cos(a + kd)$$

where a is an initial phase and d is the angular increment.

Using **Euler's formula**, $e^{i\theta} = \cos\theta + i\sin\theta$, we express the cosine as the real part of a complex exponential:

$$\cos(a + kd) = \operatorname{Re}\left(e^{i(a+kd)}\right)$$

Therefore, the sum becomes:

$$\sum_{k=0}^{n-1} \cos(a+kd) = \text{Re}\left(\sum_{k=0}^{n-1} e^{i(a+kd)}\right)$$
$$= \text{Re}\left(e^{ia}\sum_{k=0}^{n-1}\left(e^{id}\right)^k\right)$$

Let $z = e^{id}$. Since $|z| = 1$ and d is real, $z \neq 1$ unless d is an integer multiple of 2π. Applying the geometric sum formula (C.1):

$$\sum_{k=0}^{n-1}\left(e^{id}\right)^k = \frac{1-e^{iNd}}{1-e^{id}}, \quad \text{for } d \text{ not an integer multiple of } 2\pi$$

Thus:

$$\sum_{k=0}^{n-1} \cos(a+kd) = \text{Re}\left(e^{ia}\cdot\frac{1-e^{iNd}}{1-e^{id}}\right)$$

We simplify this expression using trigonometric identities. First, factor out half-angles from numerator and denominator:

- Numerator:
$$1-e^{iNd} = e^{iNd/2}\left(e^{-iNd/2}-e^{iNd/2}\right) = -2ie^{iNd/2}\sin\left(\frac{Nd}{2}\right)$$

- Denominator:
$$1-e^{id} = e^{id/2}\left(e^{-id/2}-e^{id/2}\right) = -2ie^{id/2}\sin\left(\frac{d}{2}\right)$$

Substituting:

$$\frac{1-e^{iNd}}{1-e^{id}} = \frac{-2ie^{iNd/2}\sin\left(\frac{Nd}{2}\right)}{-2ie^{id/2}\sin\left(\frac{d}{2}\right)} = e^{i(n-1)d/2}\cdot\frac{\sin\left(\frac{Nd}{2}\right)}{\sin\left(\frac{d}{2}\right)}$$

Now multiply by e^{ia}:

$$e^{ia}\cdot\frac{1-e^{iNd}}{1-e^{id}} = e^{i\left(a+\frac{(n-1)d}{2}\right)}\cdot\frac{\sin\left(\frac{Nd}{2}\right)}{\sin\left(\frac{d}{2}\right)}$$

Taking the real part:

$$\text{Re}\left(e^{i\left(a+\frac{(n-1)d}{2}\right)}\cdot\frac{\sin\left(\frac{Nd}{2}\right)}{\sin\left(\frac{d}{2}\right)}\right) = \frac{\sin\left(\frac{Nd}{2}\right)}{\sin\left(\frac{d}{2}\right)}\cdot\cos\left(a+\frac{(n-1)d}{2}\right)$$

Thus, we obtain the desired identity:

$$\sum_{k=0}^{n-1} \cos(a+kd) = \frac{\sin\left(\frac{nd}{2}\right)\cos\left(a+\frac{(n-1)d}{2}\right)}{\sin\left(\frac{d}{2}\right)}$$

This formula is valid when d is not an integer multiple of 2π.

Analogous Result for Sine

A nearly identical derivation applies to the sine sum. Starting from:

$$\sin(a+kd) = \text{Im}\left(e^{i(a+kd)}\right)$$

we follow the same steps:

$$\begin{aligned}\sum_{k=0}^{n-1} \sin(a+kd) &= \text{Im}\left(e^{ia}\cdot\frac{1-e^{iNd}}{1-e^{id}}\right)\\ &= \text{Im}\left(e^{i\left(a+\frac{(n-1)d}{2}\right)}\cdot\frac{\sin\left(\frac{Nd}{2}\right)}{\sin\left(\frac{d}{2}\right)}\right)\\ &= \frac{\sin\left(\frac{Nd}{2}\right)}{\sin\left(\frac{d}{2}\right)}\cdot\sin\left(a+\frac{(n-1)d}{2}\right)\end{aligned}$$

Hence:

$$\boxed{\sum_{k=0}^{n-1} \sin(a+kd) = \frac{\sin\left(\frac{nd}{2}\right)\sin\left(a+\frac{(n-1)d}{2}\right)}{\sin\left(\frac{d}{2}\right)}} \tag{C.2}$$

The structure is identical to the cosine case — only the outer trigonometric function changes, reflecting the real vs. imaginary part decomposition.

Conclusion

These derivations show how powerful the combination of **complex exponentials** and **geometric series** is in analyzing periodic and oscillatory sequences. The resulting summation formulas are not only elegant but also essential in signal processing, particularly in understanding orthogonality, spectral leakage, and energy compaction in discrete transforms such as the DCT and DFT.

C.2 Fourier Transform

C.2.1 Derivation of the Dirac Delta Function from the Fourier Integral

The integral

$$\int_{-\infty}^{\infty} e^{i\omega(t-t')}d\omega$$

is a fundamental result in Fourier analysis. It evaluates to a Dirac delta function:

$$\int_{-\infty}^{\infty} e^{i\omega(t-t')} d\omega = 2\pi\delta(t-t').$$

Below we derive this result rigorously by regularizing the integral using a limiting process. Consider the modified version of the integral:

$$I_\epsilon = \int_{-\infty}^{\infty} e^{i\omega(t-t')} e^{-\epsilon\omega^2} d\omega,$$

where $e^{-\epsilon\omega^2}$ is a Gaussian damping factor where $\epsilon > 0$. As $\epsilon \to 0$, this damping factor approaches 1, and we recover the original integral.

Step 3.1: Evaluate the Regularized Integral

Looking at the exponent:

$$i\omega(t-t') - \epsilon\omega^2 = -\epsilon\left(\omega^2 - \frac{i(t-t')}{\epsilon}\omega\right).$$

Complete the square inside the parentheses:

$$\omega^2 - \frac{i(t-t')}{\epsilon}\omega = \left(\omega - \frac{i(t-t')}{2\epsilon}\right)^2 - \frac{(t-t')^2}{4\epsilon^2}.$$

Substitute this back into the integral:

$$I_\epsilon = \int_{-\infty}^{\infty} e^{-\epsilon\left(\omega - \frac{i(t-t')}{2\epsilon}\right)^2} e^{\frac{\epsilon(t-t')^2}{4\epsilon^2}} d\omega.$$

The first term is a Gaussian centered at $\omega = \frac{i(t-t')}{2\epsilon}$ [1], and the second term is a constant factor. The Gaussian integral evaluates to:

$$\int_{-\infty}^{\infty} e^{-\epsilon(\omega-c)^2} d\omega = \sqrt{\frac{\pi}{\epsilon}}.$$

So:

$$I_\epsilon = \sqrt{\frac{\pi}{\epsilon}} e^{-\frac{(t-t')^2}{4\epsilon}}.$$

Step 3.2: Take the Limit $\epsilon \to 0$

As $\epsilon \to 0$, the prefactor $\sqrt{\frac{\pi}{\epsilon}}$ diverges, and the exponential $e^{-\frac{(t-t')^2}{4\epsilon}}$ becomes sharply peaked around $t = t'$. In the limit, this behaves like a Dirac delta function:

$$\lim_{\epsilon\to 0} I_\epsilon = 2\pi\delta(t-t').$$

Proof is completed.

C.2.2 Proof of the Inverse Fourier Transform

For readers interested in the mathematical derivation, here is how the inverse Fourier Transform is derived.

[1] In Complex Analysis, a normal distribution can have a complex mean, such as $\omega = \frac{i(t-t')}{2\epsilon}$ in this case. This extends the standard real-valued Gaussian but is omitted here for simplicity. For further reading, see "Complex Normal Distribution" at Wikipedia.

Step 1: Start with the Fourier Transform

The Fourier Transform is:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt.$$

Substitute this into the formula for the inverse Fourier Transform:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(t')e^{-i\omega t'}dt'\right]e^{i\omega t}d\omega.$$

Step 2: Rearrange the Integrals

Swap the order of integration:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t')\left[\int_{-\infty}^{\infty} e^{i\omega(t-t')}d\omega\right]dt'.$$

Step 3: Evaluate the Inner Integral

The inner integral is:

$$\int_{-\infty}^{\infty} e^{i\omega(t-t')}d\omega.$$

Using the properties of the Dirac delta function, this evaluates to:

$$\int_{-\infty}^{\infty} e^{i\omega(t-t')}d\omega = 2\pi\delta(t-t').$$

Substitute this back:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t')\left[2\pi\delta(t-t')\right]dt'.$$

Step 4: Simplify Using the Sifting Property

The 2π cancels out:

$$f(t) = \int_{-\infty}^{\infty} f(t')\delta(t-t')dt'.$$

Using the sifting property of the delta function:

$$f(t) = f(t).$$

Thus, the inverse Fourier Transform correctly reconstructs the original signa

Appendix D

Statistics Derivations

D.1 Covariance Matrix and Eigen Values

We show that if $\mathbf{v}$ is an eigenvector of the covariance matrix Σ with eigenvalue λ, then the variance of the projection of the random vector $\mathbf{X}$ onto $\mathbf{v}$ equals λ:

$$\mathrm{Var}(\mathbf{v}^\top \mathbf{X}) = \lambda.$$

Here $\mathbf{v}$ denotes a column vector.

Step 1: Define the Projection

Let $\mathbf{X} = [X_1, X_2, \ldots, X_n]^\top$ be a random vector in $\mathbb{R}^n$, and let $\mathbf{v} = [v_1, v_2, \ldots, v_n]^\top$ be a unit eigenvector of Σ. If $\mathbf{X}$ represents a 2D point, then $n = 2$.

The scalar quantity $\mathbf{v}^\top \mathbf{X}$ is a linear combination of the components of $\mathbf{X}$ — in other words, the projection of $\mathbf{X}$ onto the direction defined by $\mathbf{v}$.

Step 2: Use Basic Properties of Variance

Recall the formula for the variance of a sum of random variables (**??**):

$$\begin{aligned}\mathrm{Var}\left(\sum_{i=1}^{n} a_i X_i\right) &= \mathrm{Cov}\left(\sum_{i=1}^{n} a_i X_i, \sum_{i=1}^{n} a_i X_i\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \,\mathrm{Cov}(X_i, X_j),\end{aligned}$$

where a_i are constants.

Applying this to $v^\top X = \sum_{i=1}^{n} v_i X_i$, we get:

$$\mathrm{Var}(v^\top X) = \sum_{i=1}^{n}\sum_{j=1}^{n} v_i v_j \,\mathrm{Cov}(X_i, X_j).$$

Notice that the double sum is just a matrix multiplication:

$$\mathrm{Var}(v^\top X) = \mathbf{v}^\top \Sigma \mathbf{v},$$

where Σ is the covariance matrix of $\mathbf{X}$, with entries $\Sigma_{ij} = \mathrm{Cov}(X_i, X_j)$.

Step 3: Plug in the Eigenvalue Equation

Now assume that $\mathbf{v}$ is an eigenvector of Σ, with corresponding eigenvalue λ. That means:

$$\Sigma \mathbf{v} = \lambda \mathbf{v}.$$

Then :

$$\text{Var}(\mathbf{v}^\top \mathbf{X}) = \mathbf{v}^\top \Sigma \mathbf{v} = \mathbf{v}^\top (\lambda \mathbf{v}) = \lambda (\mathbf{v}^\top \mathbf{v}).$$

Since $\mathbf{v}$ is a unit vector, $\mathbf{v}^\top \mathbf{v} = 1$. Therefore,

$$\text{Var}(\mathbf{v}^\top \mathbf{X}) = \lambda.$$

D.2 Derivation of Normal Distribution Properties

D.2.1 Verification of the Validity of the Normal Distribution as a PDF

The probability density function (PDF) of a normal distribution with mean μ and variance σ^2 is given by:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{D.1}$$

To verify that this is a valid PDF, we must show:

$$\int_{-\infty}^{\infty} f(x)\, dx = 1$$

As before, we make the substitution $z = \frac{x-\mu}{\sigma}$, so that $x = \mu + \sigma z$ and $dx = \sigma\, dz$. The integral becomes:

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z^2}{2}} \sigma\, dz = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{z^2}{2}}\, dz$$

Let us define:

$$I = \int_{-\infty}^{\infty} e^{-\frac{z^2}{2}}\, dz$$

Then:

$$I^2 = \left(\int_{-\infty}^{\infty} e^{-\frac{z^2}{2}}\, dz \right)^2 = \iint_{\mathbb{R}^2} e^{-\frac{x^2+y^2}{2}}\, dx\, dy$$

Now convert to polar coordinates using:

$$x = r\cos\theta, \quad y = r\sin\theta, \quad dx\, dy = r\, dr\, d\theta$$

In these coordinates, $x^2 + y^2 = r^2$, so:

$$I^2 = \int_0^{2\pi} \int_0^{\infty} e^{-\frac{r^2}{2}} r\, dr\, d\theta$$

Evaluate the inner integral first:

$$\int_0^{\infty} e^{-\frac{r^2}{2}} r\, dr$$

Make the substitution $u = \frac{r^2}{2} \Rightarrow du = r\,dr$, so:

$$\int_0^\infty e^{-\frac{r^2}{2}} r\,dr = \int_0^\infty e^{-u}\,du = 1$$

Thus:

$$I^2 = \int_0^{2\pi} 1\,d\theta = 2\pi \Rightarrow I = \sqrt{2\pi}$$

Returning to our original expression:

$$\frac{1}{\sqrt{2\pi}} \cdot \sqrt{2\pi} = 1$$

Therefore:

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\,dx = 1$$

This confirms that the normal distribution integrates to one over the entire real line, and hence is a valid probability density function.

D.2.2 Derivation of the Moment Generating Function for a Normal Distribution

We are given the definition of the moment generating function (MGF) for a random variable $X \sim \mathcal{N}(\mu, \sigma^2)$:

$$M_X(t) = \int_{-\infty}^{\infty} e^{tx} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx.$$

This expression combines the exponential term e^{tx} with the probability density function of the normal distribution. Our goal is to simplify and evaluate this integral.

Step 1: Combine Exponents

We combine the exponents inside the integral:

$$M_X(t) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(tx - \frac{(x-\mu)^2}{2\sigma^2}\right) dx.$$

Expanding the quadratic in the exponent:

$$\begin{aligned} tx - \frac{(x-\mu)^2}{2\sigma^2} &= tx - \frac{1}{2\sigma^2}(x^2 - 2\mu x + \mu^2) \\ &= -\frac{x^2}{2\sigma^2} + \left(\frac{\mu}{\sigma^2} + t\right) x - \frac{\mu^2}{2\sigma^2} \\ &= -\frac{1}{2\sigma^2}\left(x - (\mu + \sigma^2 t)\right)^2 + \frac{(\mu + \sigma^2 t)^2}{2\sigma^2}. \end{aligned}$$

Step 2: Evaluate the Integral

Now plug this back into the original expression:

$$M_X(t) = \exp\left(\mu t + \frac{1}{2}\sigma^2 t^2\right) \cdot \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - (\mu + \sigma^2 t))^2\right) dx.$$

The integral represents the integral of a normal density function with mean $\mu + \sigma^2 t$ and variance σ^2, which integrates to 1.

Therefore, the moment generating function is:

$$M_X(t) = \exp\left(\mu t + \frac{1}{2}\sigma^2 t^2\right)$$

This is the MGF of a normally distributed random variable with mean μ and variance σ^2.

D.3 Proof of Central Limit Theorem

To prove the CLT, we analyze the moment-generating function (MGF) of $Z_n = \frac{\sum_{i=1}^{n}(X_i - \mu)}{\sigma\sqrt{n}}$ and show that it converges to the MGF of a standard normal random variable.

MGF of Z_n

The MGF of Z_n is given by:

$$M_{Z_n}(t) = \mathbb{E}\left[e^{tZ_n}\right].$$

Substituting the definition of Z_n:

$$M_{Z_n}(t) = \mathbb{E}\left[e^{\frac{t}{\sigma\sqrt{n}}\sum_{i=1}^{n}(X_i - \mu)}\right].$$

Since the X_i's are i.i.d., the sum can be factored into a product:

$$M_{Z_n}(t) = \prod_{i=1}^{n} \mathbb{E}\left[e^{\frac{t}{\sigma\sqrt{n}}(X_i - \mu)}\right].$$

Because all X_i's have the same distribution, this simplifies to (in the context of this expectation, t is a constant here):

$$M_{Z_n}(t) = \left(\mathbb{E}\left[e^{\frac{t}{\sigma\sqrt{n}}(X_1 - \mu)}\right]\right)^n.$$

Approximation for Large n

Let $Y = \frac{t}{\sigma\sqrt{n}}(X_1 - \mu)$. For small Y, expand e^Y using a Taylor series around 0:

$$e^Y \approx 1 + Y + \frac{Y^2}{2} + \cdots.$$

Thus:

$$\mathbb{E}\left[e^{\frac{t}{\sigma\sqrt{n}}(X_1-\mu)}\right] \approx \mathbb{E}\left[1 + \frac{t}{\sigma\sqrt{n}}(X_1-\mu) + \frac{1}{2}\left(\frac{t}{\sigma\sqrt{n}}(X_1-\mu)\right)^2 + \cdots\right].$$

Compute each term:

- The first term is $\mathbb{E}[1] = 1$.
- The second term is:
$$\mathbb{E}\left[\frac{t}{\sigma\sqrt{n}}(X_1-\mu)\right] = \frac{t}{\sigma\sqrt{n}}\mathbb{E}[X_1-\mu] = 0,$$
since $\mathbb{E}[X_1-\mu] = 0$.
- The third term is:
$$\mathbb{E}\left[\frac{1}{2}\left(\frac{t}{\sigma\sqrt{n}}(X_1-\mu)\right)^2\right] = \frac{1}{2}\frac{t^2}{\sigma^2 n}\mathbb{E}[(X_1-\mu)^2] = \frac{t^2}{2n},$$
because $\mathbb{E}[(X_1-\mu)^2] = \sigma^2$.

Higher-order terms involve powers of $\frac{1}{\sqrt{n}}$, which vanish as $n \to \infty$. Thus, for large n:

$$\mathbb{E}\left[e^{\frac{t}{\sigma\sqrt{n}}(X_1-\mu)}\right] \approx 1 + \frac{t^2}{2n}.$$

MGF of Z_n

Raise the above expression to the power n:

$$M_{Z_n}(t) = \left(1 + \frac{t^2}{2n}\right)^n.$$

For large n, use the approximation $(1 + x/n)^n \to e^x$ as $n \to \infty$:

$$M_{Z_n}(t) \to e^{t^2/2}.$$

The limiting MGF of Z_n is $e^{t^2/2}$, which is the MGF of a standard normal random variable $Z \sim N(0,1)$. By the uniqueness of MGFs, this implies that Z_n converges in distribution to $Z \sim N(0,1)$.

D.4 Derivation of KL Divergence Properties

A fundamental property of the Kullback–Leibler (KL) divergence is its **non-negativity**: it is always greater than or equal to zero, with equality if and only if the two distributions being compared are identical almost everywhere. This result can be elegantly established using *Jensen's inequality*.

Later, we will derive the KL divergence of two Gaussians.

D.4.1 Convex Functions and Jensen's Inequality

A function $f : I \to \mathbb{R}$, defined on an interval $I \subseteq \mathbb{R}$, is called **convex** if for all $x_1, x_2 \in I$ and $\lambda \in [0, 1]$, it satisfies:

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$$

If the inequality is strict for all $\lambda \in (0, 1)$ and $x_1 \neq x_2$, then f is said to be **strictly convex**.

For twice-differentiable functions, convexity can also be characterized in terms of derivatives:

- f is convex on I if and only if $f''(x) \geq 0$ for all $x \in I$.
- If $f''(x) > 0$ for all $x \in I$, then f is strictly convex.

This definition can be generalized to any finite number of points via mathematical induction. Specifically, for weights $\lambda_i \geq 0$ such that $\sum_{i=1}^{n} \lambda_i = 1$, we have:

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i)$$

This inequality naturally extends to expectations when X is a real-valued random variable and λ_i are interpreted as probabilities. That is, for a convex function f and a random variable X with finite expectation:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$

This is known as **Jensen's inequality**, introduced by Danish mathematician **Johan Ludwig William Valdemar Jensen** in 1906.

For strictly convex functions, equality holds if and only if X is constant almost surely.

In what follows, we will apply Jensen's inequality to the strictly convex function $f(x) = -\log x$, which is defined and strictly convex on $(0, \infty)$. This will allow us to prove the non-negativity of the KL divergence.

D.4.2 Proof Non-negativity of KL Divergence

The continuous form of the Kullback–Leibler (KL) divergence between two probability density functions $p(x)$ and $q(x)$ is defined as:

$$D_{\mathrm{KL}}(p \parallel q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

We can rewrite this expression as:

$$D_{\mathrm{KL}}(p \parallel q) = -\int_{-\infty}^{\infty} p(x) \log\left(\frac{q(x)}{p(x)}\right) dx$$

Now consider the function $f(t) = -\log t$, which is strictly convex on $(0, \infty)$. Applying Jensen's inequality yields:

$$-\log\left(\int_{-\infty}^{\infty} p(x)\frac{q(x)}{p(x)}\, dx\right) \leq -\int_{-\infty}^{\infty} p(x) \log\left(\frac{q(x)}{p(x)}\right) dx$$

Simplifying the left-hand side:

$$-\log\left(\int_{-\infty}^{\infty} q(x)\,dx\right) = -\log(1) = 0$$

Therefore:

$$D_{\mathrm{KL}}(p \parallel q) \geq 0$$

Extension to the Discrete Case In the discrete case, the KL divergence between two probability mass functions $p(x)$ and $q(x)$ is given by:

$$D_{\mathrm{KL}}(p \parallel q) = \sum_{x} p(x) \log\left(\frac{p(x)}{q(x)}\right)$$

The same proof applies, replacing integrals with sums and interpreting $p(x)$ and $q(x)$ as discrete probabilities. The strict convexity of $-\log x$ ensures that:

$$D_{\mathrm{KL}}(p \parallel q) \geq 0$$

with equality if and only if $p(x) = q(x)$ for all x.

D.4.3 Derivation of the KL Divergence Between Two Gaussians

We now derive the closed-form expression for the Kullback–Leibler (KL) divergence between two univariate Gaussian distributions. Let:

$$p(x) = \mathcal{N}(x; \mu_1, \sigma_1^2) = \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}},$$

$$q(x) = \mathcal{N}(x; \mu_2, \sigma_2^2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}.$$

The KL divergence from p to q is defined as:

$$D_{\mathrm{KL}}(p \parallel q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx = \mathbb{E}_p\left[\log p(x) - \log q(x)\right].$$

We compute this by evaluating the expectations of $\log p(x)$ and $\log q(x)$ under $p(x)$.

Step 1: Logarithm of the densities First, take the logarithm of each density:

$$\log p(x) = -\frac{1}{2}\log(2\pi\sigma_1^2) - \frac{(x-\mu_1)^2}{2\sigma_1^2},$$

$$\log q(x) = -\frac{1}{2}\log(2\pi\sigma_2^2) - \frac{(x-\mu_2)^2}{2\sigma_2^2}.$$

Thus, the log-ratio is:

$$\log \frac{p(x)}{q(x)} = \log p(x) - \log q(x) = \frac{1}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - \frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2}.$$

Step 2: Take expectation under $p(x)$ Now compute the expectation of this expression with respect to $p(x)$:

$$D_{\mathrm{KL}}(p \parallel q) = \mathbb{E}_p\left[\frac{1}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right)\right] - \mathbb{E}_p\left[\frac{(x-\mu_1)^2}{2\sigma_1^2}\right] + \mathbb{E}_p\left[\frac{(x-\mu_2)^2}{2\sigma_2^2}\right].$$

We evaluate each term separately.

1. The first term is constant:
$$\mathbb{E}_p\left[\frac{1}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right)\right] = \frac{1}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right).$$

2. The second term uses $\mathbb{E}_p[(x-\mu_1)^2] = \sigma_1^2$:
$$\mathbb{E}_p\left[\frac{(x-\mu_1)^2}{2\sigma_1^2}\right] = \frac{1}{2\sigma_1^2}\cdot\sigma_1^2 = \frac{1}{2}.$$

3. For the third term, expand $(x-\mu_2)^2$:
$$(x-\mu_2)^2 = (x-\mu_1+\mu_1-\mu_2)^2 = (x-\mu_1)^2 + 2(x-\mu_1)(\mu_1-\mu_2) + (\mu_1-\mu_2)^2.$$
Taking expectation under p:
$$\mathbb{E}_p[(x-\mu_2)^2] = \mathbb{E}_p[(x-\mu_1)^2] + 0 + (\mu_1-\mu_2)^2 = \sigma_1^2 + (\mu_1-\mu_2)^2.$$
Therefore:
$$\mathbb{E}_p\left[\frac{(x-\mu_2)^2}{2\sigma_2^2}\right] = \frac{1}{2\sigma_2^2}\left[\sigma_1^2 + (\mu_1-\mu_2)^2\right].$$

Step 3: Combine all terms Now substitute all expectations back:

$$D_{\mathrm{KL}}(p \parallel q) = \frac{1}{2}\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) - \frac{1}{2} + \frac{1}{2\sigma_2^2}\left(\sigma_1^2 + (\mu_1-\mu_2)^2\right).$$

Rearranging:

$$D_{\mathrm{KL}}(p \parallel q) = \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) + \frac{\sigma_1^2}{\sigma_2^2} + \frac{(\mu_1-\mu_2)^2}{\sigma_2^2} - 1\right].$$

This is the desired result.

Index