Matrix Calculus Derivatives: An Introduction

A work-in-progress introductory tutorial on understanding derivatives in matrix calculus with explanations for those with rusty linear algebra, geometry, and calculus knowledge

Chris Snow

Table of contents

1 Prerequisites

Before diving into matrix calculus, you should be comfortable with:

- **Basic calculus**: derivatives, chain rule, partial derivatives
- **Linear algebra fundamentals**: vectors, matrices, matrix multiplication, transpose
- **Basic programming**: familiarity with Python/NumPy is helpful but not required

Quick refresher resources:

- Khan Academy: Linear Algebra and Calculus courses
- 3Blue1Brown: "Essence of Linear Algebra" YouTube series
- NumPy documentation for computational examples

2 Introduction

Matrix calculus is a powerful mathematical tool that extends ordinary calculus to handle multiple variables at once.

Think of it this way: ordinary calculus deals with functions like $f(x) = x^2$, where you have one input and one output.

Matrix calculus deals with more complex situations:

- Functions with multiple inputs (like $f(x, y, z) = x^2 + y^2 + z^2$)
- Functions with multiple outputs (like position in 3D space)
- Functions involving entire arrays of numbers (matrices)

2.1 Why Does This Matter?

In the real world, most problems involve many variables simultaneously.

Examples:

- A neural network might have millions of parameters that all need to be optimized together
- The flight path of an airplane depends on altitude, speed, wind direction, and many other factors
- Economic models consider prices, supply, demand, and hundreds of other variables

Matrix calculus gives us the tools to handle these complex, multi-variable situations efficiently.

2.2 Why Matrix Calculus Matters

Matrix calculus is essential for:

Machine Learning:

- Computing gradients for backpropagation in neural networks
- This means figuring out how to adjust millions of parameters to reduce prediction errors

Optimization:

- Finding the best solution when you have many variables to consider
- Like finding the minimum cost when you can adjust production, shipping, marketing, etc.

Statistics:

- Analyzing data with many variables (like predicting house prices based on size, location, age, etc.)
- Deriving formulas for statistical methods like regression

Engineering:

- Designing control systems that manage multiple inputs and outputs
- Signal processing for audio, video, and communications

Physics:

- Describing how fields (like electromagnetic fields) behave in space
- Quantum mechanics calculations

3 Mathematical Foundations

Before we dive into matrix calculus, let's review the basic building blocks.

Don't worry if these concepts feel rusty - we'll build up slowly and explain everything step by step.

3.1 What Are Scalars, Vectors, and Matrices?

Scalars: A scalar is just a single number.

Examples: 5, -2.7, π, 0

Think of it as a quantity that has magnitude but no direction.

Examples: temperature (70°F), mass (150 pounds), price ($50)

Vectors: A vector is an ordered list of numbers.

We usually write vectors as columns:

$$\mathbf{x} = \begin{bmatrix} 3 \\ 1 \\ 4 \end{bmatrix}$$

Or sometimes as rows: $\mathbf{x} = [3, 1, 4]$

Physical interpretation: Vectors represent quantities that have both magnitude and direction.

Examples: velocity (50 mph northeast), force (10 newtons upward)

Mathematical interpretation: Vectors represent a point in multi-dimensional space.

The vector $[3, 1, 4]$ represents a point that's 3 units along the x-axis, 1 unit along the y-axis, and 4 units along the z-axis.

Matrices: A matrix is a rectangular array of numbers arranged in rows and columns.

Example of a 3×2 matrix (3 rows, 2 columns):

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

Physical interpretation: Matrices can represent transformations (like rotations, scaling, or shearing).

They can also represent systems of equations or relationships between multiple variables.

Mathematical interpretation: Matrices are a way to organize and manipulate many numbers at once.

They're especially useful for representing linear relationships between multiple variables.

3.2 Notation and Conventions

Important: We use consistent notation throughout this document to avoid confusion.

Definition - Scalar: A scalar is a single real number: $x \in \mathbb{R}$

The symbol $\mathbb{R}$ represents the set of all real numbers (positive, negative, and zero).

Definition - Vector: A vector is an ordered array of scalars: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$

The superscript T means "transpose" - it flips the vector from a row to a column or vice versa.

$\mathbb{R}^n$ means "n-dimensional real space" - the set of all possible vectors with n real number components.

Definition - Matrix: A matrix is a rectangular array of scalars: $\mathbf{X} \in \mathbb{R}^{m \times n}$ with elements X_{ij}.

$\mathbb{R}^{m \times n}$ means the set of all matrices with m rows and n columns.

X_{ij} represents the element in the i-th row and j-th column.

Scalars: lowercase letters (x, y, z, a, b, c)

- These represent single numbers
- Example: $x = 5$

Vectors: lowercase bold letters ($\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$)

- These represent lists of numbers (column vectors by default)
- Example: $\mathbf{x} = [1, 2, 3]^T$

Matrices: uppercase bold letters ($\mathbf{A}$, $\mathbf{X}$, $\mathbf{Y}$)

- These represent rectangular arrays of numbers
- Example: $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

Functions:

- f, g, h for scalar-valued functions (output a single number)
- $\mathbf{f}$, $\mathbf{g}$ for vector-valued functions (output multiple numbers)
- $\mathbf{F}$, $\mathbf{G}$ for matrix-valued functions (output a matrix)

Derivatives:

- ∂ (partial derivative symbol)
- ∇ (gradient operator)
- $\mathbf{J}$ (Jacobian matrix)

- **H** (Hessian matrix)

3.3 Single-Variable Calculus Review

In single-variable calculus:

- The derivative $f'(x)$ tells us how much $f(x)$ changes when we make a small change to x.

Geometric interpretation:

- $f'(x)$ is the slope of the tangent line to the curve $y = f(x)$.

Physical interpretation:

- If $f(x)$ represents position at time x, then $f'(x)$ is velocity.

Rate interpretation:

- If $f(x)$ represents profit when you sell x items, then $f'(x)$ tells you how much extra profit you get from selling one more item.

3.4 Multiple-Variable Calculus

In multiple variables: When we have functions of multiple variables, the situation becomes more complex.

Consider $f(x, y) = x^2 + y^2$. This function takes two inputs and gives one output.

The question becomes: How does f change when we change x? When we change y? When we change both?

This leads to partial derivatives:

- $\frac{\partial f}{\partial x}$ tells us how f changes when we change x while keeping y fixed
- $\frac{\partial f}{\partial y}$ tells us how f changes when we change y while keeping x fixed

When we arrange these partial derivatives into a vector, we get the gradient:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The gradient tells us:

- The direction of steepest increase of the function
- How fast the function increases in that direction

> **Key Insight**
>
> In matrix calculus, the derivative must capture how each component of the output changes with respect to each component of the input. This means we need to keep track of many partial derivatives at once, which is where matrices become essential.

4 The Matrix Calculus Derivatives Table

The heart of matrix calculus can be summarized in a simple table.

This table shows what type of mathematical object you get when you take derivatives of different combinations of inputs and outputs.

Function Type	**Scalar Variable**	**Vector Variable**	**Matrix Variable**
Scalar Function	$\frac{df}{dx}$	$\frac{\partial f}{\partial \mathbf{x}}$	$\frac{\partial f}{\partial \mathbf{X}}$
Vector Function	$\frac{d\mathbf{f}}{dx}$	$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$	—
Matrix Function	$\frac{d\mathbf{F}}{dx}$	—	—

How to read this table:

Rows represent what you're differentiating (the function).

- "Scalar Function" means a function that outputs a single number
- "Vector Function" means a function that outputs multiple numbers (a vector)
- "Matrix Function" means a function that outputs a matrix

Columns represent what you're differentiating with respect to (the variable).

- "Scalar Variable" means the input is a single number
- "Vector Variable" means the input is a vector
- "Matrix Variable" means the input is a matrix

Entries show the notation used for that type of derivative.

Dashes (—) indicate cases that are either rarely used in practice or require advanced tensor notation.

Let's understand each entry:

- $\frac{df}{dx}$: A scalar function of a scalar variable. This is ordinary calculus, one input, one output, one derivative.
- $\frac{\partial f}{\partial \mathbf{x}}$: A scalar function of a vector variable gives us a gradient vector.
- $\frac{\partial f}{\partial \mathbf{X}}$: A scalar function of a matrix variable gives us a matrix of partial derivatives.
- $\frac{d\mathbf{f}}{dx}$: A vector function of a scalar variable - we differentiate each component.
- $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$: A vector function of a vector variable gives us the Jacobian matrix.
- $\frac{d\mathbf{F}}{dx}$: A matrix function of a scalar variable - we differentiate each matrix element.

Don't worry if this seems abstract now - we'll go through each case with detailed examples.

5 Detailed Analysis of Each Case

Now let's examine each entry in the table with detailed explanations, examples, and interpretations.

5.1 Case 1: Scalar Function, Scalar Variable ($\frac{\partial f}{\partial x}$)

> **i** Quick Reference
>
> - **Function Type**: $f : \mathbb{R} \to \mathbb{R}$
> - **Example**: $f(x) = x^2$
> - **Input**: Scalar x (e.g., $x = 2$)
> - **Output**: Scalar $\frac{\partial f}{\partial x}$ (e.g., $2x = 4$)
> - **Interpretation**: Rate of change

This is the familiar case from single-variable calculus that you learned in your first calculus course.

Definition: Given a scalar function $f : \mathbb{R} \to \mathbb{R}$, the derivative with respect to scalar x is:

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

What this definition means: We're asking: "If I change x by a tiny amount h, how much does $f(x)$ change?"

The ratio $\frac{f(x+h)-f(x)}{h}$ gives us the average rate of change over the interval h.

As h gets smaller and smaller (approaches 0), this ratio approaches the instantaneous rate of change.

Simple Example: Let $f(x) = x^3 + 2x^2 - 5x + 7$

To find the derivative, we use the power rule:

- The derivative of x^3 is $3x^2$
- The derivative of $2x^2$ is $4x$
- The derivative of $-5x$ is -5
- The derivative of 7 (a constant) is 0

Therefore: $\frac{\partial f}{\partial x} = 3x^2 + 4x - 5$

What this derivative tells us: At any point x, the derivative gives us the instantaneous rate of change of f.

At $x = 0$: $\frac{\partial f}{\partial x} = -5$ (the function is decreasing at a rate of 5 units per unit of x)

At $x = 1$: $\frac{\partial f}{\partial x} = 3(1)^2 + 4(1) - 5 = 2$ (the function is increasing at a rate of 2 units per unit of x)

Geometric interpretation: The derivative represents the slope of the tangent line to the curve $y = f(x)$ at any point x.

Physical interpretation: If $f(x)$ represents position at time x, then $f'(x)$ is velocity.

Economic interpretation: If $f(x)$ represents profit when selling x items, then $f'(x)$ is marginal profit (extra profit from selling one more item).

5.2 Case 2: Scalar Function, Vector Variable ($\frac{\partial f}{\partial \mathbf{x}}$)

> **i** Quick Reference
>
> - **Function Type**: $f : \mathbb{R}^n \to \mathbb{R}$
> - **Example**: $f(\mathbf{x}) = x_1^2 + x_2^2$
> - **Input**: Vector $\mathbf{x} \in \mathbb{R}^n$ (e.g., $\mathbf{x} = [1, 2]^T$)
> - **Output**: Gradient vector $\nabla f \in \mathbb{R}^n$
> - **Result**: $f(\mathbf{x}) = 1^2 + 2^2 = 5$
> - **Interpretation**: Direction of steepest increase

This is where things get more interesting and where matrix calculus really begins.

We have a function that takes multiple inputs (arranged in a vector) and produces a single output.

Definition: Given a scalar function $f : \mathbb{R}^n \to \mathbb{R}$ and vector $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$, the gradient is:

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

What this means: Instead of one derivative, we now have n derivatives - one for each input variable.

Each partial derivative $\frac{\partial f}{\partial x_i}$ tells us how f changes when we change x_i while holding all other variables constant.

The gradient is the vector that collects all these partial derivatives.

Detailed Example: Let $f(\mathbf{x}) = x_1^2 + 3x_1x_2 + x_2^2$ where $\mathbf{x} = [x_1, x_2]^T$

Step 1: Find $\frac{\partial f}{\partial x_1}$ Treat x_2 as a constant and differentiate with respect to x_1: $\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}(x_1^2 + 3x_1x_2 + x_2^2) = 2x_1 + 3x_2$

Step 2: Find $\frac{\partial f}{\partial x_2}$ Treat x_1 as a constant and differentiate with respect to x_2: $\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}(x_1^2 + 3x_1x_2 + x_2^2) = 3x_1 + 2x_2$

Step 3: Form the gradient vector $\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} 2x_1 + 3x_2 \\ 3x_1 + 2x_2 \end{bmatrix}$

Numerical example at a specific point: At the point $\mathbf{x} = [1, 2]^T$:

$$\left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=[1,2]^T} = \begin{bmatrix} 2(1) + 3(2) \\ 3(1) + 2(2) \end{bmatrix} = \begin{bmatrix} 8 \\ 7 \end{bmatrix}$$

What this means: At the point $(1, 2)$, if we increase x_1 by a small amount while keeping $x_2 = 2$, the function increases at a rate of 8.

If we increase x_2 by a small amount while keeping $x_1 = 1$, the function increases at a rate of 7.

Geometric interpretation: The gradient vector points in the direction of steepest increase of the function.

Physical interpretation: If f represents elevation on a hill and $\mathbf{x}$ represents your position, the gradient points uphill in the steepest direction.

Magnitude interpretation: The magnitude (length) of the gradient vector tells us how steep the hill is in that direction.

Optimization connection: In optimization, we often want to find where $\nabla f = \mathbf{0}$ (the zero vector).

These are critical points where the function might have local minima, maxima, or saddle points.

Key Point

The gradient is fundamental to gradient descent optimization.
To minimize $f(\mathbf{x})$, we update $\mathbf{x}$ in the direction opposite to the gradient:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)$$

where $\alpha > 0$ is the learning rate (step size).
This is like rolling a ball downhill - it naturally moves in the direction opposite to the gradient.

5.3 Case 3: Scalar Function, Matrix Variable ($\frac{\partial f}{\partial \mathbf{X}}$)

Quick Reference

- **Function Type**: $f : \mathbb{R}^{m \times n} \to \mathbb{R}$
- **Example**: $f(\mathbf{X}) = \text{tr}(\mathbf{X})$
- **Input**: Matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ (e.g., $\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$)
- **Output**: Matrix $\frac{\partial f}{\partial \mathbf{X}} \in \mathbb{R}^{m \times n}$
- **Result**: $f(\mathbf{X}) = \text{tr}(\mathbf{X}) = 1 + 4 = 5$
- **Interpretation**: Sensitivity to each matrix element

Now we consider functions that take an entire matrix as input and produce a single scalar output.

This might seem abstract, but it's actually very common in applications.

Definition: Given a scalar function $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ and matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, the derivative is:

$$\frac{\partial f}{\partial \mathbf{X}} = \left[\frac{\partial f}{\partial X_{ij}} \right] \in \mathbb{R}^{m \times n}$$

What this means: We compute the partial derivative of f with respect to each element of the matrix $\mathbf{X}$.

The result is a matrix of the same size as $\mathbf{X}$, where each element is a partial derivative.

Example 1: The Trace Function The trace of a square matrix is the sum of its diagonal elements.

For a 3×3 matrix: $\text{tr}(\mathbf{X}) = X_{11} + X_{22} + X_{33}$

Let $f(\mathbf{X}) = \text{tr}(\mathbf{X})$ where $\mathbf{X} \in \mathbb{R}^{n \times n}$.

Step 1: Understand what tr($\mathbf{X}$) means $\text{tr}(\mathbf{X}) = X_{11} + X_{22} + \cdots + X_{nn} = \sum_{i=1}^{n} X_{ii}$

Step 2: Find the partial derivatives $\frac{\partial f}{\partial X_{ij}} = \frac{\partial}{\partial X_{ij}}(X_{11} + X_{22} + \cdots + X_{nn})$

Case 1: $i = j$ **(diagonal elements)** $\frac{\partial f}{\partial X_{ii}} = 1$ (because X_{ii} appears once in the sum)

Case 2: $i \neq j$ **(off-diagonal elements)** $\frac{\partial f}{\partial X_{ij}} = 0$ (because X_{ij} doesn't appear in the sum at all)

Step 3: Form the derivative matrix $\frac{\partial f}{\partial \mathbf{X}} = \mathbf{I}_n$ (the $n \times n$ identity matrix)

What this means: The trace function is "sensitive" only to changes in diagonal elements.

If you change any diagonal element by 1, the trace increases by 1.

If you change any off-diagonal element, the trace doesn't change at all.

Example 2: The Frobenius Norm Squared The Frobenius norm of a matrix is like the "length" of the matrix when viewed as a big vector.

$f(\mathbf{X}) = \|\mathbf{X}\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij}^2$

This is the sum of squares of all elements in the matrix.

Step 1: Find the partial derivative $\frac{\partial f}{\partial X_{ij}} = \frac{\partial}{\partial X_{ij}} \left(\sum_{k=1}^{m} \sum_{l=1}^{n} X_{kl}^2 \right) = 2X_{ij}$

Step 2: Form the derivative matrix $\frac{\partial f}{\partial \mathbf{X}} = 2\mathbf{X}$

What this means: The rate of change of the Frobenius norm squared is proportional to the matrix itself.

Large elements contribute more to the rate of change than small elements.

Applications: This result is crucial in matrix optimization problems and regularization techniques.

For example, in machine learning, we often add a term like $\lambda\|\mathbf{W}\|_F^2$ to prevent weights from getting too large.

5.4 Case 4: Vector Function, Scalar Variable ($\frac{d\mathbf{f}}{dx}$)

> **Quick Reference**
>
> - **Function Type**: $\mathbf{f} : \mathbb{R} \to \mathbb{R}^m$
> - **Example**: $\mathbf{f}(x) = [x^2, x^3]^T$
> - **Input**: Scalar x (e.g., $x = 2$)
> - **Output**: Vector $\frac{d\mathbf{f}}{dx} \in \mathbb{R}^m$
> - **Result**: $\frac{d\mathbf{f}}{dx} = [2x, 3x^2]^T = [4, 12]^T$
> - **Interpretation**: Rate of change for each component

Now we consider functions that take a single scalar input and produce multiple scalar outputs (arranged in a vector).

Definition: Given a vector function $\mathbf{f} : \mathbb{R} \to \mathbb{R}^m$ with $\mathbf{f}(x) = [f_1(x), f_2(x), \ldots, f_m(x)]^T$, the derivative is:

$$\frac{d\mathbf{f}}{dx} = \begin{bmatrix} \frac{df_1}{dx} \\ \frac{df_2}{dx} \\ \vdots \\ \frac{df_m}{dx} \end{bmatrix} \in \mathbb{R}^m$$

What this means: We have m different functions, each depending on the same scalar variable x.

We differentiate each function separately with respect to x.

The result is a vector where each component is the derivative of the corresponding component function.

Detailed Example: Let $\mathbf{f}(t) = [\cos(t), \sin(t), t^2]^T$ where t is a scalar parameter.

Step 1: Identify the component functions

- $f_1(t) = \cos(t)$
- $f_2(t) = \sin(t)$
- $f_3(t) = t^2$

Step 2: Differentiate each component

- $\frac{df_1}{dt} = \frac{d}{dt}[\cos(t)] = -\sin(t)$
- $\frac{df_2}{dt} = \frac{d}{dt}[\sin(t)] = \cos(t)$
- $\frac{df_3}{dt} = \frac{d}{dt}[t^2] = 2t$

Step 3: Form the derivative vector $\frac{d\mathbf{f}}{dt} = \begin{bmatrix} -\sin(t) \\ \cos(t) \\ 2t \end{bmatrix}$

Physical interpretation: If $\mathbf{f}(t)$ represents the position vector of a particle moving in 3D space as a function of time t, then $\frac{d\mathbf{f}}{dt}$ is the velocity vector.

Geometric interpretation: If $\mathbf{f}(t)$ traces out a curve in 3D space, then $\frac{d\mathbf{f}}{dt}$ is the tangent vector to that curve.

Component analysis: At any time t:

- The x-component of velocity is $-\sin(t)$
- The y-component of velocity is $\cos(t)$
- The z-component of velocity is $2t$

Specific example at $t = \frac{\pi}{2}$**:** $\left.\frac{d\mathbf{f}}{dt}\right|_{t=\frac{\pi}{2}} = \begin{bmatrix} -\sin\left(\frac{\pi}{2}\right) \\ \cos\left(\frac{\pi}{2}\right) \\ 2\left(\frac{\pi}{2}\right) \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ \pi \end{bmatrix}$

What this means: At time $t = \frac{\pi}{2}$, the particle is moving in the negative x-direction at speed 1, not moving in the y-direction, and moving in the positive z-direction at speed π.

5.5 Case 5: Vector Function, Vector Variable ($\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$)

> Quick Reference
>
> - **Function Type**: $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$
> - **Example**: $\mathbf{f}(\mathbf{x}) = [x_1^2 + x_2, x_1 x_2]^T$
> - **Input**: Vector $\mathbf{x} \in \mathbb{R}^n$ (e.g., $\mathbf{x} = [1, 2]^T$)
> - **Output**: Jacobian matrix $\mathbf{J} \in \mathbb{R}^{m \times n}$
> - **Result**: $\mathbf{J} = \begin{bmatrix} 2x_1 & 1 \\ x_2 & x_1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}$
> - **Interpretation**: Linear approximation of function

This is one of the most important cases in matrix calculus.

We have a function that takes multiple inputs and produces multiple outputs.

This produces the Jacobian matrix, which is fundamental to multivariable calculus and optimization.

Definition: Given a vector function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ with $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T$, the Jacobian matrix is:

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

What this means: We have m functions, each depending on n variables.

The Jacobian is an $m \times n$ matrix where:

- Each row contains the gradient of one component function
- Each column shows how all outputs change with respect to one input

The (i, j) element of the Jacobian is $\frac{\partial f_i}{\partial x_j}$:

- This tells us how the i-th output changes when we change the j-th input

Detailed Example: Let $\mathbf{f}(\mathbf{x}) = [x_1^2 + x_2, x_1 x_2, \sin(x_1) + \cos(x_2)]^T$ where $\mathbf{x} = [x_1, x_2]^T$

Step 1: Identify the component functions

- $f_1(\mathbf{x}) = x_1^2 + x_2$
- $f_2(\mathbf{x}) = x_1 x_2$
- $f_3(\mathbf{x}) = \sin(x_1) + \cos(x_2)$

Step 2: Compute partial derivatives for each row

Row 1 (gradient of f_1):

- $\frac{\partial f_1}{\partial x_1} = \frac{\partial}{\partial x_1}(x_1^2 + x_2) = 2x_1$
- $\frac{\partial f_1}{\partial x_2} = \frac{\partial}{\partial x_2}(x_1^2 + x_2) = 1$

Row 2 (gradient of f_2):

- $\frac{\partial f_2}{\partial x_1} = \frac{\partial}{\partial x_1}(x_1 x_2) = x_2$
- $\frac{\partial f_2}{\partial x_2} = \frac{\partial}{\partial x_2}(x_1 x_2) = x_1$

Row 3 (gradient of f_3):

- $\frac{\partial f_3}{\partial x_1} = \frac{\partial}{\partial x_1}(\sin(x_1) + \cos(x_2)) = \cos(x_1)$
- $\frac{\partial f_3}{\partial x_2} = \frac{\partial}{\partial x_2}(\sin(x_1) + \cos(x_2)) = -\sin(x_2)$

Step 3: Form the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} 2x_1 & 1 \\ x_2 & x_1 \\ \cos(x_1) & -\sin(x_2) \end{bmatrix}$$

Numerical example at a specific point: At the point $\mathbf{x} = [1, 0]^T$:

$$\mathbf{J}\Big|_{\mathbf{x}=[1,0]^T} = \begin{bmatrix} 2(1) & 1 \\ 0 & 1 \\ \cos(1) & -\sin(0) \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ \cos(1) & 0 \end{bmatrix}$$

What each element means:

- $J_{11} = 2$: If we increase x_1 slightly, f_1 increases at rate 2
- $J_{12} = 1$: If we increase x_2 slightly, f_1 increases at rate 1
- $J_{21} = 0$: If we increase x_1 slightly, f_2 doesn't change (at this point)
- $J_{22} = 1$: If we increase x_2 slightly, f_2 increases at rate 1
- And so on...

i Theorem (Chain Rule for Jacobians)

If $\mathbf{g} : \mathbb{R}^p \to \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, then the Jacobian of the composition $\mathbf{h}(\mathbf{x}) = \mathbf{f}(\mathbf{g}(\mathbf{x}))$ is:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{g}}\right)\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right)$$

What the chain rule means: If you have a composition of functions (one function feeding into another), the derivative of the composition is the product of the individual Jacobians.
This is the foundation of backpropagation in neural networks: Neural networks are compositions of many simple functions, and we use the chain rule to compute how the final output depends on all the parameters.

Applications of the Jacobian:

Newton's Method for solving equations: To solve $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, we use the update rule: $\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}(\mathbf{x}_k)^{-1}\mathbf{f}(\mathbf{x}_k)$

Linear approximation: Near a point $\mathbf{a}$, we can approximate: $\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{a}) + \mathbf{J}(\mathbf{a})(\mathbf{x} - \mathbf{a})$

Change of variables in integration: When changing variables in multiple integrals, the determinant $|\det(\mathbf{J})|$ appears as the "scaling factor."

5.6 Case 6: Matrix Function, Scalar Variable ($\frac{d\mathbf{F}}{dx}$)

> Quick Reference
>
> - **Function Type**: $\mathbf{F} : \mathbb{R} \to \mathbb{R}^{m \times n}$
> - **Example**: $\mathbf{F}(x) = \begin{bmatrix} x & x^2 \\ x^3 & x^4 \end{bmatrix}$
> - **Input**: Scalar x (e.g., $x = 2$)
> - **Output**: Matrix $\frac{d\mathbf{F}}{dx} \in \mathbb{R}^{m \times n}$
> - **Result**: $\frac{d\mathbf{F}}{dx} = \begin{bmatrix} 1 & 2x \\ 3x^2 & 4x^3 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 12 & 32 \end{bmatrix}$
> - **Interpretation**: Element-wise rate of change

Finally, we consider functions that take a scalar input and produce a matrix output.

Definition: Given a matrix function $\mathbf{F} : \mathbb{R} \to \mathbb{R}^{m \times n}$ with elements $F_{ij}(x)$, the derivative is:

$$\frac{d\mathbf{F}}{dx} = \left[\frac{dF_{ij}}{dx}\right] \in \mathbb{R}^{m \times n}$$

What this means: Each element of the matrix $\mathbf{F}$ is a function of the scalar x.

We differentiate each element separately.

The result is a matrix of the same size, where each element is the derivative of the corresponding element in $\mathbf{F}$.

Detailed Example: Let $\mathbf{F}(t) = \begin{bmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{bmatrix}$ (a 2×2 rotation matrix)

Step 1: Identify each matrix element as a function of t

- $F_{11}(t) = \cos(t)$
- $F_{12}(t) = \sin(t)$
- $F_{21}(t) = -\sin(t)$
- $F_{22}(t) = \cos(t)$

Step 2: Differentiate each element

- $\frac{dF_{11}}{dt} = \frac{d}{dt}[\cos(t)] = -\sin(t)$
- $\frac{dF_{12}}{dt} = \frac{d}{dt}[\sin(t)] = \cos(t)$
- $\frac{dF_{21}}{dt} = \frac{d}{dt}[-\sin(t)] = -\cos(t)$
- $\frac{dF_{22}}{dt} = \frac{d}{dt}[\cos(t)] = -\sin(t)$

Step 3: Form the derivative matrix $\frac{d\mathbf{F}}{dt} = \begin{bmatrix} -\sin(t) & \cos(t) \\ -\cos(t) & -\sin(t) \end{bmatrix}$

Physical interpretation: The original matrix $\mathbf{F}(t)$ represents a rotation by angle t.

The derivative $\frac{d\mathbf{F}}{dt}$ represents the rate of rotation.

Geometric interpretation: As t changes, the matrix $\mathbf{F}(t)$ rotates vectors in the plane.

The derivative tells us how fast this rotation is happening.

6 The Shape Rule: A Universal Principle

One of the most important concepts in matrix calculus is understanding the dimensions (shape) of derivatives.

This helps you check your work and understand the structure of the mathematics.

> **Theorem** (The Shape Rule)
>
> The derivative $\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$ has dimensions that are determined by the dimensions of $\mathbf{Y}$ and $\mathbf{X}$.
> If $\mathbf{Y}$ has $p \times q$ elements and $\mathbf{X}$ has $r \times s$ elements, then the derivative conceptually lives in a space with $p \times q \times r \times s$ dimensions.
> In practice, we organize these derivatives in a way that makes computational sense.

Don't worry if this seems abstract - let's look at practical rules:

6.1 Practical Shape Rules

Rule 1: Scalar function of vector variable If $f : \mathbb{R}^n \to \mathbb{R}$, then $\frac{\partial f}{\partial \mathbf{x}} \in \mathbb{R}^n$

Why this makes sense:

- We have 1 output (scalar)
- We have n inputs (vector components)
- So we need n partial derivatives (one for each input)
- Result: a vector with n components

Rule 2: Vector function of vector variable If $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, then $\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \in \mathbb{R}^{m \times n}$

Why this makes sense:

- We have m outputs (vector components)
- We have n inputs (vector components)
- For each output, we need the partial derivative with respect to each input

- So we need $m \times n$ partial derivatives
- Result: an $m \times n$ matrix (m rows, n columns)

Rule 3: Scalar function of matrix variable If $f : \mathbb{R}^{m \times n} \to \mathbb{R}$, then $\frac{\partial f}{\partial \mathbf{X}} \in \mathbb{R}^{m \times n}$

Why this makes sense:

- We have 1 output (scalar)
- We have $m \times n$ inputs (matrix elements)
- We need one partial derivative for each matrix element
- Result: a matrix with the same shape as the input matrix

6.2 Memory Aid for Shape Rules

The key insight: The derivative has the same "shape structure" as the variable you're differentiating with respect to, combined with the output structure.

Simple way to remember:

1. Look at what you're differentiating (the function output)
2. Look at what you're differentiating with respect to (the variable)
3. The derivative combines information about both

Examples to cement the concept:

Gradient example:

- Function: $f(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^3$ (3-dimensional vector)
- Output: scalar (1 number)
- Derivative: $\nabla f \in \mathbb{R}^3$ (3 numbers - one partial derivative for each input component)

Jacobian example:

- Function: $\mathbf{f}(\mathbf{x})$ where $\mathbf{f} \in \mathbb{R}^2$ and $\mathbf{x} \in \mathbb{R}^3$
- Output: 2 numbers (vector with 2 components)
- Input: 3 numbers (vector with 3 components)
- Derivative: $\mathbf{J} \in \mathbb{R}^{2 \times 3}$ (2x3 matrix - for each of the 2 outputs, we need partial derivatives with respect to each of the 3 inputs)

> **Note**
>
> **Key Point**: The shape rule helps you quickly verify if your derivative calculations are correct.
> If the dimensions don't match what the shape rule predicts, you've made an error somewhere.

Always check dimensions as a sanity check!

7 Important Derivative Formulas

Here are the most commonly used derivative formulas in matrix calculus.

Don't try to memorize these all at once - focus on understanding the patterns and refer back to this section as needed.

7.1 Linear Forms

Linear forms are expressions where variables appear to the first power only (no squares, products, etc.).

Theorem (Linear Form Derivatives)

Let $\mathbf{a}$ be a constant vector and $\mathbf{A}$ be a constant matrix. Then:
Formula 1: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{a}^T\mathbf{x}) = \mathbf{a}$
Formula 2: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{a}) = \mathbf{a}$
Formula 3: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x}) = \mathbf{A}^T$
Formula 4: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}) = \mathbf{A}$

Example for Formula 1: Let $\mathbf{a} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

Then $\mathbf{a}^T\mathbf{x} = 2x_1 + 3x_2 + x_3$

Taking partial derivatives:

- $\frac{\partial}{\partial x_1}(2x_1 + 3x_2 + x_3) = 2$
- $\frac{\partial}{\partial x_2}(2x_1 + 3x_2 + x_3) = 3$
- $\frac{\partial}{\partial x_3}(2x_1 + 3x_2 + x_3) = 1$

So $\frac{\partial}{\partial \mathbf{x}}(\mathbf{a}^T\mathbf{x}) = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} = \mathbf{a}$

7.2 Quadratic Forms

Quadratic forms involve variables raised to the second power or products of variables.

Theorem (Quadratic Form Derivatives)

Let $\mathbf{A}$ be a constant matrix. Then:
Formula 1: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$
Formula 2: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{x}) = 2\mathbf{x}$

Detailed Example for Formula 1: Let $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Step 1: Expand the quadratic form

$$\begin{aligned} \mathbf{x}^T\mathbf{A}\mathbf{x} &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2x_1 + x_2 \\ 3x_1 + 4x_2 \end{bmatrix} \\ &= 2x_1^2 + x_1x_2 + 3x_1x_2 + 4x_2^2 \\ &= 2x_1^2 + 4x_1x_2 + 4x_2^2 \end{aligned}$$

Step 2: Take partial derivatives $\frac{\partial}{\partial x_1}(2x_1^2 + 4x_1x_2 + 4x_2^2) = 4x_1 + 4x_2$ $\frac{\partial}{\partial x_2}(2x_1^2 + 4x_1x_2 + 4x_2^2) = 4x_1 + 8x_2$

So $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = \begin{bmatrix} 4x_1 + 4x_2 \\ 4x_1 + 8x_2 \end{bmatrix}$

Step 3: Verify using the formula $\mathbf{A} + \mathbf{A}^T = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 4 & 8 \end{bmatrix}$

$(\mathbf{A} + \mathbf{A}^T)\mathbf{x} = \begin{bmatrix} 4 & 4 \\ 4 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4x_1 + 4x_2 \\ 4x_1 + 8x_2 \end{bmatrix}$

Special case - when A is symmetric: If $\mathbf{A} = \mathbf{A}^T$ (symmetric matrix), then $\mathbf{A}+\mathbf{A}^T = 2\mathbf{A}$

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

7.3 Matrix Trace Derivatives

The trace of a matrix is the sum of its diagonal elements: $\text{tr}(\mathbf{A}) = A_{11} + A_{22} + \cdots + A_{nn}$

Theorem (Trace Derivatives)

Formula 1: $\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{X}) = \mathbf{I}$
Formula 2: $\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{AX}) = \mathbf{A}^T$
Formula 3: $\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{XA}) = \mathbf{A}^T$
Formula 4: $\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{AXB}) = \mathbf{A}^T\mathbf{B}^T$

Example for Formula 2: Let $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$

$$\mathbf{AX} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} = \begin{bmatrix} x_{11} + 2x_{21} & x_{12} + 2x_{22} \\ 3x_{11} + 4x_{21} & 3x_{12} + 4x_{22} \end{bmatrix}$$

$\text{tr}(\mathbf{AX}) = (x_{11} + 2x_{21}) + (3x_{12} + 4x_{22}) = x_{11} + 2x_{21} + 3x_{12} + 4x_{22}$

Taking partial derivatives:

- $\frac{\partial \text{tr}(\mathbf{AX})}{\partial x_{11}} = 1$
- $\frac{\partial \text{tr}(\mathbf{AX})}{\partial x_{12}} = 3$
- $\frac{\partial \text{tr}(\mathbf{AX})}{\partial x_{21}} = 2$
- $\frac{\partial \text{tr}(\mathbf{AX})}{\partial x_{22}} = 4$

So $\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{AX}) = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = \mathbf{A}^T$

7.4 Determinant and Inverse Derivatives

These formulas are more advanced but very important in statistics and optimization.

Theorem (Determinant and Inverse Derivatives)

For an invertible matrix $\mathbf{X}$:
Formula 1: $\frac{\partial}{\partial \mathbf{X}} \det(\mathbf{X}) = \det(\mathbf{X})(\mathbf{X}^{-1})^T$
Formula 2: $\frac{\partial}{\partial \mathbf{X}} \log\det(\mathbf{X}) = (\mathbf{X}^{-1})^T$

What this means: The derivative of the determinant involves both the determinant itself and the inverse transpose of the matrix.

Applications: - Maximum likelihood estimation for multivariate normal distributions - Optimization problems involving covariance matrices - Regularization in machine learning

8 Applications in Machine Learning

Now let's see how matrix calculus is used in real machine learning applications.

These examples show why understanding derivatives is crucial for modern AI and data science.

8.1 Linear Regression

Linear regression is one of the most fundamental machine learning algorithms.

The goal is to find the best line (or hyperplane) that fits through data points.

The Problem: Given data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, find the best linear relationship $y \approx \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$

In matrix form: We want to solve $\mathbf{y} \approx \mathbf{X}\beta$ where:

- $\mathbf{y} \in \mathbb{R}^n$ is the vector of target values
- $\mathbf{X} \in \mathbb{R}^{n \times (p+1)}$ is the matrix of input features (with a leading column of ones for the intercept β_0)
- $\beta \in \mathbb{R}^{p+1}$ is the vector of parameters we want to find

The objective function: We minimize the sum of squared errors: minimize $J(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|^2$

Step 1: Expand the objective function $J(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$

Let's expand this step by step: $J(\beta) = (\mathbf{y}^T - \beta^T\mathbf{X}^T)(\mathbf{y} - \mathbf{X}\beta) = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta$

Note: $\mathbf{y}^T\mathbf{X}\beta$ is a scalar, so $\mathbf{y}^T\mathbf{X}\beta = (\mathbf{y}^T\mathbf{X}\beta)^T = \beta^T\mathbf{X}^T\mathbf{y}$

Therefore: $J(\beta) = \mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta$

Step 2: Take the gradient with respect to β Using our derivative formulas:

- $\frac{\partial}{\partial\beta}(\mathbf{y}^T\mathbf{y}) = \mathbf{0}$ (constant with respect to β)
- $\frac{\partial}{\partial\beta}(-2\beta^T\mathbf{X}^T\mathbf{y}) = -2\mathbf{X}^T\mathbf{y}$ (linear form)
- $\frac{\partial}{\partial\beta}(\beta^T(\mathbf{X}^T\mathbf{X})\beta) = 2\mathbf{X}^T\mathbf{X}\beta$ (quadratic form with symmetric matrix $\mathbf{X}^T\mathbf{X}$)

So: $\nabla J = \frac{\partial J}{\partial\beta} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta$

Step 3: Set the gradient to zero and solve $\nabla J = \mathbf{0}$ $-2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\beta = \mathbf{0}$ $\mathbf{X}^T\mathbf{X}\beta = \mathbf{X}^T\mathbf{y}$ $\beta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

This is the famous normal equation for linear regression!

What this means: The optimal parameters β^* can be computed directly using matrix operations.

No iterative optimization needed - just matrix multiplication and inversion.

Geometric interpretation: The vector $\mathbf{X}\beta^*$ is the orthogonal projection of $\mathbf{y}$ onto the column space of $\mathbf{X}$.

Practical note: In practice, we often use QR decomposition or SVD instead of computing $(\mathbf{X}^T\mathbf{X})^{-1}$ directly for numerical stability.

8.2 Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that finds the directions of maximum variance in data.

The Problem: Given data points in high-dimensional space, find a lower-dimensional representation that preserves as much information as possible.

Mathematical formulation: Find the direction $\mathbf{w}$ (unit vector) that maximizes the variance of the data when projected onto $\mathbf{w}$.

The optimization problem: maximize $\mathbf{w}^T\mathbf{S}\mathbf{w}$ subject to $\mathbf{w}^T\mathbf{w} = 1$

where $\mathbf{S}$ is the sample covariance matrix of the data.

Step 1: Set up the Lagrangian We use the method of Lagrange multipliers to handle the constraint.

$L(\mathbf{w}, \lambda) = \mathbf{w}^T\mathbf{S}\mathbf{w} - \lambda(\mathbf{w}^T\mathbf{w} - 1)$

where λ is the Lagrange multiplier.

Step 2: Take the gradient with respect to w $\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T\mathbf{S}\mathbf{w}) - \lambda\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T\mathbf{w})$

Using our quadratic form formulas:

- $\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T\mathbf{S}\mathbf{w}) = (\mathbf{S} + \mathbf{S}^T)\mathbf{w} = 2\mathbf{S}\mathbf{w}$ (since $\mathbf{S}$ is symmetric)
- $\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T\mathbf{w}) = 2\mathbf{w}$

So: $\frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{S}\mathbf{w} - 2\lambda\mathbf{w}$

Step 3: Set the gradient to zero $2\mathbf{S}\mathbf{w} - 2\lambda\mathbf{w} = \mathbf{0}$ $\mathbf{S}\mathbf{w} = \lambda\mathbf{w}$

This is the eigenvalue equation!

What this means: The optimal direction $\mathbf{w}$ is an eigenvector of the covariance matrix $\mathbf{S}$.

The corresponding eigenvalue λ tells us how much variance is captured in that direction.

Complete solution:

- The first principal component is the eigenvector with the largest eigenvalue
- The second principal component is the eigenvector with the second largest eigenvalue
- And so on…

Geometric interpretation: PCA finds the axes along which the data varies the most.

Physical interpretation: If the data represents physical measurements, PCA finds the "natural coordinate system" of the data.

8.3 Neural Network Backpropagation

Backpropagation is the algorithm used to train neural networks.

It's essentially a systematic application of the chain rule to compute gradients efficiently.

Simple network example: Consider a single layer: $\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}$, $\mathbf{a} = \sigma(\mathbf{z})$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the input vector
- $\mathbf{W} \in \mathbb{R}^{m \times n}$ is the weight matrix
- $\mathbf{b} \in \mathbb{R}^m$ is the bias vector
- $\mathbf{z} \in \mathbb{R}^m$ is the pre-activation
- σ is an activation function (applied element-wise)
- $\mathbf{a} \in \mathbb{R}^m$ is the activation (output)

The goal: Compute $\frac{\partial L}{\partial \mathbf{W}}$ where L is some scalar loss function.

Step 1: Apply the chain rule We can think of this as $\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{W}}$.

Step 2: Compute each term

The term $\frac{\partial L}{\partial \mathbf{z}}$ represents the gradient of the loss with respect to the pre-activations. It is computed from upstream layers. Let's call this vector $\delta = \frac{\partial L}{\partial \mathbf{z}} \in \mathbb{R}^m$.

The term $\frac{\partial \mathbf{z}}{\partial \mathbf{W}}$ is more complex. Let's find the partial derivative of an element z_i with respect to an element W_{ij}. Since $z_i = \sum_k W_{ik} x_k + b_i$, we have $\frac{\partial z_i}{\partial W_{jk}} = \delta_{ij} x_k$, where δ_{ij} is the Kronecker delta.

Step 3: Combine using matrix operations A more direct approach is to consider the contribution of each weight W_{ij} to the loss L.

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}} = \delta_i x_j$$

If we arrange these into a matrix, we get:

$$\frac{\partial L}{\partial \mathbf{W}} = \delta \mathbf{x}^T$$

where $\delta = \frac{\partial L}{\partial \mathbf{z}} \in \mathbb{R}^m$ and $\mathbf{x}^T \in \mathbb{R}^{1 \times n}$, resulting in an $m \times n$ matrix, which is the correct shape for the gradient with respect to $\mathbf{W}$.

Key insight: This gradient computation can be done efficiently for networks with millions of parameters by systematically applying the chain rule layer by layer.

Why this works: The chain rule allows us to decompose the complex dependency of the loss on the weights into simple, local computations.

9 Advanced Topics and Extensions

Now let's explore some more advanced concepts that build on the foundation we've established.

9.1 Higher-Order Derivatives: The Hessian Matrix

Just as we can take second derivatives in single-variable calculus, we can take second derivatives in matrix calculus.

The most important second-order derivative is the Hessian matrix.

> **Definition**
>
> The **Hessian matrix** of a scalar function $f : \mathbb{R}^n \to \mathbb{R}$ is:
>
> $$\mathbf{H} = \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}^T} = \left[\frac{\partial^2 f}{\partial x_i \partial x_j} \right] \in \mathbb{R}^{n \times n}$$
>
> **What this means:** The Hessian is a square matrix where each element (i, j) is the second partial derivative $\frac{\partial^2 f}{\partial x_i \partial x_j}$.
>
> **Element-by-element:** $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$
> This tells us how the gradient in the i-th direction changes when we move in the j-th direction.
>
> **Important property:** If f is twice continuously differentiable, then the Hessian is symmetric: $\mathbf{H} = \mathbf{H}^T$. This is because $\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$ (Clairaut's theorem).

Detailed Example: Let $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}$ where $\mathbf{A}$ is a symmetric matrix.

Step 1: Find the gradient From our earlier formulas: $\nabla f = 2\mathbf{A}\mathbf{x}$

Step 2: Find the Hessian The Hessian is the Jacobian of the gradient vector: $\mathbf{H} = \frac{\partial}{\partial \mathbf{x}^T}(\nabla f) = \frac{\partial}{\partial \mathbf{x}^T}(2\mathbf{A}\mathbf{x})$ Using the linear form rule $\frac{\partial}{\partial \mathbf{x}}(\mathbf{M}\mathbf{x}) = \mathbf{M}^T$ and adapting for our layout convention: $\mathbf{H} = 2\mathbf{A}$

General result: For $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$ where $\mathbf{A}$ is symmetric, the Hessian is $\mathbf{H} = 2\mathbf{A}$.

9.1.1 Applications of the Hessian

Newton's Method for Optimization: To find the minimum of $f(\mathbf{x})$, Newton's method uses both first and second derivatives:

$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$

Geometric interpretation: Newton's method fits a quadratic approximation to the function at each step and jumps to the minimum of that quadratic.

Convergence properties: Newton's method converges much faster than gradient descent when near the optimum, but requires computing and inverting the Hessian.

Analyzing Critical Points: When $\nabla f(\mathbf{x}^*) = \mathbf{0}$ (critical point), the eigenvalues of $\mathbf{H}(\mathbf{x}^*)$ tell us about the nature of the critical point:

- **All eigenvalues positive:** $\mathbf{x}^*$ is a local minimum (the function curves upward in all directions)
- **All eigenvalues negative:** $\mathbf{x}^*$ is a local maximum (the function curves downward in all directions)
- **Mixed eigenvalues:** $\mathbf{x}^*$ is a saddle point (curves up in some directions, down in others)

Quadratic Approximations: Near a point $\mathbf{a}$, we can approximate f using the Taylor expansion:

$f(\mathbf{x}) \approx f(\mathbf{a}) + \nabla f(\mathbf{a})^T(\mathbf{x} - \mathbf{a}) + \frac{1}{2}(\mathbf{x} - \mathbf{a})^T\mathbf{H}(\mathbf{a})(\mathbf{x} - \mathbf{a})$

What each term means:

- $f(\mathbf{a})$: the function value at the expansion point
- $\nabla f(\mathbf{a})^T(\mathbf{x} - \mathbf{a})$: first-order (linear) change
- $\frac{1}{2}(\mathbf{x} - \mathbf{a})^T\mathbf{H}(\mathbf{a})(\mathbf{x} - \mathbf{a})$: second-order (quadratic) change

9.2 Matrix Differential Calculus

An alternative approach to matrix calculus uses matrix differentials.

This can be more intuitive for some calculations, especially when dealing with complex matrix expressions.

> **i** Definition
>
> The **differential** of a function $f(\mathbf{X})$ is defined by the first-order approximation: $f(\mathbf{X} + d\mathbf{X}) - f(\mathbf{X}) \approx df$ where df is a linear function of $d\mathbf{X}$. The relationship between the differential and the gradient is $df = \text{tr}\left(\left(\frac{\partial f}{\partial \mathbf{X}}\right)^T d\mathbf{X}\right)$.

What this means: Instead of computing partial derivatives directly, we find a linear approximation for how small changes in $\mathbf{X}$ (represented by $d\mathbf{X}$) affect f.

Basic differential rules:

Linearity: $d(\alpha\mathbf{A} + \beta\mathbf{B}) = \alpha d\mathbf{A} + \beta d\mathbf{B}$

Product rule: $d(\mathbf{AB}) = (d\mathbf{A})\mathbf{B} + \mathbf{A}(d\mathbf{B})$

This is the matrix version of the familiar product rule from calculus.

Inverse rule: $d(\mathbf{A}^{-1}) = -\mathbf{A}^{-1}(d\mathbf{A})\mathbf{A}^{-1}$

Example derivation: Since $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, taking differentials: $d(\mathbf{A}\mathbf{A}^{-1}) = d(\mathbf{I}) = \mathbf{0}$

Using the product rule: $(d\mathbf{A})\mathbf{A}^{-1} + \mathbf{A}(d\mathbf{A}^{-1}) = \mathbf{0}$

Solving for $d\mathbf{A}^{-1}$: $\mathbf{A}(d\mathbf{A}^{-1}) = -(d\mathbf{A})\mathbf{A}^{-1}$ $d\mathbf{A}^{-1} = -\mathbf{A}^{-1}(d\mathbf{A})\mathbf{A}^{-1}$

Trace rule: $d(\text{tr}(\mathbf{A})) = \text{tr}(d\mathbf{A})$

Determinant rule: $d(\det(\mathbf{A})) = \det(\mathbf{A})\text{tr}(\mathbf{A}^{-1}d\mathbf{A})$

9.3 Vectorization and Kronecker Products

When dealing with complex matrix derivatives, vectorization becomes a powerful tool.

> **i** Definition
>
> The **vectorization** operator $\text{vec}(\mathbf{A})$ stacks the columns of matrix $\mathbf{A}$ into a single column vector.

Example: If $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$, then $\text{vec}(\mathbf{A}) = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$

Key property: Vectorization converts matrix equations into vector equations, which are often easier to manipulate.

i Definition

The **Kronecker product** $\mathbf{A} \otimes \mathbf{B}$ is defined as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Example: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 2 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \\ 3 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 4 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{bmatrix}$

i Theorem (Vectorization Identity)

For matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ of appropriate dimensions:

$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$$

What this means: Matrix multiplication can be expressed as a linear transformation of the vectorized middle matrix.

Applications: This identity is crucial for:

- Converting matrix equations into vector form for optimization
- Deriving gradients for complex matrix expressions
- Solving matrix equations using vector methods

Example application: To solve the matrix equation $\mathbf{AXB} = \mathbf{C}$ for $\mathbf{X}$:

Step 1: Vectorize both sides $\text{vec}(\mathbf{AXB}) = \text{vec}(\mathbf{C})$

Step 2: Apply the vectorization identity $(\mathbf{B}^T \otimes \mathbf{A})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{C})$

Step 3: Solve the linear system $\text{vec}(\mathbf{X}) = (\mathbf{B}^T \otimes \mathbf{A})^{-1}\text{vec}(\mathbf{C})$

Step 4: Reshape back to matrix form $\mathbf{X} = \text{reshape}(\text{vec}(\mathbf{X}), \text{original dimensions})$

10 Computational Considerations

Understanding the theory is only half the battle - implementing matrix calculus efficiently and accurately is crucial for practical applications.

10.1 Automatic Differentiation (Autodiff)

Modern machine learning frameworks like TensorFlow, PyTorch, and JAX use automatic differentiation to compute derivatives.

This is different from both symbolic differentiation (like Mathematica) and numerical differentiation (finite differences).

The key insight: Every computer program, no matter how complex, is built from elementary operations ($+$, $-$, $\times$, $\div$, exp, log, sin, cos, etc.).

If we know the derivatives of these elementary operations, we can use the chain rule to compute derivatives of arbitrarily complex programs.

10.1.1 Forward Mode Autodiff

How it works: We compute derivatives alongside the original computation, moving forward through the computation graph.

Example: Suppose we want to compute $f(x_1, x_2) = \sin(x_1) + x_1 x_2$ and its partial derivatives.

Forward pass (computing function and derivatives simultaneously):

Step 1: Input values and seed derivatives

- $x_1 = 2$, $\dot{x}_1 = 1$ (for $\frac{\partial}{\partial x_1}$)
- $x_2 = 3$, $\dot{x}_2 = 0$

Step 2: Compute $\sin(x_1)$

- $v_1 = \sin(x_1) = \sin(2) \approx 0.909$
- $\dot{v}_1 = \cos(x_1) \cdot \dot{x}_1 = \cos(2) \cdot 1 \approx -0.416$

Step 3: Compute $x_1 x_2$

- $v_2 = x_1 x_2 = 2 \cdot 3 = 6$
- $\dot{v}_2 = \dot{x}_1 x_2 + x_1 \dot{x}_2 = 1 \cdot 3 + 2 \cdot 0 = 3$

Step 4: Compute $v_1 + v_2$

- $f = v_1 + v_2 = 0.909 + 6 = 6.909$
- $\dot{f} = \dot{v}_1 + \dot{v}_2 = -0.416 + 3 = 2.584$ (This is $\frac{\partial f}{\partial x_1}$)

To get $\frac{\partial f}{\partial x_2}$, we need another pass with $\dot{x}_1 = 0, \dot{x}_2 = 1$.

When forward mode is efficient: Forward mode is efficient when the number of inputs is much smaller than the number of outputs.

Example: $f : \mathbb{R}^2 \to \mathbb{R}^{1000}$ (2 inputs, 1000 outputs)

- Forward mode: 2 passes (one for each input)
- Reverse mode: 1000 passes (one for each output)

10.1.2 Reverse Mode Autodiff (Backpropagation)

How it works: We first compute the function value in a forward pass, then compute derivatives in a backward pass through the computation graph.

The key idea: Use the chain rule systematically, starting from the output and working backward.

Example (same function): $f(x_1, x_2) = \sin(x_1) + x_1 x_2$

Forward pass (compute function only):

- $x_1 = 2, x_2 = 3$
- $v_1 = \sin(x_1) \approx 0.909$
- $v_2 = x_1 x_2 = 6$
- $f = v_1 + v_2 = 6.909$

Backward pass (compute derivatives):

Step 1: Start with the "seed" $\bar{f} = \frac{\partial f}{\partial f} = 1$

Step 2: Backpropagate through addition ($f = v_1 + v_2$)

- $\bar{v}_1 = \bar{f} \frac{\partial f}{\partial v_1} = 1 \cdot 1 = 1$
- $\bar{v}_2 = \bar{f} \frac{\partial f}{\partial v_2} = 1 \cdot 1 = 1$

Step 3: Backpropagate through sin ($v_1 = \sin(x_1)$) and product ($v_2 = x_1 x_2$)

- $\bar{x}_1 = \bar{v}_1 \frac{\partial v_1}{\partial x_1} + \bar{v}_2 \frac{\partial v_2}{\partial x_1} = 1 \cdot \cos(x_1) + 1 \cdot x_2 = \cos(2) + 3 \approx 2.584$
- $\bar{x}_2 = \bar{v}_2 \frac{\partial v_2}{\partial x_2} = 1 \cdot x_1 = 2$

The results are $\frac{\partial f}{\partial x_1} = \bar{x}_1$ and $\frac{\partial f}{\partial x_2} = \bar{x}_2$.

When reverse mode is efficient: Reverse mode is efficient when the number of outputs is much smaller than the number of inputs.

Example: $f : \mathbb{R}^{10^6} \to \mathbb{R}$ (1 million inputs, 1 output)

- Forward mode: 1,000,000 passes
- Reverse mode: 1 pass

This is why neural networks can be trained efficiently!

> **Key Point**
>
> Reverse mode autodiff is why we can efficiently train neural networks with millions of parameters.
> It computes all partial derivatives of a scalar loss function in time roughly proportional to the forward pass.
> Without this, modern deep learning would be computationally infeasible.

10.2 Numerical Stability

When implementing matrix calculus operations, numerical errors can accumulate and cause serious problems.

Here are the most important considerations:

10.2.1 Matrix Inversion

The problem: Never compute $(\mathbf{X}^T\mathbf{X})^{-1}$ directly when $\mathbf{X}$ is tall and thin (more rows than columns).

Why this fails: The condition number of $\mathbf{X}^T\mathbf{X}$ is the square of the condition number of $\mathbf{X}$.

If $\mathbf{X}$ is slightly ill-conditioned, $\mathbf{X}^T\mathbf{X}$ becomes very ill-conditioned.

The solution: Use QR decomposition instead to solve the linear system $(\mathbf{X}^T\mathbf{X})\beta = \mathbf{X}^T\mathbf{y}$.

QR decomposition: Any matrix $\mathbf{X}$ can be written as $\mathbf{X} = \mathbf{QR}$ where:

- $\mathbf{Q}$ has orthonormal columns ($\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$)
- $\mathbf{R}$ is upper triangular

The system becomes $\mathbf{R}^T\mathbf{Q}^T\mathbf{QR}\beta = \mathbf{R}^T\mathbf{Q}^T\mathbf{y}$, which simplifies to $\mathbf{R}\beta = \mathbf{Q}^T\mathbf{y}$. This can be solved efficiently and stably using back-substitution.

10.2.2 Log-Sum-Exp Trick

The problem: Computing $\log(\exp(a_1) + \exp(a_2) + \cdots + \exp(a_n))$ directly can cause overflow or underflow.

Example: If $a_1 = 1000$, then $\exp(1000)$ is astronomically large and will overflow.

The solution: Factor out the maximum value: Let $a_{\max} = \max_i a_i$.

```
\begin{aligned}
\log(\sum_i \exp(a_i)) &= \log(\exp(a_{\max})\sum_i \exp(a_i - a_{\max})) \\
&= a_{\max} + \log(\sum_i \exp(a_i - a_{\max}))
\end{aligned}
```

Why this works: All the terms $\exp(a_i - a_{\max})$ are ≤ 1, so no overflow. At least one term equals 1, so the sum is at least 1, avoiding underflow in the log.

10.3 Implementation Tips

Vectorization: Use vectorized operations in libraries like NumPy/PyTorch instead of explicit loops whenever possible. This is significantly faster.

Bad (slow):

```
import numpy as np
A = np.random.rand(1000, 1000)
B = np.random.rand(1000, 1000)
C = np.zeros((1000, 1000))
for i in range(1000):
    for j in range(1000):
        C[i,j] = A[i,j] + B[i,j]
```

Good (fast):

```
C = A + B  # Vectorized operation
```

Broadcasting: Understand how broadcasting works in your framework to avoid unnecessary memory allocation and make code cleaner.

Batch operations: Process multiple data examples simultaneously when possible. This leverages parallel hardware (GPUs) for massive speedups.

11 Common Mistakes and Pitfalls

Learning matrix calculus involves avoiding several common traps.

Here are the most frequent mistakes and how to avoid them.

11.1 Dimension Mismatches

The problem: Matrix operations are only defined when dimensions are compatible.

Common mistake 1: Forgetting to transpose

Wrong: If $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{x} \in \mathbb{R}^m$, then $\mathbf{Ax}$ is undefined.

Right: You probably meant $\mathbf{A}^T\mathbf{x}$ (if you want the result in $\mathbb{R}^n$) or you have a mismatch in your variable definitions.

How to avoid dimension mistakes:

Always write dimensions explicitly: Instead of writing $\mathbf{Ax}$, write $\mathbf{A}_{m \times n}\mathbf{x}_{n \times 1} = \mathbf{y}_{m \times 1}$ during your derivation.

11.2 Chain Rule Errors

The problem: Matrix multiplication is not commutative, so order matters in the chain rule.

Wrong: If $\mathbf{y} = f(\mathbf{u})$ and $\mathbf{u} = g(\mathbf{x})$, writing $\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \frac{\partial \mathbf{y}}{\partial \mathbf{u}}$.

Right: $\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$.

How to avoid chain rule errors:

Always check dimensions: If $\mathbf{y} \in \mathbb{R}^m, \mathbf{u} \in \mathbb{R}^n, \mathbf{x} \in \mathbb{R}^p$:

- $\frac{\partial \mathbf{y}}{\partial \mathbf{u}}$ is $m \times n$
- $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ is $n \times p$
- $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ must be $m \times p$
- Check: $(m \times n)(n \times p) \to (m \times p)$

11.3 Forgetting Symmetry

The problem: Many important matrices (like Hessians and covariance matrices) are symmetric, but it's easy to forget this property and use a more complex general formula.

Missed optimization: If $\mathbf{A}$ is symmetric, then $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = 2\mathbf{A}\mathbf{x}$, which is simpler than the general form $(\mathbf{A} + \mathbf{A}^T)\mathbf{x}$.

How to avoid: Always check for symmetry before computing derivatives.

12 Practice Problems

Test your understanding with these carefully designed problems.

12.1 Basic Problems

Problem 1: Simple Gradients Find the gradient of $f(\mathbf{x}) = 3x_1^2 + 2x_1x_2 + x_2^2$ where $\mathbf{x} = [x_1, x_2]^T$.

Click for solution

Solution: $\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}(3x_1^2 + 2x_1x_2 + x_2^2) = 6x_1 + 2x_2$ $\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}(3x_1^2 + 2x_1x_2 + x_2^2) = 2x_1 + 2x_2$

Therefore: $\nabla f = \begin{bmatrix} 6x_1 + 2x_2 \\ 2x_1 + 2x_2 \end{bmatrix}$

Problem 2: Simple Jacobian Find the Jacobian of $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_1 + x_2 \\ x_1 - x_2 \end{bmatrix}$ where $\mathbf{x} = [x_1, x_2]^T$.

Click for solution

Solution: The function $\mathbf{f}$ has 2 outputs and 2 inputs, so the Jacobian is 2×2. $f_1(\mathbf{x}) = x_1 + x_2$ $f_2(\mathbf{x}) = x_1 - x_2$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

12.2 Intermediate Problems

Problem 3: Quadratic Forms Find $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x})$ where $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$.

Click for solution

Solution: Since $\mathbf{A}$ is symmetric, we can use the simplified rule $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = 2\mathbf{A}\mathbf{x}$.
Therefore: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = 2\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4x_1 + 2x_2 \\ 2x_1 + 6x_2 \end{bmatrix}$

12.3 Advanced Problems

Problem 4: Optimization Application Use matrix calculus to find the value of $\mathbf{x}$ that minimizes $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{c}^T\mathbf{x}$, where $\mathbf{Q}$ is a symmetric positive definite matrix.

Click for solution

Solution: To find the minimum, we must find where the gradient is zero.
Step 1: Find the gradient $\nabla f = \frac{\partial}{\partial \mathbf{x}}(\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}) - \frac{\partial}{\partial \mathbf{x}}(\mathbf{c}^T\mathbf{x})$ Using our formulas for quadratic and linear forms (and since $\mathbf{Q}$ is symmetric): $\nabla f = \frac{1}{2}(2\mathbf{Q}\mathbf{x}) - \mathbf{c} = \mathbf{Q}\mathbf{x} - \mathbf{c}$
Step 2: Set gradient to zero $\nabla f = \mathbf{0}$ $\mathbf{Q}\mathbf{x} - \mathbf{c} = \mathbf{0}$ $\mathbf{Q}\mathbf{x} = \mathbf{c}$ $\mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{c}$
Step 3: Verify it's a minimum The Hessian is $\mathbf{H} = \frac{\partial}{\partial \mathbf{x}^T}(\mathbf{Q}\mathbf{x} - \mathbf{c}) = \mathbf{Q}$. Since $\mathbf{Q}$ is positive definite, all its eigenvalues are positive, confirming that $\mathbf{x}^*$ is a local (and in this case, global) minimum.

13 Summary and Key Takeaways

13.1 The Fundamental Structure

The derivative's structure depends on the function's input and output:

1. **Scalar → Scalar:** Ordinary derivative (a scalar)
2. **Vector → Scalar:** Gradient (a vector)
3. **Matrix → Scalar:** Matrix of partials (a matrix)
4. **Vector → Vector:** Jacobian (a matrix)

13.2 The Shape Rule is Your Friend

The dimensions of derivatives follow predictable patterns. Use this to check your work.

- Gradient: $f : \mathbb{R}^n \to \mathbb{R}$ gives derivative $\in \mathbb{R}^n$
- Jacobian: $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ gives derivative $\in \mathbb{R}^{m \times n}$

13.3 Essential Formulas to Remember

- **Linear forms:** $\frac{\partial}{\partial \mathbf{x}}(\mathbf{a}^T\mathbf{x}) = \mathbf{a}$
- **Quadratic forms (A symmetric):** $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = 2\mathbf{A}\mathbf{x}$
- **Matrix Trace:** $\frac{\partial}{\partial \mathbf{X}}\text{tr}(\mathbf{A}\mathbf{X}) = \mathbf{A}^T$

13.4 The Big Picture

> **i** Final Key Point
>
> Matrix calculus is not just a mathematical abstraction - it's the computational foundation that makes modern machine learning possible. Every time you train a neural network, optimize a complex function, or analyze multivariate data, you're using these principles. The most important skill is to think systematically about how small changes in inputs propagate through functions to affect the final outputs.

14 Quick Reference Guide

14.1 Common Derivatives

Expression	Derivative	Notes
$\mathbf{a}^T\mathbf{x}$	$\mathbf{a}$	Linear form
$\mathbf{x}^T\mathbf{A}\mathbf{x}$	$(\mathbf{A} + \mathbf{A}^T)\mathbf{x}$	Quadratic form (general)
$\mathbf{x}^T\mathbf{A}\mathbf{x}$	$2\mathbf{A}\mathbf{x}$	Quadratic form (A is symmetric)
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$	Special case: $\mathbf{A} = \mathbf{I}$
$\text{tr}(\mathbf{A}\mathbf{X})$	$\mathbf{A}^T$	Trace of product
$\det(\mathbf{X})$	$\det(\mathbf{X})(\mathbf{X}^{-1})^T$	Determinant
$\log\det(\mathbf{X})$	$(\mathbf{X}^{-1})^T$	Log determinant
$\mathbf{A}\mathbf{x}$	$\mathbf{A}$ (Jacobian, denominator layout) or $\mathbf{A}^T$ (numerator layout)	Linear transformation

14.2 Shape Rules Quick Reference

Function Type	Variable Type	Derivative Shape	Example
Scalar	Vector $\in \mathbb{R}^n$	Vector $\in \mathbb{R}^n$	$f : \mathbb{R}^n \to \mathbb{R}$, $\frac{\partial f}{\partial \mathbf{x}}$
Scalar	Matrix $\in \mathbb{R}^{m \times n}$	Matrix $\in \mathbb{R}^{m \times n}$	$f : \mathbb{R}^{m \times n} \to \mathbb{R}$, $\frac{\partial f}{\partial \mathbf{X}}$
Vector $\in \mathbb{R}^m$	Vector $\in \mathbb{R}^n$	Matrix $\in \mathbb{R}^{m \times n}$	$\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$