

Chapter 1: Introduction

This chapter introduces the reader to tensor representations and their operations from the mathematical representation to the Python data structures. Starting from vectors, the presentation progresses to matrices and tensors. The focus is on studying vectors and matrices using Linear Algebra concepts that get generalised to Tensors that will be introduced in chapter three. Several references provide examples and applications of these concepts in their entirety. The primary operations that will be discussed in subsequent chapters are explained with some preliminary examples and applications. Every chance to visualise these concepts vividly is presented.

1.1 History

Structured data is used in computer algorithms in many different ways. The primary data structures for most algorithms are scalars, linear arrays, matrices, and multi-dimensional arrays. Multi-dimensional arrays are not tensor structures but share some of their properties. Tensors can be defined as functions of any point in space coordinates, which transform linearly between coordinate systems. The three-dimensional space has $3r$ components, where r is the rank. The tensors of rank zero are scalars, the tensors of rank one are vectors, and the tensors of rank two are matrices.

Since tensors transform linearly between coordinates, they are commonly used in differential non-Euclidean geometry to study curves and surfaces in three-dimensional space by using calculus techniques applicable in higher-dimensional spaces (Pressley, 2010; Fortney, 2018). A tensor may refer to different objects in different domains, for instance, a stress tensor, moment of inertia tensor, field tensor, metric tensor, and tensor product, which are defined in physics and are not what we aim to explain in this book. Tensors are rarely defined carefully, and the definition usually has to do with transformation properties and domain-specific definitions, making it difficult to visualise what these objects are. We focus on tensor definitions related to data mining and machine learning, including deep learning. Data conversion into information is aided by differential and integral calculus. Differential calculus is used for studying the rates of change, while integral calculus is used to study the definitions, properties, and applications of these two related concepts. In chapter three, we will explain more.

Motivation: What are the benefits of studying tensors in computer science disciplines? To answer this question, we need to discuss how tensor applications are implemented and how many algorithms have been developed to process data in tensor spaces. This book attempts

CHAPTER 1

to answer this question with various applications' essential mathematical backgrounds, algorithms, and code examples. A motivational example problem appears in section 1.5. In each chapter, more applications are discussed, with chapter six devoted entirely to applications.

Absolute differential calculus is the earliest foundation of tensor theory. It was developed by Gregorio Ricci-Curbastro in 1887–96 and subsequently popularised in a paper (Ricci and Levi-Civita, 1900) written with his student Tullio Levi-Civita. The general relativity theory described the geometry of gravitation in space-time curvature with respect to the energy and momentum of the matter and radiation in a system of partial differential equations. The following is a detailed timeline for developing tensor computing techniques. More theoretical concepts are mentioned in the timeline that this book will not cover but are referenced for the readers interested in a deeper dive.

1846: “Tensor” was first introduced by William Ron Hamilton and later became known to scientists through the publication of Levi-Civita's book “The Absolute Differential Calculus”.

1853: Matrix, Matrix Theory, and the principles of Universal Algebra are developed by Joseph Sylvester and Arthur Cayley.

1874: Set theory is developed by George Cantor; it represents collections of abstract objects, including notions like Venn diagrams and set memberships.

1908: Axiomatic Set Theory is developed by Ernst Zermelo, reformulating the now "naive set theory" in first-order logic to resolve its paradoxes, for example, Russell's paradox, the Burali-Forti paradox, and Cantor's paradox. This theory does not allow the construction of ordinal numbers, while most ordinary mathematics can be developed without using ordinals. The latter is an essential tool in most set-theoretic investigations.

1922: Abraham Fraenkel and Thoralf Skolem propose operationalising a definite property as one is formulated in first-order logic, with all atomic formulae involving set membership or identity. This adds the axioms of replacement and regularity, yielding the theory of ZF. Then adding the axiom of choice becomes the ZFC theory. This cannot be axiomatised by a finite set of axioms because of the replacement axiom.

1925: Werner Heisenberg, Max Born, and Pascual Jordan formulate matrix mechanics, a formulation of quantum mechanics.

1922-1940: Von Neumann-Bernays-Godel (NBG) set theory can be finitely axiomatised. The ontology of NBG includes classes as well as sets; a set is a class that is a member of another class. NBG and ZFC are equivalent set theories such that any theorem about sets is provable in NBG if and only if it is in ZFC.

1942-1945: Samuel Eilenberg and Saunders Mac Lane first introduced the category theory in connection with the algebraic topology. It has several aspects, such as “general abstract nonsense”. The latter refers to the high abstraction level compared to more classical branches of mathematics. Homological algebra is a category theory in organising and suggesting calculations in abstract algebra. Diagram chasing is a visual method of arguing with abstract "arrows". The topos theory is a form of abstract sheaf theory with geometric origins; it leads to ideas such as the pointless topology.

1964: Iverson uses the Array Programming Language (APL) APL notation to describe IBM's system 360. (FALKOFF, AD, Iverson, KE, Sussenguth, EH, 1964)

CHAPTER 1

1957 to 1965: APL is the first homogenous simple array programming language designed by Kenneth E. Iverson. The language works on entire arrays simultaneously, like the SIMD architecture's vector instruction set. It yields smaller and more concise programs though no iteration is involved.

1973: Based on APL, Trenched More proposes an array theory that offers a robust set of operators and operations on nested, heterogeneous rectangular arrays (MORE, T, 1973).

1979: Programming Language/Systems PL/S II or AT/370 languages is developed to implement More's array theory operations, using an APL interface. NIAL2 (Nested Interactive Array Language) is developed as a programming language based on array theory and its applications to make it easy to rapidly develop loop-free data-driven algorithms (JENKINS, MA, Franksen, OI, 1992). APL2 was another implementation of More's nested Array theory. For a while, it was IBM's strategic language for HPC.

1988: Mathematics of Arrays and the Ψ -Calculus were first introduced in the PhD thesis of Dr Lenore Mullin in Computer and Information Science at Syracuse University, Syracuse, NY. The thesis introduces an algebraic formulation representing all data structures invariant of dimensionality and shape. An MoA structure describes scalars as rank 0, linear arrays (vectors) as rank one, 2-D arrays (matrices) as rank two, and similar higher rank structures. The representation is stored in memory in a linear structure with elements stored in a row or column-major order in a linear array, a dimensionality scalar, and a shape vector. A list of constructs is provided. The Ψ -Calculus is a way to combine expressions in the MoA algebra by composing indices; it uses the Ψ -Function as its foundation. Mullin's dissertation put closure on work started by Phil Abrams ("An APL Machine", Stanford '72, Harold Stone advisor), who believed there was formalism for array reasoning based on shapes and indexing. His work was augmented by Hassett and Lyon, Guibas and Wyatt, Perlis, Miller, Minter, Tu, Gerhart, Berkling, and Budd, to name a few.

1990: A Comparison of Array Theory with Mathematics of Arrays is presented by L. Mullin and M. Jenkins. (JENKINS, MA, Mullin, LR, 1991).

1993: L. Mullin and G. Hains show how to use the Bird-Meertens Formalism to define MoA. (HAINS, G, Mullin, LR, 1993).

2000: MoA library was built as a dynamic link library (dll); it is then used in basic image and video processing applications in an MSc. thesis (Helal, 2001).

2001: Faster Fast Fourier Transform (FFT) and generalised Radix n FFT using MoA are presented by L. Mullin and S. Small (MULLIN, LR, Small, S, 2002).

2004: "Multi-Way Analysis with Applications in the Chemical Sciences" book detailed how multi-way PCA and Multi-way Factor analysis and other methods are applied to various computational chemistry problems. (Smilde, Bro and Geladi, 2004)

2005: L. Mullin used MoA and the Ψ -Calculus to map Digital Signal Processing (DSP) algorithms to multiple processor/memory hierarchies. "A Uniform way of reasoning about array-based computation in radar: Algebraically connecting the hardware/software boundary" presented by Mullin (MULLIN, LR, 2005). Mullin and Reynolds applied the Conformal Computing Techniques by using MoA and Ψ -Calculus to solve problems in Computational Physics (MULLIN, LR, Reynolds, J, 2005).

CHAPTER 1

2009: A comprehensive survey of multi-way analysis and their applications was presented (Kolda and Bader, 2009).

2009: The NSF held a workshop to decide future trends in tensor computation. In this workshop, researchers from mathematics, physics and computing presented state-of-the-art in the field.

2010: A PhD thesis applied the MoA methods in the high-dimensional scientific computation problem “Multiple Sequence Alignment in Bioinformatics” in the MSA dynamic programming algorithm to score a tensor of alignments. Partitioning is processed in parallel providing automatic load balancing (Helal, 2009).

2014: “Multilinear subspace Learning” detailed the advances from linear subspace learning applying dimensionality reduction algorithms based on linear algebra and how it scales to multilinear subspace learning through tensor projections and decompositions (Lu, Plataniotis and Venetsanopoulos, 2014).

2006 - onwards: various neural networks and data mining applications applied tensor decomposition proving advances in accuracy and efficient computation, using less memory and time. This led to being coined “Compressive Neural Networks” due to the performance benefits of employing tensor decomposition techniques.

1.2 Linear Algebra

Linear algebra can be better reviewed or studied for the first time with complete books such as (Chahal, 2018), (Carter, 1995) and (Dym, 2007). The Online Edx platform course Linear Algebra for Frontiers by The University of Texas at Austin is a valuable resource for learning efficient computation of linear algebra concepts (Geijn and Quintana-Ort, 2008). The AI algorithms’ mathematical foundations and their Python packages are covered in (Farrell, 2020). The intuition that this book is trying to build is to vividly understand the notion of projections from lower to higher spaces or from higher to lower spaces using linear algebra tools, then expanding to other types of Algebra in later chapters. Vector and matrix operations, including linear transformations and independence, must be understood to understand higher-order tensor operations and properties. Vector addition, subtraction, normalisation, dot product, cross product, outer product, and derivatives (rate of change) are the basic operations on vectors required for machine learning. The next section will focus on the main matrix operations and properties that will be necessary for the following chapters. The accompanying python notebook `ch1.ipynb` has a helpful review of the operations discussed in these two sections with some preliminary operations and visualisations.

Motivation: Machine learning (ML) algorithms aim to explain the dynamics of a given dataset of any sample. A training dataset is often represented as a matrix, with rows as m entities and columns as n features. Each entity is a row vector of all features describing the entity. Each feature is a column vector, representing the domain and distribution of values that any entity can take. When data is linear, it is easy to describe it with a linear equation in the form $y=f(x) = w \cdot x + b$ that satisfies all rows equations (samples in the dataset). This is also described as

CHAPTER 1

inferring a relationship between (x,y) pairs that reflect a hypothesis to which an accuracy measure is required from a testing dataset. The weights/ coefficients/parameters w measures the correlation and is estimated by the ML algorithm, which is the slope in a line equation or gradients in higher dimensions. The independent features/predictors x is read from the dataset along with the dependent/outcome/target value y . b is the y -intercept in the line equation and is the bias in higher dimensions, and is usually represented as an extra weight element in the weights vector rather than a variable of its own. Solving a set of equations, as shown below, is a deterministic approach to identifying the parameters or the weights (the unknowns) with the given values in the dataset (the knowns or observations) has many algorithms. This approach will not converge in a reasonable time, even for a small dataset. Deterministic approaches would create a lookup table for predicting a y for a test x , as defined from the training dataset. This is not the aim of machine learning, in which a dataset is just samples, and the algorithm needs to generalise to unseen test sets. Failing to generalise to unseen data is called overfitting. In classification models, y is the dependent discrete variable/feature in the prelabelled dataset extracted from the x vector. In binary classification, y can be the set $\{1, -1\}$, or more codes for more classes. In regression models, $y \in \mathbb{R}$ is the predicted continuous value. The last section in this chapter will explain both models in more detail.

This equation maps $x \in \mathbb{R}^n$ to $y \in \mathbb{R}$ and is called functionals $F: \mathbb{R}^n \rightarrow \mathbb{R}$, which will be further explained in chapter three. For now, these different spaces of y and x are vector spaces, domains and the range of the function that maps between them. These linear mapping functions have the linearity property:

$f(w_1x_1 + w_2x_2) = w_1f(x_1) + w_2f(x_2)$, (in the higher dimension $f(WX) = \sum_{i=1}^n w_i x_i$, which is equivalent to:

1. $f(x_1 + x_2) = f(x_1) + f(x_2)$: Addition
2. $f(wx) = wf(x)$: Scaling

Intuition: Linear equations draw a line for one-dimensional space when x is a scalar value, a plane for 2-dimensional space when x is a vector of 2 values, and a hyper-plane for higher dimensions. An animation can be found in <https://youtu.be/slBISYuVUTM> to visualise regression in the higher-dimensional space.

1.2.1 Vector Operations:

Let us define vectors and their calculus. A vector $v \in \mathbb{R}^N$ is not just a one-dimensional array of N scalars; it is a trajectory with the given magnitude (elements values) along N coordinates

CHAPTER 1

corresponding to each element. It represents displacement and velocities compared to scalar values such as temperature and mass.

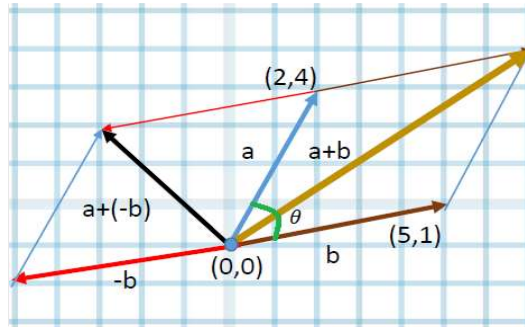


Figure 1: Geometric interpretations of vector addition, subtraction, and dot products. Angle θ is meant to represent the angle between vectors a and b , but it is positioned visually as if it was between vectors $(a+b)$ and b . A green arc highlights this angle.

1.2.1.1 Vector Addition, Subtraction and Normalisation, and Transposition

Vector operations have geometric interpretations. The addition of two vectors measures the length of the longer diagonal of the parallelogram formed by these two vectors, as shown in Figure 1.

The p -norm of a vector is a positive-definite scalar function defined as $\|v\|_p = (\sum_{i=1}^N |v_i|^p)^{\frac{1}{p}} \geq 0, \forall p \geq 1$, where $|v_i|$ is the absolute value of each element v_i .

This means that 1-norm is the sum of the absolute values of the elements. The 2-norm is the magnitude of the vector $v \in \mathbb{R}^N$, which is its length (Frobenius norm) and is denoted $\|v\|_2$ or $\|v\|_F$. It is the Euclidean distance from the origin to the point reached by the vector and calculated as follows $= \sqrt{\sum_{i=1}^N v_i^2}$. The infinity-norm is defined as the case where $p \rightarrow \infty$, as

$$\|v\|_\infty = \lim_{p \rightarrow \infty} (\sum_{i=1}^N |v_i|^p)^{\frac{1}{p}} = \max(|v_i|). \text{ For example, given } v = \begin{bmatrix} 10 \\ 2 \\ -6 \end{bmatrix}, \text{ then } \|v\|_1 = 18,$$

$$\|v\|_2 = 11.83, \|v\|_\infty = 10.$$

A vector is normalised such that adding all its elements equals one. Dividing all its elements by the vector's length normalises the vector, such that normalised $v = \frac{v}{\|v\|}$. The transposition of a vector does not change the order of its elements but changes its representation as a row or column vector.

CHAPTER 1

1.2.1.2 Vector Dot (Inner) Product

Vectors are generally represented as column vectors. A dot product between vectors v and $u \in \mathbb{R}^N$ is denoted $\langle v, u \rangle$ or $v^T u$ and measures the similarity between similar dimensions. For Example, given:

$$v = \begin{bmatrix} 10 \\ 2 \\ -6 \end{bmatrix}, u = \begin{bmatrix} -3 \\ 0 \\ -2 \end{bmatrix}, \text{ then } v^T u = [10 \ 2 \ -6] \cdot \begin{bmatrix} -3 \\ 0 \\ -2 \end{bmatrix} = (10 \times -3) + (20) + (-6 \times -2) = -18.$$

The general rule of the dot product is, for any two given vectors v and $u \in \mathbb{R}^n$: $\sum_{i=1}^n v_i u_i$. Another formula for the cross product is $\langle v, u \rangle = \|u\| \|v\| \cos \theta$, where θ is the angle between vectors v and u . The result of a dot product is a scalar that denotes the projection of one vector over the other. It gives a measure of similarity between two vectors. When θ is 90° between any given two vectors, i.e. they are perpendicular to each other, their dot product will be 0. This means that these vectors are orthogonal.

1.2.1.3 Vector Cross Product

The cross product is denoted \times and it measures the similarity between the different dimensions. It is performed as follows for the same v and $u \in \mathbb{R}^3$ given above: $(2(-2) - (-6)(0), -6(-3) - 10(-2), 10(0) - (-2)(-3)) = (-4, 38, 6)$.

The general rule for the cross-product in \mathbb{R}^3 is: $(v_1, v_2, v_3) \times (u_1, u_2, u_3) = (v_2 u_3 - v_3 u_2, v_3 u_1 - v_1 u_3, v_1 u_2 - v_2 u_1)$. The output is a third vector in \mathbb{R}^3 that is perpendicular to both input vectors. The length of the output vector is equal to the area of the parallelogram formed by the input vectors. The general rule in \mathbb{R}^n , it is $v \times u = \|u\| \|v\| \sin \theta n$, where θ is the angle between vectors v and u , and n is the unit vector perpendicular to the plane containing vectors v and u and given by the right-hand rule.

1.2.1.4 Vector Outer and Hadamard Products

The Hadamard product \odot measures the interactions between elements in the same order and position between two vectors and is defined as follows:

$$v \odot u = [v_1 u_1 \ \dots \ v_n u_n] = [10 \times -3 \ 2 \times 0 \ -6 \times -2] = [-30 \ 0 \ 12]$$

The outer product (also called the dyadic/external) product) is denoted \otimes and is performed as follows for the same $v \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ produces a Matrix $M \in \mathbb{R}^{n \times m}$. For example, given v and u above:

$$v \otimes u = \begin{bmatrix} v_1 u_1 & \dots & v_1 u_n \\ \vdots & \ddots & \vdots \\ v_n u_1 & \dots & v_n u_n \end{bmatrix} = \begin{bmatrix} 10 \times -3 & 10 \times 0 & 10 \times -2 \\ 2 \times -3 & 2 \times 0 & 2 \times -2 \\ -6 \times -3 & -6 \times 0 & -6 \times -2 \end{bmatrix} = \begin{bmatrix} -30 & 0 & -20 \\ -6 & 0 & -4 \\ 18 & 0 & 12 \end{bmatrix} \in \mathbb{R}^{3 \times 3}$$

CHAPTER 1

The general rule for the outer product for \mathbb{R}^n is shown above with the \mathbb{R}^3 example. The output is a matrix, called a dyad, with rows and columns equal to the number of elements in the input vectors, which is generalised to tensor of order two and rank one as will be explained below. The outer product is generalised as the tensor product. The multiplications of every element in the first vector with every element in the second vector measure the interaction between all elements in the first vector with all elements in the second vector. The Dirac notation or bra-ket notation use this dyadic algebra in quantum mechanics, such that a ket denoted $|v\rangle$ is a vector in an abstract (complex) vector space V (will be explained below), and a bra denoted $\langle f|$ is a linear mapping $f: V \rightarrow \mathbb{C}$, to each vector v in V to complex plane \mathbb{C} .

1.2.1.5 Vector Calculus: Derivatives and Gradients

Calculus measures the rate of change of a function, such as the rate of change of x as a ratio to the rate of change of y in a linear equation. This measures the slope of the line tangent to the curve at the point x_0 . Differentiation measures the rate of change as the delta change goes to zero is:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x_0+h) - f(x_0)}{h}$$

$f'(x)$ is called the first derivative of function $f(x)$. If the derivative can be formed at each point of a subdomain of the domain of f , then f is said to be differentiable on that subdomain. $dy = f'(x) dx$ is called the differential of y or $f(x)$. Therefore, $f'(x) = \frac{dy}{dx}$. Many calculus books such as (Banner, 2007) and online cheat sheets and calculators can give the rules of differentiation for different functions, and various exercises on the chain rule are applied when many functions are composed together.

The second (order) derivative $f''(x)$ of a function is the derivative of the derivative of the function. On the graph of a function, the second derivative corresponds to the curvature of the graph.

For functions of two or more variables, the partial derivative is the derivative with respect to one of those variables, keeping all other variables constant. For example $f(x, y) = 3x^2y^3$, we can differentiate with respect to x , $\frac{df}{dx}(x, y) = 6xy^3$ or with respect to y , $\frac{df}{dy}(x, y) = 9x^2y^2$.

The gradient of a function, for example, $f(x, y, z)$, is a vector function of all first partial derivatives of all its variables. $\nabla f = [\frac{\delta f}{\delta x}(x, y, z), \frac{\delta f}{\delta y}(x, y, z), \frac{\delta f}{\delta z}(x, y, z)]$. More on this will be explained in chapter three.

CHAPTER 1

1.2.1.6 Vector Field, Spaces and Independence

A field is a set \mathbb{F} with at least two elements, 0 and 1 and two functions: addition and multiplication. We can define $\mathbb{F}: \mathbb{F}^n \rightarrow \mathbb{F}^m \{0, 1, x, v | x + y \in \mathbb{F} \text{ and } xy \in \mathbb{F}\}$. For example, the field \mathbb{Q} of rationals, that is, fractions of the form $\frac{m}{n}$, where m, n are integers and $n > 0$, the field \mathbb{R} of real numbers, and the field $\mathbb{C} = \{x + iy | x, y \in \mathbb{R}\}$ of complex numbers. Vector was defined earlier as a magnitude and direction. This makes adding any two vectors or scaling any of them, or both by a factor create new vectors that are linearly dependent on the input vectors and belong to the same vector space defined over a field $\mathbb{F} = \mathbb{R}, \mathbb{Q}, \mathbb{C}, \dots$, and so forth, or in the span of the input vectors. For a complete definition of vector space properties, spans and linear dependence and independence, please review complete books like (Carter, 1995) and (Deisenroth, Faisal and Ong, 2019). You can also follow the link to “the Jupyter Guide to Linear Algebra” from the ch1.ipynb notebook. We will summarise this critical concept as vector space is a subset, S , of \mathbb{R}^n with the following properties:

- $0 \in S$ (the zero vector of size n is in the set S); and
- If $v, w \in S$ then $(v+w) \in S$; and addition
- If $\alpha \in \mathbb{R}$ and $v \in S$, then $\alpha v \in S$. scalar multiplication

Any linear combinations can be defined on the unit base vectors describing the vector span of all members of the vector space. The Vector span is the set containing all linearly dependent vectors on a given vector.

The vector span is explained as:

$$\left\{ \alpha_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \alpha_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mid \alpha_1, \alpha_2 \in \mathbb{R} \right\}$$

is the set of all linear combinations of the unit basis vectors $e_1, e_2 \in \mathbb{R}^2$. The basis vectors have two fundamental properties: completeness, such that every vector can be written as a linear combination of basis vectors, and uniqueness, such that the coefficients in the expansion of vectors are unique. For example, all vectors in \mathbb{R}^n (an uncountable infinite set) can be

described with just these n basis vectors. For example, given $v = \begin{bmatrix} 10 \\ 2 \\ -6 \end{bmatrix}$ in \mathbb{R}^3 , it is expressed as:

$$v = 10 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} - 6 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

This is generalised to \mathbb{R}^n by: $x_1 e_1 + x_2 e_2 + \dots + x_n e_n$. Let $\{v_1, v_2, \dots, v_n\} \in \mathbb{R}^n$. $e_i \in \mathbb{R}^n$ is defined as the unit basis for dimension $i=1, 2, \dots, n$ such that only the i^{th} position is equal to 1, and all

other values = 0, for $1 \leq i \leq n$, $e_i = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$. Then the span of these vectors, $(\{v_1, v_2, \dots, v_n\})$, is said to

be the set of all vectors that are formed by a linear combination of the given set of vectors.

CHAPTER 1

Given two vectors, v above, and $u = \begin{bmatrix} 20 \\ 4 \\ -12 \end{bmatrix}$, both are said to be linearly dependent as $u = 2v$, under scalar multiplication, such that the scalar is 2.

Let $\{v_1, v_2, \dots, v_n\} \in \mathbb{R}^n$. Then this set of vectors is said to be linearly independent if $x_1 v_1 + \dots + x_n v_n = 0$. This implies that $x_1 = \dots = x_n = 0$. A set of linearly dependent vectors is defined as such when at least one of the vectors can be expressed as a linear combination of another.

Example of a vector space that satisfies a plane of vectors = $\begin{bmatrix} 0 \\ x_1 \\ x_2 \end{bmatrix}$, is it a subspace of \mathbb{R}^3 ? we

need to test the three conditions. 1) the zero vector is included in the set since variables x_1 and x_2 are meant to accept any values, including zero. 2) if u and v are two vectors in

this set, will $u+v$ be in the set? Yes because $\begin{bmatrix} 0+0 \\ u_1+v_1 \\ u_2+v_2 \end{bmatrix}$, and all elements satisfy the set

definition. 3) if $\alpha \in \mathbb{R}$ and v are in the set, will αv be in the set? Yes because $\begin{bmatrix} \alpha 0 = 0 \\ \alpha v_1 \\ \alpha v_2 \end{bmatrix}$, and all elements satisfy the set definition. Then this set is a subspace of \mathbb{R}^3 .

Another example of a vector space that satisfies a plane of vectors = $\begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$, will not be a subspace of \mathbb{R}^3 the zero vector is not in the set.

Intuition: Dot products are functionals that reduce the dimensionality of any two vectors to 1 scalar, measuring similarity between them. Cross products keep the dimensionality as is measuring the area of the parallelogram formed by the two vectors. Outer products increase the dimensionality by 1 to show the correspondence of each element of the first vector by each element of the second vector. Independent vectors are vectors that can not be linearly composed of each other. Being perpendicular to each other indicates that the two vectors are uncorrelated and independent. The first derivative of one dimension variable is the slope of the line formed by the vector equation. At the same time, the first derivative of the multivariate vector is comprised of a partial derivative of each variable, forming the gradient vector. Second derivatives are defined to check for minimum or maximum, or saddle points of the first derivatives.

1.2.2: Matrix Operations

As introduced in the motivation section earlier, matrices are linear mapping between the rows and the columns. In data science, this would be a map between the entities and their features. A matrix $M \in \mathbb{R}^{m \times n}$ is composed of m rows forming the entities as m vectors v , each containing N elements $v_i \in \mathbb{R}^n$ for $1 \leq i \leq N$. The n elements or features describe the rows and form the columns. Vectors are special kinds of matrices that contain one row or one

CHAPTER 1

column, such that either the m or n is equal to one, and the other is the number of elements it contains. Element-wise operations, such as addition and subtraction, require matrices of equal dimensions. Scalar multiplication and division require a matrix and a scalar. The transposition of a matrix turns its rows into columns and vice versa. For a given matrix $M \in \mathbb{R}^{m \times n}$, the transpose is defined as: $M^T \in \mathbb{R}^{n \times m}$. We will explain the matrix multiplication, orthonormal matrices, determinants, inverse matrices, and Hessian and Jacobian Matrices.

1.2.2.1: Matrix Multiplication

As the motivation section illustrates, linear algebra studies linear maps or function dynamics. For example, if given three breakfast recipes (R), Pancakes (P), Biscuits (B), and Waffles (W) that feed 3 people. Pancake ingredients are 2 cups of baking mix (BM), 2 eggs (E), and 1 cup of Milk (M). Biscuits' ingredients are 2.25 cups of baking mix and 0.75 cups of milk. Waffles ingredients are 2 cups of baking mix, 1 egg, 1.3 cups of milk, and 2 tablespoons of oil (O). We can represent this data using the three vectors: $[2, 2, 1, 0]$, $[2.25, 0, 0.75, 0]$ and $[2, 1, 1.3, 2]$. To unify the representation, we form a matrix $M \in \mathbb{R}^{3 \times 4}$ in which the three recipes are the entity of the rows, the ingredient quantities needed per recipe are in the columns, and each has its weight metric defined in the dataset metadata. This will produce the following dataset matrix (labels in the bold first row are usually omitted):

$$M = \begin{bmatrix} \mathbf{BM} & \mathbf{E} & \mathbf{M} & \mathbf{O} \\ 2 & 2 & 1 & 0 \\ 2.25 & 0 & 0.75 & 0 \\ 2 & 1 & 1.3 & 2 \end{bmatrix} \begin{cases} R \\ P \\ B \\ W \end{cases}$$

Element a_{ij} denotes the value in the i^{th} row and j^{th} column starting indexing from 0 for both rows and columns. For example, $a_{1,1}$ in $M = 0$, which is the number of eggs required for making 1 serve of Biscuits. Matrix addition can be defined as having more data about the same entities in the same order of features, and it is safe to do element-wise addition. For example, to double the ingredients to feed six people, $M+M$, which is also equal to $2M$, calculates the ingredients:

$$2M = \begin{bmatrix} 4 & 4 & 2 & 0 \\ 4.5 & 0 & 1.5 & 0 \\ 4 & 2 & 2.6 & 4 \end{bmatrix}$$

If we want to feed 1 person a pancake, 12 people biscuits, and 9 people waffles, we need to make one batch of pancakes, 4 batches of biscuits, and 3 batches of waffles. To find out the total ingredients needed for this, we can use a vector $v = [1, 4, 3]$ and multiply it by M to produce:

$$\begin{aligned} v \times M &= [1, 4, 3] \times \begin{bmatrix} 2 & 2 & 1 & 0 \\ 2.25 & 0 & 0.75 & 0 \\ 2 & 1 & 1.3 & 2 \end{bmatrix} \\ &= [1 \times 2 + 4 \times 2.25 + 3 \times 2 \quad 1 \times 2 + 4 \times 0 + 3 \times 1 \quad 1 \times 1 + 4 \times 0.75 + 3 \times 1.3 \quad 1 \times 0 + 4 \times 0 + 3 \times 2] \\ &= [17, 5, 8, 6] \end{aligned}$$

We need 17 cups of baking mix, 5 eggs, 8 cups of milk, and 6 tablespoons of oil.

CHAPTER 1

This is a matrix multiplication between a vector v , which is a special matrix $\in \mathbb{R}^{1 \times 3}$ and a matrix $M \in \mathbb{R}^{3 \times 4}$. We did a dot product (inner product) between each row of the first matrix (here was only one) and each column of the second matrix. Matrix multiplication requires that the number of elements of the rows in the first matrix matches the number of elements of the columns in the second matrix. This means the dimension of the columns of the first matrix should be equal to the dimension of the rows of the second matrix. For example, the matrix multiplication $M_1 \in \mathbb{R}^{1 \times 3} \times M_2 \in \mathbb{R}^{3 \times 4} = M_{out} \in \mathbb{R}^{1 \times 4}$. The inner dimensions of the input matrices should match to produce an output matrix with the outer dimension of both matrices. Using the labels and the meaning of the multiplication, as illustrated in Figure 2, we can see that matrix multiplication combines information from two matrices to calculate the contribution of the rows entities of the first matrix to the features columns of the second matrix.

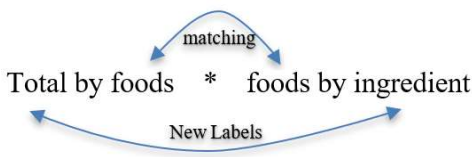


Figure 2: Matrix Multiplication Label Matching and meaning

Another linear mapping example is when given two gas producers (A and B) that send different proportions of the gas they produce to three suppliers (1, 2 and 3), with proportions defined in matrix $P \in \mathbb{R}^{2 \times 3}$. Each supplier then forwards gas in different proportions to four gas stations (X, Y, Z and W) as defined in matrix $S \in \mathbb{R}^{3 \times 4}$. To find out the proportion of each producer output that goes to each gas station, matrix multiplication computes this as follows:

$$P \times S = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.0 & 0.4 & 0.6 \end{bmatrix} \times \begin{bmatrix} 0.4 & 0.6 & 0.0 & 0.0 \\ 0.0 & 0.7 & 0.3 & 0.0 \\ 0.0 & 0.5 & 0.0 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.20 & 0.59 & 0.06 & 0.15 \\ 0.00 & 0.58 & 0.12 & 0.30 \end{bmatrix} \in \mathbb{R}^{2 \times 4}$$

The general rule of matrix multiplication is:

$C = A \times B$ such that Matrix C contains elements c indexed by i and j $c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$. For example, the above $v \in \mathbb{R}^{1 \times 3}$ multiplied by $M \in \mathbb{R}^{3 \times 4}$ $c_{12} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} = 1 \times 2 + 4 \times 2.25 + 3 \times 2 = 5$, then repeat for all c_{ij} $1 \leq i \leq 1$ and $1 \leq j \leq 4$.

Lie Product is a square matrix produced by the difference of the simple product order of two square matrices of the same order $((A, B) \in \mathbb{F}^{n \times n})$, using the following rule:

$$[[A, B]]_{i,j} = \sum_{k=1}^n \{a_{ik}b_{kj} - b_{ik}a_{kj}\} \Rightarrow [A, B] = AB - BA \in \mathbb{F}^{n \times n}$$

CHAPTER 1

1.2.2.2 Systems of Equations Solving

The most straightforward way to solve a system of equations is by substitution. Substitution requires moving all variables on the right-hand side of the equal sign and leaving only one variable on the left-hand side. Then to find a variable solution, simplifying the equation and substituting backwards in other equations assigns a specific value to the variable on the left-hand side. This process is repeated for the remaining variables using the solved variables' values until all are solved. Solving a system of equations can be represented in matrix form. For example: Given two equations:

$$\begin{aligned}5x_1 + 3x_2 &= 93 \\ -4x_1 - 2x_2 &= -66\end{aligned}$$

can be written as Matrix A and vector x and outputs vector b, $Ax=b$.

$$A = \begin{bmatrix} 5 & 3 \\ -4 & -2 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, b = \begin{bmatrix} 93 \\ -66 \end{bmatrix},$$

This is represented as an augmented matrix combining A and b:

$$\left[\begin{array}{cc|c} 5 & 3 & 93 \\ -4 & -2 & -66 \end{array} \right]$$

1.2.2.2.1 Elementary Row Operations

Using Elementary Row Operations (ERO) such as: interchanging any rows, multiplying any row by a non-zero scalar, and replacing any row by the sum of that row and any other row. Any ERO can be chosen in any order until matrix A is converted into an identity matrix I, such that only ones are on the diagonal and all other elements are zeros. The diagonal elements represent the unknowns per column that become equal to the output in the rows of b after the vertical bar. The procedure relies on the fact that its solution does not change if:

1. An equation in the system is modified by subtracting a multiple of another equation in the system from it; and/or
2. Both sides of an equation in the system are scaled by a non-zero.
3. Equations (rows) can be reordered to maintain the strictly lower triangle equal to zero.

These three basic rules are an effort to reduce the system to an upper triangular system, which is easier to solve. An upper triangular matrix is defined as having every entry below the diagonal be zero, i.e. $a_{ij} = 0$ if $i > j$. It is lower triangular $a_{ij} = 0$ for $i < j$. When there are no zeros in the diagonal, the columns are linearly independent. This is formalised algorithmically using Gauss Jordan Elimination as illustrated in the python notebook ([xxx](#)). Additional Rules of the Gauss-Jordan Elimination to get to **reduced row echelon** form are:

4. make each row starts its non-zero coefficients with 1. This is the pivot of the row.

CHAPTER 1

5. move all rows consisting of only zeroes to the bottom of the matrix.

Intuitively, the row that contains 1 in the first column should be moved to be the first row. If non is available, divide the first row by the value of the first column to get 1 in the first diagonal element. To turn a non-diagonal element into a zero, find another row that when scaled by a value, will produce a negative value to the element to zero out and add them together. The following steps solve the previous example:

1. Form the Augmented Matrix:

$$\left[\begin{array}{cc|c} 5 & 3 & 93 \\ -4 & -2 & -66 \end{array} \right]$$

2. Divide Row 1 by 5: $r1 \div 5$

$$\left[\begin{array}{cc|c} 1 & 0.6 & 18.6 \\ -4 & -2 & -66 \end{array} \right]$$

3. We now have a 1 as the first entry in row 1, column 1. Now let us obtain a 0 in row 2, column 1. This can be accomplished by multiplying row 1 by 4 and then adding the result to row 2, leaving row 1 unaffected: $4 * r1 + r2$.

$$\left[\begin{array}{cc|c} 1 & 0.6 & 18.6 \\ 0 & 0.4 & 8.4 \end{array} \right]$$

4. To have 1 in the second diagonal element, we divide row 2 by 0.4: $r2 \div 0.4$

$$\left[\begin{array}{cc|c} 1 & 0.6 & 18.6 \\ 0 & 1 & 21 \end{array} \right]$$

5. To have zero in the non-diagonal elements in row 1, we multiply row 2 by -6 and then add the result to row 1, leaving row 2 unaffected: $-6 * r2 + r1$.

$$\left[\begin{array}{cc|c} 1 & 0 & 6 \\ 0 & 1 & 21 \end{array} \right]$$

This forms the equations:

$$\begin{aligned} 1x_1 + 0x_2 &= 6 \\ 0x_1 + 1x_2 &= 21 \end{aligned}$$

Making $x_1 = 6$ and $x_2 = 21$. We can stop at step four since we can continue by backward substitution forming the Gaussian Elimination only to reduce the computational steps, which is helpful for larger matrices.

1.2.2.2 Matrix Inverse

A matrix inverse is defined A^{-1} such that $AA^{-1}=I$, where I is the identity matrix as defined above. For solving equations of the form $Ax=b$, then $x = A^{-1}b$ is an equivalent representation to solve for x . This makes a system of equations solved by multiplication rather than by Gauss Jordan Elimination. To find A^{-1} for matrix A defined above, we can follow the same steps as above to solve $A^{-1}A = I$:

CHAPTER 1

1. Form the Augmented Matrix:

$$\left[\begin{array}{cc|cc} 5 & 3 & 1 & 0 \\ -4 & -2 & 0 & 1 \end{array} \right]$$

2. $r1 \div 5$

$$\left[\begin{array}{cc|cc} 1 & 0.6 & 0.2 & 0 \\ -4 & -2 & 0 & 1 \end{array} \right]$$

3. $4 * r1 + r2$.

$$\left[\begin{array}{cc|cc} 1 & 0.6 & 0.2 & 0 \\ 0 & 0.4 & 0.8 & 1 \end{array} \right]$$

4. $r2 \div 0.4$

$$\left[\begin{array}{cc|cc} 1 & 0.6 & 0.2 & 0 \\ 0 & 1 & 2 & 2.5 \end{array} \right]$$

5. $-6 * r2 + r1$.

$$\left[\begin{array}{cc|cc} 1 & 0 & -1 & -1.5 \\ 0 & 1 & 2 & 2.5 \end{array} \right]$$

Therefore $A^{-1} = \begin{bmatrix} -1 & -1.5 \\ 2 & 2.5 \end{bmatrix}$

Finding the inverse of a matrix computationally for large matrices is inefficient. Matrix factorisation/decomposition algorithms are used instead, as explained in chapter two. A matrix inverse is not always defined for all given matrices, just as much as scalar zero has no inverse. This happens when the columns of the matrix are not linearly independent, as defined in the previous section. The parallelogram formed by these vectors has an area that equals zero and is labelled a degenerate matrix.

The general formula to find an inverse of a matrix $A \in \mathbb{R}^{2 \times 2} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ can be found by following the same steps above:

1. Form the Augmented Matrix:

$$\left[\begin{array}{cc|cc} a & b & 1 & 0 \\ c & d & 0 & 1 \end{array} \right]$$

2. $r1 \div a$

$$\left[\begin{array}{cc|cc} 1 & \frac{b}{a} & \frac{1}{a} & 0 \\ c & d & 0 & 1 \end{array} \right]$$

3. $-c * r1 + r2$.

$$\left[\begin{array}{cc|cc} 1 & \frac{b}{a} & \frac{1}{a} & 0 \\ 0 & \frac{ad-bc}{a} & \frac{-c}{a} & 1 \end{array} \right]$$

4. $r2 \div \frac{ad-bc}{a}$

$$\left[\begin{array}{cc|cc} 1 & \frac{b}{a} & \frac{1}{a} & 0 \\ 0 & 1 & \frac{-c}{ad-bc} & \frac{a}{ad-bc} \end{array} \right]$$

CHAPTER 1

$$5. \frac{b}{a} * r2 + r1.$$

$$\begin{bmatrix} 1 & 0 & \frac{d}{ad-bc} & \frac{-b}{ad-bc} \\ 0 & 1 & \frac{-c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix}$$

$$\text{Therefore } A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

1.2.2.3 Orthonormal Matrix


Orthogonal matrix A of order n is defined such that the inner product of all its column or row = 0 if different or a value if the same index.

$$\langle a_i, a_j \rangle = \begin{cases} 0, & \text{if } i \neq j \\ \alpha_i = \|a_i\|^2 > 0, & \text{if } i = j \end{cases} \text{ where } a_i, a_j \text{ are any two rows in the matrix, or any two columns, and } \alpha_i \text{ is the square of the euclidian norms of the given row or column.}$$

An orthogonal matrix, or orthonormal matrix, is a real square matrix whose columns and rows are orthonormal vectors that are orthogonal and normalised such that $\alpha_i = 1$. A matrix $A \in \mathbb{R}^{m \times n}$ orthonormal with respect to the rows if $A \cdot A^T = I \in \mathbb{R}^{m \times m}$, and orthonormal with respect to the columns if $A \cdot A^T = I \in \mathbb{R}^{n \times n}$.

Knowing that a matrix is orthonormal means, its determinant is equal to one and it has an inverse, $A^{-1} = A^T$. This makes solving a system of equations $Ax = b \rightarrow A^{-1}Ax = A^{-1}b \rightarrow A^T Ax = A^T b \rightarrow x = A^T b \rightarrow Ix = A^T b$.

1.2.2.3 Matrix Determinant

The square matrix determinant measures the area of the parallelogram formed by the column vectors of the matrix and gives valuable information about the matrix. A determinant of a 1×1 matrix that contains only one element is just the value of this element, which is interpreted as the length of one dimension. A determinant of a 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is defined as $ad - bc$. This is visually defined as: . This formula is used as a divisor of the inverse formula above. This means that if the determinant is equal to zero, the matrix has no inverse. To generalise, we subtract the products of the diagonals from each other, beginning from the main diagonal that goes from the top left to the bottom right. In the 2×2 matrix, this covered all elements of the matrix and did triangularisation of the matrix by replacing the second row by the result of itself minus the first row weighted with factor c/a , yielding:

$$\begin{bmatrix} a & b \\ c - \frac{c}{a}a & d - \frac{c}{a}b \end{bmatrix} = \begin{bmatrix} a & b \\ 0 & d - \frac{c}{a}b \end{bmatrix}.$$

CHAPTER 1

A determinant of a 3×3 matrix can not be produced using the subtraction of the diagonal products method that will cover only 4 out of the 9 elements of the matrix. We can augment the first two columns to the right of the matrix such that all elements fall on diagonals, as illustrated visually below:

$$\begin{bmatrix} a & b & c & a & b \\ d & e & f & d & e \\ g & h & i & g & h \end{bmatrix}$$

Using the diagonals that have 3 numbers only, we can deduce the 3×3 matrix determinant formula to be: $(a \times e \times i + b \times f \times g + c \times d \times h) - (c \times e \times g + a \times f \times h + b \times d \times i)$.

This procedure does not generalise well to the $n \times n$ matrices when $n > 3$. The expansion by minor method works for all values of n . It works by computing a determinant for the "minor" which is a submatrix $A_{ij} \in \mathbb{R}^{(n-1) \times (n-1)}$ that does not include row i and column j from the original matrix, using alternating sign and cofactors C_{ij} . We either take each element of a column by making j spans the columns while i is fixed to the column we are expanding on, or take all rows and make j fixed to the row we are expanding on. It uses the formula:

$$C_{ij} = s_{ij}M_{ij}$$

$s_{11} = 1$, then as we increment i along with the columns and j along with the rows, it keeps changing the sign.

$$\text{For a } 3 \times 3 \text{ matrix, } S = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

For $A \in \mathbb{R}^{3 \times 3} = \begin{bmatrix} 4 & 2 & 3 \\ 0 & 2 & 4 \\ 1 & 3 & 6 \end{bmatrix}$, if we expand on column 1, we can calculate the determinant as:

$$\begin{aligned} \det(A) &= |A| = a_{11}s_{11}M_{11} + a_{21}s_{21}M_{21} + a_{31}s_{31}M_{31} \\ &= 4(+1) \begin{vmatrix} 2 & 4 \\ 3 & 6 \end{vmatrix} + 0(-1) \begin{vmatrix} 2 & 3 \\ 3 & 6 \end{vmatrix} + 1(+1) \begin{vmatrix} 2 & 3 \\ 0 & 4 \end{vmatrix} \\ &= 4(12 - 12) - 0(12 - 9) + 1(8 - 6) \\ &= 4 \times 0 - 0 \times 3 + 1 \times 2 = 2 \end{aligned}$$

For $A \in \mathbb{R}^{4 \times 4} = \begin{bmatrix} 5 & 4 & 6 & 3 \\ 0 & 2 & 1 & 0 \\ 9 & 7 & 4 & 6 \\ 2 & 8 & 1 & 3 \end{bmatrix}$, Then, for $M_{1,1} = \begin{bmatrix} 4 & 6 & 3 \\ 2 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 0 \\ 7 & 4 & 6 \\ 8 & 1 & 3 \end{bmatrix}$. The

complete determinant is calculated as:

$$\begin{aligned} \det(A) &= |A| = 0(-1) \begin{vmatrix} 4 & 6 & 3 \\ 7 & 4 & 6 \\ 8 & 1 & 3 \end{vmatrix} + 2(+1) \begin{vmatrix} 5 & 6 & 3 \\ 9 & 4 & 6 \\ 2 & 1 & 3 \end{vmatrix} + 1(-1) \begin{vmatrix} 5 & 4 & 3 \\ 9 & 7 & 6 \\ 2 & 8 & 3 \end{vmatrix} \\ &\quad + 0(+1) \begin{vmatrix} 5 & 4 & 6 \\ 9 & 7 & 4 \\ 2 & 8 & 1 \end{vmatrix} \end{aligned}$$

For a 4×4 matrix,

CHAPTER 1

$$S = \begin{bmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$$

For $A = \begin{bmatrix} 5 & 4 & 6 & 3 \\ 0 & 2 & 1 & 0 \\ 9 & 7 & 4 & 6 \\ 2 & 8 & 1 & 3 \end{bmatrix}$, Then, for $M_{2,1} = \begin{bmatrix} 5 & 4 & 6 & 3 \\ 0 & 2 & 1 & 0 \\ 9 & 7 & 4 & 6 \\ 2 & 8 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 6 & 3 \\ 7 & 4 & 6 \\ 8 & 1 & 3 \end{bmatrix}$. If we expand on row

2 since it contains two zeros, the complete determinant is calculated as:

$$\det(A) = |A| = a_{21}s_{21}M_{21} + a_{22}s_{22}M_{22} + a_{23}s_{23}M_{23} + a_{24}s_{24}M_{24}$$

$$= 0(-1) \begin{vmatrix} 4 & 6 & 3 \\ 7 & 4 & 6 \\ 8 & 1 & 3 \end{vmatrix} + 2(+1) \begin{vmatrix} 5 & 6 & 3 \\ 9 & 4 & 6 \\ 2 & 1 & 3 \end{vmatrix} + 1(-1) \begin{vmatrix} 5 & 4 & 3 \\ 9 & 7 & 6 \\ 2 & 8 & 3 \end{vmatrix} + 0(+1) \begin{vmatrix} 5 & 4 & 6 \\ 9 & 7 & 4 \\ 2 & 8 & 1 \end{vmatrix}$$

Then further reduce until we find the $\det(A) = |A| = -93$.

The general formula for the determinant of any square matrix $A \in \mathbb{R}^{n \times n}$ is,

$$\det(A) = |A| = a_{i1}s_{i1}M_{i1} + a_{i2}s_{i2}M_{i2} + \dots + a_{in}s_{in}M_{in}$$

$$= a_{1j}s_{1j}M_{1j} + a_{2j}s_{2j}M_{2j} + \dots + a_{nj}s_{nj}M_{nj}$$

$\det(A) = |A| = \sum_{j=1}^n a_{ij}s_{ij}M_{ij}$ for the chosen $i = \sum_{i=1}^n a_{ij}s_{ij}M_{ij}$ for the chosen j .

Choosing the row or column with the most zeros saves much work.

Placing all cofactors in matrix C, the formula for the inverse $A^{-1} = \frac{C^T}{\det(A)}$, such that $(A^{-1})_{ij} = \frac{c_{ji}}{\det(A)}$.

This process is still computationally inefficient since it requires $n!$ operations ($n \times (n - 1) \times (n - 1) \dots \times 2 \times 1$). It is tractable only for small values of n .

Another method to compute the determinant is by using ERO knowing the following properties:

1. Interchanging any two adjacent rows changes the sign of the determinant. Non-adjacent rows interchanges require counting the number of adjacent rows swappings; an even number of swaps will result in a positive determinant, and an odd number of swaps will result in a negative determinant.
2. Multiplying a row by a scalar multiplies the determinant by the same scalar.
3. Replacing any row by the sum of that row and any other row does not change the determinant.
4. The determinant of a triangular matrix (upper or lower) is the product of the diagonal elements.

For a 2×2 matrix,

$$A = \begin{bmatrix} 5 & 3 \\ -4 & -2 \end{bmatrix} \rightarrow \det(A) = D_1$$

CHAPTER 1

1. $r_1 \div 5$
$$\begin{bmatrix} 1 & 0.6 \\ -4 & -2 \end{bmatrix} \rightarrow \det(A) = D_2 \div 5$$

2. $4 * r_1 + r_2$.
$$\begin{bmatrix} 1 & 0.6 \\ 0 & 0.4 \end{bmatrix} \rightarrow \det(A) = D_3 \div 5$$

$D_3 = 0.4 \rightarrow \det(A) = 0.4 \div 5 = 2$

The original determinant formula is $\det(A) = 5(-2) - 3(-4) = -10 + 12 = 2$

For a 3×3 matrix,

$$A = \begin{bmatrix} 0 & 2 & 4 \\ 4 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix} \rightarrow \det(A) = D_1$$

1. Swap r_1 and r_3 , two adjacent row exchanges, with no change to the determinant sign.

$$\begin{bmatrix} 1 & 3 & 6 \\ 4 & 2 & 3 \\ 0 & 2 & 4 \end{bmatrix} \rightarrow \det(A) = D_2$$

2. $-4 * r_1 + r_2$.
$$\begin{bmatrix} 1 & 3 & 6 \\ 0 & -10 & -21 \\ 0 & 2 & 4 \end{bmatrix} \rightarrow \det(A) = D_3$$

3. $r_2 \div (-10)$.
$$\begin{bmatrix} 1 & 3 & 6 \\ 0 & 1 & 2.1 \\ 0 & 2 & 4 \end{bmatrix} \rightarrow \det(A) = D_4 \div (-10)$$

4. $-2 * r_2 + r_3$.
$$\begin{bmatrix} 1 & 3 & 6 \\ 0 & 1 & 2.1 \\ 0 & 0 & -0.2 \end{bmatrix} \rightarrow \det(A) = D_5 \div (-10)$$

$D_5 = -0.2 \rightarrow \det(A) = -0.2 \div (-10) = -2$

The ERO is computationally more efficient for large n and is used more often. There is also modular triangularisation of any square matrix, such that determinants or 2×2 matrices is used in the equations of the 3×3 and so on. This recursive calculation is defined from blocks of triangulated matrices. The determinant of an $n \times n$ matrix becomes the product of the diagonal elements of the triangulated matrix: $\det(A) = \prod_{i=1}^n \det(A_{ii})$.

A singular matrix is a square matrix order n , such that its determinant is equal to zero; otherwise, it is non-singular.

CHAPTER 1

1.2.2.4 Consistent and Inconsistent Systems of Equations

The system of equations example that was given previously is a **consistent system** since it has only one valid solution. This solution is found at the point of intersection of the two lines formed by the two given equations. For three unknowns and three equations, the solution is at the point of intersection of the three planes formed by each equation. The same concept applies to the higher dimensions. Systems $Ax = b$ with one unique solution are defined as when b is in the column space of A . For practising Gaussian Elimination, these online calculators show all intermediate steps:

- <http://ulaff.s3.amazonaws.com/GaussianEliminationPractice/index.html>
- <https://onlinesechool.com/math/assistance/equation/haus/>

Sometimes there is more than one point of intersection, such that the system is defined to be a **consistent dependent system**. This case happens when the system $Ax = b$ has $Ax_s = b$ and $Ax_n = 0$, then $x_s + x_n$ is a solution for $A(x_s + x_n) = b$ (we have many solutions), which signifies that there are linear combinations between the column vectors of A . For example:

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \text{and } b = \begin{bmatrix} 4 \\ 8 \end{bmatrix},$$

We need to find x such that $Ax = 2x$. To solve, we subtract $2x$ from both sides $Ax - 2x = 0$ and have this matrix to apply Gaussian Elimination on:

$$\begin{aligned} x_1 + x_2 &= 4 \\ 2x_1 + 2x_2 &= 8 \end{aligned}$$

1. Form the augmented form $\begin{bmatrix} 1 & 1 & | & 4 \\ 2 & 2 & | & 8 \end{bmatrix}$

2. $R_2 - 2R_1 \rightarrow R_2$ (multiply 1 row by 2 and subtract it from 2 row)

$$\begin{bmatrix} 1 & 1 & | & 4 \\ 0 & 0 & | & 0 \end{bmatrix}$$

If we have a zero on the diagonal, then we have fewer equations than variables, i.e. we have more than one solution, but we have the upper equation that describes the possible solutions as the solution set: $x_1 + x_2 = 4$, which means $x_1 = 4 - x_2$, which gives a bound on the values of x_1 and x_2 . x can be

$$\begin{bmatrix} 1 \\ 3 \end{bmatrix} \text{ or } \begin{bmatrix} 2 \\ 2 \end{bmatrix} \text{ or } \begin{bmatrix} 3 \\ 1 \end{bmatrix} \text{ and so forth.}$$

To formalise this process for any variables, we say we are going to make the last variable a “free variable”, meaning that it can take on any value in \mathbb{R} , and we will see how to describe

CHAPTER 1

the “bound variables” using the free variable. In the exercise above, we say $x_2 = b$ and $x_1 = 4 - b$. Therefore $x = \begin{bmatrix} 4 - b \\ b \end{bmatrix}$.

We now claim that this captures all solutions of the system of linear equations. We will call this the **general solution**. Try different values of b and substitute in the original matrices to find that they always produce the same results. i.e. this is the vector space of the solution.

Because there is a non-trivial solution to $Ax = 0$, the null space of A has more than just the zero vector, and A 's columns are linearly dependent.

Sometimes there is no point of intersection, such that the system is defined to be inconsistent. When system $Ax = b$, b is not in the column span of A , there will be no solution. For example, solve $Ax = b$, for the following:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \text{ and } b = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Reduce it to row echelon form by hand or use the online calculators to find out that the last row contains all zeros, while the last element in b is -2 since zero is not equal to -2 , then b is not in the columns space of A , and there is no solution to this system of equation. The appended form of final output after Gaussian Elimination will be:

$$\left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -2 \end{array} \right]$$

Other ways to solve a system of equations are to reduce column echelon form.

1.2.2.5 Linear Combinations and Linear Transformations

Revisiting vector spaces and generalising to linear matrix transformation, as explained in (Geijn and Quintana-Ort', 2008), a linear transformation L takes a vector in R^n space and transform it to R^m can be expressed as a matrix of size m rows and n columns. A transformation L is linear if the following properties hold for α, β as scalars, and u and v as vectors:

1. $L(\alpha u) = \alpha L(u)$
2. $L(u+v) = L(u) + L(v)$ The transformation is distributive with respect to vector addition.
3. $L(\alpha u + \beta v) = L(\alpha u) + L(\beta v)$
4. $L(\alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n) = L(\alpha_1 v_1) + L(\alpha_2 v_2) + \dots + L(\alpha_n v_n)$

CHAPTER 1

The unit vectors e_j as defined earlier when multiplied by linear transformation matrix L , produces the required output vector. A linear transformation is defined as $L(\sum_{j=0}^{n-1} x_j e_j)$ for input vector x of n elements. The following matrix describes this transformation:

$$\begin{aligned} y \in \mathbb{R}^m &= L(x) \in \mathbb{R}^{m \times n} = L(x_0 e_0 + x_1 e_1 + \dots + x_{n-1} e_{n-1}) = x_0 L(e_0) + x_1 L(e_1) + \dots + x_{n-1} L(e_{n-1}) \\ &= x_0 a_0 + x_1 a_1 + \dots + x_{n-1} a_{n-1}, \end{aligned}$$

Such that a_j is the output of the transformation L at the j^{th} unit vector e_j and size m each, which is the size of the output vector y . This linear transformation can be expressed as a matrix A as follows;

$$y = L(x) = Ax = \begin{bmatrix} a_{0,0} & \dots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{m-1,0} & \dots & a_{m-1,n-1} \end{bmatrix} x = \begin{bmatrix} a_{0,0}x_0 & + \dots + & a_{0,n-1}x_{n-1} \\ \vdots & \ddots & \vdots \\ a_{m-1,0}x_0 & + \dots + & a_{m-1,n-1}x_{n-1} \end{bmatrix}$$

Example: Rotation Linear Transformation matrix:

To define the Linear transformation R_ϕ that performs the rotation of a vector on the 2D plane by ϕ angle, we need to define the output of the transformation on all unit basis vectors.

For the first unit vector, $e_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ visualised as illustrated in Figure 3 a, a point with coordinates $(1, 0)$ is transformed to the $R_\phi(1, 0)$ using trigonometry rules.

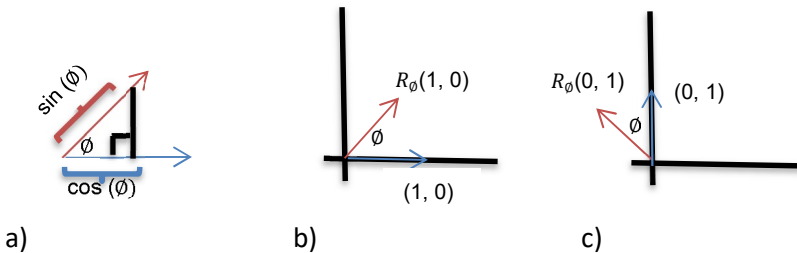


Figure 3: a) trigonometry rules to calculate the length of a side of the right angle triangle b) Rotation of first Unit basis using trigonometry, c) rotation of second basis vector (Geijn, 2012)

$$\begin{aligned} R_\phi(1, 0) &= a_0 = \begin{bmatrix} \cos \phi \\ \sin \phi \end{bmatrix} \\ R_\phi(0, 1) &= a_1 = \begin{bmatrix} -\sin \phi \\ \cos \phi \end{bmatrix} \end{aligned}$$

Using the matrix notation defined previously to express the linear rotation transformation, the following Matrix A is composed of the a_i output produced from the transformation of the unit basis:

$$R_\phi = \begin{bmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{bmatrix}$$

For rotating a vector $x = (3, 5)$ for angle $\phi = 45$, the following matrix-vector multiplication is performed to produce the new output vector

CHAPTER 1

$$R_{\emptyset}(x) = \begin{bmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 3 \cos(45) - 5 \sin(45) \\ 3 \sin(45) + 5 \cos(45) \end{bmatrix} = \begin{bmatrix} 5.6568 \\ 1.4142 \end{bmatrix}$$

In computer graphics, objects are defined by their vertices coordinates in matrix form. Various transformation matrices are used to apply rotation, reflection on an axis, translation by shifting points along a coordinate, skewing, and sheering are applied by matrix multiplication. More examples of forming linear transformation matrices for spatial images are found in (Ashburner and Friston, 1997), or generally, to do ERO in solving systems of equations are found in (Chahal, 2018). These concepts are visualised in <https://youtu.be/lrggOvOSzr4>.

1.2.2.6 Matrix Calculus

The Jacobian Matrix is defined to be a matrix of partial derivatives of n functions, d variables, such as n functions and d variables as follows:

$$J = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \frac{\delta f_1}{\delta x_2} & \cdots & \frac{\delta f_1}{\delta x_d} \\ \frac{\delta f_2}{\delta x_1} & \frac{\delta f_2}{\delta x_2} & \cdots & \frac{\delta f_2}{\delta x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta f_n}{\delta x_1} & \frac{\delta f_n}{\delta x_2} & \cdots & \frac{\delta f_n}{\delta x_d} \end{bmatrix} = \begin{bmatrix} \nabla f_1 \\ \nabla f_2 \\ \vdots \\ \nabla f_n \end{bmatrix}$$

The following link has more examples to practice: <http://mathonline.wikidot.com/the-jacobian-matrix-of-differentiable-functions-examples-1>. This link contains more advanced topics using the Jacobian matrix in artificial neural networks computations “Jacobian Matrix –the Joys of the Jacobian”: <http://chalkdustmagazine.com/features/the-joys-of-the-jacobian/>.

The Hessian matrix of f is a square $n \times n$ symmetric matrix of second-partial derivatives of f that is defined as follows:

$$\mathcal{H}f(x_1, x_2, \dots, x_n) = \begin{bmatrix} \frac{\delta^2 f}{\delta x_1^2} & \frac{\delta^2 f}{\delta x_1 x_2} & \cdots & \frac{\delta^2 f}{\delta x_1 x_n} \\ \frac{\delta^2 f}{\delta x_2 x_1} & \frac{\delta^2 f}{\delta x_2^2} & \cdots & \frac{\delta^2 f}{\delta x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta^2 f}{\delta x_n x_1} & \frac{\delta^2 f}{\delta x_n x_2} & \cdots & \frac{\delta^2 f}{\delta x_n^2} \end{bmatrix}$$

The symmetry of $\frac{\delta^2 f}{\delta x_i x_j} = \frac{\delta^2 f}{\delta x_j x_i}$ simplifies the computation. Hessian Matrices are helpful in finding extreme values of multivariate functions using the matrix eigenvalues. The extreme values of a function are the local minima, local maxima, and saddle points, which define the

CHAPTER 1

function curvature. These values are useful in optimisation algorithms such as gradient descent in neural networks. The determinant of the hessian matrix (D-test) provides a function discriminant. The inverse of this matrix identifies the least relevant components of a function to use in pruning and reducing model complexity (Singh and Alistarh, 2020). More details can be viewed at: <http://mathonline.wikidot.com/hessian-matrices>.

Intuition: A matrix represents entities' labels in rows and features values in columns. Matrix addition and subtraction are defined as element-wise increasing or decreasing values by known augmenting datasets. A scalar product scales the values element-wise by a given magnitude. A vector and matrix multiplication is a special case of matrix-matrix multiplication, in which the inner dimensions (with their corresponding labels) should match, and the outer dimensions are the output dimension. A matrix multiplication by a vector represents a linear transformation on the vector, such as projection to a higher or lower dimension, rotation, reflection, scaling, or any linear combination. This transformation is defined in terms of the basis vectors of the corresponding coordinates. To solve a linear system of equations in matrix form, we apply elementary row operations to decompose a matrix representing the equations to a diagonal matrix.

References

Ashburner, J. and Friston, K.J. (1997) 'Spatial Transformation of Images', in *Human Brain Function. First Edition. Academic Press USA*, p. 36. Available at: <http://www.fil.ion.ucl.ac.uk/spm/doc/books/hbf1/>.

Banner, A.D. (2007) *The calculus lifesaver: all the tools you need to excel at calculus. Princeton, NJ: Princeton University Press (A Princeton lifesaver study guide)*.

Carter, T.A. (1995) *Linear Algebra, An Introduction to Linear Algebra for Pre-Calculus Students. Rice University*.

Chahal, J.S. (2018) *Fundamentals of Linear Algebra: With Applications in Computer Science, Economics, Engineering, Mathematics, and Physics. Boca Raton: CRC Press, Taylor and Francis Group*.

Cichocki, A. et al. (2016) 'Low-Rank Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Problems: Perspectives and Challenges PART 1', *Foundations and Trends® in Machine Learning*, 9(4–5), pp. 249–429. Available at: <https://doi.org/10.1561/22000000059>.

Deisenroth, M.P., Faisal, A.A. and Ong, C.S. (2019) *Mathematics for Machine Learning. Cambridge University Press*.

Downey, A. (2013) *Think Bayes. First edition. Sebastopol, CA: O'Reilly*.

CHAPTER 1

Dym, H. (2007) *Linear algebra in action*. Providence, R.I: American Mathematical Society (*Graduate studies in mathematics*, v. 78).

Farrell, P. (2020) *The statistics and calculus workshop a comprehensive introduction to mathematics in Python for artificial intelligence applications*. Available at: <http://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9781800208360> (Accessed: 25 October 2021).

Geijn, R. van de (2012) *ULAFF: Linear Algebra: Foundations to Frontiers*. Available at: <http://www.ulaff.net/> (Accessed: 6 March 2021).

Geijn, R.A. van de and Quintana-Ort', E.S. (2008) *The Science of Programming Matrix Computations*. www.lulu.com. Available at: <http://z.cs.utexas.edu/wiki/LA.wiki/books/TSoPMC/>.

Gelß, P. (2017) *The Tensor-Train Format and Its Applications*. PhD Dissertation. Universität Berlin Institut für Mathematik.

Ghosal, S. and Vaart, A.W. van der (2017) *Fundamentals of nonparametric Bayesian inference*. Cambridge ; New York: Cambridge University Press (*Cambridge series in statistical and probabilistic mathematics*, 44).

Kolda, T.G. and Bader, B.W. (2009) 'Tensor Decompositions and Applications', *SIAM Review*, 51(3), pp. 455–500. Available at: <https://doi.org/10.1137/07070111X>.

Lu, H., Plataniotis, K.N. and Venetsanopoulos, A.N. (2014) *Multilinear subspace learning: dimensionality reduction of multidimensional data*. Boca Raton, Florida: CRC Press/Taylor & Francis Group (*Chapman & Hall/CRC machine learning & pattern recognition series*).

Singh, S.P. and Alistarh, D. (2020) 'WoodFisher: Efficient Second-Order Approximation for Neural Network Compression'. *arXiv*. Available at: <http://arxiv.org/abs/2004.14340> (Accessed: 19 September 2022).

Sun, L.-H., Huang, Xin-Wei, Alqawba, Mohammed S. and Kim, J.-M., Emura, Takeshi (2020) *Copula-Based Markov Models for Time Series: Parametric Inference and Process Control*. Singapore: Springer Singapore : Imprint: Springer.

Watts, S. (2016) 'The Gaussian Copula and the Financial Crisis: A Recipe for Disaster or Cooking the Books?', p. 25.