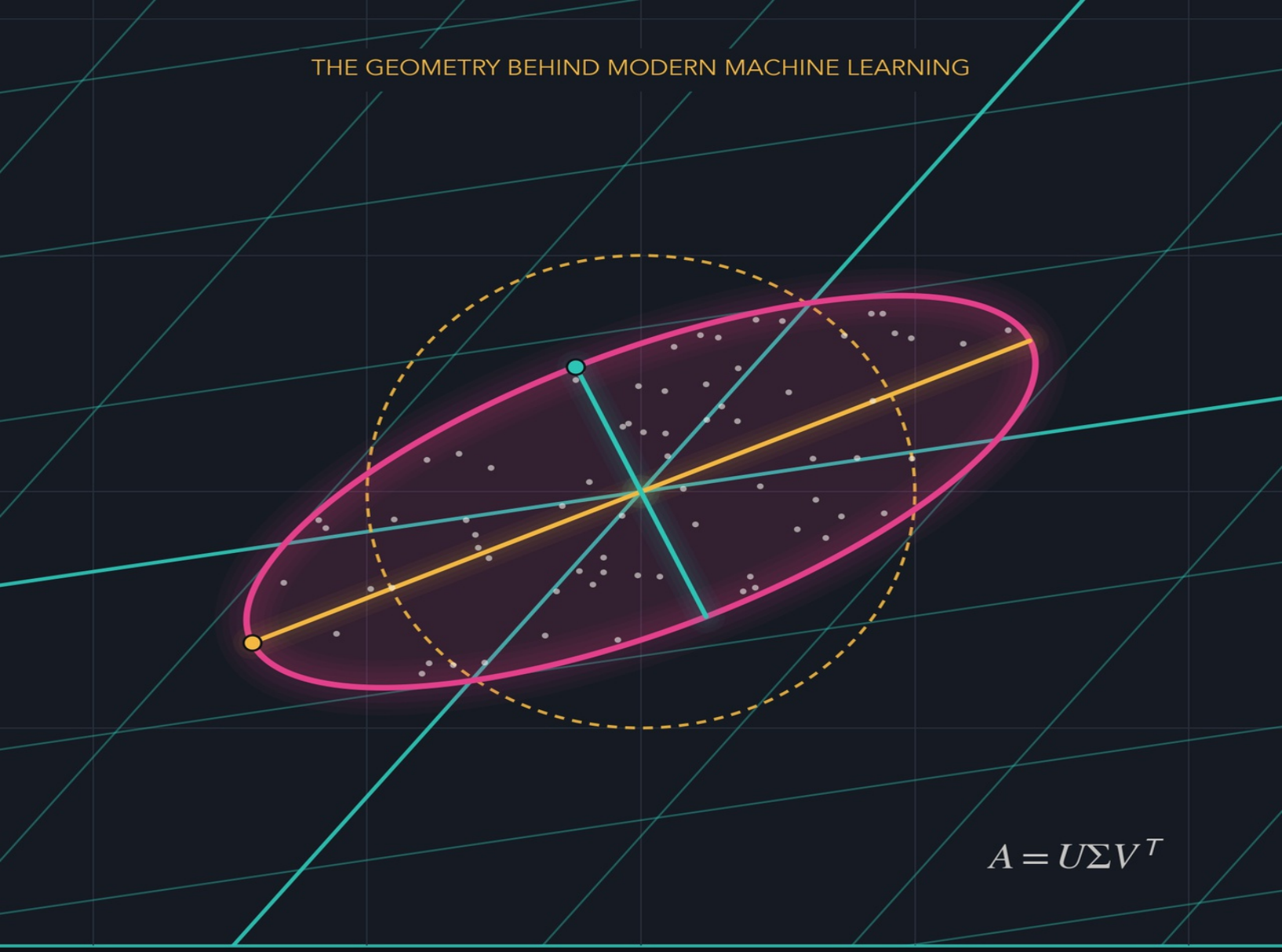
THE GEOMETRY BEHIND MODERN MACHINE LEARNING



LINEAR ALGEBRA

FOR AI AND MACHINE LEARNING

A Visual and Conceptual Guide to
Vectors, Matrices, and Eigenvalues

GEORGE CHU

LINEAR ALGEBRA

FOR AI AND MACHINE LEARNING

A Visual and Conceptual Guide to

Vectors, Matrices, and Eigenvalues

George Chu

References to research published in 2024–2026 reflect the literature available to the author in the first half of 2026. The reader should assume that these, and not the mathematics, are the part of this book that will date.

First Edition, 2026

Body text set in Charter; mathematics in Cambria Math and STIX.

Mathematics is the art of giving

the same name to different things.

HENRI POINCARÉ, 1908

A matrix is a geometric action on space.

Everything else in this book follows.

CONTENTS

The contents of this edition are in your reader's navigation menu, and every entry there is a link.

PREFACE

Linear algebra is usually taught twice, and the two courses never meet. The first is a sequence of procedures: reduce this matrix to echelon form, take that determinant, find the eigenvalues of a three-by-three. It is taught without a single picture, and most students leave it able to compute and unable to say what they computed. The second arrives years later, inside a course on machine learning, as a sequence of shapes: an embedding is a vector of length 768, a layer is a weight matrix, the data has a covariance, the loss has a gradient. It is taught without the theory, because there is no time, and most students leave it able to use the shapes and unable to reason about them.

This book is written for the place where those two courses ought to have met. It is built on a single observation, and the observation is old: **a matrix is a geometric action on space.** It rotates, stretches, shears, reflects, flattens. Every theorem in linear algebra is a statement about what that action does, and every linear-algebraic step inside a learning machine is one of those actions applied to data. Once the action is visible, the procedures stop being procedures. Row reduction is a way of finding which directions a map destroys. A determinant is a volume. An eigenvector is a direction the map does not turn. The singular value decomposition says that every matrix, of any shape, is a rotation, followed by a stretch along perpendicular axes, followed by another rotation — and almost everything else in the subject can be read off those stretches.

What this book asserts

Five structural propositions organise the whole. They are stated here so that the reader can watch them accumulate; each is re-derived in an unfamiliar setting several times before the book ends.

1. **A vector is a point in a space of possibilities, and the dot product is a measure of agreement.** Length, angle, projection, cosine similarity and the score an attention layer computes are one operation seen from five sides.

2. **A matrix is a map, its columns are where the basis vectors land, and multiplication is composition.** A deep network is a long composition of such maps with a simple fold between each pair, and every shape error in a model is a composition error.
3. **Every matrix is rotate, stretch, rotate.** Rank is the number of stretches that are not zero; the determinant is their product; the condition number is the ratio of the largest to the smallest; the best low-rank approximation keeps the largest and discards the rest.
4. **Eigenvectors are the directions a map does not turn, and symmetric matrices have a perpendicular set of them.** Covariances, Hessians and graph Laplacians are symmetric, which is the single reason so much of learning is tractable.
5. **High dimension is not three dimensions with more axes.** In a space of a thousand dimensions almost every pair of directions is almost perpendicular, and that fact is what allows a learned representation to hold far more distinct features than it has coordinates.

By Chapter 15 the reader should be able to look at any matrix and describe, without computing anything, the ellipse it turns a circle into. By Chapter 20 the reader should be able to open the description of a modern model — its embeddings, its attention layers, its low-rank adapters, its optimiser — and recognise every piece as a construction from this book.

How the material is presented

Six devices recur, and they carry different kinds of information.

- **Figures.** Ninety-one of them, and they are the spine rather than the decoration. Where the text says a map shears the plane, a figure shows the grid being sheared; where it says a high-dimensional ball is mostly skin, a figure shows exactly how much. A reader who studies only the figures and their captions will have read a coherent short version of the book.

- **Definitions and theorems**, in shaded boxes with a blue rule, each theorem followed by its proof. The proofs are short, because the right picture usually makes them short, and none is omitted for being "beyond the scope".
- **Worked examples**, in boxes with a teal rule, carried out by hand on two-by-two and three-by-three matrices so that every number can be checked with a pencil. Understanding in this subject is built by doing one small example completely, and the examples are chosen so that the arithmetic stays out of the way of the idea.
- **In machine learning**, short notes that connect a result to the place it lives inside a model: the dot product to attention, the projection to regression, the low-rank approximation to fine-tuning, the eigenvalues of the Hessian to the speed of training.
- **Exercises** at the end of every chapter, answered in Appendix C. Each one is meant to be done.
- **Key ideas** at the end of every chapter: five sentences stating what it established, for review and for finding one's way back.

Prerequisites

Comfort with algebra, the idea of a function, and a first course in calculus — enough to be untroubled by a derivative and a sum. No previous linear algebra is assumed, and neither is any machine learning; Chapter 1 begins with what a vector is. A reader who has taken a procedural linear algebra course will move quickly through Part I and should not skip it, because the geometric language set up there is used everywhere afterwards.

Appendix A collects the notation and every fact the later chapters rely on. Appendix B places the major decompositions side by side in a single table — what each one says, when it exists, what it costs, and where it appears in machine learning — and is meant to be photocopied and kept. Appendix E is a glossary of the terms the book uses, each with the chapter that introduces it. Appendix F closes the book with sixteen problems that cross the chapters, each worked in full, and a table that starts from the questions practice raises and points to the tool that answers each.

On what is inherited and what is added

Roughly half of this book is the settled core of its subject, restated so that the parts connect. The attributions are given in the text and collected in Appendix D: the four fundamental subspaces; Gaussian elimination and the LU factorisation; Gram–Schmidt and QR; the normal equations of least squares; the spectral theorem; the singular value decomposition and the Eckart–Young theorem; principal component analysis; the Johnson–Lindenstrauss lemma; the chain rule as a product of Jacobians; the convergence theory of gradient descent; the attention mechanism.

The other half is the work of connecting them and bringing them forward. One picture — a unit sphere and the ellipsoid a matrix turns it into — is used as the lens for rank, determinant, conditioning, eigenvalues, singular values, least squares, principal components and the speed of optimisation, so that eight topics usually taught separately become one. The book derives a clean counting bound for how many nearly perpendicular directions fit in a space of a given dimension, and uses it to explain why learned representations can hold more features than coordinates. It works out the parameter arithmetic of low-rank adaptation and says exactly when it pays. It shows that the orthogonalised update used by a family of optimisers adopted for large-scale training in 2025 is steepest descent measured in the right norm. It writes attention as a bilinear form whose rank is capped by the key dimension. Where this book advances a claim on its own account, the claim is set apart in bold against a gold rule and collected in the Index of Propositions at the back.

How to read it

In order, for the first reading. The book is cumulative in a way that rewards patience: Part IV is short because Parts I to III did the work, and Part V is a series of recognitions rather than new material.

A reader in a hurry may take a shorter route. Chapters 2, 5, 7, 13 and 15 are the spine: the dot product, the matrix as a map, the four subspaces, eigenvectors, and the SVD. Everything else is used by at most two later chapters.

The tone of this book is optimistic, and the optimism is meant to be earned. Linear algebra is one of the few subjects in which the deepest results are also the most useful and the most beautiful, and in which a student who understands a single picture — a circle turned into an ellipse — understands, in a real sense, the working parts of the most capable learning systems yet built. That is an extraordinary return on a short and elegant body of mathematics, and it is available to anyone willing to look carefully at what a matrix does.

G. C.

— ● —

PART

I

VECTORS

the objects

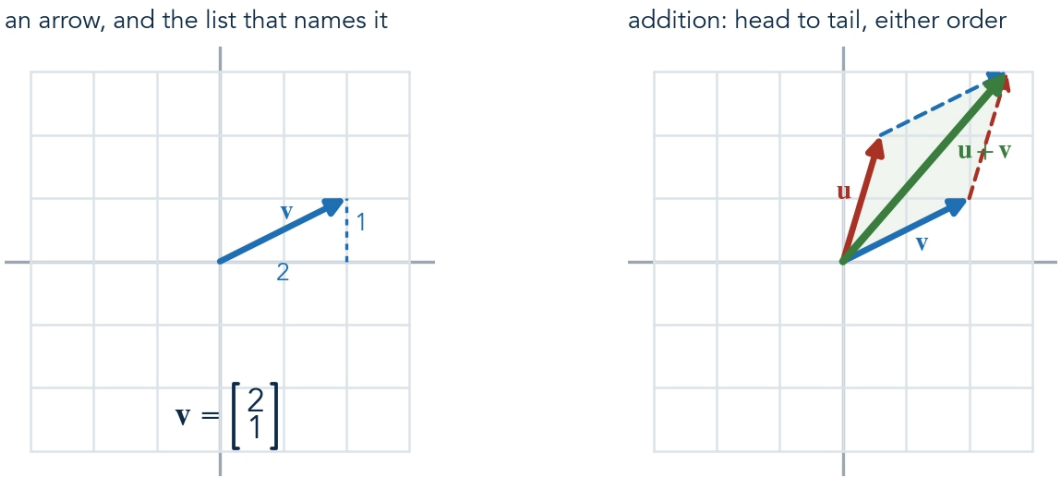
CHAPTER ONE

1 What a Vector Is

Every idea in this book is built from one kind of object, and the object has two faces. Seen one way, a vector is an arrow: it has a length and a direction, and it can be slid anywhere without changing. Seen the other way, it is a list of numbers. The whole power of linear algebra comes from moving freely between these two faces — computing with the list, reasoning with the arrow — and this chapter establishes both, and the rules that connect them.

1.1 Two faces of one object

Start in the plane. An arrow that runs two units to the right and one unit up is a vector. So is the list $(2, 1)$ that records those two numbers. They are the same thing, provided we have agreed in advance which directions "right" and "up" are and how long a unit is.



the same object seen twice. The arrow is what a vector does; the list is how we write it down once a grid has been chosen. Chapter 3 is about that choice.

FIGURE 1.1 Left: the arrow and the list that names it once a grid is fixed. Right: adding two vectors by placing them head to tail. The result is the diagonal of the parallelogram they span, and the order does not matter.

The arrow is what the vector *is*; the list is how we *write it down*. This distinction is not pedantry. In Chapter 3 we will change the grid, and the list will change while the arrow stays exactly where it was. In machine learning the same point is made every time a model re-expresses data in new coordinates: the information has not moved, only its description.

Definition 1.1 Vector in $\mathbb{R}^n$

A vector in $\mathbb{R}^n$ is an ordered list of n real numbers, written as a column. The numbers are its components, and n is the dimension of the space the vector lives in.

We write vectors in bold, $\mathbf{v}$, and their components with subscripts, v_1, v_2, ..., v_n. As a column, a vector in $\mathbb{R}^3$ looks like this:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \qquad \mathbf{v}^T = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \tag{1.1}$$

The transpose $\mathbf{v}^T$ lays the column on its side. It will matter a great deal in Chapter 2, where a row meeting a column is exactly the dot product.

1.2 Two operations, and nothing else

There are only two things you can do to vectors in linear algebra, and everything else is built from them.

Definition 1.2 Addition and scalar multiplication

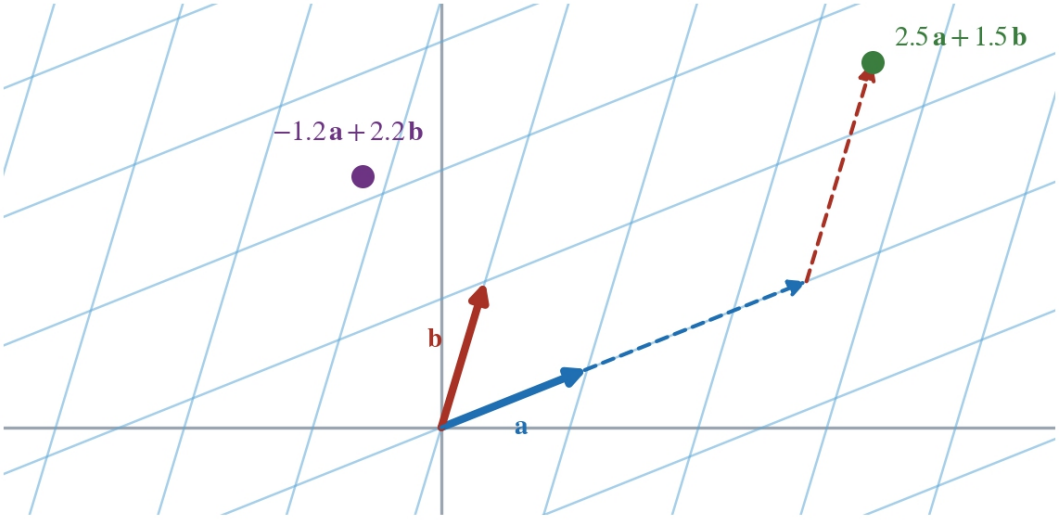
Vectors of the same dimension are added component by component, and a vector is multiplied by a number c by multiplying every component by c.

$\mathbf{u} + \mathbf{v} = (u_1 + v_1, \ldots, u_n + v_n)$ and $c\mathbf{v} = (c\,v_1, \ldots, c\,v_n)$.

Geometrically, addition is head-to-tail and scaling stretches the arrow, flipping it round when c is negative. Put the two together and you have the single most important construction in the subject.

Definition 1.3 Linear combination

A linear combination of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is any vector of the form $c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_k\mathbf{v}_k$, where the c_i are numbers called the coefficients.



the slanted grid is every combination $s\mathbf{a}+t\mathbf{b}$ with whole-number s, t. Fill in the fractions and it covers the plane: two vectors that do not line up reach everywhere.

FIGURE 1.2 Two vectors a and b that do not lie on one line lay down a slanted grid. Every point of the plane sits somewhere on that grid, so every point is a combination of a and b, with its own pair of coefficients.

Worked Example 1.1 Finding the coefficients

Write $\mathbf{w} = (3, 5)$ as a combination of $\mathbf{a} = (1, 1)$ and $\mathbf{b} = (1, 3)$.

We need numbers s and t with $s(1, 1) + t(1, 3) = (3, 5)$. Component by component that says $s + t = 3$ and $s + 3t = 5$. Subtracting the first equation from the second gives $2t = 2$, so $t = 1$ and $s = 2$.

Check: $2(1, 1) + 1(1, 3) = (2 + 1, 2 + 3) = (3, 5)$. The point w is two steps along a and one along b.

The last example already contains the question at the heart of Part III. Asking whether a vector can be written as a combination of others, and with which coefficients, is the same as solving a system of linear equations. We have just solved one without naming it.

1.3 What makes a space a vector space

The plane and $\mathbb{R}^n$ are the spaces this book uses most, but the rules above apply to far more than lists of numbers, and it is worth seeing why.

Definition 1.4 Vector space

A vector space is a collection of objects that can be added to each other and multiplied by numbers, where the results stay in the collection and obey the familiar rules of arithmetic: addition is commutative and associative, there is a zero vector, every vector has a negative, and scaling distributes over addition.

Polynomials of degree at most three form a vector space: add two and you get another, scale one and you get another. So do all 3×3 matrices, all continuous functions on an interval, and all images of a fixed size, each regarded as a long list of pixel intensities. The conclusions of the later chapters hold in every one of these, because they are derived from the rules, not from the arrows.

> **The definition of a vector space is not a list of properties a vector happens to have. It is a license: any collection that obeys those few rules inherits every theorem in this book, which is why the same mathematics describes arrows in the plane, sound waves, images and the internal states of a neural network.**

1.4 A vector as a point in a space of possibilities

There is a third way of seeing a vector that the machine-learning literature relies on constantly, and it is the one that makes high dimensions feel natural. A vector is a **point** — a location in a space whose axes are measurable properties.

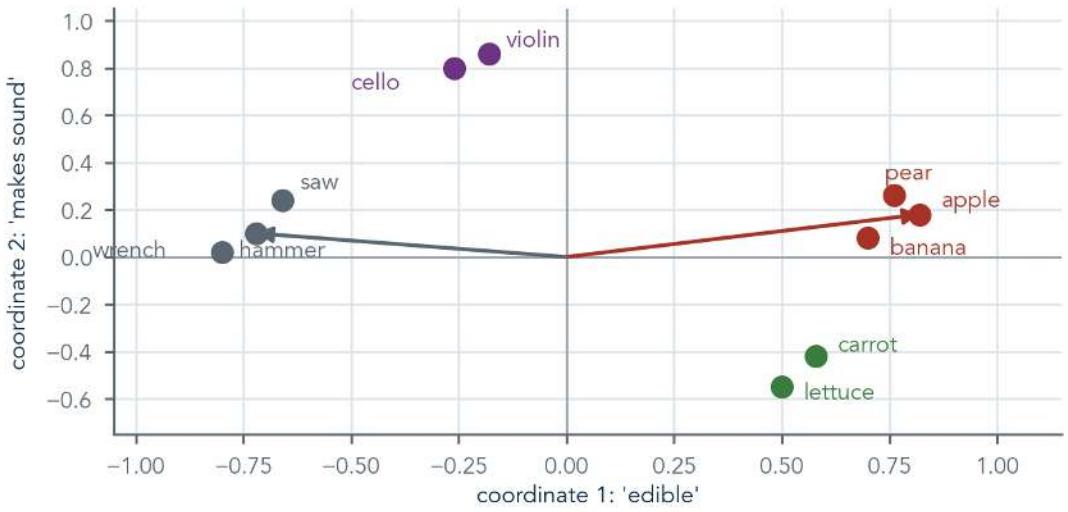


FIGURE 1.3 Ten things placed by two invented properties. Things that are alike sit close together, and the arrow from the origin carries the meaning: fruit points one way, tools another. A learned embedding is the same picture with hundreds of axes whose meanings are not assigned in advance.

A house described by floor area, number of rooms, age and distance to the nearest school is a point in $\mathbb{R}^4$. A photograph with a million pixels is a point in $\mathbb{R}^{1000000}$. A word, in a modern model, is a point in a space of several hundred or a few thousand dimensions, placed so that words used in similar contexts lie near one another. None of these spaces can be drawn. All of them obey the rules of Section 1.3, and so every tool in this book applies to them unchanged.

> **In machine learning.** The input to almost every model is a vector, and so is almost every intermediate result. A model's "embedding dimension" or "hidden size" is simply the n of the space its internal vectors live in. When a paper says two representations are "close", it means the distance between two points in that space is small; when it says they are "aligned", it means their arrows point in similar directions — the subject of Chapter 2.

1.5 Length and distance

Two more quantities complete the basic toolkit, and both come straight from Pythagoras.

Definition 1.5 Norm and distance

The length, or Euclidean norm, of $\mathbf{v}$ is $\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$. The distance between $\mathbf{u}$ and $\mathbf{v}$ is the length of their difference, $\|\mathbf{u} - \mathbf{v}\|$.

$$\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2}, \qquad \operatorname{dist}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\| \tag{1.2}$$

In the plane this is the hypotenuse of a right triangle; in three dimensions it is the diagonal of a box; in n dimensions it is the same formula with more terms, and it still behaves like a length. It is never negative, it is zero only for the zero vector, scaling a vector by c scales its length by $|c|$, and the length of a sum is at most the sum of the lengths — the triangle inequality, which says that a detour is never shorter than the direct route.

Worked Example 1.2 Length and distance in four dimensions

Let $\mathbf{u} = (1, 2, 2, 0)$ and $\mathbf{v} = (3, 2, 0, 1)$.

$\|\mathbf{u}\| = \sqrt{1 + 4 + 4 + 0} = 3$. $\|\mathbf{v}\| = \sqrt{9 + 4 + 0 + 1} = \sqrt{14} \approx 3.74$.

$\mathbf{u} - \mathbf{v} = (-2, 0, 2, -1)$, so the distance is $\sqrt{4 + 0 + 4 + 1} = 3$. The two points are three units apart, even though neither space nor distance can be drawn.

A vector of length one is a **unit vector**. Dividing any nonzero vector by its length produces the unit vector pointing the same way, $\mathbf{v}/\|\mathbf{v}\|$, a step called normalisation. It strips out size and keeps only direction, and it is performed so often in machine learning that many models do it automatically inside every layer.

TABLE 1.1 The basic vocabulary of this chapter, in the three languages it will be spoken in. The rightmost column is the reason the first two are worth learning.

idea	geometry	algebra	in a learning machine
vector	an arrow, or a point	a list of n numbers	an input, an embedding, a hidden state
addition	head to tail	componentwise sum	combining signals, residual connections
scaling	stretch or flip	multiply every component	weighting, learning-rate steps
linear combination	reach a point on a grid	$c_1\mathbf{v}_1 + \ldots + c_k\mathbf{v}_k$	a weighted average, the output of a layer
norm	length	$\sqrt{\sum v_i^2}$	magnitude, normalisation
distance	how far apart	$\lVert\mathbf{u} - \mathbf{v}\rVert$	similarity, nearest neighbours

Worked Example 1.3 Normalising, and the triangle inequality

Let $\mathbf{u} = (3, 4)$ and $\mathbf{v} = (1, 0)$.

$\lVert\mathbf{u}\rVert = \sqrt{9 + 16} = 5$, so the unit vector in the direction of $\mathbf{u}$ is $(0.6, 0.8)$, whose length is $\sqrt{0.36 + 0.64} = 1$.

$\mathbf{u} + \mathbf{v} = (4, 4)$ has length $4\sqrt{2} \approx 5.66$, which is less than $\lVert\mathbf{u}\rVert + \lVert\mathbf{v}\rVert = 6$. The two would be equal only if $\mathbf{u}$ and $\mathbf{v}$ pointed the same way.

1.6 Why this is the right foundation

It would be possible to begin a book like this with matrices, since that is where most of the computation happens. But matrices are maps from vectors to vectors, and a map cannot be understood before the things it maps. Every later chapter asks a question about vectors — which ones a matrix destroys, which directions it leaves alone, which combination best approximates a target — and every answer is a vector or a set of vectors.

> **A learning machine is, to a first approximation, a device for moving points around in a high-dimensional space**

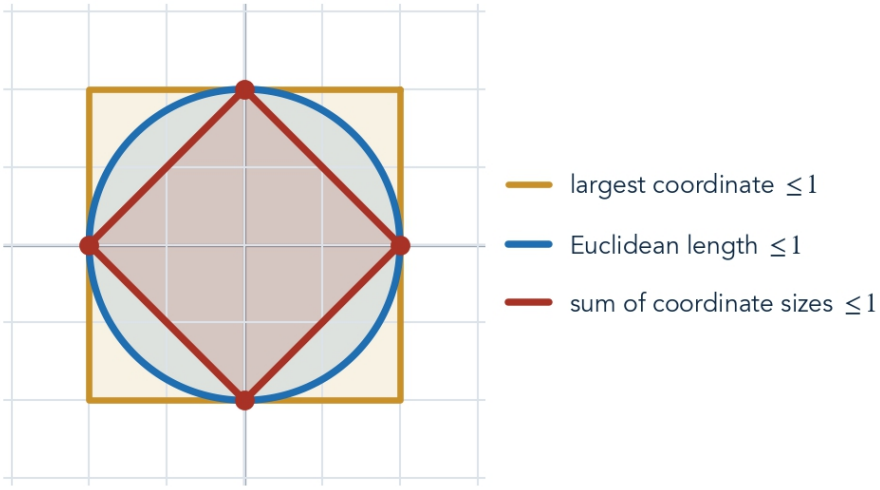
until the points that should be near each other are near and the ones that should be far apart are far. Linear algebra is the geometry of that space, and it is the only geometry the machine has.

Worked Example 1.4 Three ways to measure length

The Euclidean length of Definition 1.5 is not the only useful one. Take $\mathbf{v} = (3, -4)$.

Its Euclidean length is $\sqrt{9 + 16} = 5$. The sum of the sizes of its coordinates, called the ℓ_1 length, is $3 + 4 = 7$. The size of its largest coordinate, the ℓ_∞ length, is 4.

Figure 1.4 shows the vectors of length at most one under each rule: a circle, a diamond and a square. Methods that penalise the ℓ_1 length tend to produce vectors with many zero coordinates, because the diamond's corners lie on the axes, where the other coordinates vanish.



the three unit balls of the plane. The diamond's corners sit on the axes, which is why penalising the sum of coordinate sizes tends to push coordinates to exactly zero.

FIGURE 1.4 Vectors of length at most one under three rules: the Euclidean length (circle), the sum of coordinate sizes (diamond) and the largest coordinate size (square). All three are legitimate lengths, and they disagree about which vectors are short.

Key ideas

- A vector is both an arrow and a list; the list depends on a chosen grid, the arrow does not.
- Addition and scaling are the only operations, and the linear combinations built from them are the central construction of the subject.
- Any collection that obeys the vector-space rules inherits every theorem in the book: arrows, polynomials, images and the internal states of a network alike.
- A vector is also a point in a space of possibilities, where nearness means similarity — the view every embedding relies on.
- Length and distance come from Pythagoras in any dimension, and normalising keeps direction while discarding size.

Exercises

1. Draw $\mathbf{u} = (1, 2)$ and $\mathbf{v} = (3, -1)$, then draw $\mathbf{u} + \mathbf{v}$, $\mathbf{u} - \mathbf{v}$ and $2\mathbf{u} - \mathbf{v}$. Check each against the componentwise formula.
2. Write $(4, 2)$ as a combination of $(1, 0)$ and $(1, 1)$. Then write it as a combination of $(2, 1)$ and $(1, 3)$. Why are the coefficients different although the point is the same?
3. Find the length of $(2, -1, 2)$ and the unit vector pointing in its direction.
4. Show that the set of vectors (x, y, z) with $x + y + z = 0$ is a vector space, and that the set with $x + y + z = 1$ is not.
5. A house is described by (area in square metres, rooms, age in years). Explain why the distance between two houses is dominated by one coordinate, and suggest how to fix it.
6. Show from the definition that scaling a vector by c multiplies its length by the size of c.
7. Find the distance between $(1, 0, -1, 2)$ and $(0, 2, 1, 2)$.
8. Explain why every vector space must contain the zero vector.

CHAPTER TWO

2 The Dot Product

One operation connects the algebra of lists to the geometry of arrows, and it is the most used calculation in machine learning. Multiply two vectors component by component and add the results. The number that comes out measures length, angle, projection and agreement all at once, and a single layer of a modern model computes millions of them.

2.1 Two definitions that agree

Definition 2.1 Dot product

The dot product of $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^n$ is the number $\mathbf{u} \cdot \mathbf{v} = u_1\, v_1 + u_2\, v_2 + \ldots + u_n\, v_n = \mathbf{u}^T\mathbf{v}$.

That is the algebraic definition, and it takes one line to compute. The geometric statement is different in appearance and identical in content.

Theorem 2.1 The geometric dot product

If θ is the angle between nonzero vectors $\mathbf{u}$ and $\mathbf{v}$, then $\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\|\, \|\mathbf{v}\| \cos\theta$.

Proof

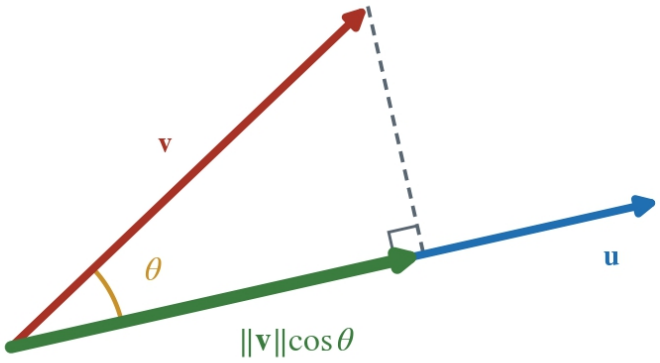
Apply the law of cosines to the triangle with sides $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{u} - \mathbf{v}$: $\|\mathbf{u} - \mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - 2\|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$.

Expand the left side with the algebraic definition: $\|\mathbf{u} - \mathbf{v}\|^2 = \sum(u_i - v_i)^2 = \sum u_i^2 + \sum v_i^2 - 2\sum u_i v_i$.

Comparing the two expressions, the squared lengths cancel and $\sum u_i v_i = \|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$. ■

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i = \|\mathbf{u}\| \, \|\mathbf{v}\| \cos\theta \tag{2.1}$$

The theorem is worth pausing over. On the left is a sum anyone can compute in any number of dimensions. On the right is an angle, which in a space of a thousand dimensions no one can see. The equation lets us *define* the angle in high dimensions by computing the sum — and every notion of similarity in this book rests on that move.



drop a perpendicular from $\mathbf{v}$ onto the line of $\mathbf{u}$. The green shadow, times the length of $\mathbf{u}$, is the dot product -- a sum of coordinate products and a statement about angle at once.

FIGURE 2.1 Drop a perpendicular from $\mathbf{v}$ onto the line of $\mathbf{u}$. The green segment has length $\|\mathbf{v}\|\cos\theta$, and multiplying it by $\|\mathbf{u}\|$ gives the dot product. The dot product is a length times a shadow.

2.2 Projection

The shadow in Figure 2.1 is important enough to have its own name and formula.

Definition 2.2 Projection onto a vector

The projection of $\mathbf{v}$ onto the line through a nonzero vector $\mathbf{u}$ is the vector $\text{proj}_{\mathbf{u}}\mathbf{v} = \frac{\mathbf{u}\cdot\mathbf{v}}{\mathbf{u}\cdot\mathbf{u}}\mathbf{u}$.

$$\text{proj}_{\mathbf{u}}\,\mathbf{v} = \frac{\mathbf{u}^T\mathbf{v}}{\mathbf{u}^T\mathbf{u}}\,\mathbf{u}, \qquad (\mathbf{v} - \text{proj}_{\mathbf{u}}\,\mathbf{v})\cdot\mathbf{u} = 0 \tag{2.2}$$

The second equation is the defining property: what remains after the projection is removed is perpendicular to $\mathbf{u}$. That single fact — *the error of the best approximation is perpendicular to what you approximated with* — will return as the Gram–Schmidt process in

Chapter 10, as least squares in Chapter 11, and as principal component analysis in Chapter 17.

Worked Example 2.1 A projection by hand

Project $\mathbf{v} = (2, 3)$ onto $\mathbf{u} = (4, 0)$, then onto $\mathbf{w} = (1, 1)$.

Onto $\mathbf{u}$: $\mathbf{u}\cdot\mathbf{v} = 8$ and $\mathbf{u}\cdot\mathbf{u} = 16$, so the projection is $8/16(4, 0) = (2, 0)$. The residual $(0, 3)$ is perpendicular to $\mathbf{u}$, as it must be.

Onto $\mathbf{w}$: $\mathbf{w}\cdot\mathbf{v} = 5$ and $\mathbf{w}\cdot\mathbf{w} = 2$, so the projection is $5/2(1, 1) = (2.5, 2.5)$. The residual is $(-0.5, 0.5)$, and $(-0.5)(1) + (0.5)(1) = 0$.

2.3 Perpendicularity and the Cauchy–Schwarz inequality

Two vectors are **orthogonal** when their dot product is zero. In the plane that means they meet at a right angle; in high dimensions we simply take it as the definition. Orthogonal vectors share no component: knowing how far you have gone along one tells you nothing about the other.

The geometric formula also bounds the dot product, since a cosine never exceeds one in size.

Theorem 2.2 Cauchy–Schwarz

For all vectors $\mathbf{u}$ and $\mathbf{v}$, $|\mathbf{u}\cdot\mathbf{v}| \le \|\mathbf{u}\|\,\|\mathbf{v}\|$, with equality exactly when one is a multiple of the other.

Proof

If $\mathbf{u} = \mathbf{0}$ both sides vanish. Otherwise, for every number t the squared length $\|\mathbf{v} - t\mathbf{u}\|^2 = \|\mathbf{v}\|^2 - 2t\,\mathbf{u}\cdot\mathbf{v} + t^2\|\mathbf{u}\|^2$ is at least zero.

A quadratic in t that is never negative has a discriminant that is at most zero: $4(\mathbf{u}\cdot\mathbf{v})^2 - 4\|\mathbf{u}\|^2\|\mathbf{v}\|^2 \le 0$, which is the inequality.

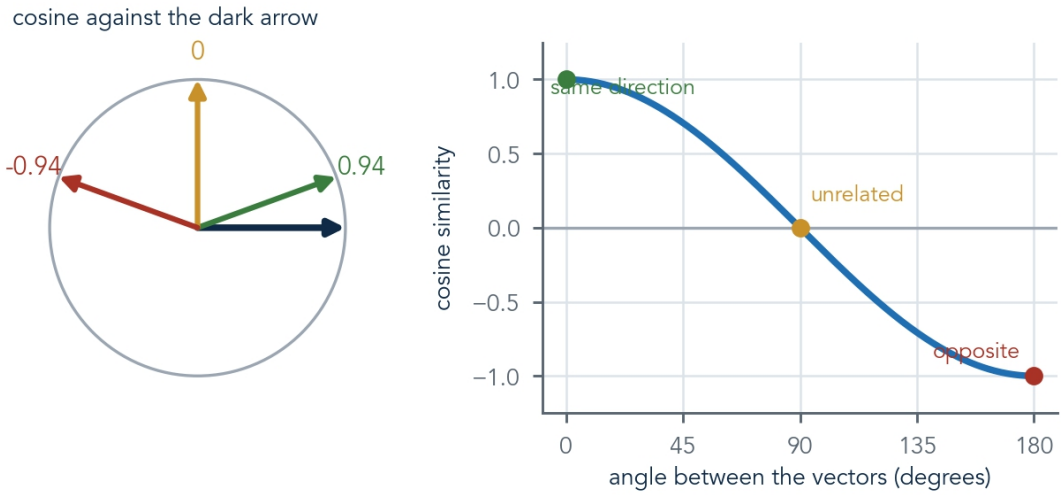
Equality means the quadratic touches zero, so $\mathbf{v} = t\mathbf{u}$ for some t. ■

The proof never mentions an angle, which is exactly why it matters: it guarantees that $\mathbf{u}\cdot\mathbf{v}/(\|\mathbf{u}\|\|\mathbf{v}\|)$ always lies between -1 and 1 in any dimension, so it is always the cosine of some angle. The definition of angle in $\mathbb{R}^{1000}$ is legitimate because of these three lines.

2.4 Cosine similarity

Divide the dot product by both lengths and what remains depends only on direction.

$$\cos\theta = \frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\,\|\mathbf{v}\|} \in [-1, 1] \tag{2.3}$$



cosine similarity ignores length and reads only direction: +1 agree, 0 unrelated, -1 opposed. Retrieval, recommendation and attention all start from this one number.

FIGURE 2.2 Cosine similarity reads only direction. Arrows that agree score near $+1$, unrelated ones near 0, opposed ones near -1, regardless of how long they are.

Ignoring length is often exactly right. Two documents about the same subject should be similar whether one is a paragraph and the other a book; two customers with the same tastes should match whether one buys often or rarely. Cosine similarity compares the *shape* of two vectors and discards their *size*.

Worked Example 2.2 Which is more similar?

Let $\mathbf{q} = (1, 2, 0)$, $\mathbf{a} = (2, 4, 1)$ and $\mathbf{b} = (0, 1, 3)$.

$\mathbf{q}\cdot\mathbf{a} = 2 + 8 + 0 = 10$, with $\|\mathbf{q}\| = \sqrt{5}$ and $\|\mathbf{a}\| = \sqrt{21}$, so $\cos\theta_a = 10/\sqrt{105} \approx 0.976$.

$\mathbf{q}\cdot\mathbf{b} = 0 + 2 + 0 = 2$, with $\|\mathbf{b}\| = \sqrt{10}$, so $\cos\theta_b = 2/\sqrt{50} \approx 0.283$.

$\mathbf{a}$ points almost exactly the way $\mathbf{q}$ does; $\mathbf{b}$ points mostly elsewhere. A search that ranks by cosine returns $\mathbf{a}$ first.

Retrieval, recommendation and nearest-neighbour search are, at their core, the same calculation: compute dot products between one vector and many, and rank.

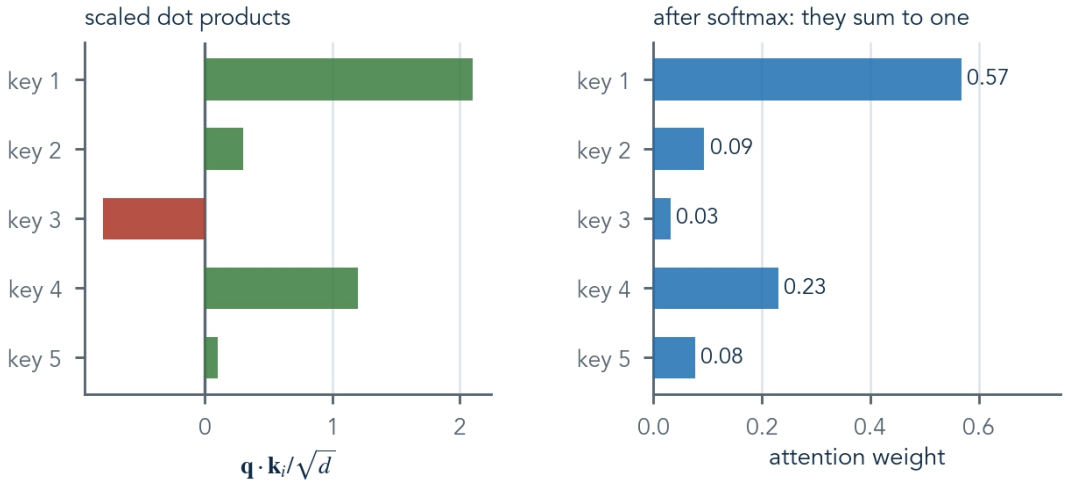
> **Every improvement in those systems is either a better way of placing the vectors or a faster way of finding the largest dot products, never a different notion of similarity.**

2.5 The dot product inside attention

The most consequential use of the dot product in current machine learning is the attention mechanism, and it can be understood completely with what this chapter has built.

Suppose a sequence of items — words, image patches, measurements — has been turned into vectors. To update the representation of one item, the model forms a **query** vector $\mathbf{q}$ for it and a **key** vector $\mathbf{k}_i$ for every item it might attend to. The relevance of item i is a dot product.

$$s_i = \frac{\mathbf{q} \cdot \mathbf{k}_i}{\sqrt{d}}, \qquad w_i = \frac{e^{s_i}}{\sum_j e^{s_j}}, \qquad \text{output} = \sum_i w_i \, \mathbf{v}_i \tag{2.4}$$



an attention layer asks one question of many vectors with a dot product, and turns the answers into a weighted average. Chapter 20 writes the whole layer as matrices.

FIGURE 2.3 One query scored against five keys. The dot products on the left are turned into positive weights that sum to one by the softmax, and the output is the weighted average of the items' value vectors.

Three pieces of this chapter appear in that equation. The score is a dot product, so it is large when the query and key point the same

way. The division by $\sqrt{d}$, where d is the dimension, keeps the scores from growing with dimension — and Chapter 4 explains why they would. The output is a linear combination (Definition 1.3) of value vectors with coefficients chosen by agreement. Attention is a learned, soft version of "find the most similar items and average them".

In machine learning. Chapter 20 writes a whole attention layer with matrices: the queries, keys and values for every item are stacked into matrices Q, K and V, and all the dot products at once are the entries of QK^T. That matrix product is the reason attention is both powerful and expensive: for a sequence of n items it contains n^2 dot products.

Worked Example 2.3 Attention weights by hand

One query $\mathbf{q} = (1, 0)$ attends to three items with keys $\mathbf{k}_1 = (2, 0)$, $\mathbf{k}_2 = (0, 1)$, $\mathbf{k}_3 = (1, 1)$ and values $\mathbf{v}_1 = (1, 0)$, $\mathbf{v}_2 = (0, 1)$, $\mathbf{v}_3 = (1, 1)$, with $d = 2$.

The scaled scores $\mathbf{q}\cdot\mathbf{k}_i/\sqrt{2}$ are 1.414, 0 and 0.707. Their exponentials are 4.11, 1 and 2.03, which sum to 7.14, so the weights are 0.58, 0.14 and 0.28.

The output is $0.58(1, 0) + 0.14(0, 1) + 0.28(1, 1) = (0.86, 0.42)$. The first item, whose key agrees most with the query, dominates, but every item contributes.

2.6 A table of what the dot product measures

TABLE 2.1 One operation, many readings. The formula in each row is the same sum of products; only the question being asked of it changes.

reading	formula	used for
length squared	$\mathbf{v}\cdot\mathbf{v} = \|\mathbf{v}\|^2$	norms, normalisation
angle	$\cos\theta = \mathbf{u}\cdot\mathbf{v}/(\|\mathbf{u}\|\|\mathbf{v}\|)$	similarity, alignment
perpendicularity	$\mathbf{u}\cdot\mathbf{v} = 0$	independent directions, bases
shadow	$(\mathbf{u}\cdot\mathbf{v})/\|\mathbf{u}\|$	projection, least squares
weighted sum	$\mathbf{w}\cdot\mathbf{x}$	a linear model's score, a neuron
agreement	$\mathbf{q}\cdot\mathbf{k}$	attention, retrieval

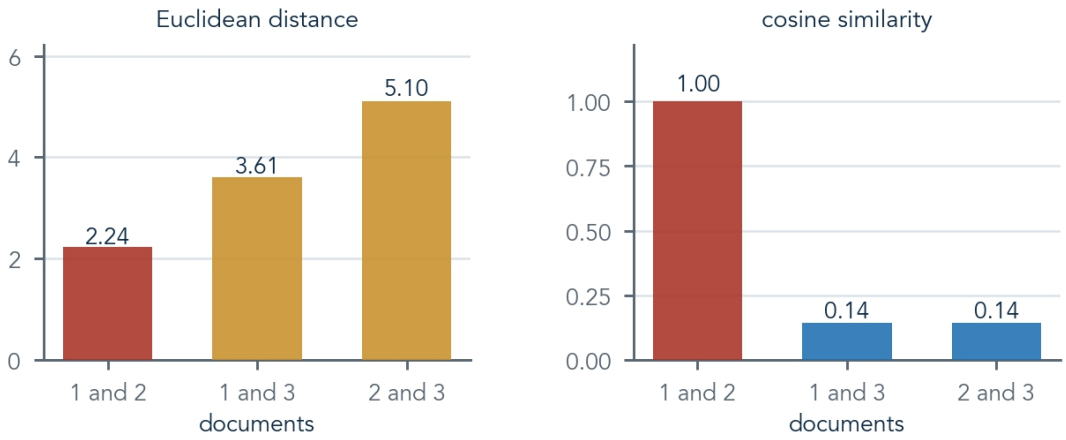
The fifth row deserves a final word. A linear model predicts $\hat{y} = \mathbf{w}\cdot\mathbf{x}$: it projects the input onto a learned direction $\mathbf{w}$ and reads off how far along that direction the input lies. A single artificial neuron does the same, then applies a simple non-linear function. Training such a model is the search for the direction whose shadows best predict the target.

Worked Example 2.4 Distance or direction?

Three short documents are described by counts of the words "matrix", "vector" and "poem": $\mathbf{d}_1 = (2, 1, 0)$, $\mathbf{d}_2 = (4, 2, 0)$ and $\mathbf{d}_3 = (0, 1, 3)$.

The second document is the first written out twice. Their Euclidean distance is $\|(2, 1, 0)\| = \sqrt{5} \approx 2.24$, which says they differ. Their cosine similarity is $(8 + 2)/(\sqrt{5}\sqrt{20}) = 10/10 = 1$, which says they are about exactly the same thing.

The first and third documents have cosine similarity $1/(\sqrt{5}\sqrt{10}) \approx 0.14$: different subjects. For comparing meaning, direction is the right measure and length is a distraction. Figure 2.4 sets the two measures side by side.



for documents 1 and 2 (red), distance reports a clear difference while the cosine reports perfect agreement. The second document is the first one repeated; only the cosine sees that.

FIGURE 2.4 The three documents of Example 2.4 compared two ways: Euclidean distance (left) and cosine similarity (right). Distance separates a document from its own repetition; the cosine recognises them as the same.

Key ideas

- The dot product is a sum of coordinate products and also $\|u\|\,\|v\|\cos\theta$, which defines angle in any dimension.
- Projection onto a vector leaves a residual perpendicular to it — the idea behind Gram–Schmidt, least squares and PCA.
- The Cauchy–Schwarz inequality guarantees that cosine similarity always lies between -1 and 1.
- Cosine similarity compares direction and ignores size, which is why retrieval and recommendation rely on it.
- Attention scores are scaled dot products turned into weights by a softmax, and the output is a weighted average of values.

Exercises

1. Compute $\mathbf{u}\cdot\mathbf{v}$ and the angle between $\mathbf{u} = (1, 0, 1)$ and $\mathbf{v} = (0, 1, 1)$.
2. Find a nonzero vector orthogonal to both $(1, 1, 0)$ and $(0, 1, 1)$.
3. Project $(3, 1, 2)$ onto $(1, 1, 1)$ and verify that the residual is orthogonal to $(1, 1, 1)$.
4. Show that $\|\mathbf{u} + \mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2$ exactly when $\mathbf{u}\cdot\mathbf{v} = 0$. What theorem is this in the plane?
5. In the attention formula, what happens to the weights w_i if every score s_i is multiplied by ten? By one tenth? Relate your answer to the division by $\sqrt{d}$.
6. Compute the cosine similarity of $(1, 2, 2)$ and $(2, 1, -2)$.
7. Show that for a unit vector $\mathbf{u}$ the projection of $\mathbf{v}$ is simply $(\mathbf{u}\cdot\mathbf{v})\,\mathbf{u}$.
8. In Example 2.3, recompute the weights when every key is doubled, and describe the change.

CHAPTER THREE

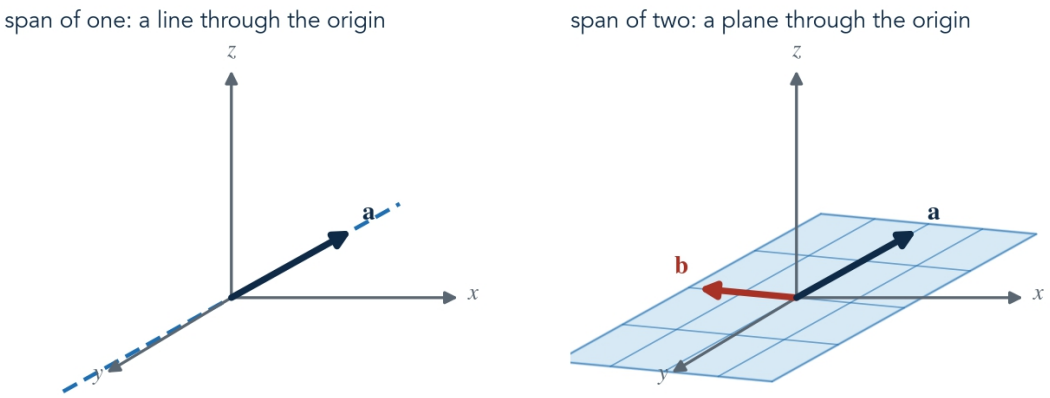
3 Span, Basis and Dimension

How many numbers does it take to describe a point? The obvious answer, "as many as the space has coordinates", turns out to be a question about a set of vectors rather than about the space. This chapter makes it precise, and in doing so shows that coordinates are a choice — one of the most liberating facts in the subject, and the one that machine learning exploits every time it re-describes data.

3.1 Span: everything a set of vectors can reach

Definition 3.1 Span

The span of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is the set of all their linear combinations, $c_1\mathbf{v}_1 + \ldots + c_k\mathbf{v}_k$ as the coefficients range over all real numbers.



the span is every combination the vectors can reach. A third vector adds a dimension only if it points out of the plane the first two already fill.

FIGURE 3.1 In three dimensions the span of one nonzero vector is a line through the origin, and the span of two vectors that point different ways is a plane through the origin. Neither reaches the rest of the space.

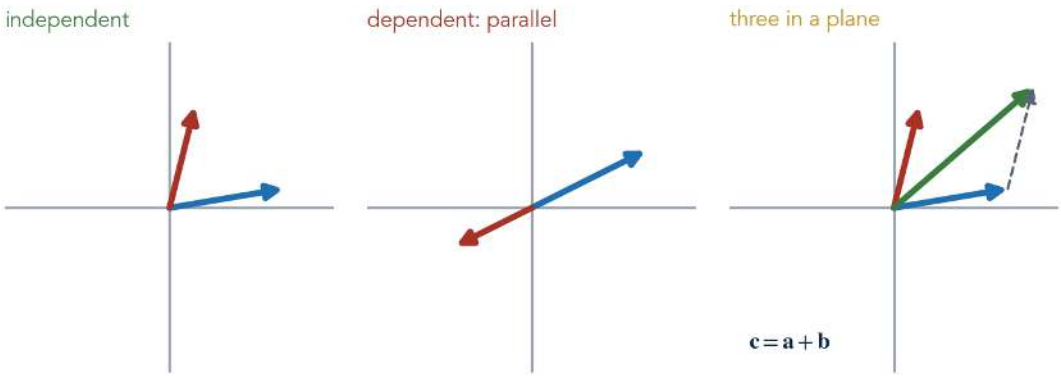
A span always contains the zero vector (take every coefficient zero), and it is always closed under the two operations of Chapter 1, so it is itself a vector space sitting inside the larger one. Such a thing is called a **subspace**. Lines and planes through the origin are subspaces of $\mathbb{R}^3$; a plane that misses the origin is not, because it does not contain $\mathbf{0}$.

3.2 Independence: when a vector adds a direction

Adding a vector to a set can enlarge its span or leave it unchanged. The difference is whether the new vector points somewhere the old ones could not already reach.

Definition 3.2 Linear independence

Vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are linearly independent if the only combination equal to zero is the trivial one: $c_1\mathbf{v}_1 + \ldots + c_k\mathbf{v}_k = \mathbf{0}$ forces every $c_i = 0$. Otherwise they are dependent.



a set is dependent when one member is a combination of the others -- it adds a vector without adding a direction. In the plane, any third vector is already reachable.

FIGURE 3.2 Two vectors that point different ways are independent. Two parallel vectors are dependent: one is a multiple of the other. And in the plane any three vectors are dependent, since the third can always be reached from the first two.

The definition is phrased around zero for a reason. If some nontrivial combination is zero, one of the vectors with a nonzero coefficient can be solved for in terms of the others — so it contributes nothing new to the span. Independence means no vector is redundant.

Worked Example 3.1 Testing independence

Are $\mathbf{a} = (1, 2, 0)$, $\mathbf{b} = (0, 1, 1)$ and $\mathbf{c} = (1, 4, 2)$ independent?

Look for $x\mathbf{a} + y\mathbf{b} + z\mathbf{c} = \mathbf{0}$. The first components give $x + z = 0$, the third give $y + 2z = 0$, and the second give $2x + y + 4z = 0$.

From the first two, $x = -z$ and $y = -2z$. Substituting into the third: $-2z - 2z + 4z = 0$, which holds for every z. Taking $z = 1$ gives $-\mathbf{a} - 2\mathbf{b} + \mathbf{c} = \mathbf{0}$.

So $\mathbf{c} = \mathbf{a} + 2\mathbf{b}$: the three vectors are dependent, and they span only a plane.

3.3 Basis and coordinates

Put the two ideas together: enough vectors to reach everything, and no more than necessary.

Definition 3.3 Basis

A basis of a vector space is a set of vectors that is linearly independent and spans the space.

Theorem 3.1 Unique coordinates

If $\mathbf{b}_1, \ldots, \mathbf{b}_n$ is a basis, every vector $\mathbf{x}$ in the space can be written as $\mathbf{x} = c_1\mathbf{b}_1 + \ldots + c_n\mathbf{b}_n$ in exactly one way.

Proof

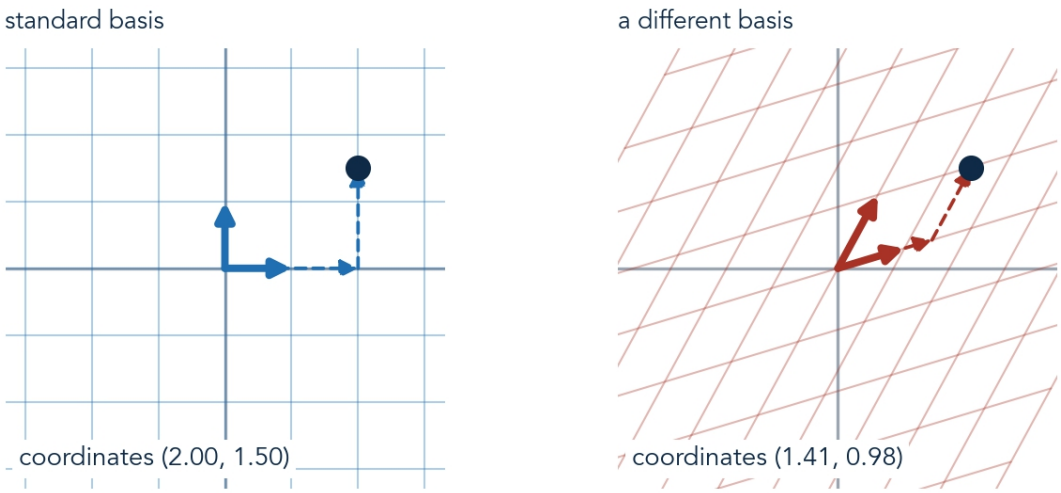
Because the basis spans, at least one such expression exists. Suppose there were two, with coefficients c_i and d_i.

Subtracting them gives $(c_1 - d_1)\mathbf{b}_1 + \ldots + (c_n - d_n)\mathbf{b}_n = \mathbf{0}$.

Because the basis is independent, every coefficient here is zero, so $c_i = d_i$ for all i. ■

The unique numbers $c_1, \ldots, c_n$ are the **coordinates** of $\mathbf{x}$ in that basis.

The standard basis of $\mathbb{R}^n$ — the vectors $\mathbf{e}_1 = (1, 0, \ldots, 0)$ through $\mathbf{e}_n = (0, \ldots, 0, 1)$ — gives the coordinates we have been writing all along. But it is only one basis among infinitely many.



the black point does not move. Its coordinates are an address, and an address depends on the street plan: change the basis and the same point gets different numbers.

FIGURE 3.3 The same point in two coordinate systems. In the standard basis it has coordinates $(2, 1.5)$; in the slanted basis on the right it has different coordinates. The point never moved.

Coordinates are an address, not a property. A point has no coordinates until a basis is chosen, and much of the art of linear algebra — and of representation learning — is choosing the basis in which a problem becomes simple.

Worked Example 3.2 Coordinates in a new basis

Find the coordinates of $\mathbf{x} = (5, 1)$ in the basis $\mathbf{b}_1 = (1, 1)$, $\mathbf{b}_2 = (1, -1)$.

Solve $c_1(1, 1) + c_2(1, -1) = (5, 1)$: that is $c_1 + c_2 = 5$ and $c_1 - c_2 = 1$. Adding gives $c_1 = 3$ and so $c_2 = 2$.

In the new basis $\mathbf{x}$ has coordinates $(3, 2)$. Because $\mathbf{b}_1$ and $\mathbf{b}_2$ happen to be orthogonal, there is a shortcut: each coordinate is a dot product divided by a squared length, $c_1 = \mathbf{x}\cdot\mathbf{b}_1 / \|\mathbf{b}_1\|^2 = 6/2 = 3$ and $c_2 = 4/2 = 2$.

The shortcut in that example is not a coincidence, and it is the reason orthogonal bases are so prized. In an orthogonal basis the coordinates are projections (Section 2.2), computed one at a time with no system of equations to solve. Chapter 10 builds such bases on demand.

3.4 Dimension

Every basis of a given space has the same number of vectors. The proof is a counting argument — in a space spanned by m vectors, any $m + 1$ vectors must be dependent — and the consequence is a number that belongs to the space itself.

> **Definition 3.4** Dimension
>
> The dimension of a vector space is the number of vectors in any basis.

$$\dim \mathbb{R}^n = n, \qquad \dim\{\text{linethrough}\mathbf{0}\} = 1, \qquad \dim\{\text{polynomialsofdegree} \le 3\} = 4 \tag{3.1}$$

Dimension is the number of independent quantities needed to pin a point down, and it is often far smaller than the number of coordinates used to write it. A set of measurements of 50 quantities that always satisfy 47 independent relationships lives in a space of dimension three, however many numbers each measurement has.

TABLE 3.1 The vocabulary of this chapter and how each idea is tested. The third column is the question you would actually ask of a set of vectors.

idea	meaning	test
span	everything reachable by combinations	can $A\mathbf{c} = \mathbf{x}$ be solved?
independence	no vector is redundant	does $A\mathbf{c} = \mathbf{0}$ force $\mathbf{c} = \mathbf{0}$?
basis	independent and spanning	both of the above
dimension	size of any basis	count the independent vectors
coordinates	the unique coefficients in a basis	solve, or project if orthogonal

Worked Example 3.3 The dimension of a solution space

Find a basis and the dimension of the set of (x, y, z, w) with $x + y + z + w = 0$ and $x - y = 0$.

The second equation gives $y = x$, and then the first gives $w = -2x - z$. The free choices are x and z.

Taking $x = 1, z = 0$ gives $(1, 1, 0, -2)$; taking $x = 0, z = 1$ gives $(0, 0, 1, -1)$. These are independent and span the set, which has dimension 2 — four unknowns less two independent equations.

3.5 Why a change of basis is the central move of learning

Nearly every transformation applied to data in machine learning is a change of basis followed by a decision about which coordinates to keep. Principal component analysis (Chapter 17) finds a basis in which the first few coordinates carry most of the variation, then discards the rest. The Fourier transform used in signal processing expresses a signal in a basis of waves. The internal layers of a network re-express their input in bases that are learned, and a layer with fewer units than its input is forced to find a lower-dimensional description.

In machine learning. "The model learns a representation" means the model learns a new set of coordinates for its input. When those coordinates make the task simple — when, say, a straight line separates the classes — the representation is good. The mathematics of choosing coordinates is the mathematics of this chapter, and "dimensionality reduction" is the statement that the data lives in a subspace, or near one, of much smaller dimension than the space it was recorded in.

The intrinsic dimension of real data is almost always much smaller than the number of coordinates it is recorded in. A photograph has a million pixels, but

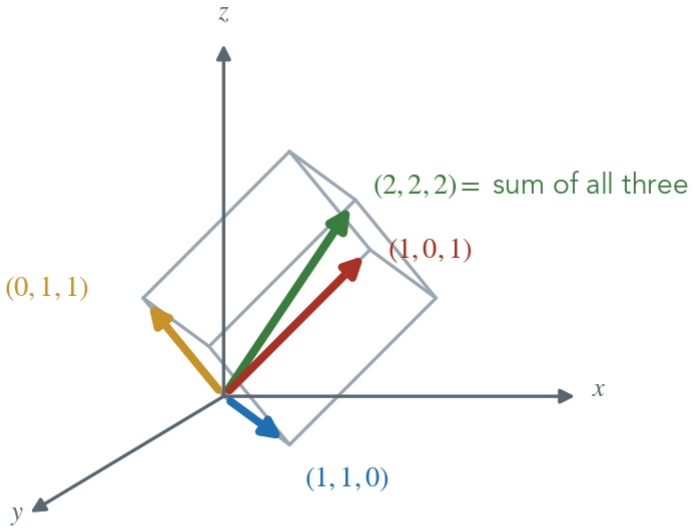
photographs of faces vary along far fewer independent directions. Learning is possible largely because of this gap, and much of linear algebra's usefulness is in finding the small space inside the large one.

Worked Example 3.4 Checking a basis of three-dimensional space

Are $(1, 1, 0)$, $(0, 1, 1)$ and $(1, 0, 1)$ a basis of $\mathbb{R}^3$, and what are the coordinates of $(2, 2, 2)$ in it?

Independence: $a(1, 1, 0) + b(0, 1, 1) + c(1, 0, 1) = \mathbf{0}$ gives $a + c = 0$, $a + b = 0$ and $b + c = 0$. Then $a = -c$ and $b = c$, so $b + c = 2c = 0$ and all three are zero. Three independent vectors in a space of dimension three form a basis.

Coordinates: the same equations with right-hand sides 2, 2, 2 add up to $2(a + b + c) = 6$, so $a + b + c = 3$, and each coefficient is $3 - 2 = 1$. Indeed $(1, 1, 0) + (0, 1, 1) + (1, 0, 1) = (2, 2, 2)$. Figure 3.4 shows the walk.



the three coloured arrows form a basis, and the box they span has $(2, 2, 2)$ as its far corner. Reaching it takes one step along each arrow, so its coordinates in this basis are $1, 1, 1$.

FIGURE 3.4 The basis of Example 3.4 and the vector $(2, 2, 2)$. One step along each basis vector reaches it, which is what the coordinates $1, 1, 1$ record.

Key ideas

- The span of a set of vectors is everything their combinations reach, and it is always a subspace through the origin.
- Vectors are independent when none is a combination of the others: each adds a new direction.
- A basis is independent and spanning, and it gives every vector unique coordinates.
- Every basis of a space has the same number of vectors, the dimension of the space.
- Coordinates are an address that depends on the basis; choosing a good basis is much of the art of linear algebra and of representation learning.

Exercises

1. Is $(1, 1, 1)$ in the span of $(1, 0, 1)$ and $(0, 1, 0)$? Is $(1, 2, 3)$?
2. Show that $(1, 2)$, $(3, 5)$ and $(0, 1)$ are dependent by finding explicit coefficients.
3. Find the coordinates of $(4, 6)$ in the basis $(1, 2)$, $(2, 1)$.
4. Give a basis for the plane $x - 2y + z = 0$ in $\mathbb{R}^3$, and state its dimension.
5. Explain why a set of 10 vectors in $\mathbb{R}^8$ cannot be independent, and why a set of 6 vectors in $\mathbb{R}^8$ cannot span.
6. Show that the set containing only $\mathbf{0}$, and the whole space, are both subspaces.
7. Find the dimension of the span of $(1, 1, 0)$, $(0, 1, 1)$ and $(1, 2, 1)$.
8. Using dot products, find the coordinates of $(3, -1)$ in the orthogonal basis $(1, 2)$, $(2, -1)$.

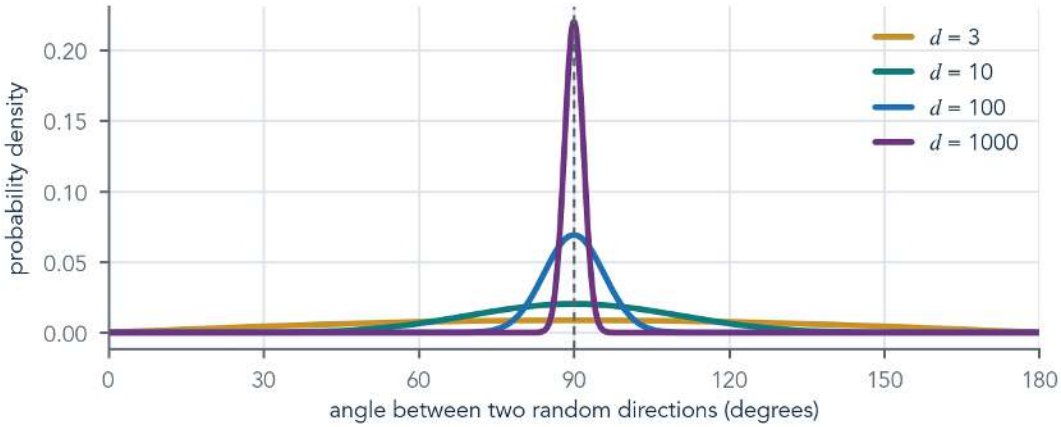
CHAPTER FOUR

4 The Geometry of High Dimensions

Everything so far has been drawn in two or three dimensions, and every formula has been written so that it works in any number. That is true, and it hides a trap. The formulas carry over; the intuition does not. A space of a thousand dimensions behaves in ways that no picture of a plane can suggest, and three of those ways are the reason modern representations work at all.

4.1 The angle between random directions

Pick two directions in the plane at random. The angle between them could be anything from 0° to 180°, and every value is equally likely. In three dimensions angles near 90° are already favoured. In a thousand dimensions something dramatic has happened.



the density is proportional to $\sin^{d-2}\theta$. In three dimensions any angle is plausible; in a thousand, two random directions are within a few degrees of perpendicular.

FIGURE 4.1 The distribution of the angle between two directions chosen uniformly at random, in three, ten, one hundred and one thousand dimensions. As the dimension grows the distribution collapses onto a right angle.

Theorem 4.1 Random directions are nearly orthogonal

If $\mathbf{u}$ and $\mathbf{v}$ are independent random unit vectors in $\mathbb{R}^d$, the density of the angle θ between them is proportional to $\sin^{d-2}\theta$, and for every $\varepsilon > 0$, $P(|\mathbf{u}\cdot\mathbf{v}| \geq \varepsilon) \leq 2e^{-d\varepsilon 2/2}$.

Proof

By symmetry we may fix $\mathbf{u}$ to be the north pole $\mathbf{e}_1$; then $\mathbf{u}\cdot\mathbf{v}$ is the first coordinate of $\mathbf{v}$.

The set of unit vectors at angle θ from the pole is a sphere of dimension $d-2$ with radius $\sin\theta$, whose size is proportional to $\sin^{d-2}\theta$; weighting each angle by that size gives the density.

For the tail bound, the fraction of the unit sphere with first coordinate at least ε is a polar cap. Its share of the sphere equals the share of the unit ball occupied by the cone joining the origin to the cap, and for $\varepsilon \leq 1/\sqrt{2}$ that cone fits inside a ball of radius $\sqrt{1-\varepsilon^2}$ (a slightly different ball handles larger ε). Comparing volumes gives a share of at most $(1-\varepsilon^2)^{d/2} \leq e^{-d\varepsilon 2/2}$. The factor 2 covers the cap at the other pole. ■

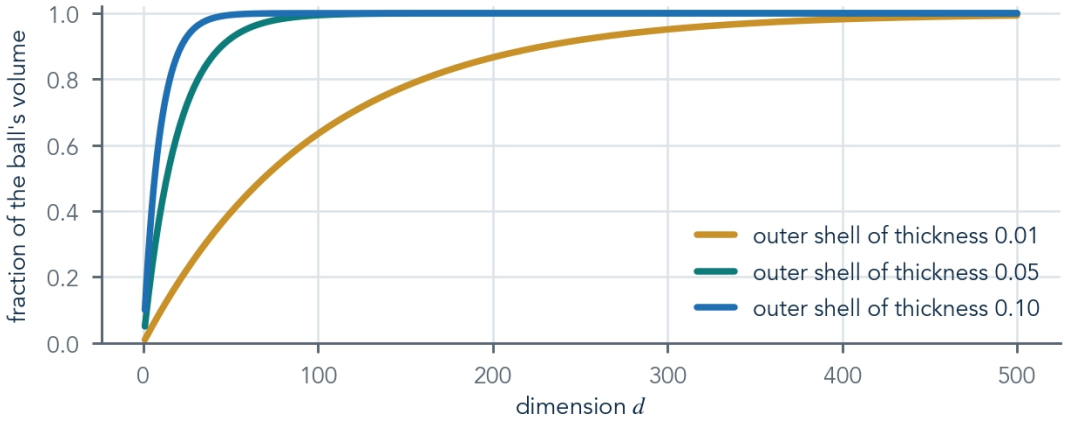
Put numbers in. In $d = 1000$ dimensions, the chance that two random directions have $|\cos\theta| \geq 0.1$ is at most $2e^{-5} \approx 0.013$; the chance that $|\cos\theta| \geq 0.2$ is at most $2e^{-20}$, about one in a quarter of a billion. Almost every pair of directions in a large space is within a few degrees of perpendicular.

> **In high dimensions, orthogonality is the default and similarity is the exception. That is why a dot product between two learned vectors carries information: agreement is rare enough that when it appears, it means something.**

4.2 Where the volume of a ball lives

The second surprise concerns volume. The volume of a ball of radius r in d dimensions is proportional to r^d. So the fraction of a unit ball lying within radius $1-\varepsilon$ of the centre is $(1-\varepsilon)^d$.

$$\frac{\text{vol}(\text{ball of radius } 1-\varepsilon)}{\text{vol}(\text{ball of radius } 1)} = (1-\varepsilon)^d \le e^{-\varepsilon d} \tag{4.1}$$



the volume inside radius $1-\varepsilon$ is $(1-\varepsilon)^d$ of the whole, which vanishes. In high dimension almost all of a ball is skin: 'typical' points sit near the surface, not near the centre.

FIGURE 4.2 The share of a ball's volume in its outer shell. In three dimensions a shell one-hundredth as thick as the radius holds 3% of the volume. In five hundred dimensions it holds 99%.

Worked Example 4.1 How thin is the skin?

In $d = 1000$ dimensions, what fraction of the unit ball lies within radius 0.99?

$(0.99)^{1000} = e^{1000 \ln 0.99} = e^{-10.05} \approx 0.000043.$

Less than five thousandths of one per cent of the volume is deeper than one per cent below the surface. A point chosen at random from the ball is, for practical purposes, on its boundary.

A related fact is easier to use. If the coordinates of a vector are independent with mean zero and variance one, its squared length is a sum of d such squares, with mean d; the typical length is about $\sqrt{d}$,

and the spread around that value stays roughly constant as d grows. So random vectors in high dimension all have nearly the same length. This is precisely why Equation 2.4 divides dot products by $\sqrt{d}$: a dot product of two such vectors has variance d, and without the scaling the attention scores would grow with dimension until the softmax saturated.

4.3 How many nearly perpendicular directions fit

Exactly perpendicular directions are scarce: $\mathbb{R}^d$ has room for d of them and no more, since they would be independent (Chapter 3). But if we relax "perpendicular" to "nearly perpendicular", the count explodes — and this is the calculation that makes the section heading a matter of practical importance.

Theorem 4.2 Exponentially many near-orthogonal directions

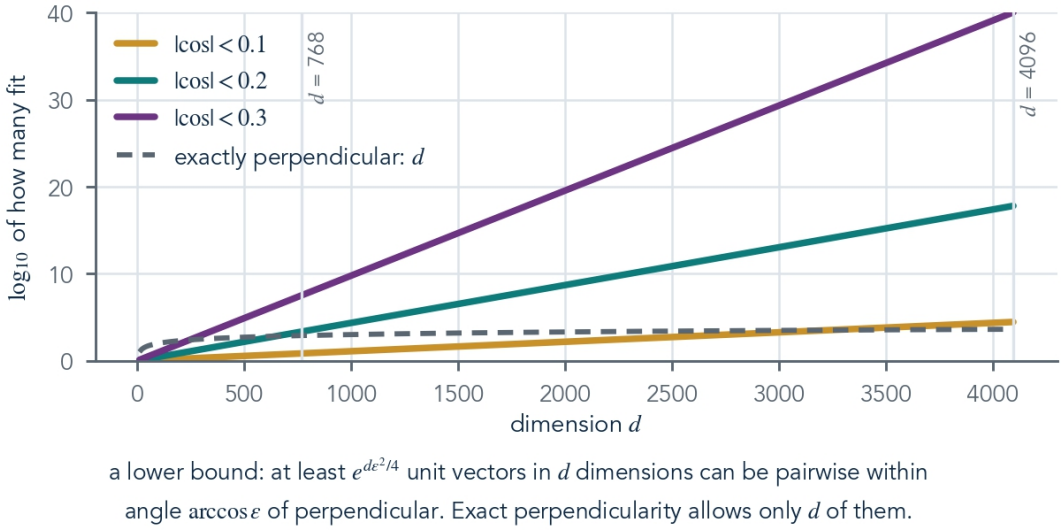
For any $0 < \varepsilon < 1$ there exist at least $N = e^{d\varepsilon 2/4}$ unit vectors in $\mathbb{R}^d$ with $|\mathbf{u}_i \cdot \mathbf{u}_j| < \varepsilon$ for every pair $i \neq j$.

Proof

Choose N unit vectors independently at random. By Theorem 4.1 each of the $N(N-1)/2$ pairs fails the condition with probability at most $2e^{-d\varepsilon 2/2}$.

The chance that some pair fails is therefore at most $N(N-1)e^{-d\varepsilon 2/2}$, which is strictly less than $N^2 e^{-d\varepsilon 2/2}$.

With $N \leq e^{d\varepsilon 2/4}$ we have $N^2 e^{-d\varepsilon 2/2} \leq 1$, so the chance of any failure is below one. With positive probability every pair succeeds, and so a set with the property exists. ■



a lower bound: at least $e^{d\varepsilon^2/4}$ unit vectors in d dimensions can be pairwise within angle $\arccos\varepsilon$ of perpendicular. Exact perpendicularity allows only d of them.

FIGURE 4.3 The logarithm of the guaranteed count $e^{d\varepsilon 2/4}$ against dimension, for three tolerances, compared with the d exactly perpendicular directions. At $d = 4096$ and $|\cos| < 0.3$, the bound exceeds 10^{39}.

TABLE 4.1 Lower bounds on the number of pairwise nearly perpendicular unit vectors, $e^{d\varepsilon 2/4}$. These are guarantees; real constructions do better.

dimension d	cosine below 0.1 in size	cosine below 0.2 in size	cosine below 0.3 in size
768	6	about 2,200	about 3.2×10^7
1,024	12	about 28,000	about 1.0×10^{10}
4,096	about 28,000	about 6.1×10^{17}	about 10^{40}

4.4 Superposition: more features than dimensions

Theorem 4.2 has a striking consequence for how learned representations can be organised, and a substantial body of research over the last few years has been devoted to testing it.

Suppose a model needs to represent many distinct properties of its input — whether a word is plural, whether an image contains a wheel, whether a sentence is about money. The simplest scheme gives each property its own direction in the representation space and represents an input by adding up the directions of the properties it has. If the directions were exactly perpendicular, a space of

dimension d could hold only d properties. Because nearly perpendicular directions are so plentiful, it can hold vastly more, provided each input has only a few properties at once: the small overlaps between directions then produce only small interference.

> **The representation space of a trained model is not a list of d slots. It is a space in which exponentially many nearly perpendicular directions are available, and a model that stores sparse features along such directions — superposition — can represent far more concepts than it has coordinates. The price is a little interference, and sparsity keeps it small.**

The **linear representation hypothesis** is the claim that high-level properties of the input are in fact encoded as directions in this way, so that moving a representation along a direction changes one property and leaves the others roughly alone. Work from 2023 onwards has tested it by training **sparse dictionaries** on the internal vectors of large transformer models — sets of many more directions than the space has dimensions, chosen so that each internal vector is well approximated by a combination of only a few of them — and reported large numbers of directions that correspond to recognisable properties. Refinements during 2024 and 2025, such as dictionaries that keep only the largest few coefficients or that are organised in nested levels of detail, have improved how cleanly those directions can be recovered. The mathematics that makes the approach possible is the geometry of this chapter.

> **In machine learning.** Three consequences are worth remembering. First, cosine similarity is informative in high dimensions because random vectors score near zero. Second, dot products must be scaled by $\sqrt{d}$ when vectors have many independent coordinates. Third, a space of a few thousand dimensions has room for an astronomical number of distinct, nearly independent directions, which is what lets a single vector carry many facts at once.

Worked Example 4.2 Why attention divides by the square root of the dimension

Suppose query and key vectors in $d = 768$ dimensions have independent coordinates with mean 0 and variance 1.

Their dot product is a sum of 768 independent products, each with mean 0 and variance 1, so its standard deviation is $\sqrt{768} \approx 27.7$. Scores that large fed to a softmax make one weight essentially 1 and the rest essentially 0.

Dividing by $\sqrt{768}$ gives scores with standard deviation 1, so the softmax produces graded weights and gradients can flow through all of them.

4.5 Preserving distances in fewer dimensions

The final fact runs in the opposite direction and is just as useful. If high-dimensional space has so much room, perhaps a set of points can be squeezed into fewer dimensions without much distortion.

Theorem 4.3 Johnson–Lindenstrauss

For any N points in any dimension and any $0 < \varepsilon < 1$, there is a linear map into $\mathbb{R}^k$ with $k = O(\varepsilon^{-2} \log N)$ that preserves every pairwise distance to within a factor $1 \pm \varepsilon$. A random projection, suitably scaled, has this property with high probability.

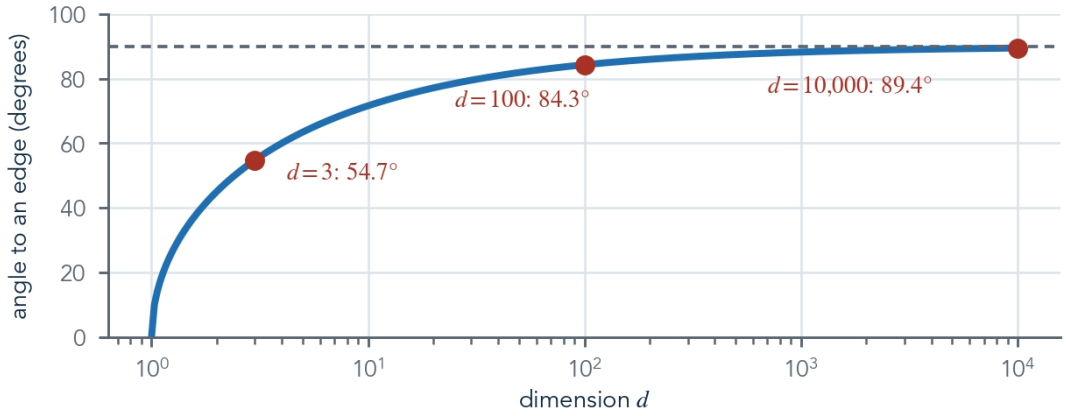
The required dimension depends on the *number of points* and the tolerance, not on the original dimension at all. A million points can be placed in a few thousand dimensions with all distances preserved to within ten per cent, whether they started in a space of ten thousand dimensions or ten billion. The proof uses the same concentration of measure as Theorem 4.1: a random projection shrinks every fixed vector by very nearly the same factor, because in high dimensions almost everything is typical.

Worked Example 4.3 The diagonal of a high-dimensional cube

In d dimensions the unit cube has a main diagonal from 0 to (1, 1, ..., 1), of length $\sqrt{d}$.

The cosine of the angle between the diagonal and any coordinate axis is $\mathbf{e}_1 \cdot (1, \ldots, 1)/\sqrt{d} = 1/\sqrt{d}$.

In three dimensions that angle is about 54.7°. In $d = 100$ the cosine is 0.1 and the angle is about 84.3°; in $d = 10{,}000$ it is about 89.4°. The line through the cube's opposite corners is nearly perpendicular to every edge — the same concentration as Theorem 4.1, seen in a single, fixed direction. Figure 4.4 plots the angle for every dimension up to ten thousand.



the long diagonal of a cube leans ever closer to a right angle with every edge as the dimension grows: a single fixed direction already shows the concentration of Chapter 4.

FIGURE 4.4 The angle between the main diagonal of a d-dimensional cube and any of its edges, from Example 4.3. It passes 84° by $d = 100$ and never stops approaching 90°.

Key ideas

- Two random directions in high dimension are almost always nearly perpendicular; the angle between them concentrates at a right angle.
- Almost all the volume of a high-dimensional ball lies in a thin shell near its surface.
- Random vectors with many independent coordinates all have nearly the same length, about $\sqrt{d}$.
- Exponentially many nearly perpendicular directions fit in d dimensions, which lets representations hold more features than coordinates.
- A random projection to about $\varepsilon^{-2}\log N$ dimensions preserves all distances among N points.

Exercises

1. Using Theorem 4.1, bound the probability that two random unit vectors in $\mathbb{R}^{500}$ have cosine similarity above 0.15 in size.
2. What fraction of a unit ball in $\mathbb{R}^{100}$ lies within radius 0.9?
3. A vector in $\mathbb{R}^d$ has independent coordinates of mean 0 and variance 1. What is the expected value of its squared length? Of its dot product with an independent vector of the same kind?
4. Use Table 4.1 to explain why storing features along exactly perpendicular directions would waste most of a 4,096-dimensional space.
5. Explain in words why the Johnson–Lindenstrauss dimension does not depend on the original dimension of the data.
6. In $d = 100$, bound the probability that two random unit vectors have cosine similarity at least 0.3 in size.
7. How many pairwise nearly perpendicular directions, with cosine below 0.1 in size, does Theorem 4.2 guarantee in $d = 2000$?
8. Explain in one sentence why dot products of random vectors grow like $\sqrt{d}$.

PART

II

MATRICES

the maps

CHAPTER FIVE

5 A Matrix Is a Linear Map

Part I was about the objects. Part II is about what can be done to them, and the answer is contained in one sentence that the rest of the book keeps returning to: a matrix is a machine that takes a vector in and puts a vector out, in a way that keeps straight lines straight. This chapter builds that sentence from nothing, and shows why the grid of numbers we call a matrix is exactly the right record of such a machine.

5.1 Linear maps

Definition 5.1 Linear map

A function T from $\mathbb{R}^n$ to $\mathbb{R}^m$ is linear if it respects the two operations of Chapter 1: $T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$ and $T(c\mathbf{v}) = c\,T(\mathbf{v})$ for all vectors and numbers.

Together the two conditions say that T commutes with linear combinations: the image of a combination is the same combination of the images. That one property has a dramatic consequence.

Theorem 5.1 A linear map is determined by the basis

If T is linear, then T is completely determined by the n vectors $T(\mathbf{e}_1), \ldots, T(\mathbf{e}_n)$.

Proof

Every $\mathbf{x}$ in $\mathbb{R}^n$ is $x_1\mathbf{e}_1 + \ldots + x_n\mathbf{e}_n$.

Applying linearity, $T(\mathbf{x}) = x_1\ T(\mathbf{e}_1) + \ldots + x_n\ T(\mathbf{e}_n)$.

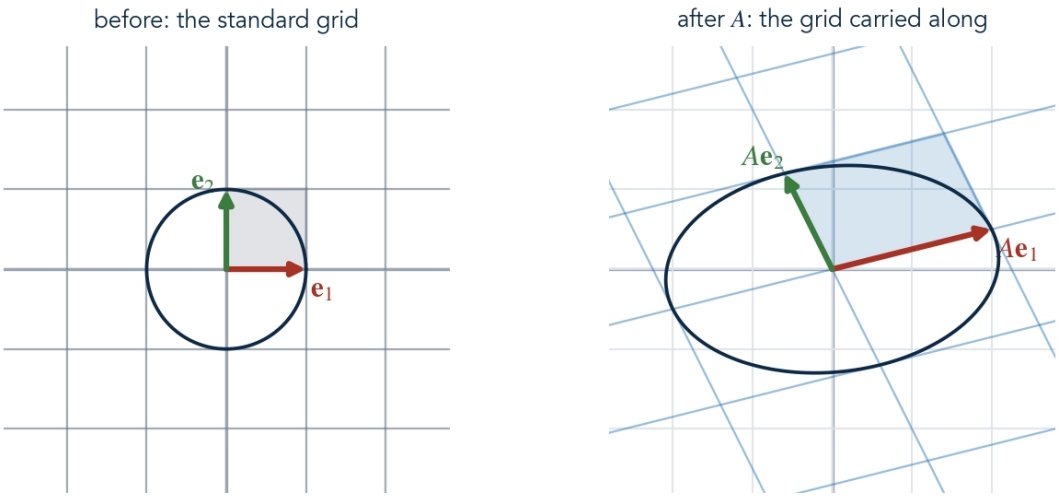
So once the images of the basis vectors are known, the image of every vector is known. ■

A function on an infinite set of inputs is pinned down by a finite amount of information. Record the n image vectors as the columns of a grid, and the grid is the whole map.

Definition 5.2 The matrix of a linear map

The matrix of T is the $m \times n$ array whose j-th column is $T(\mathbf{e}_j)$. We write $A\mathbf{x}$ for $T(\mathbf{x})$.

$$A = \begin{bmatrix} | & | & & | \\ T(\mathbf{e}_1) & T(\mathbf{e}_2) & \cdots & T(\mathbf{e}_n) \\ | & | & & | \end{bmatrix}, \qquad A\mathbf{x} = x_1 A\mathbf{e}_1 + \cdots + x_n A\mathbf{e}_n \tag{5.1}$$



$A = [A\mathbf{e}_1 \;\; A\mathbf{e}_2]$: the first column is where $\mathbf{e}_1$ goes, the second where $\mathbf{e}_2$ goes.
Grid lines stay straight, parallel and evenly spaced -- that is what 'linear' means.

FIGURE 5.1 Left: the standard grid with $\mathbf{e}_1$ in red and $\mathbf{e}_2$ in green. Right: after the map. The two arrows land at the columns of A, and the entire grid is carried along with them — straight, parallel and evenly spaced.

> **The single most useful sentence in linear algebra is "the columns of a matrix are where the basis vectors go". With it, any matrix can be read as a picture before a single multiplication is done, and most matrix identities can be checked by drawing.**

5.2 Matrix times vector, two ways

Equation 5.1 already gives one way to compute $A\mathbf{x}$: as a combination of the columns, weighted by the entries of $\mathbf{x}$. That is the **column picture**, and it is the one that carries meaning. The familiar rule — each output entry is a row of A dotted with $\mathbf{x}$ — is the **row picture**, and it is the one that is quick to compute.

$$\begin{bmatrix} 2 & -1 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} 4 \\ 1 \end{bmatrix} = 4\begin{bmatrix} 2 \\ 1 \end{bmatrix} + 1\begin{bmatrix} -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2\cdot 4 + (-1)\cdot 1 \\ 1\cdot 4 + 3\cdot 1 \end{bmatrix} = \begin{bmatrix} 7 \\ 7 \end{bmatrix} \tag{5.2}$$

Worked Example 5.1 Building a matrix from a description

Find the matrix that rotates the plane by 90° anticlockwise.

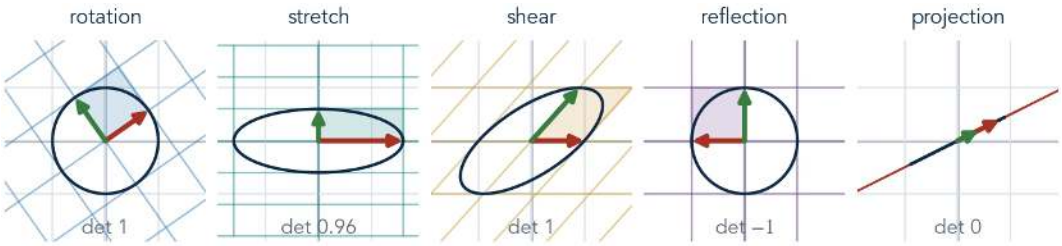
We only need to know where the basis goes. Rotating $\mathbf{e}_1 = (1, 0)$ by 90° gives $(0, 1)$; rotating $\mathbf{e}_2 = (0, 1)$ gives $(-1, 0)$.

Those are the columns, so the matrix has first column $(0, 1)$ and second column $(-1, 0)$. Check on $\mathbf{x} = (3, 2)$: the result is $3(0, 1) + 2(-1, 0) = (-2, 3)$, which is indeed $(3, 2)$ turned a quarter turn.

The rotation came from a description with no arithmetic at all. This is how matrices should be built: decide what the map does to each basis vector and write the answers down as columns.

5.3 A gallery of maps

Five kinds of map account for most of the geometry anyone needs, and every matrix can be understood as a combination of them (Chapter 15 makes that precise).



the unit square (shaded) and unit circle (dark) under five maps. Rotations and reflections keep lengths; a stretch and a shear distort them; a projection flattens the plane onto a line.

FIGURE 5.2 The unit square and the unit circle under a rotation, a stretch, a shear, a reflection and a projection. The determinant under each panel — the area factor of Chapter 8 — is a first hint at what distinguishes them.

TABLE 5.1 The gallery as matrices. Each can be verified from its columns: they are the images of $\mathbf{e}_1$ and $\mathbf{e}_2$.

map	matrix (by columns)	what it keeps
rotation by θ	$(\cos\theta, \sin\theta)$, $(-\sin\theta, \cos\theta)$	lengths and angles
stretch	$(a, 0)$, $(0, b)$	the axes' directions
shear	$(1, 0)$, $(k, 1)$	area, horizontal lines
reflection in the y-axis	$(-1, 0)$, $(0, 1)$	lengths, but not orientation
projection onto a line through $\mathbf{u}$	$\mathbf{u}\mathbf{u}^T$ for unit $\mathbf{u}$	only the line itself

The projection deserves a note, because it is the first map in the gallery that destroys information. Every point is flattened onto a line, so the whole plane lands on a set of dimension one and there is no way back. Chapter 7 names what is destroyed.

5.4 Rectangular matrices change dimension

Nothing in Definition 5.2 requires $m = n$. A 2×3 matrix takes vectors from three dimensions to two; a 3×2 matrix takes vectors from two dimensions into three, where the whole image is at most a plane. Rectangular matrices are the ordinary case in machine learning: a layer that turns a 768-dimensional vector into a 3,072-dimensional one is a 3072×768 matrix.

Worked Example 5.2 A map from three dimensions to two

Let A have columns $(1, 0)$, $(0, 1)$ and $(1, 1)$. What does it do to $(2, -1, 3)$, and which vectors does it send to zero?

The column picture gives $2(1, 0) - 1(0, 1) + 3(1, 1) = (5, 2)$.

For $A\mathbf{x} = \mathbf{0}$ we need $x_1 + x_3 = 0$ and $x_2 + x_3 = 0$, so $x_1 = x_2 = -x_3$. Every multiple of $(-1, -1, 1)$ is sent to zero: a whole line of the input space disappears.

5.5 A layer of a neural network

The most common matrix in machine learning is the weight matrix of a layer, and the picture of this chapter describes exactly what it does.

$$\mathbf{h} = \sigma(W\mathbf{x} + \mathbf{b}), \qquad W \in \mathbb{R}^{m \times n},\ \mathbf{b} \in \mathbb{R}^{m} \tag{5.3}$$

The input $\mathbf{x}$ is mapped linearly by W, shifted by $\mathbf{b}$, and then passed through a simple function σ applied to each coordinate separately — most often the rectifier, which replaces negative numbers by zero.

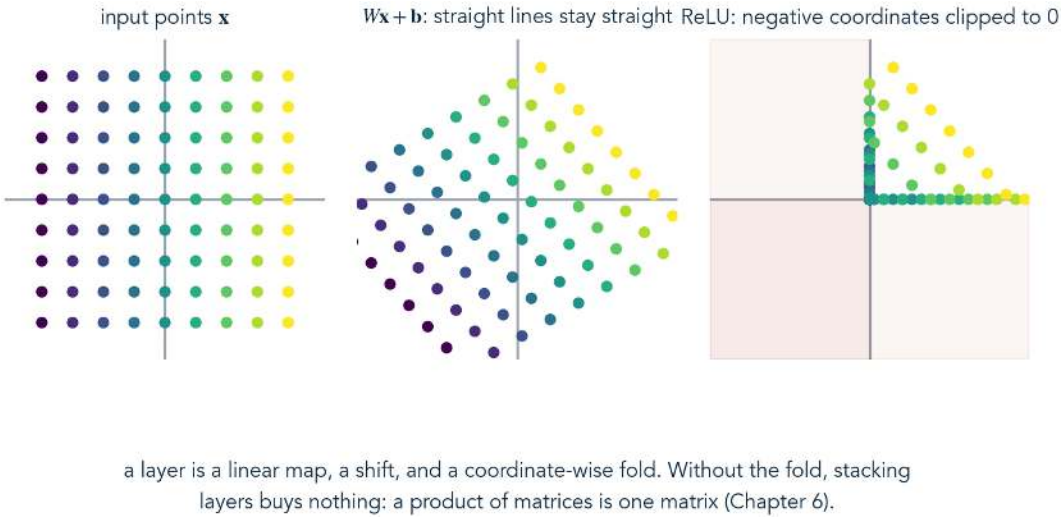


FIGURE 5.3 Points of the plane (left), after a linear map and a shift (centre), and after the rectifier (right). The linear step keeps lines straight; the fold bends the space along the coordinate axes. Alternating the two is what lets a network represent curved boundaries.

In machine learning. Each row of W is a direction, and the corresponding output coordinate of $W\mathbf{x}$ is the dot product of the input with that direction — the "neuron" is a detector for how far the input lies along its row. Read by columns, W says where each input coordinate is sent in the output space. Both readings are used: rows for asking what a unit responds to, columns for asking what an input feature contributes.

Worked Example 5.3 A layer, unit by unit

Let W have rows $(1, -1)$ and $(2, 1)$, let $\mathbf{b} = (0, -1)$, and use the rectifier.

For $\mathbf{x} = (3, 1)$: $W\mathbf{x} = (2, 7)$, adding $\mathbf{b}$ gives $(2, 6)$, and the rectifier leaves $(2, 6)$.

For $\mathbf{x} = (1, 3)$: $W\mathbf{x} = (-2, 5)$, adding $\mathbf{b}$ gives $(-2, 4)$, and the rectifier gives $(0, 4)$.

The first unit computes $x_1 - x_2$ and responds only when the first coordinate exceeds the second: its row is a direction, and the unit reports which side of the line $x_1 = x_2$ the input lies on.

5.6 Affine maps and a useful trick

The shift $\mathbf{b}$ in Equation 5.3 makes the map not quite linear: it sends $\mathbf{0}$ to $\mathbf{b}$ rather than to $\mathbf{0}$. Such maps are called **affine**. A standard trick restores linearity by adding one coordinate that is always 1.

$$\begin{bmatrix} W\mathbf{x} + \mathbf{b} \\ 1 \end{bmatrix} = \begin{bmatrix} W & \mathbf{b} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{5.4}$$

Every affine map in n dimensions is a linear map in $n + 1$. Computer graphics uses this to write rotations and translations as a single matrix, and it is why a linear model with an intercept can be analysed with exactly the same tools as one without — the intercept is the weight on a constant feature.

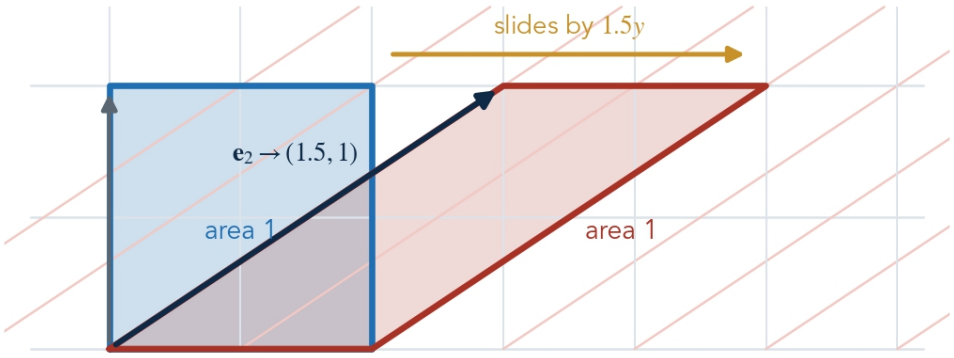
> **A matrix is not a table of numbers that happens to support a multiplication rule. It is a complete description of a geometric action, compressed into mn numbers by the fact that linearity lets the images of n vectors determine everything. The multiplication rule is a consequence, and the next chapter derives it.**

Worked Example 5.4 Reading a shear from its columns

Let A have columns $(1, 0)$ and $(1.5, 1)$.

The first column says horizontal vectors are unchanged. The second says $\mathbf{e}_2$ is pushed to $(1.5, 1)$: every point at height y slides right by $1.5y$.

The unit square's corners $(0, 0)$, $(1, 0)$, $(1, 1)$, $(0, 1)$ go to $(0, 0)$, $(1, 0)$, $(2.5, 1)$, $(1.5, 1)$: a parallelogram with base 1 and height 1, so the area is still 1. A shear slants space without changing area, which Chapter 8 records as a determinant of one. Figure 5.4 shows the square and its image.



the unit square (blue) and its image (red) under the shear of Example 5.4. Each row of the square slides sideways in proportion to its height; base and height, and so area, are unchanged.

FIGURE 5.4 The shear of Example 5.4 applied to the unit square. The first column fixes the bottom edge; the second column tips the left edge over to $(1.5, 1)$.

Key ideas

- A linear map is determined completely by where it sends the basis vectors.
- The columns of a matrix are the images of the basis vectors, so a matrix can be read as a picture.
- Matrix times vector is a combination of columns, for meaning, or a list of row dot products, for computation.
- Rectangular matrices change dimension, and some maps, such as projections, destroy information.
- A network layer is a linear map, a shift and a coordinate-wise fold; each row of its weights is a direction detector.

Exercises

1. Find the matrix that reflects the plane in the line $y = x$. Check it on $(3, 1)$.
2. Find the matrix of the map that projects every vector of the plane onto the x-axis, and describe everything it sends to $\mathbf{0}$.
3. Show that $T(x, y) = (x + 1, y)$ is not linear, and write it as a linear map in three coordinates using Equation 5.4.
4. Compute $A\mathbf{x}$ both by columns and by rows for A with columns $(1, 2, 0)$ and $(3, -1, 1)$ and $\mathbf{x} = (2, 5)$.
5. A layer maps $\mathbb{R}^{512}$ to $\mathbb{R}^{2048}$. How many numbers are in its weight matrix and bias? What is the largest possible dimension of the set of all outputs $W\mathbf{x}$?
6. Find the matrix of a rotation by $180°$.
7. Find the matrix that sends $\mathbf{e}_1$ to $(1, 1)$ and $\mathbf{e}_2$ to $(1, -1)$, and apply it to $(2, 3)$.
8. Show that every linear map sends $\mathbf{0}$ to $\mathbf{0}$.

CHAPTER SIX

6 Multiplication Is Composition

The rule for multiplying two matrices is usually presented as a definition to be memorised: row of the first against column of the second, over and over. It is not a definition. It is the unique rule that makes the product of two matrices the matrix of doing one map and then the other, and once seen that way, every property of matrix multiplication — including the ones that surprise — follows from a picture.

6.1 What the product has to be

Suppose B is an $n \times p$ matrix, taking $\mathbb{R}^p$ to $\mathbb{R}^n$, and A is $m \times n$, taking $\mathbb{R}^n$ to $\mathbb{R}^m$. Doing B and then A is a map from $\mathbb{R}^p$ to $\mathbb{R}^m$, and it is linear because both steps are. By Chapter 5 it has a matrix, whose columns are the images of the basis vectors.

> **Theorem 6.1** The product rule
>
> The matrix of "first B, then A" is the $m \times p$ matrix AB whose j-th column is A times the j-th column of B. Equivalently, its (i, j) entry is the dot product of row i of A with column j of B.

Proof

The j-th column of the composite matrix is where $\mathbf{e}_j$ ends up. The first step sends $\mathbf{e}_j$ to $B\mathbf{e}_j$, the j-th column of B.

The second step sends that vector to $A(B\mathbf{e}_j)$, which is what the theorem says.

Computing A times a column by the row picture of Section 5.2 gives, in position i, row i of A dotted with that column. ■

$$(AB)_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}, \qquad (AB)\mathbf{x} = A(B\mathbf{x}) \tag{6.1}$$

The second equation is the real content. Matrix multiplication is defined so that multiplying the matrices first and applying the result is the same as applying them one after the other. Everything else is bookkeeping.

Worked Example 6.1 A product by columns

Let A have columns $(1, 1)$ and $(0, 2)$, and B have columns $(3, 1)$ and $(1, 0)$. Compute AB.

The first column of AB is $A(3, 1) = 3(1, 1) + 1(0, 2) = (3, 5)$.

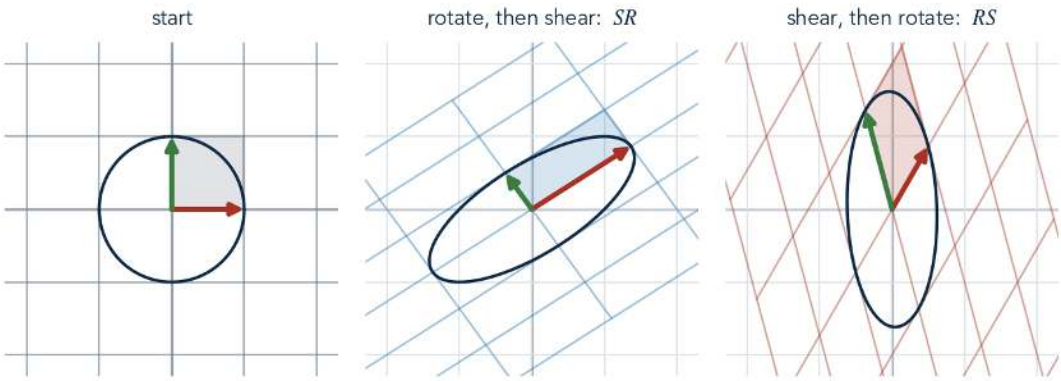
The second column is $A(1, 0) = 1(1, 1) + 0(0, 2) = (1, 1)$.

So AB has columns $(3, 5)$ and $(1, 1)$. By rows: the top-left entry is $(1, 0)\cdot(3, 1) = 3$, the bottom-left is $(1, 2)\cdot(3, 1) = 5$, agreeing with the column calculation.

$$\begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 5 & 1 \end{bmatrix} \tag{6.2}$$

6.2 Order matters

Composition of maps depends on order, and so does matrix multiplication. Rotating and then shearing is not shearing and then rotating.



the same two maps in the two orders give different results. Matrix products are read right to left, like function composition: $(SR)\mathbf{x} = S(R\mathbf{x})$.

FIGURE 6.1 The unit square and circle under a rotation followed by a shear (centre) and under the same shear followed by the same rotation (right). The results differ: $SR \neq RS$.

Products are read from right to left, like functions: in $(SR)\mathrm{x} = S(R\mathrm{x})$ the matrix nearest the vector acts first. This convention causes more confusion than any other in the subject, and the fix is to read every product aloud as "first the right-hand map, then the left".

TABLE 6.1 Which rules of ordinary arithmetic survive for matrices. Each entry in the second column can be checked with a picture of composing maps.

property	holds?	why
associativity $(AB)C = A(BC)$	always	doing three maps in order is doing three maps in order
distributivity $A(B + C) = AB + AC$	always	linearity of A
commutativity $AB = BA$	not in general	order of maps matters
cancellation $AB = AC \Rightarrow B = C$	not in general	A may destroy the difference
$AB = 0 \Rightarrow A = 0$ or $B = 0$	not in general	B may land where A sends to $\mathbf{0}$

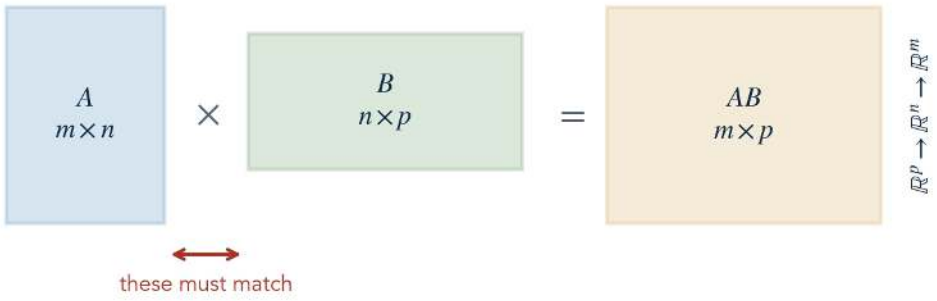
Worked Example 6.2 Two nonzero matrices with a zero product

Let A be the projection onto the x-axis, with columns $(1, 0)$ and $(0, 0)$, and B the projection onto the y-axis, with columns $(0, 0)$ and $(0, 1)$.

AB first projects onto the y-axis, then onto the x-axis. Anything on the y-axis projects to the origin, so $AB = 0$.

Neither A nor B is zero. The last row of Table 6.1 is not an exotic failure; it happens whenever the second map destroys what the first map produces.

6.3 Shapes must chain



B takes *p* numbers to *n*, and *A* takes *n* to *m*. The inner sizes must agree because the output of the first map is the input of the second: shape errors are composition errors.

FIGURE 6.2 An $m \times n$ matrix times an $n \times p$ matrix is an $m \times p$ matrix. The inner sizes agree because the output space of the right-hand map must be the input space of the left-hand one.

> **Every shape mismatch reported by a numerical library is a composition error: an attempt to feed the output of one map into a map expecting a different space. Reading shapes as spaces — "this matrix goes from 768 dimensions to 3,072" — makes such errors impossible to write rather than merely possible to debug.**

6.4 Special products worth knowing by sight

A handful of products occur so often that they should be recognised immediately.

$$\mathbf{u}^T\mathbf{v} = \text{anumber}, \qquad \mathbf{u}\mathbf{v}^T = \text{amatrixofrankone}, \qquad I\mathbf{x} = \mathbf{x}, \qquad (AB)^T = B^T A^T \tag{6.3}$$

The first is the dot product, a $1 \times n$ row times an $n \times 1$ column. The second is the **outer product**, an $m \times 1$ column times a $1 \times n$ row, which is an $m \times n$ matrix whose every column is a multiple of $\mathbf{u}$. Outer products are the atoms from which Chapter 15 builds every matrix, and every weight gradient in a neural network is a sum of them (Chapter 18). The identity matrix I, with ones on the diagonal, is the map that does nothing. The transpose of a product reverses the

order, for the same reason that taking off shoes and socks reverses putting them on.

Worked Example 6.3 An outer product

With $\mathbf{u} = (1, 2, 3)$ and $\mathbf{v} = (4, 5)$, compute $\mathbf{u}\mathbf{v}^T$.

Entry (i, j) is $u_i v_j$, so the rows are (4, 5), (8, 10) and (12, 15).

Every column is a multiple of $\mathbf{u}$ (the first is $4\mathbf{u}$, the second $5\mathbf{u}$), and every row is a multiple of $\mathbf{v}$. The 3×2 matrix is determined by five numbers, not six.

6.5 The inverse: undoing a map

Definition 6.1 Inverse

A square matrix A is invertible if there is a matrix A^{-1} with $A^{-1}A = AA^{-1} = I$. Geometrically, A^{-1} is the map that undoes A.

A map can be undone only if it loses no information: it must not send two different vectors to the same place. The projection of the gallery fails this badly. A rotation passes easily; its inverse is the rotation by the opposite angle. For 2×2 matrices there is a formula, which Chapter 8 will explain geometrically.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}, \qquad (AB)^{-1} = B^{-1}A^{-1} \tag{6.4}$$

The inverse of a product reverses the order, as it must: to undo "first B, then A", undo A first. In practice inverses are almost never computed; Chapter 9 explains why solving an equation is both faster and more accurate than inverting a matrix.

Worked Example 6.4 Undoing a product in the right order

Let A have rows $(2, 1)$ and $(1, 1)$, whose inverse has rows $(1, -1)$ and $(-1, 2)$, and let B be the quarter-turn with rows $(0, -1)$ and $(1, 0)$, whose inverse has rows $(0, 1)$ and $(-1, 0)$.

AB has rows $(1, -2)$ and $(1, -1)$. The rule $(AB)^{-1} = B^{-1}A^{-1}$ gives rows $(-1, 2)$ and $(-1, 1)$.

Check: the first row of AB against the columns of that inverse gives $(1)(-1) + (-2)(-1) = 1$ and $(1)(2) + (-2)(1) = 0$, and the second row gives 0 and 1. The product is I. The other order, $A^{-1}B^{-1}$, has rows $(1, 1)$ and $(-2, -1)$ and does not undo AB.

6.6 Depth, and why it needs non-linearity

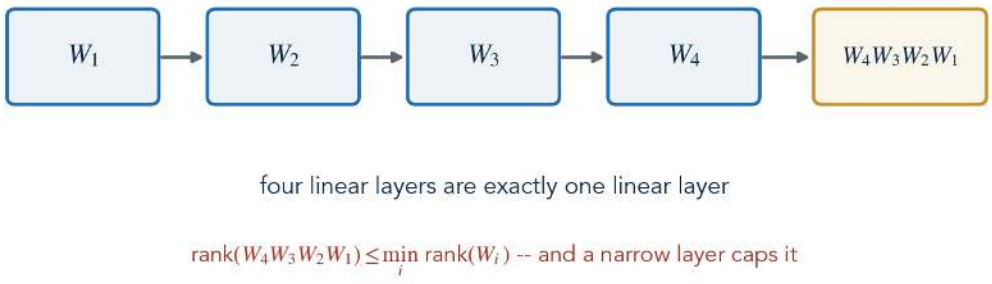


FIGURE 6.3 Four linear layers in a row are exactly one linear layer, and the rank of the whole product is capped by the narrowest layer. The non-linearity between layers is what makes depth worth having.

Chain several layers without the fold of Section 5.5 and associativity (Table 6.1) collapses them: $W_4W_3W_2W_1$ is a single matrix, and a hundred-layer linear network can represent nothing a one-layer

network cannot. The rectifier between layers is what breaks the collapse, which is why it is there.

Something survives the collapse, though, and it matters in practice. The rank of a product can be no larger than the rank of any factor (Chapter 7 proves it). A narrow layer in the middle of a network — a "bottleneck" — therefore limits how much information can pass through, whatever the layers on either side do. That limit is exploited on purpose in autoencoders and in the low-rank adapters of Chapter 16.

> **In machine learning.** Frameworks store a batch of inputs as the rows of a matrix X of size (batch) × (features) and compute a layer as XW^T, which is every input's $W\mathbf{x}$ at once, stacked as rows. Once this is recognised, the dozens of matrix products in a model's forward pass read as a single sentence: a batch of points is pushed through a chain of maps, with a fold after each.

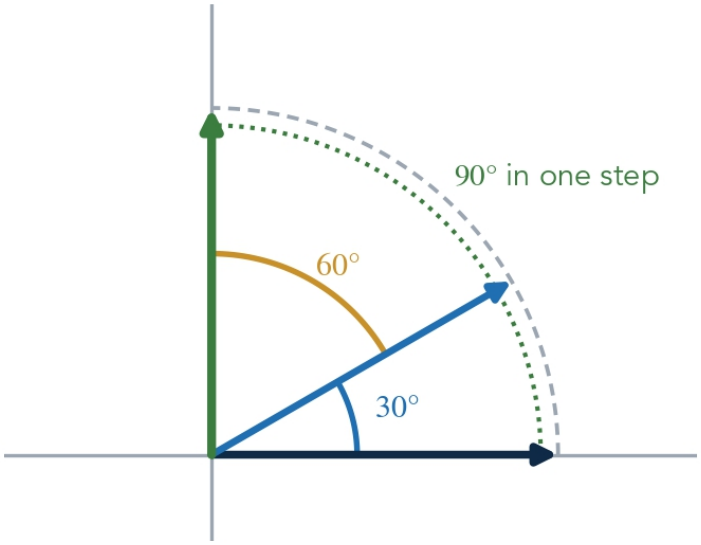
> **Deep networks are compositions, and the algebra of composition — associativity, non-commutativity, the reversal of order under transposes and inverses, the rank cap of a narrow factor — governs them completely between the non-linear folds. Understanding a network architecture is largely a matter of reading its sequence of shapes as a sequence of spaces.**

Worked Example 6.5 Composing two rotations

Rotation by 30° has rows $(0.866, -0.5)$ and $(0.5, 0.866)$; rotation by 60° has rows $(0.5, -0.866)$ and $(0.866, 0.5)$.

Their product has first row $(0.866 \times 0.5 - 0.5 \times 0.866,\ -0.866 \times 0.866 - 0.5 \times 0.5) = (0, -1)$ and second row $(0.5 \times 0.5 + 0.866 \times 0.866,\ -0.5 \times 0.866 + 0.866 \times 0.5) = (1, 0)$.

That is the quarter-turn, rotation by 90°: angles add. Multiplying in the other order gives the same result, so rotations of the plane are one of the exceptions to the third row of Table 6.1. Figure 6.4 shows the two routes arriving at the same place.



rotating by 30° and then by 60° is rotating by 90°. In the plane the angles simply add, so the two matrices commute and their product is the quarter-turn.

FIGURE 6.4 A vector rotated by 30° and then by 60° lands exactly where a single rotation by 90° sends it. Rotations of the plane compose by adding their angles, in either order.

Key ideas

- The product AB is the matrix of doing B first and then A; the row-column rule follows from that.
- Matrix multiplication is associative but not commutative, and products are read from right to left.
- Shapes must chain: an $m \times n$ matrix can follow an $n \times p$ matrix, and not otherwise.
- Transposes and inverses of products reverse the order of the factors.
- Stacked linear layers collapse into one matrix, and the narrowest layer caps the rank of the whole.

Exercises

1. Compute AB and BA for A with columns $(1, 0)$, $(1, 1)$ and B with columns $(2, 0)$, $(0, 1)$. Describe each as a composition of a shear and a stretch.
2. Find a 2×2 matrix $A \neq 0$ with $A^2 = 0$, and explain it geometrically.
3. Verify $(AB)^T = B^T A^T$ for the matrices of Example 6.1.
4. Use Equation 6.4 to invert the matrix with columns $(2, 1)$ and $(1, 1)$, and check that the product with the original is I.
5. A network has layer widths 1024, 64, 1024 with no non-linearities. What is the largest possible rank of the map from input to output?
6. For A with rows $(1, 2)$, $(0, 1)$ and B with rows $(3, 0)$, $(1, 1)$, compute $(AB)^T$ and $B^T A^T$ and compare.
7. Show that the quarter-turn rotation R satisfies $R^4 = I$, and interpret R^2.
8. Explain why a 3×2 matrix cannot have a two-sided inverse.

CHAPTER SEVEN

7 The Four Fundamental Subspaces

Every matrix sorts space into pieces. Some directions of the input survive the map, some are destroyed; some directions of the output are reached, some can never be. Those four pieces — two in the input space, two in the output — are the four fundamental subspaces, and their dimensions are tied together by a single number, the rank. Gilbert Strang has called this picture the heart of linear algebra, and this chapter shows why.

7.1 Column space: everything the map can reach

Definition 7.1 Column space

The column space of an $m \times n$ matrix A, written $\mathrm{col}(A)$, is the span of its columns. It is the set of all outputs $A\mathbf{x}$, and it is a subspace of $\mathbb{R}^m$.

That the column space is the set of outputs is just the column picture of Section 5.2: $A\mathbf{x}$ is a combination of the columns, and every combination arises from some $\mathbf{x}$. So the equation $A\mathbf{x} = \mathbf{b}$ has a solution exactly when $\mathbf{b}$ lies in the column space.

Definition 7.2 Rank

The rank of A is the dimension of its column space: the number of independent directions among its outputs.

7.2 Null space: everything the map destroys

Definition 7.3 Null space

The null space of A, written $\text{null}(A)$, is the set of all $\mathbf{x}$ with $A\mathbf{x} = \mathbf{0}$. It is a subspace of $\mathbb{R}^n$.

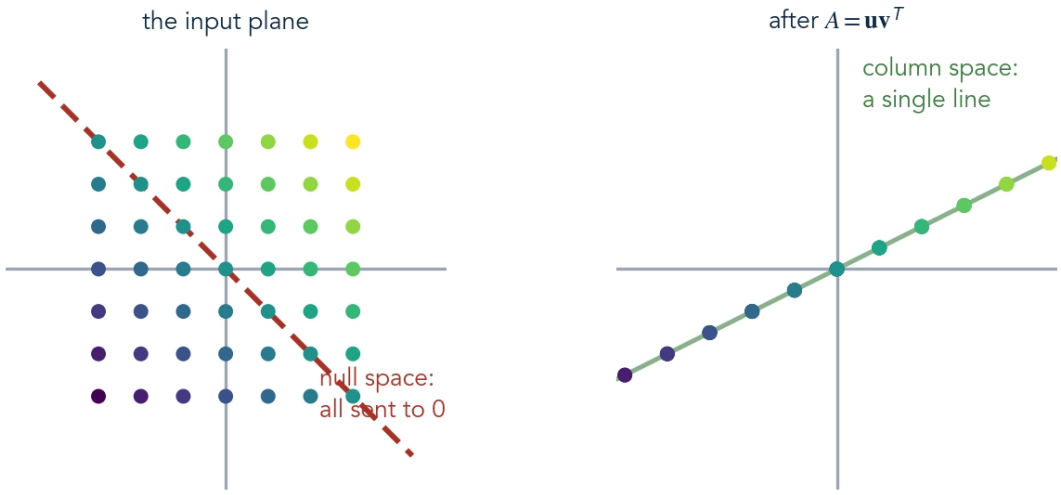


FIGURE 7.1 A rank-one matrix on the plane. Every point on a line parallel to the red null-space direction is sent to the same output, so the whole plane is flattened onto the green line, which is the column space.

The null space measures how much the map forgets. If it contains a nonzero vector $\mathbf{n}$, then $\mathbf{x}$ and $\mathbf{x} + \mathbf{n}$ produce the same output, and no observer of the output can tell them apart. A map with a null space larger than $\mathbf{0}$ cannot be inverted.

Worked Example 7.1 Finding both spaces

Let A be the 2×3 matrix with rows $(1, 2, 1)$ and $(2, 4, 3)$.

Column space: the columns are $(1, 2)$, $(2, 4)$ and $(1, 3)$. The second is twice the first, but the first and third are independent, so they span all of $\mathbb{R}^2$. The rank is 2.

Null space: we need $x_1 + 2x_2 + x_3 = 0$ and $2x_1 + 4x_2 + 3x_3 = 0$. Subtracting twice the first from the second leaves $x_3 = 0$, so $x_1 = -2x_2$. The null space is the line of multiples of $(-2, 1, 0)$, of dimension 1.

Notice in that example that the rank (2) plus the dimension of the null space (1) equals the number of columns (3). That is not a coincidence.

7.3 The rank–nullity theorem

Theorem 7.1 Rank–nullity

For an $m \times n$ matrix A, $\operatorname{rank}(A) + \operatorname{null}(A) = n$.

Proof

Choose a basis $\mathbf{n}_1, \ldots, \mathbf{n}_k$ of the null space and extend it with vectors $\mathbf{w}_1, \ldots, \mathbf{w}_r$ to a basis of all of $\mathbb{R}^n$, so $k + r = n$.

Every output is A applied to a combination of these, and the null-space vectors contribute nothing, so the outputs $A\mathbf{w}_1, \ldots, A\mathbf{w}_r$ span the column space.

They are also independent: if $\sum c_i A\mathbf{w}_i = \mathbf{0}$ then $\sum c_i \mathbf{w}_i$ lies in the null space, so it is a combination of the $\mathbf{n}_j$, and independence of the full basis forces every $c_i = 0$. So the rank is $r = n - k$. ■

$$\operatorname{rank}(A) + \dim \operatorname{null}(A) = n \qquad \text{(directions kept + directions destroyed = input dimension)} \tag{7.1}$$

The theorem is a conservation law for dimensions. Each of the n input directions is either carried faithfully into the output or destroyed; none is partly kept. A 1000×50 matrix has rank at most 50, and a 50×1000 matrix destroys at least 950 dimensions of its input, whatever its entries are.

7.4 The other two subspaces

Transposing a matrix swaps the roles of rows and columns and gives two more subspaces.

Definition 7.4 Row space and left null space

The row space of A is the span of its rows, $\operatorname{col}(A^T)$, a subspace of $\mathbb{R}^n$. The left null space is $\operatorname{null}(A^T)$, the set of $\mathbf{y}$ with $A^T\mathbf{y} = \mathbf{0}$, a subspace of $\mathbb{R}^m$.

Two facts turn four separate definitions into one picture.

Theorem 7.2 Row rank equals column rank

The row space and the column space of any matrix have the same dimension, the rank r.

Theorem 7.3 Orthogonal complements

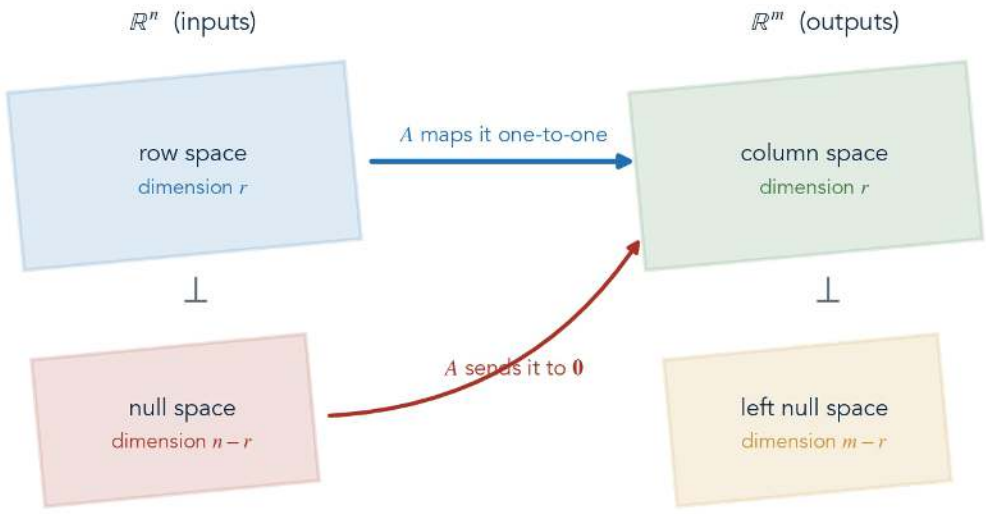
The row space and the null space are orthogonal complements in $\mathbb{R}^n$: every vector in one is perpendicular to every vector in the other, and together they account for all n dimensions. Likewise the column space and the left null space are orthogonal complements in $\mathbb{R}^m$.

Proof

For Theorem 7.3: $A\mathbf{x} = \mathbf{0}$ says that every row of A has dot product zero with $\mathbf{x}$, so $\mathbf{x}$ is perpendicular to every row and hence to every combination of rows. Conversely, a vector perpendicular to all rows satisfies $A\mathbf{x} = \mathbf{0}$.

For Theorem 7.2: the dimensions of these complements add to n, and by rank–nullity so do $\operatorname{rank}(A)$ and $\operatorname{null}(A)$. Hence the row space has dimension $\operatorname{rank}(A)$, the dimension of the column space.

Applying the same argument to A^T gives the statement in $\mathbb{R}^m$. ■



every $m \times n$ matrix of rank r splits its input space into two perpendicular pieces and its output space into two more. Dimensions add up on each side: $r+(n-r)=n$, $r+(m-r)=m$.

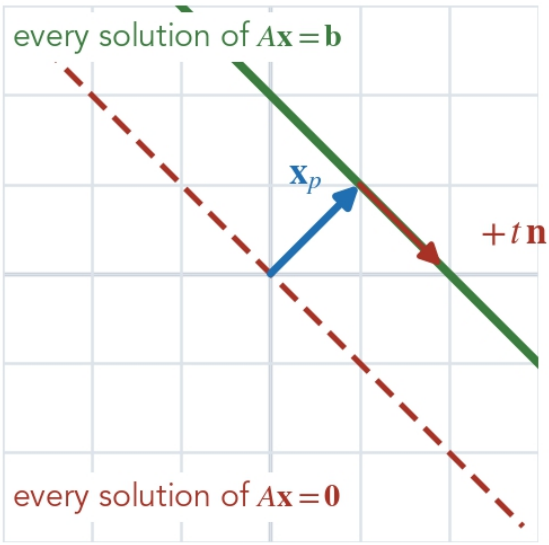
FIGURE 7.2 The four fundamental subspaces. In the input space the row space and null space meet at right angles; A carries the row space one-to-one onto the column space and sends the null space to zero. In the output space the column space and left null space meet at right angles.

TABLE 7.1 The four subspaces of an $m \times n$ matrix of rank r. The dimensions in each space add up to the dimension of that space.

subspace	lives in	dimension	what it means
column space $\text{col}(A)$	$\mathbb{R}^m$	r	outputs the map can reach
left null space $\text{null}(A^T)$	$\mathbb{R}^m$	$m-r$	output directions never reached
row space $\text{col}(A^T)$	$\mathbb{R}^n$	r	input directions the map uses
null space $\text{null}(A)$	$\mathbb{R}^n$	$n-r$	input directions destroyed

> **Every matrix, whatever its size or entries, does the same thing: it throws away a subspace of its input, maps the perpendicular remainder one-to-one onto a subspace of its output, and leaves the rest of the output unreachable. The four fundamental subspaces are not four topics; they are one picture of what a matrix does, drawn in two spaces.**

7.5 Solving equations with the picture



the solution set is the null space, shifted to pass through one particular solution. Its size is the size of the null space; whether it is empty depends on $\mathbf{b}$.

FIGURE 7.3 The solutions of $A\mathbf{x} = \mathbf{b}$, when there are any, form a copy of the null space shifted to pass through one particular solution $\mathbf{x}_p$.

The picture answers every existence and uniqueness question about linear equations at once. A solution exists exactly when $\mathbf{b}$ is in the column space. If one exists, all solutions are $\mathbf{x}_p + \mathbf{n}$ for $\mathbf{n}$ in the null space, so the solution is unique exactly when the null space is $\mathbf{0}$. And among all solutions, exactly one lies in the row space — the one with no null-space component — which is also the shortest. That shortest solution is what the pseudoinverse of Chapter 15 returns.

> **In machine learning.** The null space of a model's weight matrix is the set of input changes the layer is blind to. When there are more parameters than data points, the equations "fit every training example exactly" have a whole shifted null space of solutions, and which one training finds is decided by the method, not by the data. The shortest solution in the row space is the one gradient descent started from zero converges to for linear least

squares — an implicit preference with real consequences, taken up again in Chapter 11.

Worked Example 7.2 All four subspaces of a small matrix

Let A have rows $(1, 2)$ and $(2, 4)$, a matrix of rank one.

Column space: both columns are multiples of $(1, 2)$, so it is the line through $(1, 2)$. Left null space: $A^T\mathbf{y} = \mathbf{0}$ means $y_1 + 2y_2 = 0$, the line through $(-2, 1)$.

Row space: the line through $(1, 2)$. Null space: $x_1 + 2x_2 = 0$, the line through $(-2, 1)$.

In each space the two lines are perpendicular, since $(1, 2)\cdot(-2, 1) = 0$, and their dimensions add to 2: Table 7.1 in miniature.

7.6 Rank of a product

Theorem 7.4 Rank of a product

For matrices whose product is defined, $\operatorname{rank}(AB) \le \min(\operatorname{rank}(A), \operatorname{rank}(B))$.

Proof

Every output of AB is an output of A, so $\operatorname{col}(AB) \subseteq \operatorname{col}(A)$ and the rank of AB is at most that of A.

The same argument on transposes, using Theorem 7.2, gives $\operatorname{rank}(AB) = \operatorname{rank}(B^TA^T) \le \operatorname{rank}(B^T) = \operatorname{rank}(B)$. ■

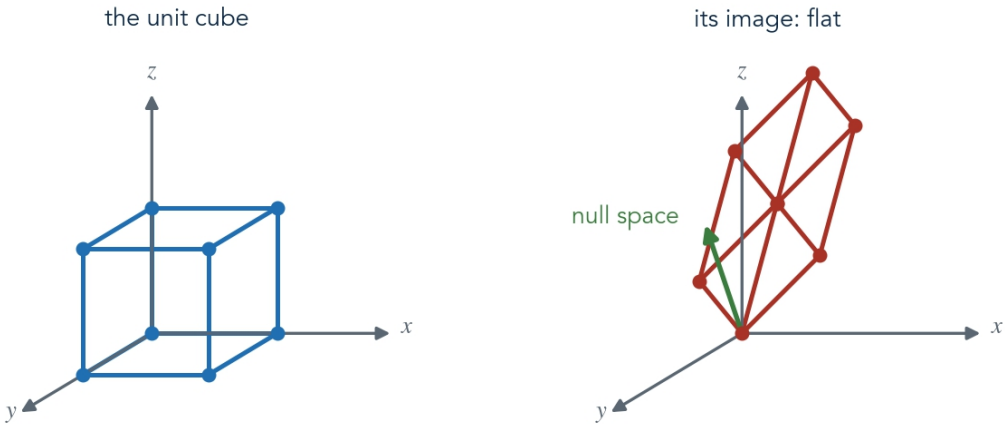
This is the fact behind the bottleneck of Section 6.6: a chain of maps can carry no more independent directions than its narrowest link.

Worked Example 7.3 A map of space with rank two

Let A have rows $(1, 0, 1)$, $(0, 1, 1)$ and $(1, 1, 2)$.

The third row is the sum of the first two, so the rank is 2. The null space solves $x + z = 0$ and $y + z = 0$: it is the line through $(-1, -1, 1)$, of dimension 1, and $2 + 1 = 3$ as rank–nullity requires.

The third column is also the sum of the first two, so the column space is the plane spanned by $(1, 0, 1)$ and $(0, 1, 1)$. Every point of space lands in that plane, and Figure 7.4 shows the unit cube flattened into it.



every corner of the cube lands in one plane, so the image has no thickness. The green direction is sent to zero, and rank two plus a null space of dimension one accounts for all three.

FIGURE 7.4 The unit cube (left) and its image under a matrix of rank two (right). The image lies flat in a plane: one whole direction of space, the null space, has been destroyed, and the cube's volume has become zero.

Key ideas

- The column space is everything a matrix can output; $A\mathbf{x} = \mathbf{b}$ is solvable exactly when $\mathbf{b}$ lies in it.
- The null space is everything the matrix sends to zero, and it measures what the map forgets.
- Rank plus the dimension of the null space equals the number of columns.
- The row space and null space are perpendicular complements, and so are the column space and left null space.
- All solutions of $A\mathbf{x} = \mathbf{b}$ are one particular solution plus the null space, and the rank of a product is at most the rank of each factor.

Exercises

1. Find the rank, a basis of the column space and a basis of the null space of the matrix with rows $(1, 2, 3)$, $(2, 4, 6)$.
2. A 7×10 matrix has rank 6. What are the dimensions of its four fundamental subspaces?
3. Show that the null space of A^TA equals the null space of A. (Hint: if $A^TA\mathbf{x} = \mathbf{0}$, consider $\mathbf{x}^TA^TA\mathbf{x}$.)
4. The system $A\mathbf{x} = \mathbf{b}$ has solution $(1, 2, 0)$ and the null space of A is spanned by $(1, -1, 1)$. Write down all solutions and find the one perpendicular to $(1, -1, 1)$.
5. Explain why a 100×3 matrix cannot have a null space of dimension zero and rank 4 at the same time, and what its maximum rank is.
6. What is the rank of the outer product $\mathbf{u}\mathbf{v}^T$ of two nonzero vectors?
7. Write down a 3×3 matrix whose null space is the line through $(1, 1, 1)$.
8. A 5×5 matrix has rank 5. What is its null space, and what does that say about $A\mathbf{x} = \mathbf{b}$?

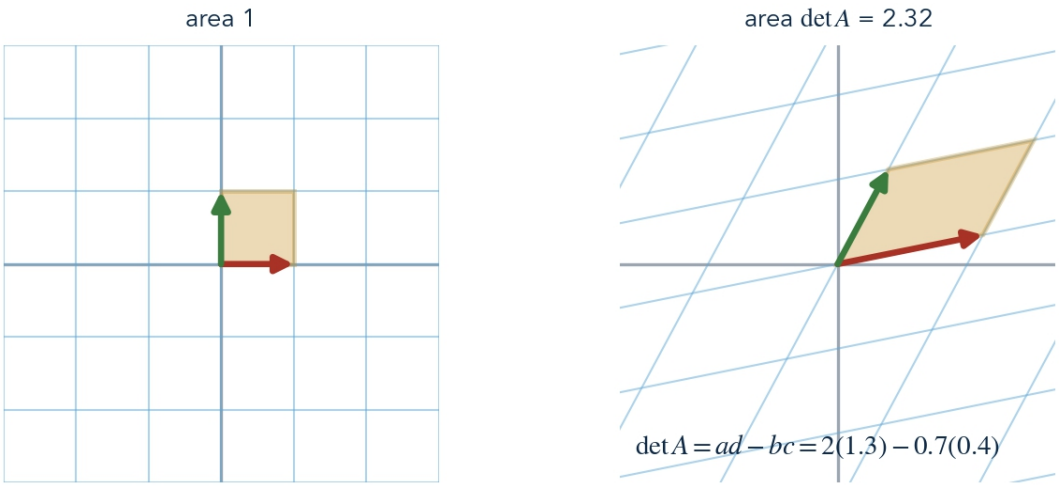
CHAPTER EIGHT

8 Determinants and Volume

The determinant is usually met as a formula — $ad - bc$ for a 2×2 matrix, and an alarming expansion for anything larger. The formula hides a single clean idea: a linear map scales every volume by the same factor, and the determinant is that factor, with a sign that says whether the map turns space inside out. Everything else about determinants follows from that sentence.

8.1 The area factor

A linear map carries the unit square to a parallelogram whose sides are the two columns of the matrix. Because the map is linear, it treats every small square of the grid the same way, so every region's area is multiplied by the same number.



every region is scaled by the same factor, so the unit square is enough to measure it. The determinant is that factor, and $ad - bc$ is the area of the parallelogram the columns span.

FIGURE 8.1 The unit square (left) and its image (right), a parallelogram spanned by the columns of A. Its area, 2.32, is the factor by which A scales the area of every region of the plane.

Theorem 8.1 The 2×2 determinant is an area

The parallelogram spanned by the columns (a, c) and (b, d) of a 2×2 matrix has area $|ad - bc|$.

Proof

Enclose the parallelogram in the rectangle of width $a + b$ and height $c + d$ (for positive entries; other cases are the same with signs).

The rectangle minus the parallelogram consists of two triangles of area $ac/2$, two of area $bd/2$ and two rectangles of area bc.

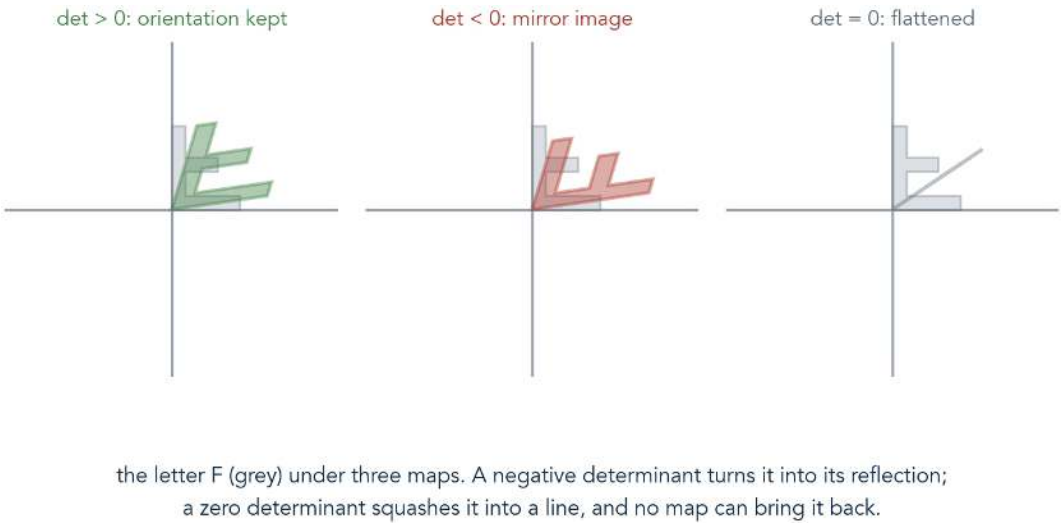
So the parallelogram's area is $(a + b)(c + d) - ac - bd - 2bc = ad - bc$. ■

Definition 8.1 Determinant

The determinant of a square matrix A, written $\det A$, is the signed factor by which A scales volume: its absolute value is the volume of the image of the unit cube, and its sign is positive if A preserves orientation and negative if it reverses it.

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc, \qquad \operatorname{vol}(A\,\Omega) = |\det A|\,\operatorname{vol}(\Omega) \tag{8.1}$$

8.2 The sign, and the zero



the letter F (grey) under three maps. A negative determinant turns it into its reflection;
a zero determinant squashes it into a line, and no map can bring it back.

FIGURE 8.2 A letter F under three maps. With a positive determinant it is distorted but still an F; with a negative determinant it becomes a mirror image; with a zero determinant it is flattened into a line segment.

Orientation is the subtle half of the definition. In the plane, a map with positive determinant keeps "anticlockwise" anticlockwise; a map with negative determinant swaps it, as a mirror does. Reflections have determinant -1, rotations $+1$.

The zero case is the important one. A determinant of zero means the unit square has been flattened to something of zero area — a line or a point — so the map has collapsed at least one dimension. That is exactly the condition for a nonzero null space, and so for non-invertibility.

> **Theorem 8.2** Invertibility
>
> For a square matrix A the following are equivalent: $\det A \neq 0$; A is invertible; the null space of A is $\mathbf{0}$; the columns of A are independent; the rank of A is n.

Five conditions that look unrelated turn out to be five descriptions of one fact: the map does not flatten space.

8.3 The product rule and its consequences

Theorem 8.3 Determinant of a product

$\det(AB) = \det(A)\det(B)$.

Proof

Applying B scales every volume by $\det B$, and then applying A scales the result by $\det A$.

Doing both scales volume by the product, and the orientation signs multiply in the same way.

So the composite map AB has determinant $\det(A)\det(B)$. ■

A proof that takes one sentence when the determinant is understood as a volume factor takes a page when it is understood as a formula. The consequences come just as quickly.

TABLE 8.1 Properties of the determinant, each read directly from "signed volume factor".

property	reason
$\det I = 1$	doing nothing scales nothing
$\det(A^{-1}) = 1/\det A$	undoing a scaling divides by it
$\det(A^T) = \det A$	rows and columns span the same volume
swapping two columns flips the sign	it reverses orientation
a repeated column gives 0	the parallelogram is flat
adding a multiple of one column to another changes nothing	that is a shear, which keeps area
$\det(cA) = c^n \det A$	every one of n directions scaled by c

Worked Example 8.1 A 3×3 determinant by shears

Find $\det A$ for the matrix with rows $(2, 1, 1)$, $(4, 3, 3)$ and $(2, 1, 3)$.

Subtract twice row 1 from row 2 and row 1 from row 3; by the shear row of Table 8.1 (applied to rows via $\det A^T = \det A$) the determinant is unchanged. The rows become $(2, 1, 1)$, $(0, 1, 1)$ and $(0, 0, 2)$.

The matrix is now upper triangular. A triangular matrix is a sequence of shears that turns a stretched box into a slanted one, so its volume is the product of the diagonal entries: $2 \times 1 \times 2 = 4$.

So $\det A = 4$. The same row operations are the first steps of Gaussian elimination, which is how determinants of large matrices are actually computed.

8.4 The general formula, and why no one uses it

For completeness: the determinant of an $n \times n$ matrix can be written as a sum over all $n!$ ways of choosing one entry from each row and column, each product signed according to how the choice permutes the columns.

$$\det A = \sum_{\pi} \operatorname{sgn}(\pi)\, a_{1\pi(1)}\, a_{2\pi(2)} \cdots a_{n\pi(n)} \tag{8.2}$$

For $n = 20$ that sum has more than 2×10^{18} terms. Elimination, as in Example 8.1, gets the same number in about $n^3/3$ operations. The formula is a definition that proves the determinant exists; the volume is the idea; elimination is the method.

Numerically, the determinant is a poor test of near-singularity. Scaling a 100×100 identity matrix by 0.1 gives a determinant of 10^{-100}, although the map is perfectly well behaved. Whether a matrix is close to flattening space is measured by the ratio of its largest and smallest stretches, the

condition number of Chapter 12, not by the size of the determinant.

Worked Example 8.2 The product rule with numbers

Let A have rows $(2, 1)$ and $(0, 3)$, so $\det A = 6$, and let B have rows $(1, 0)$ and $(4, 2)$, so $\det B = 2$.

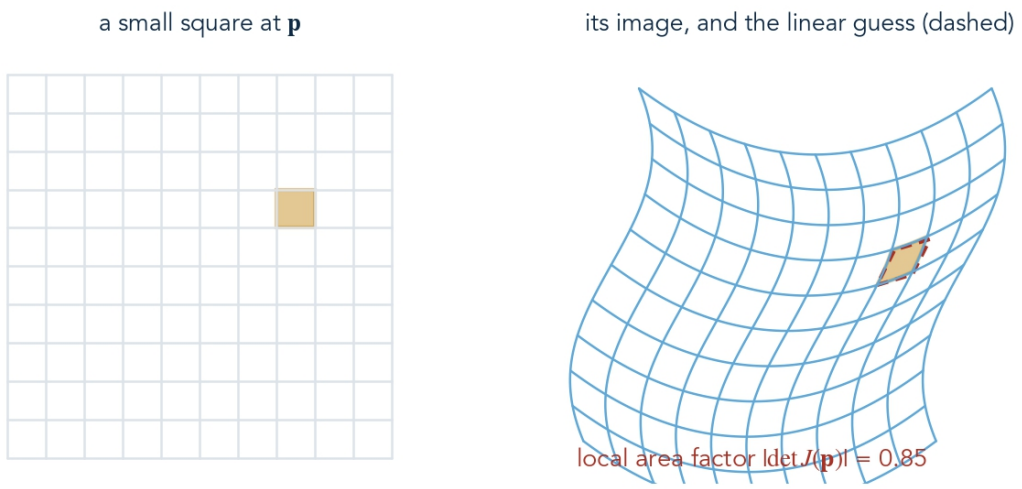
AB has rows $(6, 2)$ and $(12, 6)$, whose determinant is $36 - 24 = 12$.

That is 6×2: B doubles every area and A multiplies the result by six.

8.5 The Jacobian: a local determinant

Curved maps are not linear, but up close they are. At each point a smooth map F from $\mathbb{R}^n$ to $\mathbb{R}^n$ is approximated by a matrix, the Jacobian, whose entries are the partial derivatives of the output coordinates with respect to the input coordinates.

$$J_F(\mathbf{p}) = \begin{bmatrix} \partial F_1/\partial x_1 & \cdots & \partial F_1/\partial x_n \\ \vdots & & \vdots \\ \partial F_n/\partial x_1 & \cdots & \partial F_n/\partial x_n \end{bmatrix}, \qquad F(\mathbf{p}+\mathbf{h}) \approx F(\mathbf{p}) + J_F(\mathbf{p})\,\mathbf{h} \tag{8.3}$$



zoom in far enough and a curved map looks like a matrix: the Jacobian. Its determinant is how much area is stretched there, which is why it appears in every change of variables.

FIGURE 8.3 A small square near $\mathbf{p}$ (left) and its image under a curved map (right). The dashed parallelogram is the linear prediction from the Jacobian, and it nearly coincides with the true image. The area factor there is $|\det J_F(\mathbf{p})|$.

The Jacobian determinant is the local volume factor, and it appears wherever volume must be tracked through a change of variables. In calculus it is the factor in $\int f(F(\mathbf{u}))\,|\det J_F|\,d\mathbf{u}$. In probability it is how a density changes when its variable is transformed: if $\mathbf{y} = F(\mathbf{x})$ is invertible, then $p_Y(\mathbf{y}) = p_X(\mathbf{x})/|\det J_F(\mathbf{x})|$, because probability mass is conserved while volume is stretched.

In machine learning. Normalising flows are models that transform a simple distribution into a complicated one through a sequence of invertible maps, and they must compute the change in density at every step — a Jacobian determinant. Computing a general determinant costs n^3 operations, so these models are designed with triangular Jacobians, whose determinant is just the product of the diagonal, exactly as in Example 8.1. An architectural choice in a modern model is dictated by the cost of a determinant.

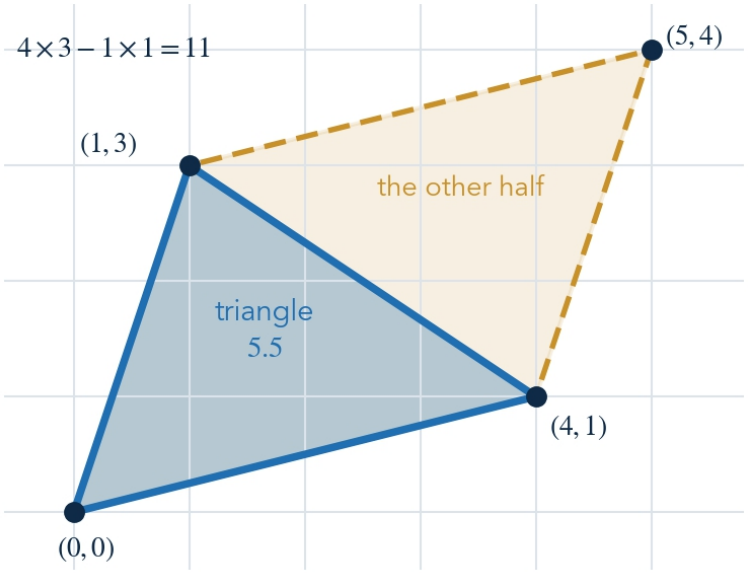
The determinant is the answer to one question — by how much, and with what orientation, does this map scale volume — and every one of its properties is that question asked about a composition, an inverse, a transpose or a shear. Treated as a formula it is a burden; treated as a volume it is almost free.

Worked Example 8.3 The area of a triangle

Find the area of the triangle with corners $(0, 0)$, $(4, 1)$ and $(1, 3)$.

The two edges from the origin are $(4, 1)$ and $(1, 3)$. The parallelogram they span has area $4 \times 3 - 1 \times 1 = 11$.

The triangle is half of that parallelogram, so its area is 5.5. The same idea in three dimensions gives the volume of a tetrahedron as one sixth of a 3×3 determinant. Figure 8.4 shows why the factor is one half.



the parallelogram on the edges $(4,1)$ and $(1,3)$ has area 11, its determinant; the diagonal cuts it into two congruent triangles, so the triangle has area 5.5.

FIGURE 8.4 The triangle of Example 8.3 inside the parallelogram spanned by two of its edges. The determinant measures the parallelogram; half of it measures the triangle.

Key ideas

- The determinant is the signed factor by which a map scales volume.
- A negative determinant reverses orientation; a zero determinant flattens space and means the matrix has no inverse.
- $\det(AB) = \det A \det B$, and the other properties follow from the volume picture.
- Determinants are computed by elimination, as the product of the pivots, and never from the formula with $n!$ terms.
- The Jacobian determinant is the local volume factor of a curved map, and it governs every change of variables.

Exercises

1. Find the area of the parallelogram spanned by $(3, 1)$ and $(1, 2)$, and decide whether the map with these columns preserves orientation.
2. Use row operations to find the determinant of the matrix with rows $(1, 2, 0)$, $(3, 1, 1)$, $(0, 1, 2)$.
3. Show that an orthogonal matrix (one with $Q^{T}Q = I$) has determinant ± 1, and interpret the result.
4. Explain with a picture why $\det(cA) = c^{2}\det A$ for 2×2 matrices.
5. The map $F(u, v) = (u^{2} - v^{2}, 2uv)$ doubles angles at the origin. Compute its Jacobian determinant at a general point and at the origin, and explain what happens there.
6. Find the determinant of the matrix with rows $(1, 2, 3)$, $(4, 5, 6)$, $(7, 8, 9)$, and explain the answer geometrically.
7. What is the determinant of a triangular 3×3 matrix with diagonal entries $2, -1$ and 4?
8. If a 4×4 matrix A has determinant 3, what is the determinant of $2A$?

PART

III

SOLVING AND FITTING

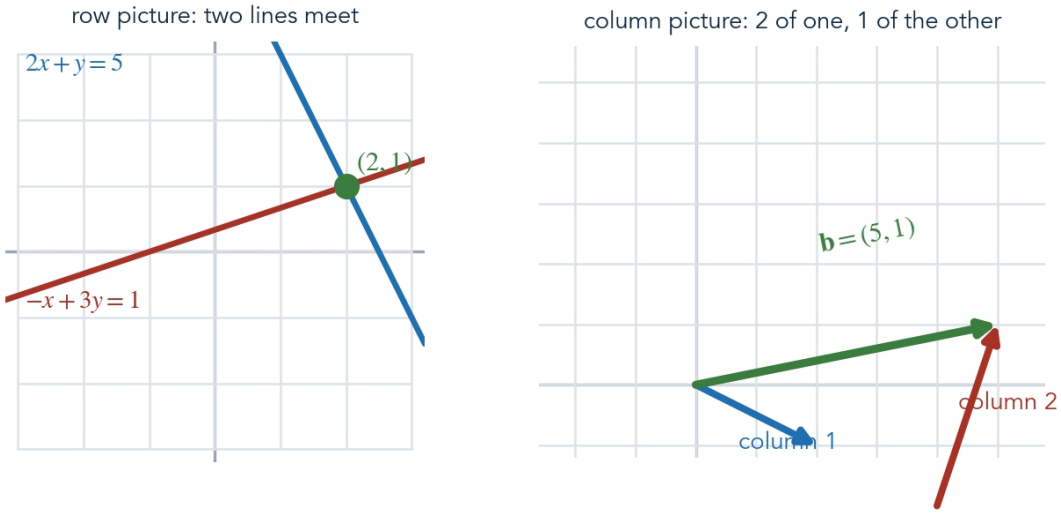
from equations to data

CHAPTER NINE

9 Solving Ax = b

Part III turns from what a matrix is to what can be done with one, and the first thing anyone wants to do is solve an equation. Given a map A and an output $\mathbf{b}$, find the input $\mathbf{x}$ that produces it. Chapter 7 already answered whether a solution exists and how many there are. This chapter answers how to find one, and why the method taught in every first course — elimination — is also the method every serious numerical library uses.

9.1 Two pictures of one system



the same two equations read two ways. By rows they are lines that cross; by columns they ask how much of each column vector reaches **b**. The column picture is the one that scales.

FIGURE 9.1 The system $2x+y=5$, $-x+3y=1$ seen by rows (left) and by columns (right). By rows it is two lines crossing at $(2,1)$. By columns it asks for the combination of $(2,-1)$ and $(1,3)$ that reaches $(5,1)$: two of the first and one of the second.

$$\begin{bmatrix} 2 & 1 \\ -1 & 3 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}=\begin{bmatrix} 5 \\ 1 \end{bmatrix} \iff x\begin{bmatrix} 2 \\ -1 \end{bmatrix}+y\begin{bmatrix} 1 \\ 3 \end{bmatrix}=\begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

(9.1)

The row picture is the one drawn in school, and it stops being drawable at three unknowns. The column picture is the one that scales: in any dimension, solving $A\mathbf{x} = \mathbf{b}$ means finding the coefficients that combine the columns of A into $\mathbf{b}$, which is possible exactly when $\mathbf{b}$ lies in the column space.

9.2 Elimination

Elimination replaces a system by an easier one with the same solutions, using three moves that cannot change the solution set: subtract a multiple of one equation from another, swap two equations, and scale an equation by a nonzero number. The goal is a triangular system, which can be solved from the bottom up.

> **Worked Example 9.1** Elimination on a 3×3 system
>
> Solve $2x + y + z = 5$, $4x - 6y = -2$, $-2x + 7y + 2z = 9$.
>
> Step 1: subtract $2 \times$ row 1 from row 2, and add row 1 to row 3. The multipliers are 2 and -1. The system becomes $2x + y + z = 5$, $-8y - 2z = -12$, $8y + 3z = 14$.
>
> Step 2: add row 2 to row 3 (multiplier -1). The third equation becomes $z = 2$.
>
> Back-substitute: $-8y - 4 = -12$ gives $y = 1$; then $2x + 1 + 2 = 5$ gives $x = 1$. The solution is $(1, 1, 2)$.
>
> Check in the original third equation: $-2 + 7 + 4 = 9$.

The numbers left on the diagonal after elimination — here 2, -8 and 1 — are the **pivots**. A system has a unique solution exactly when elimination produces n nonzero pivots, which by Chapter 8 is exactly when the determinant, the product of the pivots up to sign, is nonzero.

9.3 Elimination is a factorisation

Each elimination step is itself multiplication by a simple matrix, and recording the steps turns a procedure into a structure.

Theorem 9.1 *LU* factorisation

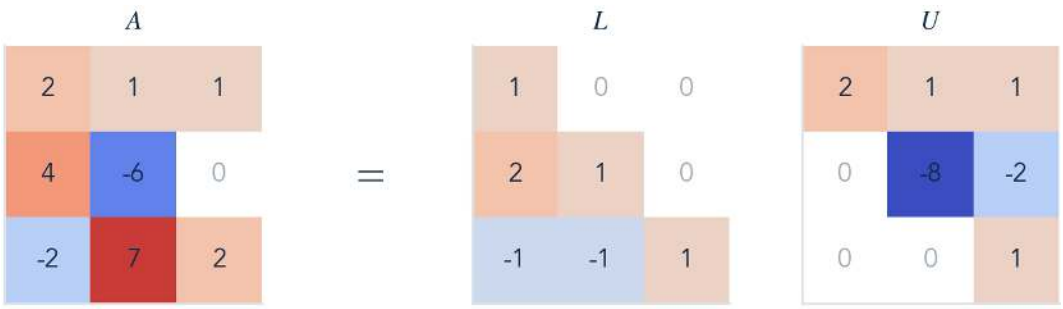
If elimination on a square matrix A needs no row exchanges, then $A = LU$, where U is the upper-triangular matrix elimination produces and L is lower triangular with ones on its diagonal and the multipliers below it, each in the position it was used.

Proof

Subtracting ℓ times row j from row i is multiplication on the left by E_{ij}, the identity with $-\ell$ in position (i, j). Elimination is therefore $E_k \cdots E_2 E_1 A = U$.

Each E_{ij} is undone by the same matrix with $+\ell$ in place of $-\ell$, so $A = E_1^{-1} E_2^{-1} \cdots E_k^{-1} U$.

When the steps are taken column by column, the product of these inverses places each multiplier in its own position without interference, which is the matrix L described. ■



elimination records its multipliers below the diagonal of L and leaves the upper-triangular U behind. Solving $A\mathbf{x}=\mathbf{b}$ becomes two triangular solves, each read off one row at a time.

FIGURE 9.2 The matrix of Example 9.1 factored as LU. The multipliers 2, -1 and -1 sit below the diagonal of L; the pivots 2, -8 and 1 sit on the diagonal of U.

$$\begin{bmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.2}$$

With $A = LU$ in hand, $A\mathbf{x} = \mathbf{b}$ becomes two triangular systems. First solve $L\mathbf{c} = \mathbf{b}$ from the top down, then $U\mathbf{x} = \mathbf{c}$ from the bottom up. Each is a single pass, costing about n^2 operations.

9.4 Pivoting: swapping rows for safety

Elimination divides by the pivot, and a pivot of zero stops it. A pivot that is merely small is almost as bad: dividing by 10^{-12} multiplies every rounding error by 10^{12}. The cure is to swap rows so that the largest available entry in the column becomes the pivot. The factorisation then reads $PA = LU$, where P is a **permutation matrix** that records the swaps.

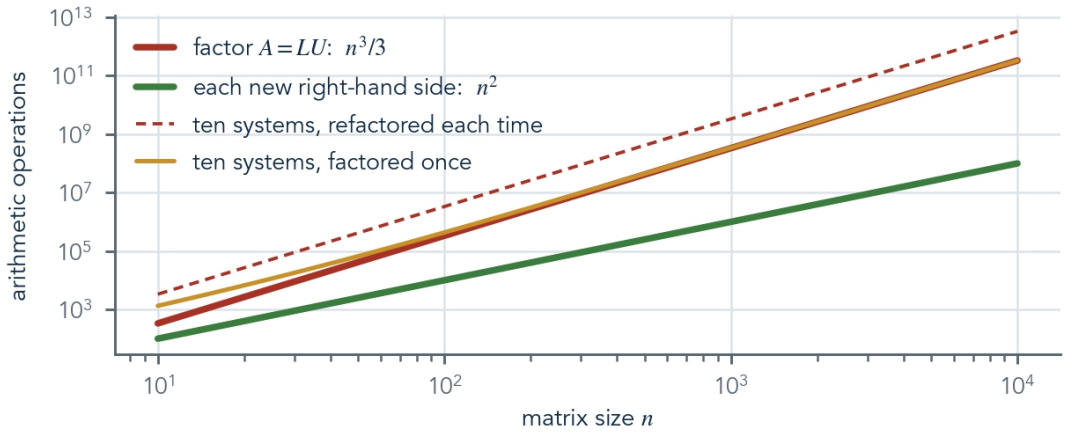
Worked Example 9.2 Why a tiny pivot is dangerous

Consider $10^{-4}x + y = 1$ and $x + y = 2$, whose solution is very nearly $x = 1$, $y = 1$.

Without swapping, the multiplier is 10^4 and the second equation becomes $(1 - 10^4)y = 2 - 10^4$. Working to three significant figures, both sides round to -10^4, giving $y = 1$ and then $10^{-4}x = 0$, so $x = 0$ — badly wrong.

Swapping first makes the multiplier 10^{-4}: the second row becomes $(1 - 10^{-4})y = 1 - 2 \times 10^{-4}$, giving $y \approx 1$ and then $x = 2 - y \approx 1$. The same arithmetic with the rows in a better order gives the right answer.

9.5 Cost, and the rule never to invert



the factorisation is the expensive part and the solves are cheap. Factor once, solve many times -- and never form A^{-1}, which costs more and is less accurate than either.

FIGURE 9.3 Arithmetic operations to factor an $n \times n$ matrix (about $n^3/3$) and to solve with the factors for each new right-hand side (about n^2). Factoring once and reusing the factors is far cheaper than starting again.

TABLE 9.1 Three ways to solve $A\mathbf{x} = \mathbf{b}$ for a dense $n \times n$ matrix, and what each costs. The last row is the one to remember.

method	operations	accuracy
Cramer's rule with determinants	roughly $n \cdot n!$ as usually taught	poor
compute A^{-1}, then $A^{-1}\mathbf{b}$	about n^3	worse than elimination
factor $PA = LU$ once, then two triangular solves	$n^3/3$, then n^2 per $\mathbf{b}$	the best a direct method gets

Computing the inverse costs about three times as much as factoring, and the result is less accurate, because forming A^{-1} and then multiplying compounds two sets of rounding errors. There is almost never a reason to form it. When a formula contains $A^{-1}\mathbf{b}$, it is an instruction to solve a system, not to invert a matrix.

> **"Solve, don't invert" is the most practically valuable rule in numerical linear algebra. An expression like $A^{-1}\mathbf{b}$ or $(X^TX)^{-1}X^T\mathbf{y}$ describes a result, not a method; the method is always a factorisation followed by triangular solves.**

Worked Example 9.3 Solving with the factors

Use the factorisation of Equation 9.2 to solve $A\mathbf{x} = (5, -2, 9)$.

First $L\mathbf{c} = \mathbf{b}$, from the top: $c_1 = 5$; $2c_1 + c_2 = -2$ gives $c_2 = -12$; $-c_1 - c_2 + c_3 = 9$ gives $c_3 = 2$.

Then $U\mathbf{x} = \mathbf{c}$, from the bottom: $z = 2$; $-8y - 2z = -12$ gives $y = 1$; $2x + y + z = 5$ gives $x = 1$.

The solution (1, 1, 2) agrees with Example 9.1, and a second right-hand side would reuse L and U unchanged.

9.6 Systems that are not square

Real problems rarely have exactly as many equations as unknowns, and Chapter 7's picture says what to expect.

TABLE 9.2 The three shapes of linear system and what the four subspaces predict for each.

shape	typical case	solutions	what to do
square, full rank	n equations, n unknowns	exactly one	$PA = LU$
tall, $m > n$	more measurements than unknowns	usually none	least squares, Chapter 11
wide, $m < n$	more unknowns than constraints	infinitely many	choose one, often the shortest

A tall system — far more data points than parameters — is the ordinary situation in statistics and in classical machine learning, and it almost never has an exact solution because measurements contain noise. A wide system — more parameters than data points — is the ordinary situation in large modern models, and it has a whole shifted null space of exact solutions. Which one a training method selects is a question about the method, and Chapter 11 answers it for gradient descent.

> **In machine learning.** Linear systems appear inside many methods that do not look linear. Each step of Newton's method solves a system with the Hessian; Gaussian-process regression solves one with a kernel matrix; second-order optimisers precondition with the solution of one. In all of them the matrix is symmetric positive definite, for which a specialised factorisation — Cholesky, $A = LL^T$ — is twice as fast as LU and needs no pivoting. Chapter 14 explains why such matrices are so well behaved.

Worked Example 9.4 When elimination meets a contradiction

Solve $x + 2y = 3$, $2x + 4y = 7$.

Subtracting twice the first equation from the second leaves $0 = 1$. The second pivot is zero and the right-hand side is not, so the system has no solution: $(3, 7)$ is not in the column space, which is the line through $(1, 2)$.

If the right-hand side were $(3, 6)$ instead, elimination would leave $0 = 0$ and every point of the line $x + 2y = 3$ would be a solution. A zero pivot always means one of these two outcomes, and the right-hand side decides which. Figure 9.4 returns to the other danger, a pivot that is small rather than zero.

the same system, the same precision. Dividing by a pivot of 10^{-4} destroys the answer for x; choosing the larger pivot keeps every digit that three significant figures can hold.

FIGURE 9.4 The two answers of Example 9.2 against the true solution. Working to three significant figures, elimination with the tiny pivot returns $x = 0$; swapping the rows first returns the right answer with the same arithmetic.

Key ideas

- A system can be read by rows, as intersecting lines and planes, or by columns, as a combination that reaches $\mathbf{b}$.
- Elimination produces a triangular system and its pivots; n nonzero pivots mean a unique solution.
- Elimination is the factorisation $A = LU$, and pivoting, $PA = LU$, protects against small pivots.
- Factor once and solve many times; never form the inverse in order to solve a system.
- Tall systems usually have no exact solution and wide ones have infinitely many, which leads to least squares and minimum-norm solutions.

Exercises

1. Solve $x + 2y = 5$, $3x + 4y = 11$ by elimination, and draw both the row and the column pictures.
2. Find the LU factorisation of the matrix with rows $(1, 2)$ and $(3, 8)$, and use it to solve $A\mathbf{x} = (5, 19)$.
3. Show that the product of two lower-triangular matrices is lower triangular.
4. For which value of k does elimination on rows $(1, 2, 3)$, $(2, k, 6)$, $(1, 0, 1)$ fail, and what does the failure mean geometrically?
5. A method needs to solve 1,000 systems with the same 500×500 matrix. Estimate the saving from factoring once, using the counts in Figure 9.3.
6. Solve $2x + y = 3$, $x + 3y = 4$ by elimination.
7. Find the pivots of the matrix with rows $(1, 1)$ and $(1, 1.0001)$, and say what they indicate.
8. Explain why solving a triangular system costs about n^2 operations rather than n^3.

CHAPTER TEN

10 Orthogonality and Projection

Some bases are better than others. In a basis of mutually perpendicular unit vectors, coordinates are dot products, lengths are Pythagoras, and the inverse of the coordinate change is a transpose. This chapter builds such bases on demand, shows what a matrix with perpendicular columns does to space, and turns projection — the shadow of Chapter 2 — into a matrix that projects onto any subspace.

10.1 Orthonormal bases

Definition 10.1 Orthonormal set

Vectors $\mathbf{q}_1, \ldots, \mathbf{q}_k$ are orthonormal if each has length one and each pair is perpendicular: $\mathbf{q}_i \cdot \mathbf{q}_j = 1$ if $i = j$ and 0 otherwise.

The payoff is immediate. If $\mathbf{q}_1, \ldots, \mathbf{q}_n$ is an orthonormal basis, the coordinates of any $\mathbf{x}$ are just dot products.

$$\mathbf{x} = (\mathbf{q}_1 \cdot \mathbf{x})\,\mathbf{q}_1 + (\mathbf{q}_2 \cdot \mathbf{x})\,\mathbf{q}_2 + \cdots + (\mathbf{q}_n \cdot \mathbf{x})\,\mathbf{q}_n, \qquad \|\mathbf{x}\|^2 = \sum_{i=1}^{n} (\mathbf{q}_i \cdot \mathbf{x})^2 \tag{10.1}$$

To find the coefficients, dot both sides with $\mathbf{q}_j$: every term but one vanishes. No system of equations is needed, which is why the Fourier series, the wavelet transform and principal components all work in orthonormal bases.

10.2 Orthogonal matrices

Stack an orthonormal basis as the columns of a square matrix Q. The dot products of Definition 10.1 are then exactly the entries of Q^TQ.

Definition 10.2 Orthogonal matrix

A square matrix Q is orthogonal if $Q^TQ = I$. Equivalently, its columns are an orthonormal basis, and $Q^{-1} = Q^T$.

Theorem 10.1 Orthogonal matrices preserve geometry

If Q is orthogonal then $\|Q\mathbf{x}\| = \|\mathbf{x}\|$ and $(Q\mathbf{x})\cdot(Q\mathbf{y}) = \mathbf{x}\cdot\mathbf{y}$ for all vectors: lengths and angles are unchanged.

Proof

$(Q\mathbf{x})\cdot(Q\mathbf{y}) = \mathbf{x}^TQ^TQ\mathbf{y} = \mathbf{x}^TI\mathbf{y} = \mathbf{x}\cdot\mathbf{y}$.

Taking $\mathbf{y} = \mathbf{x}$ gives the statement about lengths.

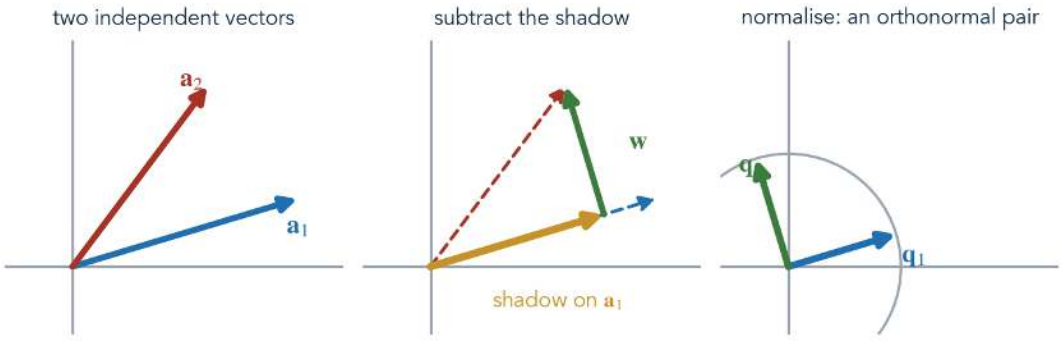
Since angles are defined through dot products and lengths (Section 2.4), they are preserved too. ■

Orthogonal matrices are the rotations and reflections of any dimension — the rigid motions that keep the origin fixed — and their determinant is ± 1. They are the perfectly conditioned matrices: they stretch nothing, so rounding errors passed through them do not grow. That is the entire reason numerical methods are built out of them.

In machine learning. A deep network whose weight matrices are orthogonal preserves the length of its signal through every layer, so activations neither explode nor die away as depth grows. Orthogonal initialisation of weights, and the family of optimisers in Chapter 19 whose updates are orthogonal matrices, both exploit Theorem 10.1.

10.3 The Gram–Schmidt process

Given any independent vectors, an orthonormal basis for their span can be built one vector at a time: remove from each new vector its shadows on the directions already chosen, then normalise what is left.



each new vector loses its component along the directions already chosen and is rescaled to length one. The span never changes; only the basis becomes perpendicular.

FIGURE 10.1 Two independent vectors (left). The second loses its shadow on the first, leaving a perpendicular remainder (centre). Both are scaled to length one (right). The span never changes; only the basis becomes perpendicular.

$$\mathbf{w}_k = \mathbf{a}_k - \sum_{j<k} (\mathbf{q}_j \cdot \mathbf{a}_k)\,\mathbf{q}_j, \qquad \mathbf{q}_k = \frac{\mathbf{w}_k}{\|\mathbf{w}_k\|} \tag{10.2}$$

Worked Example 10.1 Gram–Schmidt in three dimensions

Orthonormalise $\mathbf{a}_1 = (1, 1, 0)$ and $\mathbf{a}_2 = (1, 0, 1)$.

$\|\mathbf{a}_1\| = \sqrt{2}$, so $\mathbf{q}_1 = (1, 1, 0)/\sqrt{2}$.

The shadow of $\mathbf{a}_2$ on $\mathbf{q}_1$ has coefficient $\mathbf{q}_1 \cdot \mathbf{a}_2 = 1/\sqrt{2}$, so the shadow is $1/2(1, 1, 0)$. Subtracting: $\mathbf{w}_2 = (1, 0, 1) - (1/2, 1/2, 0) = (1/2, -1/2, 1)$.

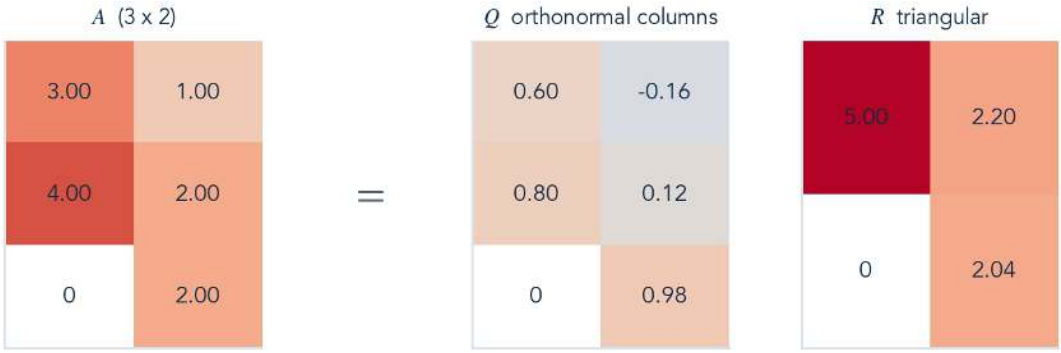
$\|\mathbf{w}_2\| = \sqrt{1/4 + 1/4 + 1} = \sqrt{3/2}$, so $\mathbf{q}_2 = (1, -1, 2)/\sqrt{6}$. Check: $\mathbf{q}_1 \cdot \mathbf{q}_2 = (1 - 1 + 0)/\sqrt{12} = 0$.

10.4 QR: Gram–Schmidt as a factorisation

As elimination became LU, Gram–Schmidt becomes a factorisation too. Each original column is a combination of the orthonormal vectors built so far — never of later ones — so the coefficients form an upper-triangular matrix.

> **Theorem 10.2** QR factorisation
>
> Any $m \times n$ matrix A with independent columns can be written $A = QR$, where Q is $m \times n$ with orthonormal columns and R is $n \times n$, upper triangular and invertible.

$$A = \begin{bmatrix} | & & | \\ \mathbf{a}_1 & \cdots & \mathbf{a}_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ | & & | \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 \cdot \mathbf{a}_1 & \cdots & \mathbf{q}_1 \cdot \mathbf{a}_n \\ & \ddots & \vdots \\ 0 & & \mathbf{q}_n \cdot \mathbf{a}_n \end{bmatrix} \tag{10.3}$$



Gram-Schmidt, written as a factorisation. Q holds the perpendicular directions; R records how much of each the original columns used. Least squares becomes $R\mathbf{x} = Q^T\mathbf{b}$.

FIGURE 10.2 A 3×2 matrix factored as QR. The columns of Q are orthonormal; R records how much of each direction the original columns used, and it is triangular because each column uses only the directions built before it.

In practice QR is computed not by Gram–Schmidt as written, which slowly loses perpendicularity to rounding, but by a sequence of reflections that are exactly orthogonal. The result is the same

factorisation, obtained stably, and it is the workhorse behind least squares in Chapter 11 and eigenvalue computation in Chapter 13.

Worked Example 10.2 A QR factorisation by hand

Factor the matrix with columns $\mathbf{a}_1 = (3, 4)$ and $\mathbf{a}_2 = (1, 2)$.

$r_{11} = \|\mathbf{a}_1\| = 5$ and $\mathbf{q}_1 = (0.6, 0.8)$. Then $r_{12} = \mathbf{q}_1 \cdot \mathbf{a}_2 = 0.6 + 1.6 = 2.2$.

$\mathbf{w}_2 = \mathbf{a}_2 - 2.2\,\mathbf{q}_1 = (1 - 1.32, 2 - 1.76) = (-0.32, 0.24)$, whose length is $r_{22} = 0.4$, so $\mathbf{q}_2 = (-0.8, 0.6)$.

Check: $5\mathbf{q}_1 = (3, 4)$ and $2.2\,\mathbf{q}_1 + 0.4\,\mathbf{q}_2 = (1, 2)$. So Q has columns $(0.6, 0.8)$ and $(-0.8, 0.6)$, and R has rows $(5, 2.2)$ and $(0, 0.4)$.

10.5 Projection onto a subspace

Chapter 2 projected onto a line. The same idea projects onto any subspace: find the point of the subspace nearest to $\mathbf{v}$, which is characterised by the leftover being perpendicular to the whole subspace.

Theorem 10.3 Projection matrices

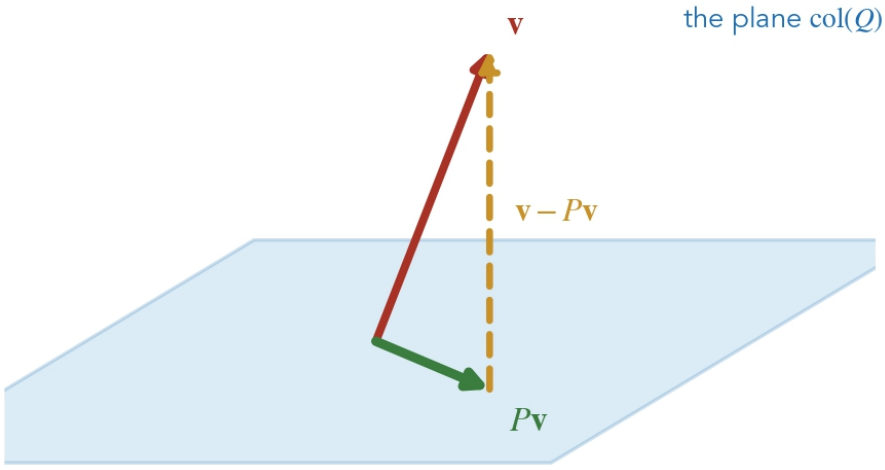
If the columns of Q are an orthonormal basis of a subspace S, the projection onto S is $P = QQ^T$. If the columns of A are merely independent, it is $P = A(A^TA)^{-1}A^T$. In both cases $P^2 = P$ and $P^T = P$.

Proof

The nearest point of $S = \text{col}(Q)$ to $\mathbf{v}$ is $Q\mathbf{c}$ for the $\mathbf{c}$ that makes $\mathbf{v} - Q\mathbf{c}$ perpendicular to every column: $Q^T(\mathbf{v} - Q\mathbf{c}) = \mathbf{0}$. Since $Q^TQ = I$, $\mathbf{c} = Q^T\mathbf{v}$ and the projection is $QQ^T\mathbf{v}$.

With a general basis A the same condition reads $A^T(\mathbf{v} - A\mathbf{c}) = \mathbf{0}$, so $\mathbf{c} = (A^TA)^{-1}A^T\mathbf{v}$.

$P^2 = QQ^TQQ^T = QQ^T$, because a point already in S is its own projection; and $P^T = P$ is visible in either formula. ■



$$P = QQ^T, \quad P^2 = P, \quad P^T = P$$

the projection is the point of the plane nearest to $\mathbf{v}$, and the leftover is perpendicular to the whole plane. Projecting twice changes nothing: that is $P^2 = P$.

FIGURE 10.3 Projection onto a plane. The projection $P\mathbf{v}$ is the nearest point of the plane, and the residual $\mathbf{v} - P\mathbf{v}$ is perpendicular to the entire plane.

$$P = QQ^T = A(A^TA)^{-1}A^T, \quad P^2 = P, \quad P^T = P, \quad I - P \text{ projects onto } S^\perp \tag{10.4}$$

Projection is the geometric form of "the best approximation from a restricted set", and the condition that characterises it — the error is perpendicular to everything you were allowed to use — is the same condition in regression, in principal component analysis, in the Fourier series and in the Kalman filter. Recognising the condition is recognising that a problem is a projection.

TABLE 10.1 Three kinds of structured matrix met so far, and what their structure buys.

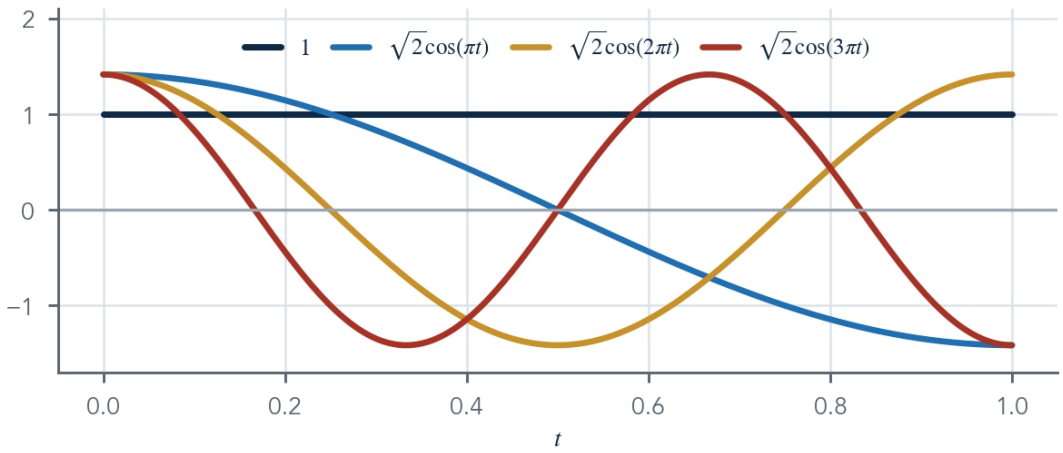
kind	defining property	what it does	why it matters
orthogonal Q	$Q^TQ = I$	rotate or reflect	stable; $Q^{-1} = Q^T$
projection P	$P^2 = P = P^T$	drop the perpendicular part	best approximation
triangular R, U	zeros below the diagonal	each coordinate uses earlier ones only	solve by substitution

Worked Example 10.3 An orthonormal basis of functions

On the interval from 0 to 1, replace the dot product by $\langle f, g\rangle = \int_0^1 f(t)g(t)\,dt$. The functions 1, $\sqrt{2}\cos(\pi t)$ and $\sqrt{2}\cos(2\pi t)$ are then orthonormal.

Take $f(t) = 1 + \cos(\pi t)$. Its coordinates are dot products, as in Equation 10.1: $\langle f, 1\rangle = 1$, $\langle f, \sqrt{2}\cos\pi t\rangle = \sqrt{2}\int_0^1 \cos^2(\pi t)\,dt = 1/\sqrt{2}$, and $\langle f, \sqrt{2}\cos 2\pi t\rangle = 0$.

So $f = 1\cdot 1 + 1\sqrt{2}\cdot\sqrt{2}\cos(\pi t)$, as it must be. Figure 10.4 shows the first few of these functions; the same bookkeeping, with enough of them, describes any reasonable signal.



each pair of these functions has integral of product zero over the interval, and each has integral of square one: an orthonormal basis, in which every coefficient is one integral.

FIGURE 10.4 The functions 1 and $\sqrt{2}\cos(k\pi t)$ for $k = 1, 2, 3$ on the interval from 0 to 1. With the dot product replaced by an integral they are orthonormal, so the coefficients of a signal are found one at a time, exactly as in Equation 10.1.

Key ideas

- In an orthonormal basis, coordinates are dot products and no system of equations is needed.
- Orthogonal matrices, with $Q^TQ = I$, preserve lengths and angles and are perfectly conditioned.
- Gram–Schmidt removes each vector's shadows on earlier directions and normalises, giving an orthonormal basis of the same span.
- Gram–Schmidt written as a factorisation is $A = QR$, computed stably in practice by reflections.
- Projection onto a subspace is $P = QQ^T$, with $P^2 = P = P^T$, and its residual is perpendicular to the subspace.

Exercises

1. Check that the columns $(3, 4)/5$ and $(-4, 3)/5$ form an orthogonal matrix, and write $(10, 5)$ in that basis using dot products.
2. Apply Gram–Schmidt to $(1, 0, 1)$, $(1, 1, 0)$, $(0, 1, 1)$.
3. Find the projection matrix onto the line through $(1, 2, 2)$ and verify $P^2 = P$.
4. Show that if P is a projection then so is $I - P$, and describe the subspace it projects onto.
5. Why is $\det P = 0$ for a projection onto a proper subspace? Give the answer in terms of volume.
6. Show that the matrix with columns $(1, 1)/\sqrt{2}$ and $(1, -1)/\sqrt{2}$ is orthogonal, and find its determinant.
7. Find the projection of $(1, 2, 3)$ onto the plane $z = 0$, and the matrix that performs it.
8. Explain what goes wrong in Gram–Schmidt if the starting vectors are dependent.

CHAPTER ELEVEN

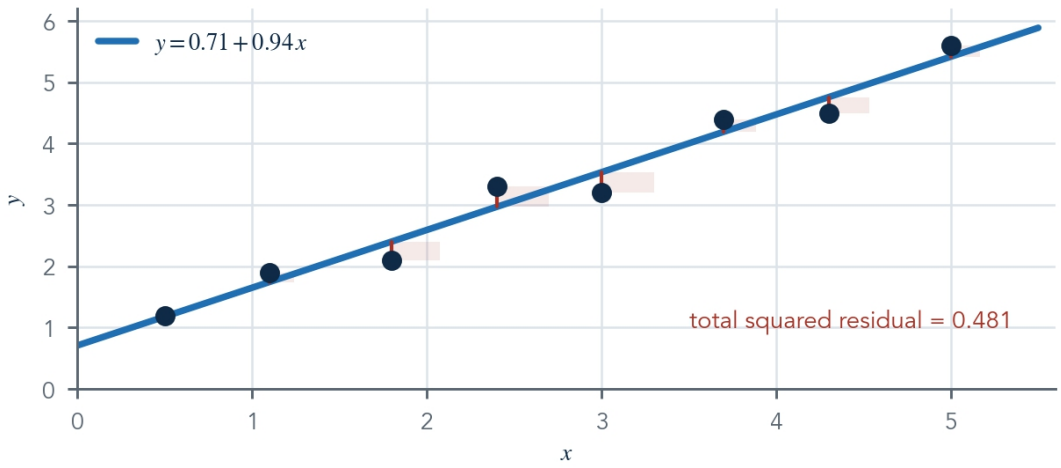
11 Least Squares

Real data does not lie on a line. A hundred measurements of two quantities that ought to be linearly related scatter around the line, and the system that asks the line to pass through every point has no solution. Least squares asks a better question — which line comes closest? — and the answer is a projection. It is the oldest method of machine learning, still one of the most used, and the cleanest place to see linear algebra turn data into a model.

11.1 The problem

Fitting $y = c_0 + c_1 x$ to m points (x_i, y_i) asks for $c_0 + c_1 x_i = y_i$ for every i: a tall system of m equations in two unknowns.

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \approx \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \qquad \min_{\mathbf{c}} \|A\mathbf{c} - \mathbf{y}\|^2 = \min_{\mathbf{c}} \sum_{i=1}^{m} (c_0 + c_1 x_i - y_i)^2 \tag{11.1}$$



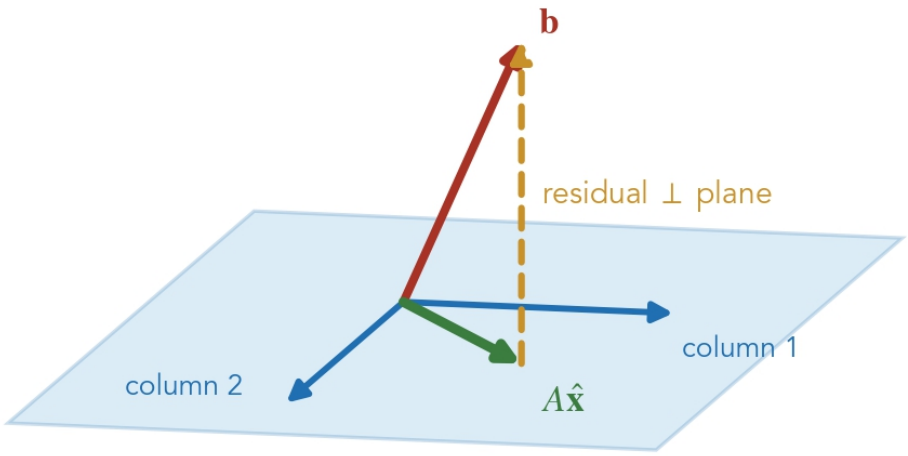
eight points, one line. Each red square has area equal to a squared residual, and the fitted line is the unique one that makes their total as small as possible.

FIGURE 11.1 Eight points and the line that minimises the sum of the squared vertical distances. Each red square has an area equal to one squared residual; no other line makes their total smaller.

The same formulation covers far more than lines. A quadratic adds a column of x_i^2; a model with twenty input features has twenty columns plus one of ones. As long as the model is linear in its coefficients, the problem is exactly Equation 11.1 with a wider A.

11.2 The geometric solution

Look at the problem in $\mathbb{R}^m$, the space with one axis per data point. The vector $\mathbf{y}$ is one point there. Every choice of coefficients produces an output $A\mathbf{c}$, and the set of all such outputs is the column space of A — a plane, for a line fit, inside a space of m dimensions.



$$A^T(\mathbf{b} - A\hat{\mathbf{x}}) = \mathbf{0}$$

b is not in the plane of the columns, so no exact solution exists. The best one is the projection, and 'residual perpendicular to every column' is the normal equation.

FIGURE 11.2 The fit seen in the space of data. The target $\mathbf{y}$ lies off the plane of the columns, so no exact solution exists. The best approximation is its projection $A\mathbf{c}$, and the residual is perpendicular to the plane.

Minimising $\|A\mathbf{c} - \mathbf{y}\|$ means finding the point of the column space nearest to $\mathbf{y}$, which by Theorem 10.3 is the projection, characterised by a residual perpendicular to every column.

Theorem 11.1 The normal equations

$\mathbf{c}$ minimises $\|A\mathbf{c} - \mathbf{y}\|^2$ if and only if $A^TA\,\mathbf{c} = A^T\mathbf{y}$. If the columns of A are independent the solution is unique.

Proof

Geometrically: the residual $\mathbf{y} - A\mathbf{c}$ must be perpendicular to every column, which is $A^T(\mathbf{y} - A\mathbf{c}) = \mathbf{0}$.

By calculus: the gradient of $\|A\mathbf{c} - \mathbf{y}\|^2 = \mathbf{c}^T A^T A\mathbf{c} - 2\mathbf{c}^T A^T\mathbf{y} + \mathbf{y}^T\mathbf{y}$ is $2A^TA\mathbf{c} - 2A^T\mathbf{y}$, which vanishes at the same point.

If the columns are independent then A^TA is invertible, since its null space equals that of A (Exercise 7.3), and the solution is unique. ■

$$A^TA\hat{\mathbf{c}} = A^T\mathbf{y}, \qquad \hat{\mathbf{y}} = A\hat{\mathbf{c}} = A(A^TA)^{-1}A^T\mathbf{y} = P\mathbf{y} \tag{11.2}$$

Worked Example 11.1 A line through three points

Fit $y = c_0 + c_1 x$ to $(0, 1)$, $(1, 2)$, $(2, 4)$.

A has rows $(1, 0)$, $(1, 1)$, $(1, 2)$, so A^TA has rows $(3, 3)$ and $(3, 5)$, and $A^T\mathbf{y} = (1 + 2 + 4, 0 + 2 + 8) = (7, 10)$.

The normal equations $3c_0 + 3c_1 = 7$ and $3c_0 + 5c_1 = 10$ give $c_1 = 1.5$ and $c_0 = 5/6 \approx 0.833$.

Fitted values 0.833, 2.333, 3.833 leave residuals 0.167, -0.333, 0.167. They sum to zero — perpendicular to the column of ones — and $0(0.167) + 1(-0.333) + 2(0.167) = 0$, perpendicular to the column of x. The geometry is exact.

The two orthogonality checks in that example are not coincidences to admire; they are the normal equations, written out. A least-squares residual always sums to zero when the model has an intercept, and it is always uncorrelated with every feature.

For the eight points of Figure 11.1 the same calculation gives the line $y = 0.71 + 0.94\,x$ with a total squared residual of 0.481.

11.3 Solving it well

The normal equations are the clearest statement of the answer and the worst way to compute it. Forming A^TA squares the condition number of the problem, discarding digits that the data contained; Chapter 12 measures how many. The stable method uses the QR factorisation of Chapter 10.

$$A = QR \implies A^TA\,\hat{\mathbf{c}} = A^T\mathbf{y} \iff R\hat{\mathbf{c}} = Q^T\mathbf{y} \tag{11.3}$$

Substituting $A = QR$ into the normal equations and cancelling R^T leaves a triangular system solved by substitution, and A^TA is never formed. Every reliable least-squares routine does this, or uses the singular value decomposition of Chapter 15, which is slower and more robust still.

11.4 When there are more unknowns than data

If A is wide, or its columns are dependent, the normal equations have infinitely many solutions — the shifted null space of Section 7.5 — and all of them fit the data equally well. One solution is distinguished: the shortest, which has no component in the null space and lies entirely in the row space.

> **Theorem 11.2** Gradient descent finds the shortest solution
>
> For least squares started from $\mathbf{c} = 0$, gradient descent with a small enough step converges to the minimum-norm solution.

Proof

Each step adds a multiple of the gradient $2A^T(A\mathbf{c} - \mathbf{y})$, which lies in the row space $\text{col}(A^T)$.

Starting from $\mathbf{0}$, every iterate therefore stays in the row space and never acquires a null-space component.

The limit is a least-squares solution lying in the row space, and there is exactly one such solution: the shortest. ■

An over-parameterised model trained by gradient descent does not pick an arbitrary member of its infinite family of perfect fits. For linear least squares it provably picks the shortest one, and that implicit preference for small solutions — invisible in the objective, supplied by the method — is part of why models with more parameters than data can still predict well.

Worked Example 11.2 The shortest of infinitely many solutions

The single equation $x_1 + x_2 = 2$ has the solutions $(t, 2 - t)$ for every t.

The squared length $t^2 + (2 - t)^2$ is smallest at $t = 1$, giving $(1, 1)$.

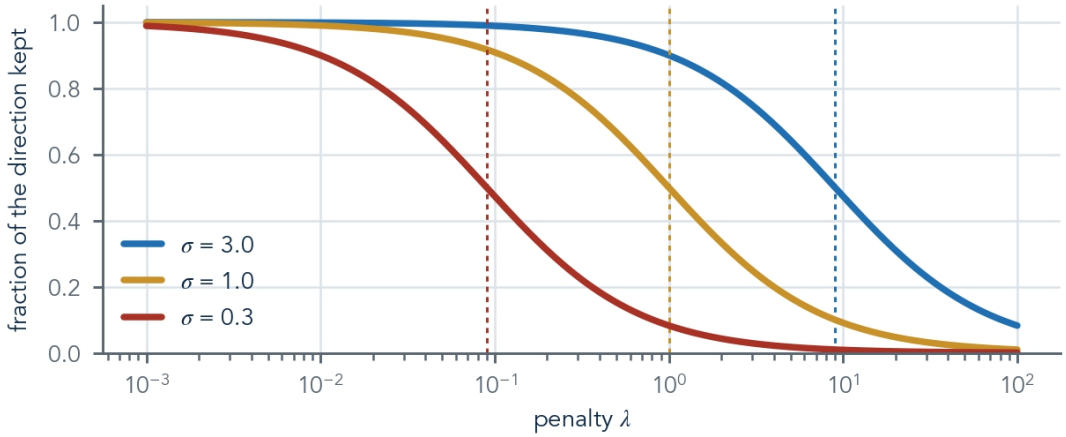
That solution lies along $(1, 1)$, which spans the row space of $A = [1 \quad 1]$, exactly as Theorem 11.2 requires. The pseudoinverse of Chapter 15 gives it directly: $A^T(AA^T)^{-1}$ times 2 is $(1, 1)/2$ times 2, which is $(1, 1)$.

11.5 Ridge regression

When the columns of A are nearly dependent the least-squares coefficients become enormous and unstable, trading huge positive and negative weights to fit noise. Ridge regression adds a penalty on size.

$$\hat{\mathbf{c}}_\lambda = \arg\min_{\mathbf{c}} \|A\mathbf{c}-\mathbf{y}\|^2 + \lambda\|\mathbf{c}\|^2 = (A^TA + \lambda I)^{-1}A^T\mathbf{y} \tag{11.4}$$

Adding λI lifts every eigenvalue of A^TA by λ, so the matrix is always invertible and never badly conditioned. In the singular value picture of Chapter 15 its effect is exact and easy to state: the component of the solution along the i-th singular direction is multiplied by $\sigma_i^2/(\sigma_i^2 + \lambda)$.



ridge keeps $\sigma^2/(\sigma^2+\lambda)$ of each singular direction. Strong directions survive; weak ones, where the data says little, are shrunk first. Each dashed line marks where half is kept.

FIGURE 11.3 The fraction of each singular direction that ridge keeps, for singular values 3, 1 and 0.3. Directions where the data is strong are barely touched; directions where it is weak, and where noise would dominate, are shrunk first.

In machine learning. Linear regression is least squares, and ridge regression is the same thing as "weight decay" in neural-network training: an $\lambda\|\mathbf{c}\|^2$ term added to the loss. The final layer of many large models is trained, or probed, with exactly this closed form, because it is cheap and exact once the features from the earlier layers are fixed.

TABLE 11.1 Ways to compute a least-squares fit, from the most transparent to the most robust.

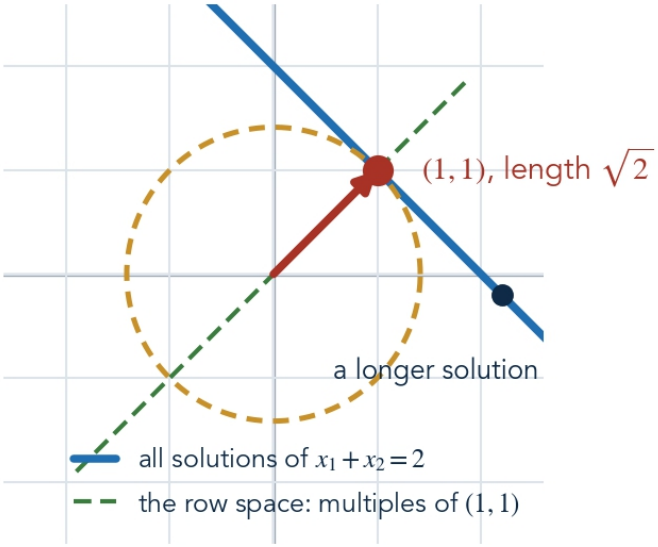
method	computes	conditioning	use when
normal equations	$A^T A\,\mathbf{c} = A^T \mathbf{y}$	$(A)^2$	small, well-scaled problems, or by hand
QR	$R\mathbf{c} = Q^T \mathbf{y}$	(A)	the default
SVD	$\mathbf{c} = V\Sigma^{+} U^T \mathbf{y}$	(A), and reveals rank	rank-deficient or nearly so
ridge	$(A^T A + \lambda I)\mathbf{c} = A^T \mathbf{y}$	improved by λ	many correlated features
gradient descent	iterates toward the fit	speed set by $(A)^2$	problems too large to factor

Worked Example 11.3 One bad point

Fit a line to (0, 0), (1, 1), (2, 2) and (3, 9), where the last point should have been (3, 3).

With $\sum x = 6$, $\sum y = 12$, $\sum x^2 = 14$ and $\sum xy = 32$, the normal equations give slope $(4 \times 32 - 6 \times 12)/(4 \times 14 - 36) = 2.8$ and intercept $(12 - 2.8 \times 6)/4 = -1.2$.

With the correct point the fit would be exactly $y = x$. One error has nearly tripled the slope, because squaring makes a residual of 6 count as much as thirty-six residuals of 1. Least squares is optimal for well-behaved noise and fragile against outliers. Figure 11.4 returns to Example 11.2, where geometry picks the shortest of many exact fits.



the circle of radius $\sqrt{2}$ just touches the line of solutions at $(1,1)$. Every other solution adds a component along the null space, perpendicular to $(1,1)$, and is longer.

FIGURE 11.4 The solutions of $x_1 + x_2 = 2$ form a line. The shortest is where the smallest circle about the origin touches it, at $(1,1)$ — the one solution that lies in the row space.

Key ideas

- Least squares chooses the coefficients that make $\|A\mathbf{c} - \mathbf{y}\|^2$ as small as possible.
- The best fit is the projection of $\mathbf{y}$ onto the column space, and its residual is perpendicular to every column: the normal equations.
- With an intercept, a least-squares residual sums to zero, and it is uncorrelated with every feature.
- QR solves least squares stably without ever forming A^TA.
- Gradient descent from zero finds the minimum-norm solution, and ridge regression shrinks the weak singular directions first.

Exercises

1. Fit $y = c_0 + c_1 x$ to $(1, 2)$, $(2, 3)$, $(3, 5)$, $(4, 6)$ by solving the normal equations, and verify that the residuals sum to zero.
2. Show that the least-squares line always passes through the point $(\bar{x}, \bar{y})$ of the averages.
3. Set up, but do not solve, the least-squares problem for fitting $y = c_0 + c_1 x + c_2 x^2$ to five points. What are the shapes of A, $A^T A$ and $A^T \mathbf{y}$?
4. For A with a single column $(1, 1)$ and $\mathbf{y} = (1, 3)$, compute the ridge estimate as a function of λ and describe its limits as $\lambda \to 0$ and $\lambda \to \infty$.
5. Explain, using the four subspaces, why the residual of a least-squares fit lies in the left null space of A.
6. Fit $y = cx$, with no intercept, to $(1, 2)$, $(2, 3)$ and $(3, 7)$.
7. Write $A^T A$ and $A^T \mathbf{y}$ for fitting $y = c_0 + c_1 x$ to $(0, 0)$, $(1, 1)$, $(2, 1)$, and solve.
8. Explain why ridge regression with $\lambda > 0$ always has exactly one solution.

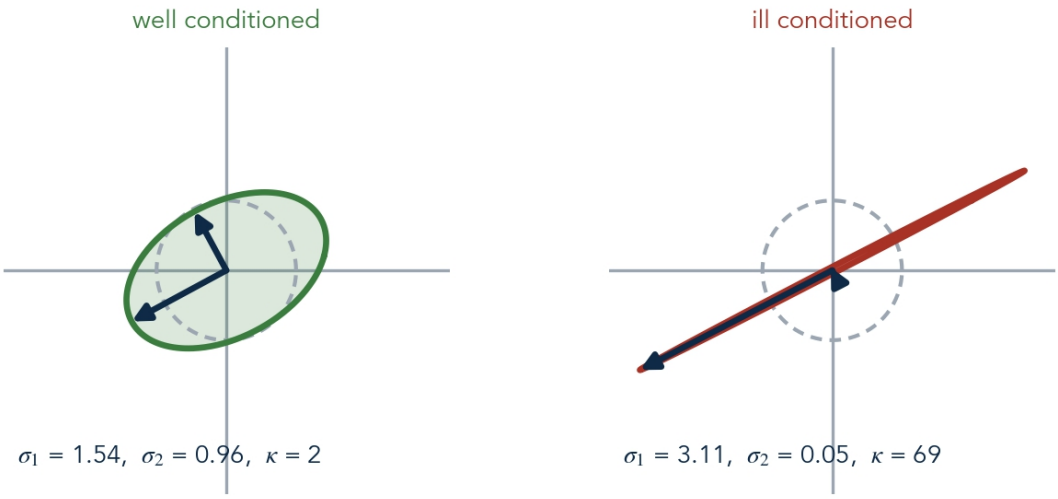
CHAPTER TWELVE

12 Conditioning

Two systems can look equally innocent on paper and behave completely differently on a computer. One returns an answer good to fifteen digits; the other returns an answer in which no digit can be trusted. The difference is not the size of the numbers, and it is not the determinant. It is the shape of an ellipse, and this chapter introduces the picture that the rest of the book uses as its lens.

12.1 The one picture

Apply a 2×2 matrix to every point of the unit circle. Because the map is linear, the image is an ellipse, possibly flattened to a segment. Its longest half-axis is the most any unit vector is stretched; its shortest half-axis is the least.



the unit circle (dashed) and its image. The longest and shortest half-axes are the singular values, and their ratio κ measures how close the map is to flattening the plane.

FIGURE 12.1 The unit circle (dashed) and its image under two matrices. The half-axes of each ellipse are the singular values $\sigma_1 \geq \sigma_2$. On the left they are similar and the map is harmless; on the right the ellipse is a sliver and the map is close to flattening the plane.

Definition 12.1 Singular values and condition number

The singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ of a matrix are the lengths of the half-axes of the image of the unit sphere. For an invertible square matrix the condition number is $(A) = \sigma_1/\sigma_n$.

Chapter 15 proves that every matrix, of any shape, carries the unit sphere to such an ellipsoid, and computes its axes. For now the picture is enough, because it already explains the determinant ($|\det A| = \sigma_1\sigma_2\cdots\sigma_n$, the volume of the ellipsoid), invertibility (no half-axis is zero), and the quantity this chapter is about.

$$\sigma_1 = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|, \qquad \sigma_n = \min_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|, \qquad \kappa(A) = \frac{\sigma_1}{\sigma_n} \geq 1 \tag{12.1}$$

12.2 Why the ratio decides accuracy

Suppose $A\mathbf{x} = \mathbf{b}$, and $\mathbf{b}$ is known only approximately: measured, rounded, or stored in finitely many digits. How much can a small error in $\mathbf{b}$ move the answer?

Theorem 12.1 Sensitivity of a linear system

If $A\mathbf{x} = \mathbf{b}$ and $A(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$, then $\|\delta\mathbf{x}\|\|\mathbf{x}\| \le (A)\,\|\delta\mathbf{b}\|\|\mathbf{b}\|$, and there are choices of $\mathbf{b}$ and $\delta\mathbf{b}$ for which equality holds.

Proof

Subtracting the two equations, $A\,\delta\mathbf{x} = \delta\mathbf{b}$, so $\|\delta\mathbf{x}\| = \|A^{-1}\delta\mathbf{b}\| \le \|\delta\mathbf{b}\|/\sigma_n$, because the inverse stretches by at most $1/\sigma_n$.

From $\mathbf{b} = A\mathbf{x}$, $\|\mathbf{b}\| \le \sigma_1\|\mathbf{x}\|$, so $1/\|\mathbf{x}\| \le \sigma_1/\|\mathbf{b}\|$.

Multiplying the two inequalities gives the bound. Equality occurs when $\mathbf{x}$ lies along the most-stretched direction and $\delta\mathbf{b}$ along the least. ■

Read the theorem as a statement about digits. A relative error of 10^{-k} in the data produces a relative error of up to $\cdot 10^{-k}$ in the answer: the solution loses about $_{10}$ significant digits. Double-precision arithmetic carries about sixteen, so a matrix with $= 10^{10}$ leaves about six trustworthy digits, and one with $= 10^{16}$ leaves none, however carefully the solver is written.

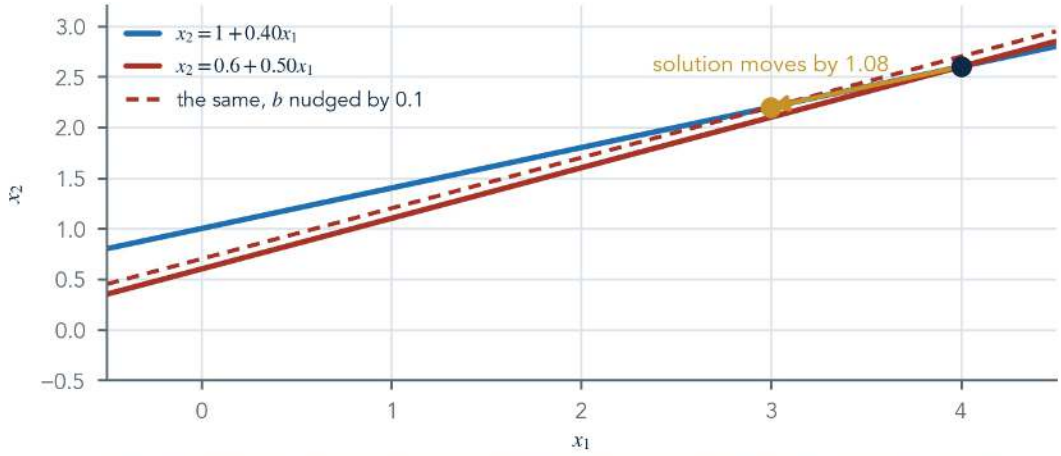


FIGURE 12.2 Two nearly parallel lines. Changing one right-hand side by 0.1 moves the crossing point by more than 1, because the lines meet at a shallow angle. The row picture of an ill-conditioned system.

Worked Example 12.1 Measuring the amplification

The lines $x_2 = 1 + 0.40x_1$ and $x_2 = 0.6 + 0.50x_1$ meet where $1 + 0.4x_1 = 0.6 + 0.5x_1$, at $(4, 2.6)$.

Raise the second intercept to 0.7. Now $1 + 0.4x_1 = 0.7 + 0.5x_1$ gives $x_1 = 3$ and the crossing moves to $(3, 2.2)$.

The data changed by 0.1 and the answer moved by $\sqrt{1^2 + 0.4^2} \approx 1.08$: an amplification of about eleven. The culprit is visible in the slopes, 0.40 and 0.50, which make the rows nearly parallel.

The determinant says whether a matrix flattens space; the condition number says how close it comes. A matrix can have a determinant of 10^{-100} and be perfectly conditioned, or a determinant of one and be useless. For every numerical question the second number is the one to ask for.

12.3 Why the normal equations lose twice as much

The singular values of A^TA are the squares of those of A — Chapter 15 shows this in one line — so its condition number is the square.

$$\kappa(A^TA) = \kappa(A)^2 \tag{12.2}$$

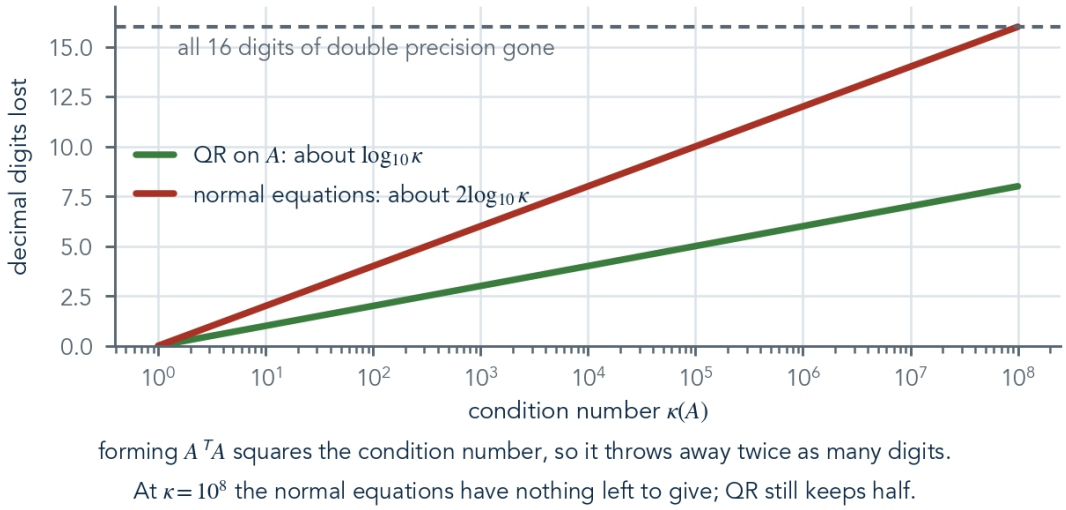


FIGURE 12.3 Decimal digits lost when solving a least-squares problem by QR on A (about $_{10}$) and through the normal equations (about 2_{10}). At $= 10^8$ the normal equations have lost every digit double precision holds.

This is the precise reason Chapter 11 recommended QR. The normal equations are mathematically equivalent and numerically a different problem. Squaring a condition number of a thousand gives a million: three lost digits become six.

12.4 Improving the condition

Conditioning is a property of how a problem is posed, and it can often be changed.

TABLE 12.1 Common causes of ill-conditioning in data problems, and the standard remedies.

cause	what the ellipse looks like	remedy
features on wildly different scales	stretched along the large-unit axis	standardise each column
nearly duplicate features	flattened across their difference	remove one, or use ridge
polynomial features $x, x^2, x^3, \ldots$	columns nearly parallel	use an orthogonal basis of polynomials
forming A^TA	the sliver made thinner	factor A directly
a genuinely weak direction	one axis near zero	regularise, or accept less precision

Worked Example 12.2 Scaling a feature

A model uses house area in square metres (around 100) and number of rooms (around 4). Suppose the columns are otherwise unrelated, with typical sizes 100 and 4.

The ellipse of A is stretched roughly 25 times more along the area direction, so is at least about 25.

Dividing each column by its typical size makes both around 1. The same information is present, the ellipse becomes round, and falls toward one. No prediction changes; only the number of reliable digits, and the speed of the iterative methods of Chapter 19, improves.

In machine learning. Conditioning governs speed as well as accuracy. Chapter 19 shows that gradient descent on a quadratic loss needs a number of steps proportional to the condition number of its Hessian, which for least squares is $(A)^2$. Standardising inputs, normalisation layers inside networks, and adaptive optimisers are all, among other things, ways of making the ellipse rounder.

Nearly every practical technique for making training faster or more stable — feature scaling, normalisation layers, careful initialisation, preconditioning, orthogonalised updates — can be understood as an intervention on the condition number of some matrix. When a method mysteriously helps, the first question to ask is which ellipse it made rounder.

Worked Example 12.3 Squaring the condition number

Let A be diagonal with entries 1 and 0.01.

Its singular values are 1 and 0.01, so $(A) = 100$: solving with A loses about two digits.

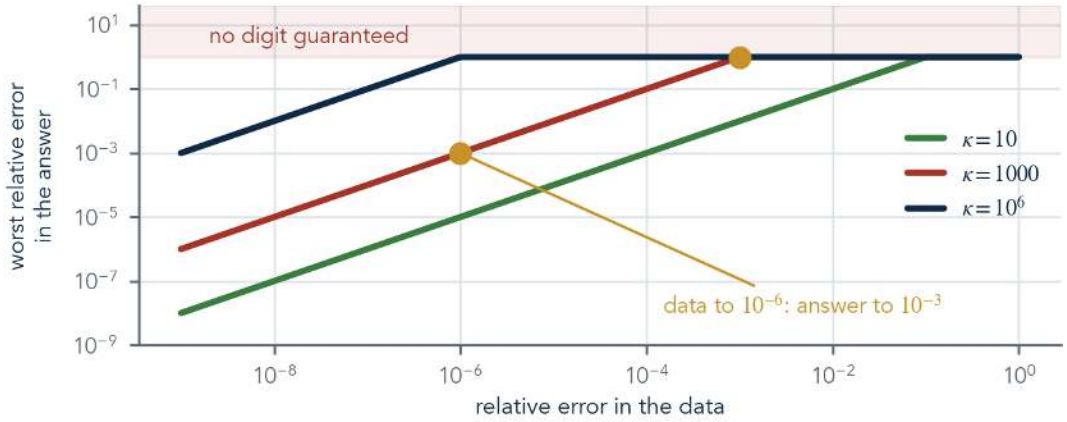
A^TA is diagonal with entries 1 and 0.0001, so $(A^TA) = 10{,}000$: going through the normal equations loses about four. The data did not change; only the route to the answer did.

Worked Example 12.4 What the bound means for real data

A system has condition number 1000.

If the right-hand side is known to a relative accuracy of 10^{-3} — a tenth of a per cent — Theorem 12.1 allows a relative error of up to $1000 \times 10^{-3} = 1$ in the answer: possibly one hundred per cent, and no digit is guaranteed.

If the data is known to 10^{-6}, the answer is good to about 10^{-3}. The same matrix is useless or excellent depending on how well the data is known, and the condition number is the exchange rate between the two. Figure 12.4 draws the exchange rate for three condition numbers.



Theorem 12.1 as an exchange rate. Each line multiplies the data's error by the condition number; the larger the number, the more accurately the data must be known to trust any digit.

FIGURE 12.4 The bound of Theorem 12.1 for three condition numbers. For $= 1000$, data known to 10^{-3} supports no digit of the answer, while data known to 10^{-6} supports three.

Key ideas

- A matrix turns the unit sphere into an ellipsoid whose half-axes are the singular values.
- The condition number, the largest singular value over the smallest, bounds how much relative error in the data is amplified in the answer.
- A solution loses about $_{10}$ significant digits.
- The determinant does not measure closeness to singularity; the condition number does.
- Forming A^TA squares the condition number, while scaling features, removing near-duplicates and regularising improve it.

Exercises

1. Find the singular values and condition number of the diagonal matrix with entries 10 and 0.1, and draw the image of the unit circle.
2. Compute $\det A$ and (A) for $A = 10^{-3} I$ in $\mathbb{R}^{50}$, and explain why the determinant is misleading.
3. A system has $= 10^{6}$ and data accurate to eight significant digits. How many digits of the solution can be trusted?
4. Explain geometrically why an orthogonal matrix has condition number one.
5. Two features are identical except for measurement noise of relative size 10^{-3}. Estimate how the condition number of the data matrix behaves, and suggest two remedies.
6. What is the condition number of the diagonal matrix with entries 1 and 10^{-6}?
7. Explain why every condition number is at least one.
8. Two standardised features have correlation ρ, and the eigenvalues of their covariance are $1 + \rho$ and $1 - \rho$. Find the condition number of the covariance when $\rho = 0.99$.

PART

IV

EIGENVALUES AND THE SVD

the structure inside every matrix

CHAPTER THIRTEEN

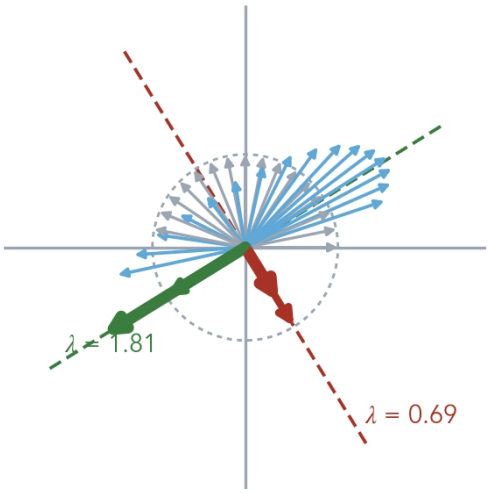
13 Eigenvalues and Eigenvectors

Most vectors are turned by a matrix: the output points somewhere different from the input. A few special directions are not. Along them the map only stretches, shrinks or flips, and in those directions a complicated matrix acts like a single number. Finding those directions turns powers of matrices into powers of numbers, explains how systems evolve over time, and ranks the pages of the web.

13.1 Directions that do not turn

Definition 13.1 Eigenvalue and eigenvector

A nonzero vector $\mathbf{v}$ is an eigenvector of a square matrix A if $A\mathbf{v} = \lambda\mathbf{v}$ for some number λ, which is the corresponding eigenvalue.



grey: unit vectors; blue: where A sends them. Every blue arrow has turned away from its grey one, except along the two dashed lines. Those are the eigenvectors, and the stretch there is the eigenvalue.

FIGURE 13.1 Unit vectors (grey) and their images (blue) under a 2×2 matrix. Nearly every image has turned away from its input. Along the two dashed lines it has not: those are the eigenvectors, and the eigenvalues 1.81 and 0.69 are the stretches along them.

The eigenvalue equation $A\mathbf{v} = \lambda\mathbf{v}$ says that on the line through $\mathbf{v}$ the matrix is just multiplication by λ. An eigenvalue of 2 doubles that direction; 0.5 halves it; -1 flips it; 0 destroys it, so the eigenvectors for eigenvalue zero are the null space.

13.2 Finding them

Rewrite the equation as $(A - \lambda I)\mathbf{v} = \mathbf{0}$. A nonzero solution exists exactly when $A - \lambda I$ has a nonzero null space — when it flattens space — which by Theorem 8.2 is when its determinant is zero.

$$\det(A-\lambda I)=0, \qquad \det\begin{bmatrix} a-\lambda & b \\ c & d-\lambda \end{bmatrix} = \lambda^2-(a+d)\lambda+(ad-bc)=0 \tag{13.1}$$

Worked Example 13.1 Eigenvalues of a 2×2 matrix

Find the eigenvalues and eigenvectors of the matrix with rows $(2, 1)$ and $(1, 2)$.

The characteristic equation is $\lambda^2 - 4\lambda + 3 = 0$, so $(\lambda - 3)(\lambda - 1) = 0$ and the eigenvalues are 3 and 1.

For $\lambda = 3$, $A - 3I$ has rows $(-1, 1)$ and $(1, -1)$, whose null space is spanned by $(1, 1)$. Check: $A(1, 1) = (3, 3) = 3(1, 1)$.

For $\lambda = 1$, $A - I$ has rows $(1, 1)$ and $(1, 1)$, with null space spanned by $(1, -1)$. Check: $A(1, -1) = (1, -1)$. The matrix stretches the diagonal by three and leaves the anti-diagonal alone.

Two facts can be read off the 2×2 characteristic polynomial and hold in every dimension: the eigenvalues add up to the trace (the sum of the diagonal entries), and they multiply to the determinant. The second is Chapter 8 again — volume is scaled by the product of the stretches along the eigenvectors.

For large matrices no one solves the characteristic polynomial; beyond degree four there is no formula, and the roots are exquisitely sensitive to rounding. Eigenvalues are computed by iterative methods built from the orthogonal factorisations of Chapter 10.

13.3 Diagonalisation: the matrix in its own coordinates

If an $n \times n$ matrix has n independent eigenvectors, use them as a basis. In that basis the matrix does nothing but scale each coordinate by its eigenvalue.

Theorem 13.1 Diagonalisation

If A has n independent eigenvectors, collected as the columns of V, then $A = V\Lambda V^{-1}$, where Λ is the diagonal matrix of eigenvalues. Consequently $A^k = V\Lambda^k V^{-1}$ for every k.

Proof

AV has columns $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$, which are the columns of $V\Lambda$. So $AV = V\Lambda$, and since V is invertible, $A = V\Lambda V^{-1}$.

Then $A^2 = V\Lambda V^{-1} V\Lambda V^{-1} = V\Lambda^2 V^{-1}$, and the same cancellation gives every power. ■

$$A = V\Lambda V^{-1} = \begin{bmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ | & & | \end{bmatrix}^{-1} \tag{13.2}$$

Read $A\mathbf{x} = V\Lambda V^{-1}\mathbf{x}$ from right to left: find the coordinates of $\mathbf{x}$ in the eigenbasis, scale each by its eigenvalue, and translate back. A matrix power that would take k multiplications becomes a power of n numbers.

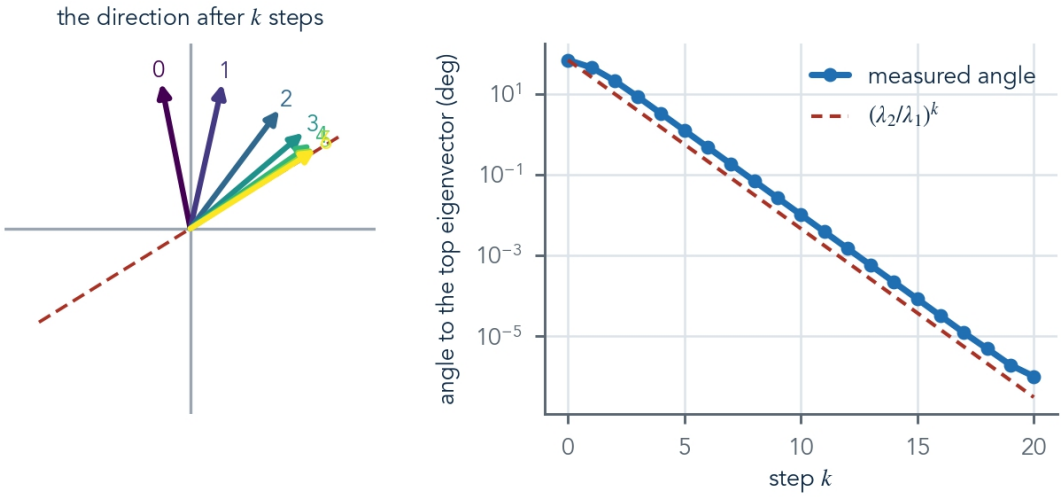
Not every matrix can be diagonalised. A shear has only one eigenvector direction, and a rotation of the plane by 90° has none among real vectors — its eigenvalues are the complex numbers $\pm i$, the algebraic signature of turning. Chapter 15 introduces a decomposition that exists for every matrix without exception.

13.4 Powers, and the dominant direction

Diagonalisation explains what happens when a matrix is applied again and again. Write $\mathbf{x}_0 = c_1\mathbf{v}_1 + \cdots + c_n\mathbf{v}_n$ in the eigenbasis, with the eigenvalues ordered so that $|\lambda_1| > |\lambda_2| \geq \cdots$.

$$A^k\mathbf{x}_0 = c_1\lambda_1^k\mathbf{v}_1 + c_2\lambda_2^k\mathbf{v}_2 + \cdots + c_n\lambda_n^k\mathbf{v}_n = \lambda_1^k\left(c_1\mathbf{v}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\mathbf{v}_2 + \cdots\right) \tag{13.3}$$

Every term except the first shrinks relative to it by a factor of $|\lambda_2/\lambda_1|$ per step. After enough steps $A^k\mathbf{x}_0$ points along $\mathbf{v}_1$, whatever $\mathbf{x}_0$ was (provided $c_1 \neq 0$). That is the **power method**: multiply, normalise, repeat.



multiply, normalise, repeat. The component along the top eigenvector grows fastest, so the direction converges to it at the rate λ_2/λ_1 per step -- the method behind PageRank.

FIGURE 13.2 Left: the direction of $A^k\mathbf{x}_0$ for $k = 0$ to 6, swinging onto the dominant eigenvector. Right: the angle to that eigenvector falls by the factor λ_2/λ_1 at every step, as Equation 13.3 predicts.

TABLE 13.1 What the eigenvalues of A say about the long-run behaviour of $\mathbf{x}_{k+1} = A\mathbf{x}_k$. The largest eigenvalue in size is the spectral radius.

largest eigenvalue, in size	behaviour of $A^k\mathbf{x}$	example
less than 1	dies away to 0	a stable system, a vanishing signal
exactly 1	settles to a fixed direction	a Markov chain's stationary distribution
greater than 1	grows without bound	compound growth, an exploding signal
complex pair	rotates as it grows or shrinks	oscillation

Worked Example 13.2 Two matrices that cannot be diagonalised with real eigenvectors

The quarter-turn with rows $(0, -1)$ and $(1, 0)$ has characteristic equation $\lambda^2 + 1 = 0$, so its eigenvalues are $\pm i$: no real direction is left unturned, as a rotation must behave.

The shear with rows $(1, 1)$ and $(0, 1)$ has characteristic equation $(1 - \lambda)^2 = 0$, so $\lambda = 1$ twice. But $A - I$ has rows $(0, 1)$ and $(0, 0)$, whose null space is only the line through $(1, 0)$.

The shear has one eigenvector direction where diagonalisation needs two. The decomposition of Chapter 15 handles both matrices without difficulty.

13.5 PageRank

The best-known eigenvector computation in the history of computing ranks web pages. Imagine a surfer who follows a random link from each page, and occasionally jumps to a page chosen at random. The fraction of time spent on each page in the long run is a measure of importance: a page is important if important pages link to it.

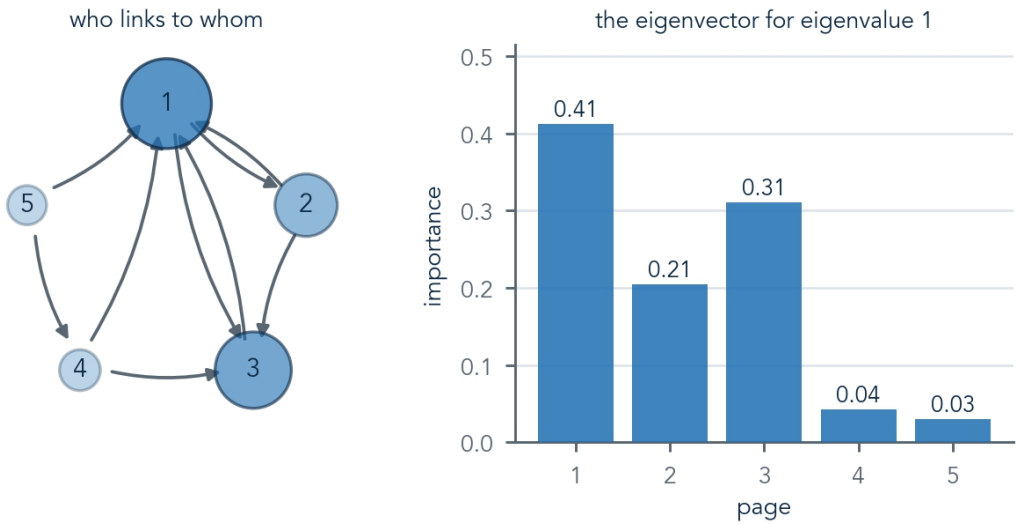


FIGURE 13.3 A web of five pages (left) and the long-run share of time a random surfer spends on each (right). The shares are the entries of the eigenvector of the link matrix for eigenvalue 1. Page 1 ranks first because pages that are themselves visited often link to it.

$$G = dM + \frac{1-d}{n}\mathbf{1}\mathbf{1}^T, \qquad G\mathbf{r} = \mathbf{r}, \qquad \sum_i r_i = 1 \tag{13.4}$$

Here M is the link matrix, whose column j spreads page j's probability equally over the pages it links to, and $d = 0.85$ is the probability of following a link rather than jumping. Every column of G sums to one, which guarantees an eigenvalue of exactly 1, and the random jumps guarantee that its eigenvector is unique and positive. The power method finds it in a few dozen multiplications, even for billions of pages, because each multiplication only follows links.

In machine learning. A recurrent network applies the same weight matrix at every time step, so by Table 13.1 its signals — and its gradients — grow or vanish like powers of the largest eigenvalue. This is the classic exploding and vanishing gradient problem, and the architectures that replaced simple recurrent networks were largely designed to keep the relevant eigenvalues near one. Spectral methods for clustering and for learning on graphs likewise work with the eigenvectors of matrices built from the connections between items.

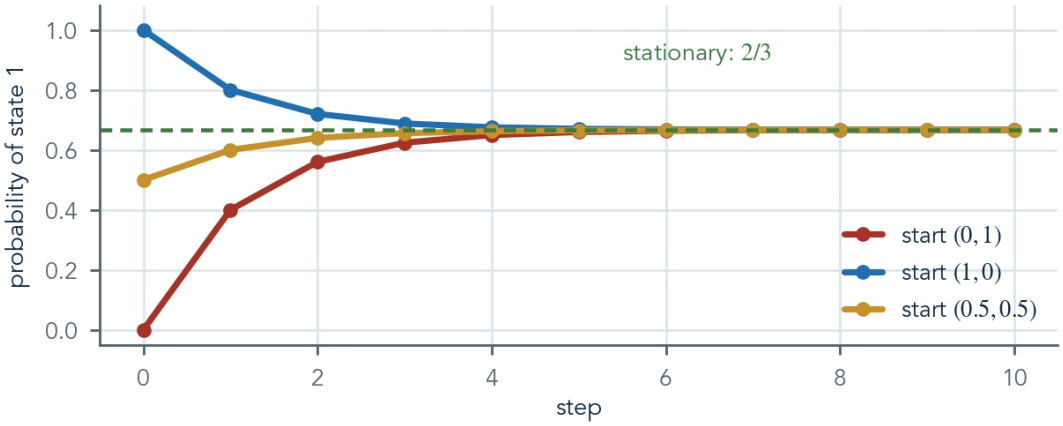
Eigenvectors turn the long-run behaviour of any repeated linear process into arithmetic. Whether a signal survives a hundred layers, whether a chain settles down, which page is most important and which direction of data carries most variance are all the same question: what does repeated application of this matrix favour, and by how much?

Worked Example 13.3 A chain settling down

A two-state chain has transition matrix with columns (0.8, 0.2) and (0.4, 0.6). Start it from (0, 1).

After one step the distribution is (0.4, 0.6); after two, (0.56, 0.44); after three, (0.624, 0.376).

The stationary distribution solves $0.2x = 0.4y$, so it is (2/3, 1/3). The first entry's distance from 2/3 — 0.667, 0.267, 0.107, 0.043 — shrinks by a factor of 0.4 each step. That factor is the second eigenvalue: the trace is 1.4 and one eigenvalue is 1, so the other is 0.4. Figure 13.4 shows the same convergence from three different starts.



whatever the starting distribution, repeated multiplication by the transition matrix settles on the eigenvector for eigenvalue 1; the leftover shrinks by the second eigenvalue, 0.4, each step.

FIGURE 13.4 The two-state chain of Example 13.3 from three different starting distributions. Every start converges to the stationary distribution $(2/3, 1/3)$, and the distance to it shrinks by the second eigenvalue, 0.4, at every step.

Key ideas

- An eigenvector is a direction a matrix does not turn, and its eigenvalue is the stretch along it.
- Eigenvalues solve $\det(A - \lambda I) = 0$; they add up to the trace and multiply to the determinant.
- With n independent eigenvectors, $A = V\Lambda V^{-1}$, and powers of A become powers of numbers.
- Repeated multiplication swings any vector toward the dominant eigenvector, at the rate λ_2/λ_1 per step.
- PageRank and the stability of repeated linear processes are both questions about the largest eigenvalue.

Exercises

1. Find the eigenvalues and eigenvectors of the matrix with rows $(4, 1)$ and $(2, 3)$, and check that their sum and product are the trace and the determinant.
2. Show that the eigenvalues of a triangular matrix are its diagonal entries.
3. Use diagonalisation to compute A^{10} for the matrix of Example 13.1, and describe what A^{10} does to $(1, 0)$.
4. Show that a projection matrix can only have eigenvalues 0 and 1, and identify the eigenvectors of each.
5. A Markov chain has transition matrix with columns $(0.9, 0.1)$ and $(0.5, 0.5)$. Find its stationary distribution.
6. Find the eigenvalues and eigenvectors of the matrix with rows $(0, 1)$ and $(1, 0)$.
7. If $A\mathbf{v} = 3\mathbf{v}$, what are $A^2\mathbf{v}$ and $A^{-1}\mathbf{v}$?
8. Find the eigenvalues of the matrix with rows $(0.5, 0.4)$ and $(0, 0.9)$, and describe the long-run behaviour of $\mathbf{x}_{k+1} = A\mathbf{x}_k$.

CHAPTER FOURTEEN

14 Symmetric Matrices and the Spectral Theorem

General matrices can have complex eigenvalues, too few eigenvectors, and eigenvectors that meet at awkward angles. Symmetric matrices have none of these problems. Their eigenvalues are real, their eigenvectors are perpendicular, and they can always be diagonalised by a rotation. The theorem that says so is one of the most useful in mathematics, and it applies to precisely the matrices machine learning cannot avoid: covariances, Hessians, kernel matrices and graph Laplacians.

14.1 The spectral theorem

Theorem 14.1 Spectral theorem

If S is a real symmetric matrix ($S^T = S$), then all its eigenvalues are real, eigenvectors for different eigenvalues are orthogonal, and $S = Q\Lambda Q^T$ for an orthogonal matrix Q whose columns are orthonormal eigenvectors.

Proof

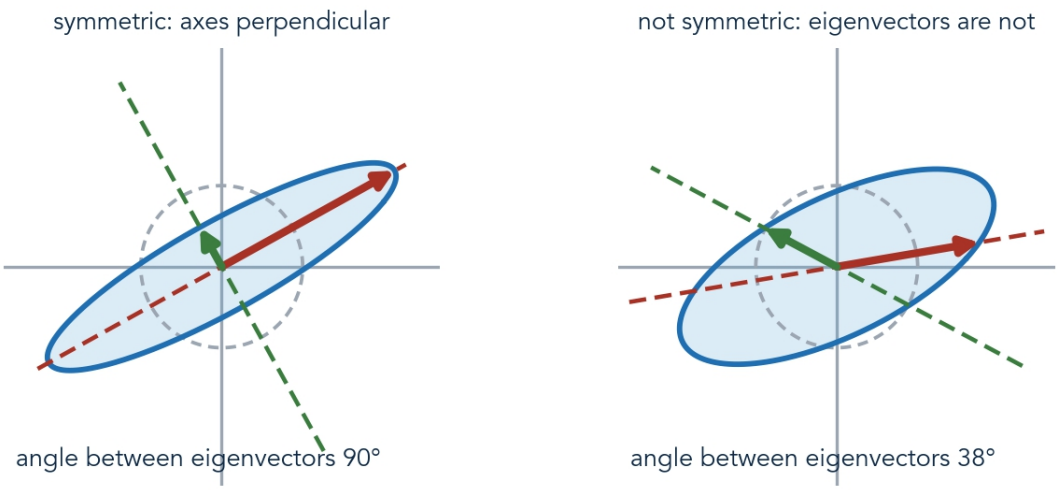
Orthogonality: suppose $S\mathbf{u} = \lambda\mathbf{u}$ and $S\mathbf{v} = \mu\mathbf{v}$ with $\lambda \neq \mu$. Then $\lambda\,\mathbf{u}\cdot\mathbf{v} = (S\mathbf{u})\cdot\mathbf{v} = \mathbf{u}^T S^T\mathbf{v} = \mathbf{u}^T S\mathbf{v} = \mu\,\mathbf{u}\cdot\mathbf{v}$, so $(\lambda - \mu)\,\mathbf{u}\cdot\mathbf{v} = 0$ and $\mathbf{u}\cdot\mathbf{v} = 0$.

Real eigenvalues: if $S\mathbf{v} = \lambda\mathbf{v}$ with complex $\mathbf{v}$, then $\mathbf{v}^T S\mathbf{v} = \lambda\,\mathbf{v}^T\mathbf{v}$, and the left side equals its own complex conjugate because S is real and symmetric, while $\mathbf{v}^T\mathbf{v} > 0$ is real. So λ is real.

A full orthonormal eigenbasis: take one unit eigenvector; symmetry makes the subspace perpendicular to it invariant under S, so the argument repeats inside that smaller space until the eigenvectors fill all n dimensions. ■

$$S = Q\Lambda Q^T = \lambda_1\mathbf{q}_1\mathbf{q}_1^T + \lambda_2\mathbf{q}_2\mathbf{q}_2^T + \cdots + \lambda_n\mathbf{q}_n\mathbf{q}_n^T \tag{14.1}$$

The second form is worth staring at. Each $\mathbf{q}_i\mathbf{q}_i^T$ is the projection onto one eigenvector direction (Theorem 10.3), so a symmetric matrix is a weighted sum of perpendicular projections: split the input along perpendicular axes, scale each piece by its eigenvalue, add them back.



on the left the eigenvectors are the axes of the ellipse and meet at a right angle. On the right they are neither: only symmetric matrices get a perpendicular eigenbasis for free.

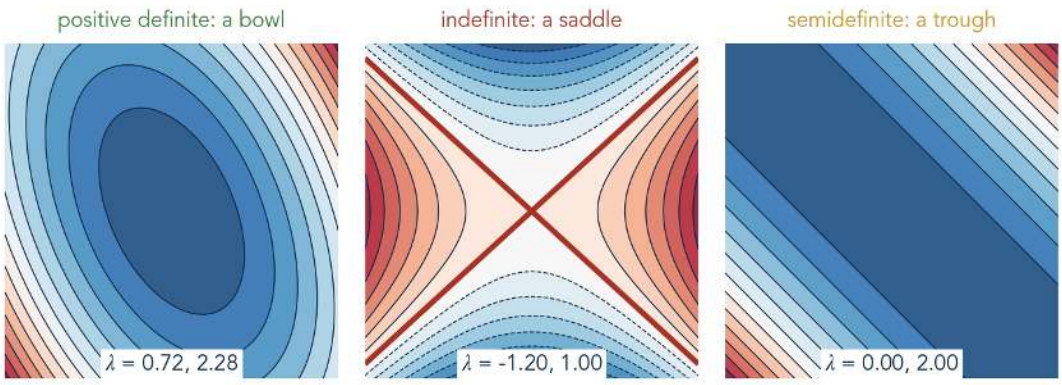
FIGURE 14.1 Left: a symmetric matrix carries the unit circle to an ellipse whose axes are its eigenvectors, meeting at a right angle. Right: a non-symmetric matrix has eigenvectors that are neither perpendicular nor the axes of its ellipse.

> **For a symmetric matrix the eigenvectors and the axes of the ellipse coincide. That coincidence is what makes symmetric matrices easy: their action is a rotation to perpendicular axes, a stretch along each, and the rotation back — no shearing, no hidden turning, nothing that cannot be drawn.**

14.2 Quadratic forms

A symmetric matrix defines a function of a vector, the quadratic form $\mathbf{x}^T S\mathbf{x}$, and its shape is decided entirely by the signs of the eigenvalues. In the eigenbasis, with coordinates $y_i = \mathbf{q}_i \cdot \mathbf{x}$, the cross terms disappear.

$$\mathbf{x}^T S\mathbf{x} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \cdots + \lambda_n y_n^2, \qquad y_i = \mathbf{q}_i \cdot \mathbf{x} \tag{14.2}$$



contours of $\mathbf{x}^T A\mathbf{x}$. The signs of the eigenvalues decide the shape: all positive, a minimum; mixed, a saddle; a zero, a flat valley. Chapter 19 reads the Hessian of a loss exactly this way.

FIGURE 14.2 Contours of three quadratic forms. All eigenvalues positive: a bowl with a single minimum. Eigenvalues of both signs: a saddle, curving up one way and down another. A zero eigenvalue: a trough, flat along one direction.

Definition 14.1 Definiteness

A symmetric matrix S is positive definite if $\mathbf{x}^T S\mathbf{x} > 0$ for every nonzero $\mathbf{x}$, and positive semidefinite if $\mathbf{x}^T S\mathbf{x} \geq 0$. Equivalently, all its eigenvalues are positive, or non-negative.

TABLE 14.1 Five tests for positive definiteness. Any one of them implies all the others for a symmetric matrix.

test	condition	convenient when
energy	$\mathbf{x}^T S\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$	proving it abstractly
eigenvalues	all $\lambda_i > 0$	the spectrum is known
pivots	all pivots of elimination positive	computing by hand
Cholesky	$S = LL^T$ with positive diagonal	computing on a machine
Gram form	$S = A^T A$ with independent columns	S is built from data

Worked Example 14.1 Is it positive definite?

Test the matrix with rows (2, 1) and (1, 2) in three ways.

Eigenvalues: from Example 13.1 they are 3 and 1, both positive.

Pivots: elimination gives 2 and then $2 - 1/2 = 3/2$, both positive.

Energy: $\mathbf{x}^T S\mathbf{x} = 2x_1^2 + 2x_1x_2 + 2x_2^2 = x_1^2 + x_2^2 + (x_1 + x_2)^2$, a sum of squares that vanishes only at the origin. Three routes, one answer.

The Gram form in the last row of Table 14.1 is the reason positive semidefinite matrices appear everywhere in data analysis: any matrix of the form A^TA satisfies $\mathbf{x}^TA^TA\mathbf{x} = \|A\mathbf{x}\|^2 \geq 0$. Every matrix of dot products between data vectors is of this kind.

14.3 The Rayleigh quotient

How large can a quadratic form be on unit vectors? The eigenvalues answer exactly.

Theorem 14.2 Rayleigh quotient

For symmetric S with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$, $\lambda_n \leq \mathbf{x}^TS\mathbf{x}\mathbf{x}^T\mathbf{x} \leq \lambda_1$ for every nonzero $\mathbf{x}$, with the maximum attained at the top eigenvector and the minimum at the bottom one.

Proof

In eigen-coordinates the quotient is $\sum \lambda_i y_i^2 / \sum y_i^2$, a weighted average of the eigenvalues with non-negative weights $y_i^2 / \sum y_j^2$.

A weighted average lies between the smallest and largest values averaged.

It equals λ_1 when all the weight is on the first coordinate, which means $\mathbf{x}$ is the first eigenvector. ■

The theorem turns an optimisation problem into an eigenvalue problem: "find the direction that maximises this quadratic" is "find the top eigenvector". Principal component analysis in Chapter 17 is exactly this, applied to a covariance.

Worked Example 14.2 The Rayleigh quotient at three points

For the matrix with rows $(2, 1)$ and $(1, 2)$, evaluate $\mathbf{x}^T S\mathbf{x} / \mathbf{x}^T\mathbf{x}$.

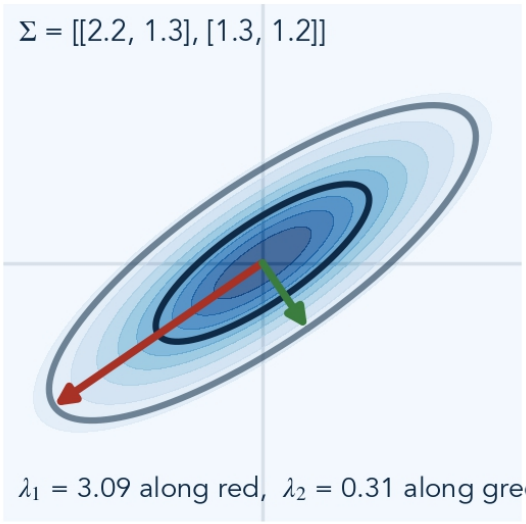
At $\mathbf{x} = (1, 0)$ the quotient is $2/1 = 2$. At $\mathbf{x} = (1, 1)$ it is $(2 + 1 + 1 + 2)/2 = 3$. At $\mathbf{x} = (1, -1)$ it is $(2 - 1 - 1 + 2)/2 = 1$.

The values 3 and 1 are the largest and smallest eigenvalues, attained at the eigenvectors, and every other direction gives something in between, as Theorem 14.2 states.

14.4 Covariance

Given data vectors with mean zero, stacked as the rows of an $m \times n$ matrix X, the covariance matrix is $C = X^T X / m$. Its (i, j) entry is the average product of features i and j, and its diagonal entries are the variances.

$$C = \tfrac{1}{m} X^T X, \qquad \operatorname{Var}(\mathbf{u} \cdot \mathbf{x}) = \mathbf{u}^T C\, \mathbf{u} \ \text{for a unit vector} \ \mathbf{u} \tag{14.3}$$



the density of two correlated measurements. Its contours are ellipses whose axes are the eigenvectors of the covariance, and each eigenvalue is the variance along its axis.

FIGURE 14.3 The density of two correlated measurements with covariance entries 2.2, 1.3 and 1.2. Its contours are ellipses whose axes are the eigenvectors of the covariance, and the eigenvalues 3.09 and 0.31 are the variances along those axes.

The second equation in 14.3 combines this chapter into one line. The variance of the data along any direction is the quadratic form of the covariance in that direction. By the Rayleigh quotient, the direction of greatest variance is the top eigenvector, and the variance there is the top eigenvalue. By the spectral theorem those directions are perpendicular. A covariance matrix is a complete description of the ellipse that summarises a cloud of data.

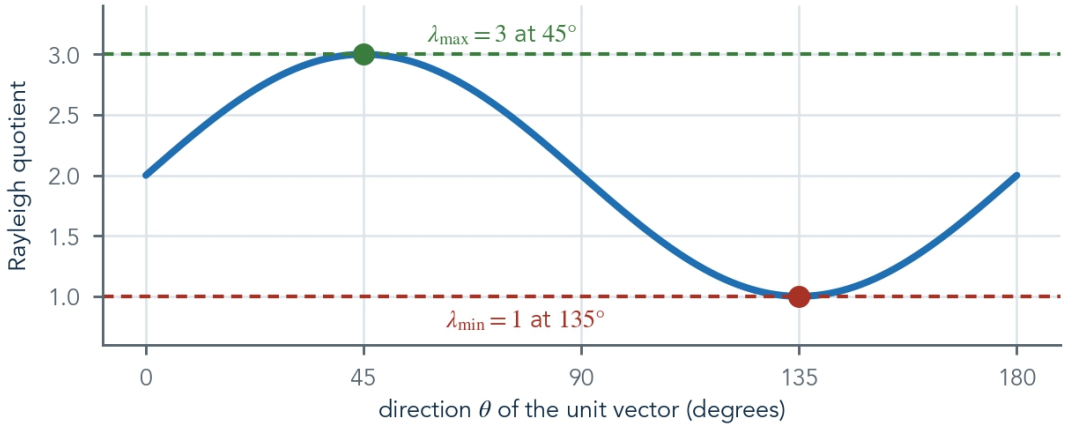
> **In machine learning.** The Hessian of a loss — the matrix of its second derivatives — is symmetric, and its eigenvalues describe the curvature of the loss surface: positive everywhere at a minimum, mixed at a saddle point, near zero along flat directions. Kernel methods rely on Gram matrices being positive semidefinite. Graph Laplacians, used in spectral clustering and in learning on graphs, are symmetric positive semidefinite, and their eigenvectors with smallest eigenvalues reveal the clusters of the graph.

Worked Example 14.3 A Cholesky factorisation by hand

Factor the matrix with rows $(4, 2)$ and $(2, 5)$ as LL^T.

The first entry of L is $\sqrt{4} = 2$, and the entry below it is $2/2 = 1$. The last entry is $\sqrt{5 - 1^2} = 2$. So L has rows $(2, 0)$ and $(1, 2)$.

Check: LL^T has rows $(4, 2)$ and $(2, 1 + 4) = (2, 5)$. Every square root was of a positive number, which is exactly the statement that the matrix is positive definite. Figure 14.4 shows a Rayleigh quotient as a function of direction.



sweeping a unit vector through every direction, the quadratic form never leaves the band between the two eigenvalues, and touches each edge exactly along an eigenvector.

FIGURE 14.4 The Rayleigh quotient of the matrix with rows $(2, 1)$ and $(1, 2)$ along the unit vector at angle θ, which works out to $2 + \sin 2\theta$. Its maximum 3 and minimum 1 are the eigenvalues, reached at $45°$ and $135°$.

Key ideas

- Symmetric matrices have real eigenvalues and perpendicular eigenvectors: $S = Q\Lambda Q^T$.
- A symmetric matrix is a weighted sum of perpendicular projections.
- The signs of the eigenvalues decide the shape of the quadratic form: a bowl, a saddle or a trough.
- The Rayleigh quotient lies between the smallest and largest eigenvalues, so maximising a quadratic form on unit vectors is an eigenvalue problem.
- A covariance matrix describes the ellipse of the data: its eigenvectors are the axes and its eigenvalues the variances along them.

Exercises

1. Diagonalise the symmetric matrix with rows (3, 1) and (1, 3) as $Q\Lambda Q^T$, and write it as a sum of two rank-one projections.
2. Decide whether the matrix with rows (1, 2) and (2, 1) is positive definite, and describe the shape of its quadratic form.
3. Show that A^TA is positive semidefinite for any A, and positive definite when A has independent columns.
4. Find the maximum of $x^2 + 4xy + y^2$ on the unit circle and the direction where it occurs.
5. A covariance matrix has eigenvalues 9 and 1. What are the standard deviations along the principal axes, and what is the total variance?
6. Is the matrix with rows (4, 2) and (2, 1) positive definite?
7. Show that the diagonal entries of a positive definite matrix are positive.
8. Write the quadratic form of the matrix with rows (1, 3) and (3, 1) in eigen-coordinates, and name its shape.

CHAPTER FIFTEEN

15 The Singular Value Decomposition

Everything in this book has been building to one theorem. Eigenvectors describe a square matrix in its own coordinates, but only some matrices have enough of them, and only symmetric ones have perpendicular ones. The singular value decomposition does the same job for every matrix, of every shape, with no exceptions and no complex numbers: every matrix is a rotation, followed by a stretch along perpendicular axes, followed by another rotation. Rank, norm, condition number, pseudoinverse and best approximation are all read off the result.

15.1 The theorem

Theorem 15.1 Singular value decomposition

Every real $m \times n$ matrix A can be written $A = U\Sigma V^T$, where U is an $m \times m$ orthogonal matrix, V is an $n \times n$ orthogonal matrix, and Σ is $m \times n$ with non-negative numbers $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ on its diagonal and zeros elsewhere.

Proof

The matrix A^TA is symmetric and positive semidefinite, so by the spectral theorem it has orthonormal eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ with eigenvalues $\sigma_1^2 \geq \cdots \geq \sigma_n^2 \geq 0$.

For each $\sigma_i > 0$ set $\mathbf{u}_i = A\mathbf{v}_i/\sigma_i$. These are orthonormal: $\mathbf{u}_i \cdot \mathbf{u}_j = \mathbf{v}_i^T A^T A \mathbf{v}_j/(\sigma_i\sigma_j) = \sigma_j^2\, \mathbf{v}_i \cdot \mathbf{v}_j/(\sigma_i\sigma_j)$, which is 1 if $i = j$ and 0 otherwise.

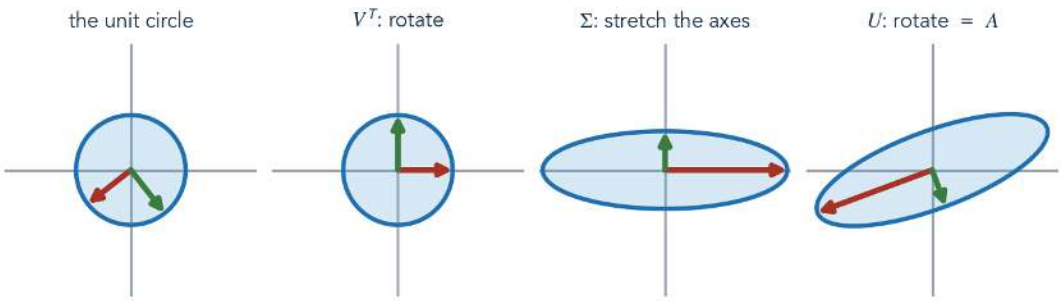
Extend the $\mathbf{u}_i$ to an orthonormal basis of $\mathbb{R}^m$. By construction $A\mathbf{v}_i = \sigma_i \mathbf{u}_i$ for every i, which in matrix form is $AV = U\Sigma$, and so $A = U\Sigma V^T$. ■

$$A = U\Sigma V^T, \qquad A\mathbf{v}_i = \sigma_i \mathbf{u}_i, \qquad A^T\mathbf{u}_i = \sigma_i \mathbf{v}_i \tag{15.1}$$

The proof also delivers the one-line fact promised in Chapter 12: the singular values of A are the square roots of the eigenvalues of A^TA, so the singular values of A^TA are σ_i^2, and $(A^TA) = (A)^2$.

15.2 Rotate, stretch, rotate

Read $A\mathbf{x} = U\Sigma V^T\mathbf{x}$ from right to left, as a composition of three maps.



the red and green arrows start as the right singular vectors. V^T turns them onto the axes, Σ stretches by 2.23 and 0.71, and U turns the result into place. Nothing else ever happens.

FIGURE 15.1 The unit circle under V^T, then Σ, then U. The first rotation turns the right singular vectors (red, green) onto the coordinate axes; Σ stretches along the axes by σ_1 and σ_2; the second rotation turns the stretched axes into place. The result is the ellipse of Chapter 12.

> **Every linear map, whatever its entries, is a rotation (or reflection), an axis-aligned stretch, and a rotation. There is no other kind of linear behaviour. Shears, projections and every "strange" matrix are this sequence with particular angles and stretches, and anything a matrix does can be described by naming its two rotations and its list of stretches.**

This is the theorem behind the one picture. The right singular vectors $\mathbf{v}_i$ are the directions in the input that end up on the axes of the ellipse; the left singular vectors $\mathbf{u}_i$ are the axes themselves; the singular values are the half-axis lengths.

Worked Example 15.1 An SVD by hand

Find the SVD of the matrix with rows (3, 0) and (4, 5).

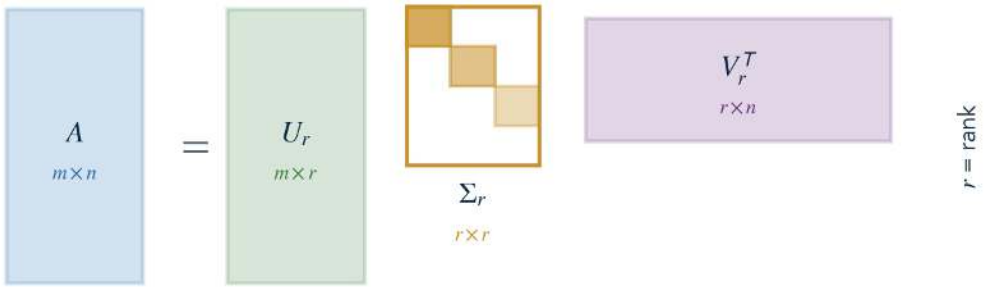
A^TA has rows (25, 20) and (20, 25), with eigenvalues 45 and 5 and eigenvectors $(1, 1)/\sqrt{2}$ and $(1, -1)/\sqrt{2}$. So $\sigma_1 = \sqrt{45} = 3\sqrt{5} \approx 6.71$ and $\sigma_2 = \sqrt{5} \approx 2.24$.

$\mathbf{u}_1 = A\mathbf{v}_1/\sigma_1 = (3, 9)/(\sqrt{2}\cdot 3\sqrt{5}) = (1, 3)/\sqrt{10}$, and $\mathbf{u}_2 = A\mathbf{v}_2/\sigma_2 = (3, -1)/(\sqrt{2}\cdot\sqrt{5}) = (3, -1)/\sqrt{10}$. These are perpendicular, as the theorem guarantees.

Checks: $\sigma_1\sigma_2 = 15 = |\det A|$, and ${\sigma_1}^2 + {\sigma_2}^2 = 50$ equals the sum of the squares of the entries, $9 + 0 + 16 + 25$.

15.3 Shapes, and the thin SVD

Only the first r singular values of a rank-r matrix are nonzero, and the columns of U and V beyond the r-th are multiplied by zeros. Dropping them loses nothing.



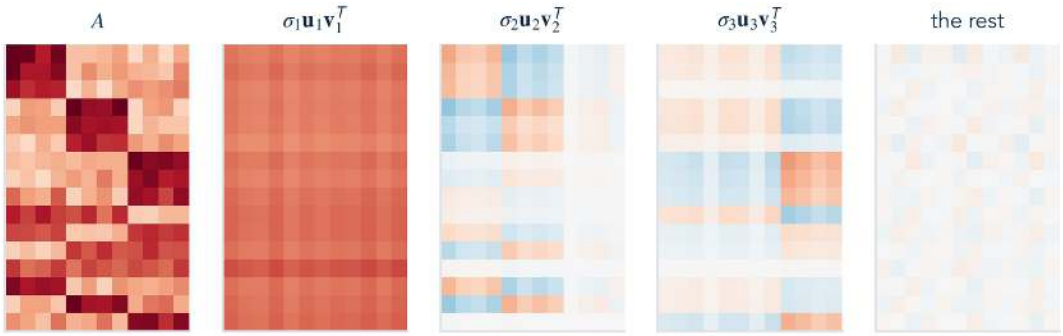
the thin SVD keeps only the r directions that carry any stretch at all. Its storage is $r(m+n+1)$ numbers against mn -- the whole case for low-rank structure in one line.

FIGURE 15.2 The thin SVD of a tall matrix of rank r: $A = U_r\Sigma_r V_r^T$ with U_r of size $m \times r$, Σ_r of size $r \times r$ and V_r of size $n \times r$.

$$A = U_r\Sigma_r V_r^T = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \cdots + \sigma_r\mathbf{u}_r\mathbf{v}_r^T$$

(15.2)

The sum form is the most important way to write the SVD. Every matrix is a sum of rank-one layers — outer products, as in Section 6.4 — each with a strength σ_i, and the layers are ordered from strongest to weakest.



a 16 x 12 table of ratings, and the first three rank-one layers of its SVD. Each layer is one taste, shared by rows and columns; what remains after three is small and has no pattern.

FIGURE 15.3 A table of ratings (left), its first three rank-one layers, and what remains. Each layer is a single pattern shared across rows and columns; the remainder after three layers is small and has no structure.

15.4 Everything read off the SVD

TABLE 15.1 Quantities that the singular value decomposition delivers directly. Most would otherwise need a separate method.

quantity	from the SVD	meaning
rank	number of $\sigma_i > 0$	stretches that are not zero
column space	first r columns of U	the ellipse's axes
null space	last $n - r$ columns of V	inputs sent to zero
row space	first r columns of V	inputs the map uses
spectral norm $\|A\|_2$	σ_1	the largest stretch
Frobenius norm $\|A\|_F$	$\sqrt{\sigma_1^2 + \cdots + \sigma_r^2}$	overall size
condition number	σ_1 / σ_r	how close to flattening
size of det A (square)	$\sigma_1 \sigma_2 \cdots \sigma_n$	volume factor
pseudoinverse A^+	$V \Sigma^+ U^T$	the best possible undoing

The four fundamental subspaces of Chapter 7 appear in the table ready-made, with orthonormal bases. The SVD is Figure 7.2 made concrete: V splits the input space into row space and null space along perpendicular axes, Σ carries the row space onto the column space stretch by stretch, and U lays the column space and left null space out in the output.

Worked Example 15.2 Reading a matrix from its singular values

A 3×3 matrix has singular values 5, 3 and 0.

Its rank is 2, so its null space and its left null space each have dimension 1. Its spectral norm is 5 and its Frobenius norm is $\sqrt{25 + 9} \approx 5.83$.

Its determinant is 0 and it has no inverse; its pseudoinverse has singular values $1/5$, $1/3$ and 0.

15.5 The pseudoinverse

A matrix that is not square or not invertible cannot be undone, but the SVD says exactly how close one can come: invert every nonzero stretch and leave the zero stretches at zero.

$$A^{+} = V\Sigma^{+}U^{T}, \qquad \Sigma^{+} = \operatorname{diag}\left(\tfrac{1}{\sigma_1}, \ldots, \tfrac{1}{\sigma_r}, 0, \ldots, 0\right), \qquad \mathbf{x}^{+} = A^{+}\mathbf{b} \tag{15.3}$$

Theorem 15.2 What the pseudoinverse solves

For any A and $\mathbf{b}$, $\mathbf{x}^{+} = A^{+}\mathbf{b}$ is the least-squares solution of $A\mathbf{x} \approx \mathbf{b}$ of smallest length.

Proof

In the coordinates given by U and V, the problem decouples into $\sigma_i y_i \approx c_i$ for $i \le r$, and $0 \cdot y_i \approx c_i$ for the rest, where $\mathbf{y} = V^{T}\mathbf{x}$ and $\mathbf{c} = U^{T}\mathbf{b}$.

The best choice for $i \le r$ is $y_i = c_i/\sigma_i$; the remaining y_i do not affect the fit, and setting them to zero minimises $\|\mathbf{y}\| = \|\mathbf{x}\|$.

That choice is exactly $\mathbf{y} = \Sigma^{+}\mathbf{c}$, so $\mathbf{x} = V\Sigma^{+}U^{T}\mathbf{b}$. ■

The pseudoinverse unifies Chapters 9, 11 and 7: for an invertible matrix it is the inverse, for a tall one it gives least squares, and for a wide one it gives the minimum-norm solution — the same solution that gradient descent from zero finds (Theorem 11.2).

One caution carries over from Chapter 12. A singular value that is tiny but not zero is inverted into an enormous one, and $A^{+}\mathbf{b}$ amplifies noise along that direction. In practice singular values below a threshold are treated as zero, which is the truncated SVD — and that idea, pushed further, is the subject of Chapter 16.

In machine learning. The SVD is the computational core of principal component analysis (Chapter 17), of latent semantic analysis and the recommender systems of Chapter 16, and of

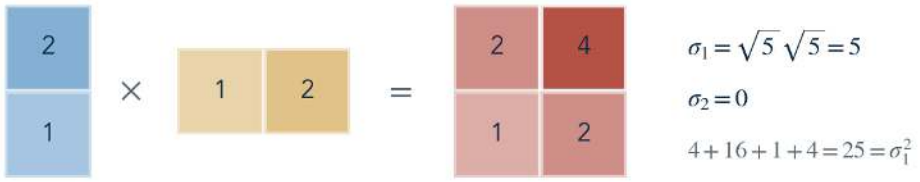
every analysis of the weight matrices of trained networks (Chapter 20). The spectral norm σ_1 also controls how much a layer can amplify a small change in its input, which is why constraining it — spectral normalisation — is used to make models smoother and more stable.

Worked Example 15.3 The SVD of a rank-one matrix

The matrix with rows (2, 4) and (1, 2) is the outer product of (2, 1) and (1, 2).

Writing it as $\sigma_1 \mathbf{u}_1 \mathbf{v}_1{}^T$ with unit vectors gives $\mathbf{u}_1 = (2, 1)/\sqrt{5}$, $\mathbf{v}_1 = (1, 2)/\sqrt{5}$ and $\sigma_1 = \sqrt{5} \times \sqrt{5} = 5$; the second singular value is 0.

Check: the squared entries sum to $4 + 16 + 1 + 4 = 25 = \sigma_1{}^2$. A rank-one matrix has exactly one nonzero stretch, and its singular vectors are the directions of the two vectors it was built from. Figure 15.4 lays the product out cell by cell.



every row of the product is a multiple of the row (1, 2) and every column a multiple of the column (2, 1). One direction in, one direction out: a single nonzero singular value.

FIGURE 15.4 The rank-one matrix of Example 15.3 as a column times a row. Its single singular value, 5, is the product of the lengths of the two vectors.

Key ideas

- Every matrix factors as $U\Sigma V^T$: a rotation, a stretch along perpendicular axes, and a rotation.
- The singular values are the square roots of the eigenvalues of A^TA.
- Every matrix is a sum of rank-one layers $\sigma_i \mathbf{u}_i \mathbf{v}_i^T$, ordered from strongest to weakest.
- Rank, norms, condition number, determinant and the four subspaces can all be read directly off the SVD.
- The pseudoinverse inverts the nonzero stretches, and it returns the minimum-norm least-squares solution.

Exercises

1. Find the singular values of the matrix with rows (1, 1) and (1, 1), and its rank-one form $\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$.
2. Show that the singular values of an orthogonal matrix are all 1, and that those of a symmetric positive semidefinite matrix are its eigenvalues.
3. For the matrix of Example 15.1 compute $\|A\|_2$, $\|A\|_F$ and (A).
4. Find the pseudoinverse of the 3×1 matrix with entries 1, 2, 2, and use it to solve $A\,x \approx (1, 1, 1)$ in the least-squares sense.
5. Explain, using Equation 15.2, why a matrix of rank r can be stored with $r(m + n + 1)$ numbers.
6. Find an SVD of the diagonal matrix with entries -2 and 1.
7. Show that $\|A\|_F^2$ equals the sum of the squared singular values.
8. What is the pseudoinverse of the projection with diagonal entries 1 and 0?

CHAPTER SIXTEEN

16 Low-Rank Approximation

The SVD writes a matrix as a sum of rank-one layers ordered from strongest to weakest. The obvious question is what happens if the weak layers are simply thrown away. The answer is one of the most consequential theorems in applied mathematics: the result is not just a good approximation but the best one possible, and the error is exactly the size of what was discarded. That theorem is behind data compression, recommender systems, noise removal, and the standard way large models are adapted to new tasks.

16.1 The Eckart–Young theorem

Definition 16.1 Truncated SVD

For $k \le r$, the rank-k truncation of $A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ is $A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$, keeping the k strongest layers.

Theorem 16.1 Eckart–Young

Among all matrices of rank at most k, A_k is closest to A, in both the spectral norm and the Frobenius norm. The errors are $\|A - A_k\|_2 = \sigma_{k+1}$ and $\|A - A_k\|_F = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}$.

Proof

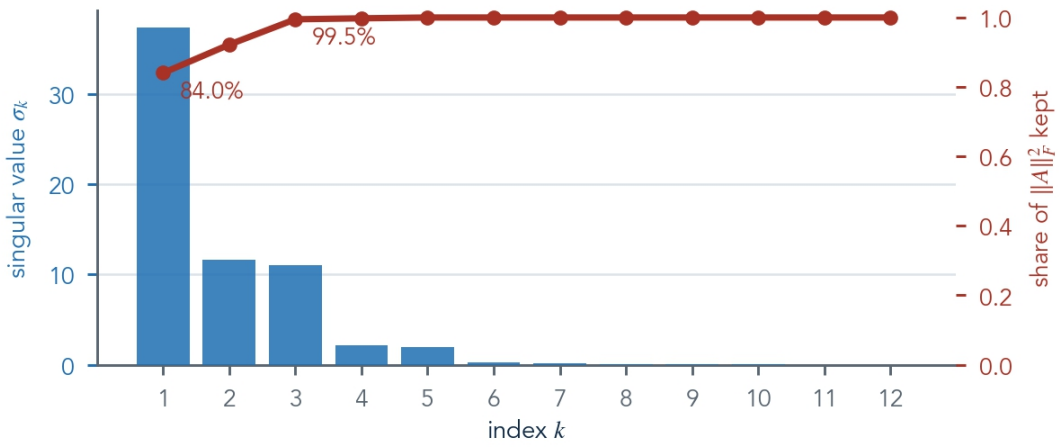
The error $A - A_k = \sum_{i>k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ has singular values $\sigma_{k+1}, \ldots, \sigma_r$, which gives both formulas for A_k.

Now let B be any matrix of rank at most k. Its null space has dimension at least $n - k$, and the span of $\mathbf{v}_1, \ldots, \mathbf{v}_{k+1}$ has dimension $k+1$, so the two subspaces share a unit vector $\mathbf{w}$.

Then $\|(A - B)\mathbf{w}\| = \|A\mathbf{w}\| \geq \sigma_{k+1}$, because $\mathbf{w}$ lies in the span of the first $k+1$ right singular vectors, where A stretches by at least σ_{k+1}. So $\|A - B\|_2 \geq \sigma_{k+1}$, and no rank-$k$ matrix beats A_k. The Frobenius case follows by applying the same argument to each singular value in turn. ■

$$\min_{\operatorname{rank}(B)\leq k} \|A-B\|_F^2 = \|A - A_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2, \qquad \frac{\|A_k\|_F^2}{\|A\|_F^2} = \frac{\sigma_1^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \cdots + \sigma_r^2} \tag{16.1}$$

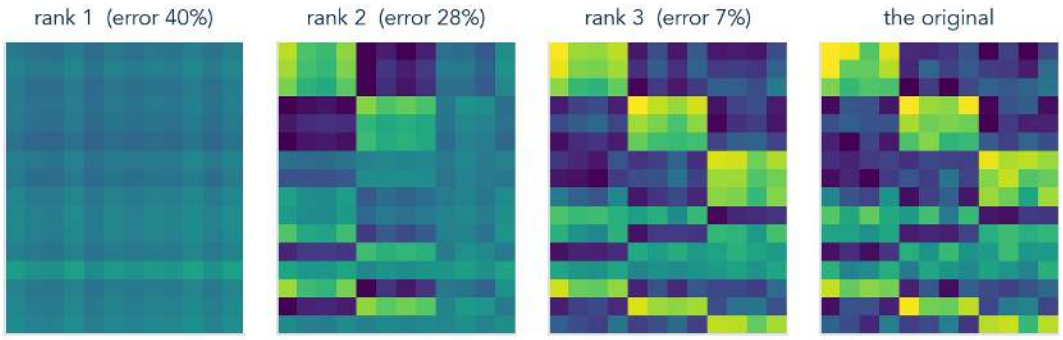
The second equation gives the quantity used in practice: the share of the matrix's squared size, often called its energy, that the first k layers retain. If the singular values fall quickly, a small k keeps almost everything.



the singular values of the ratings table. The first one carries the average level, the next two the tastes; by $k = 3$ almost all of the squared size of the matrix is accounted for.

FIGURE 16.1 The singular values of a 16×12 table of ratings (bars) and the share of its energy kept by the first k layers (line). One layer keeps 94%; three keep 99.5%.

16.2 What a low-rank approximation looks like



Eckart-Young: truncating the SVD gives the closest matrix of each rank. Rank 3 stores 3 x (16 + 12 + 1) = 87 numbers instead of 192 and is hard to tell from the original.

FIGURE 16.2 Best approximations to the ratings table of rank 1, 2 and 3, and the original. Rank 1 captures only the average level of each row and column; rank 3 is hard to tell from the original while storing fewer than half as many numbers.

Worked Example 16.1 The arithmetic of compression

A 1000×800 matrix has singular values that make the first 50 layers keep 98% of its energy. How much storage does the rank-50 approximation need, and what is its relative error?

Storing A_{50} as U_{50}, Σ_{50} and V_{50} takes $50 \times (1000 + 800 + 1) = 90{,}050$ numbers, against 800,000 for the full matrix — about 11%.

By Equation 16.1 the relative Frobenius error is $\sqrt{1 - 0.98} \approx 0.14$, so the approximation is within 14% of the matrix in overall size, while using a ninth of the storage.

Low rank is the mathematical form of "a few underlying factors explain most of what we see". When a matrix of observations has rapidly falling singular values, the observations are the combined result of a small number of hidden patterns, and the truncated SVD recovers those patterns optimally. Much of what is called structure in data is, precisely, fast singular-value decay.

16.3 Recommender systems

A table of ratings, with one row per user and one column per item, is the classic example. People's tastes are not arbitrary: a few factors — genre, mood, style — account for most of their preferences. If each user is described by k numbers and each item by k numbers, and a rating is roughly their dot product, then the rating matrix is approximately UV^T with U of size (users) $\times\, k$ and V of size (items) $\times\, k$ — a matrix of rank k.

$$R \approx UV^T, \qquad r_{ij} \approx \mathbf{u}_i \cdot \mathbf{v}_j, \qquad \min_{U,V} \sum_{(i,j)\ \text{observed}} (r_{ij} - \mathbf{u}_i \cdot \mathbf{v}_j)^2 + \lambda\left(\|U\|_F^2 + \|V\|_F^2\right) \tag{16.2}$$

In real systems most ratings are missing, so the SVD cannot be applied directly; instead U and V are fitted to the observed entries only, with a ridge penalty as in Chapter 11. The fitted low-rank matrix

then fills in the missing entries, and those predictions are the recommendations. The principle is Eckart–Young's: a low-rank model is the most economical explanation of the table.

16.4 Computing it at scale

The full SVD of an $m \times n$ matrix costs on the order of $mn\min(m, n)$ operations, which is prohibitive for large matrices when only the top k layers are wanted. Randomised methods, analysed thoroughly by Halko, Martinsson and Tropp, avoid it. Multiply A by a small random matrix Ω of size $n \times (k + p)$, with a few extra columns p for safety. Because the top singular directions dominate, the columns of $A\Omega$ span almost exactly the top-k column space. Orthonormalise them with QR to get Q, compute the SVD of the small matrix $Q^{T}A$, and map back.

$$Y = A\Omega, \quad Y = QR, \quad B = Q^{T}A = \tilde{U}\Sigma V^{T}, \quad A \approx (Q\tilde{U})\Sigma V^{T} \tag{16.3}$$

The expensive work is a few products with A, which can exploit sparsity and parallel hardware, and one SVD of a $(k + p) \times n$ matrix. The Johnson–Lindenstrauss geometry of Chapter 4 is why a random Ω works: random directions capture the dominant subspace with overwhelming probability.

16.5 Low-rank adapters

The most important use of low rank in machine learning in the last few years concerns adapting a large trained model to a new task. The model's weight matrices W_0, of size $d \times d$ with d in the thousands, are kept fixed. Instead of learning a full update ΔW with d^2 entries, one learns a low-rank update.

$$W = W_0 + \Delta W = W_0 + BA, \qquad B \in \mathbb{R}^{d \times r},\ A \in \mathbb{R}^{r \times d},\ r \ll d \tag{16.4}$$

This is the low-rank adapter introduced by Hu and colleagues in 2021, and by 2025 it had become the default way of fine-tuning large

models. Its arithmetic is simple and it is worth doing exactly.

Theorem 16.2 When a low-rank update pays

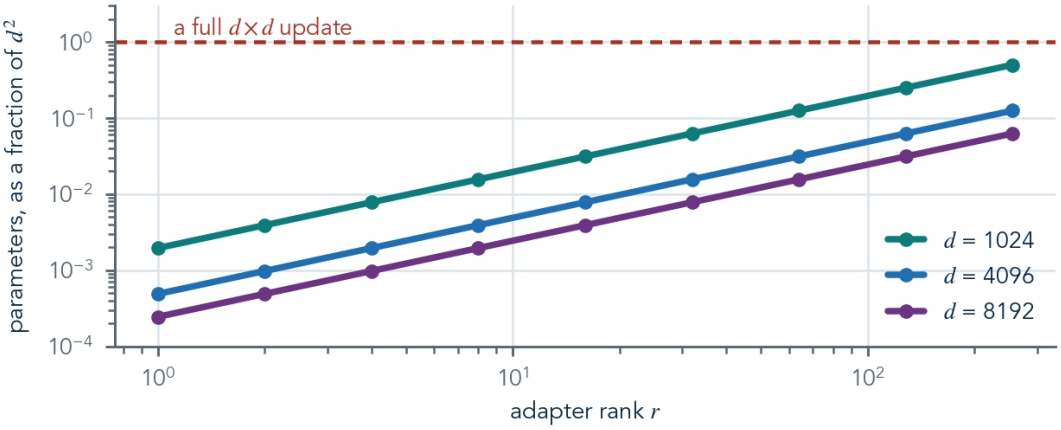
A rank-r update of a $d \times d$ matrix has $2dr$ parameters, a fraction $2r/d$ of a full update. Fitting a full update and truncating it to rank r loses exactly the energy $\sum_{i>r}\sigma_i^2$ of its discarded singular values; so the adapter loses nothing essential precisely when the useful update has rapidly falling singular values.

Proof

The factors B and A contain $dr + rd = 2dr$ numbers, against d^2.

The best rank-r approximation to any intended update ΔW^* is its truncated SVD (Theorem 16.1), whose squared error is the sum of the discarded σ_i^2.

Every matrix BA has rank at most r, so no adapter of rank r can do better than that truncation, and one that matches the truncation achieves it. ■



a low-rank update BA with B of size $d \times r$ and A of size $r \times d$ has $2dr$ parameters. At d = 4096 and r = 16 that is 0.8% of the full update -- a factor of 128 fewer numbers to learn and store.

FIGURE 16.3 The parameter count of a rank-r update as a fraction of a full $d \times d$ update. At $d = 4096$ and $r = 16$ the adapter has 0.8% of the parameters.

Worked Example 16.2 Counting an adapter

A model has 32 layers, each with four 4096×4096 attention weight matrices, and an adapter of rank 16 is attached to each.

Each adapter has $2 \times 4096 \times 16 = 131{,}072$ parameters, against $4096^2 \approx 16.8$ million for a full update: a factor of 128.

Over $32 \times 4 = 128$ matrices the adapters total about 16.8 million parameters — the size of one full matrix — while a full update of all of them would need about 2.1 billion. Many different adapters can be stored and swapped for the price of one copy of the model.

The theorem says why the method works when it works: the change needed to specialise a large model tends to be concentrated in a few directions. Later refinements keep the same algebra. Weight-decomposed variants, proposed in 2024, split each weight matrix into a magnitude for every column and a unit direction, and adapt the direction with a low-rank update — a decomposition in the spirit of the SVD's split between stretches and rotations.

In machine learning. Low rank appears on both sides of a model. On the input side, embeddings and the attention mechanism of Chapter 20 compute bilinear forms of limited rank. On the training side, weight updates and gradients are often close to low rank, which is exploited to save memory by storing them in factored form. Checking the singular-value decay of a matrix before deciding how to store, compress or adapt it is one of the most useful habits this book can recommend.

Worked Example 16.3 The best rank-one approximation of Example 15.1

For the matrix with rows $(3, 0)$ and $(4, 5)$, $\sigma_1 = 3\sqrt{5}$, $\mathbf{u}_1 = (1, 3)/\sqrt{10}$ and $\mathbf{v}_1 = (1, 1)/\sqrt{2}$.

$A_1 = \sigma_1 \mathbf{u}_1 \mathbf{v}_1{}^T$ is $3\sqrt{5}/\sqrt{20} = 1.5$ times the matrix with rows $(1, 1)$ and $(3, 3)$, which is the matrix with rows $(1.5, 1.5)$ and $(4.5, 4.5)$.

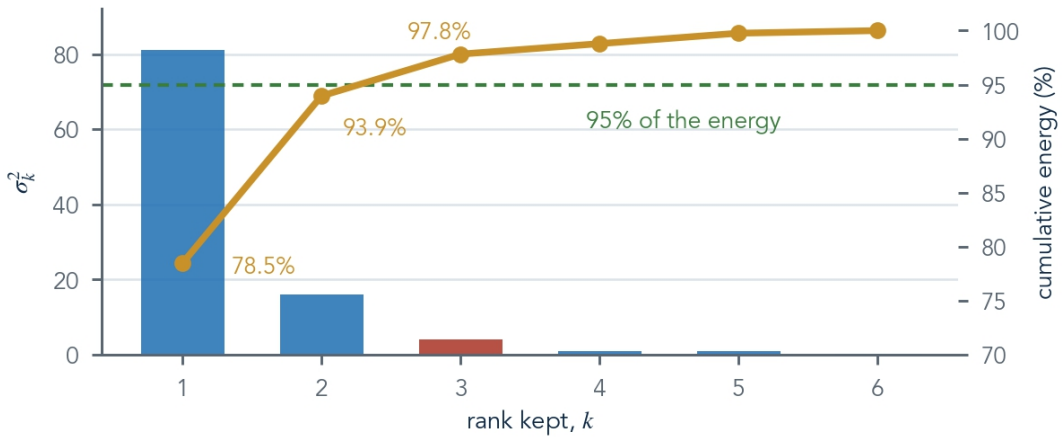
The error $A - A_1$ has rows $(1.5, -1.5)$ and $(-0.5, 0.5)$. Its squared Frobenius norm is $2.25 + 2.25 + 0.25 + 0.25 = 5 = \sigma_2{}^2$, exactly as Eckart–Young predicts.

Worked Example 16.4 Choosing a rank from the energy

A matrix has singular values 9, 4, 2, 1, 1 and 0.5. Which rank keeps 95% of its energy?

The squares are 81, 16, 4, 1, 1 and 0.25, totalling 103.25. Rank one keeps $81/103.25 \approx 78.5$, rank two keeps $97/103.25 \approx 94.0$, and rank three keeps $101/103.25 \approx 97.8$.

Rank three is the smallest that reaches 95%, and by Eckart–Young its relative error in the Frobenius norm is $\sqrt{1 - 0.978} \approx 0.15$. Figure 16.4 shows the same arithmetic as a picture.



bars: the energy σ_k^2 in each rank-one piece. Line: the share kept by the best rank-k approximation. Rank three (red) is the first to pass 95%, keeping 97.8%.

FIGURE 16.4 The singular values of Example 16.4: energy in each rank-one piece (bars) and the cumulative share kept (line). Rank three is the smallest to keep 95%.

Key ideas

- Truncating the SVD gives the best approximation of each rank, in both the spectral and the Frobenius norm.
- The error of the rank-k approximation is the size of the singular values it discards.
- Fast singular-value decay is the mathematical form of a few hidden factors explaining the data.
- Randomised methods find the leading singular directions with a handful of matrix products.
- A rank-r adapter uses $2dr$ parameters and loses only the energy of the singular values it cannot represent.

Exercises

1. A matrix has singular values 10, 5, 2, 1, 0.5. Find the spectral and Frobenius errors of its best rank-2 approximation, and the share of energy kept.
2. Explain why the best rank-1 approximation of a matrix whose rows are all multiples of one vector is exact.
3. How many numbers are needed to store the rank-20 truncation of a 5000×3000 matrix, and what fraction of the full matrix is that?
4. For $d = 1024$, find the adapter rank at which the parameters equal 10% of a full update.
5. Why must the random matrix Ω in Equation 16.3 have a few more columns than the target rank k? Answer in terms of the chance that a random direction misses part of the dominant subspace.
6. What share of the energy does the best rank-one approximation of the diagonal matrix with entries 4 and 3 keep?
7. A 1000×1000 weight matrix receives a rank-8 adapter. How many parameters is that, and what fraction of a full update?
8. Explain why truncating the SVD tends to remove noise.

CHAPTER SEVENTEEN

17 Principal Component Analysis

A table of measurements with fifty columns is a cloud of points in fifty dimensions, and nobody can look at it. Principal component analysis finds the few directions along which the cloud actually varies, rotates the data onto them, and discards the rest. It is the covariance matrix of Chapter 14 read through the singular value decomposition of Chapter 15, and it is the most widely used method in data analysis for a reason: when it applies, it is optimal, and it tells you exactly how much it has thrown away.

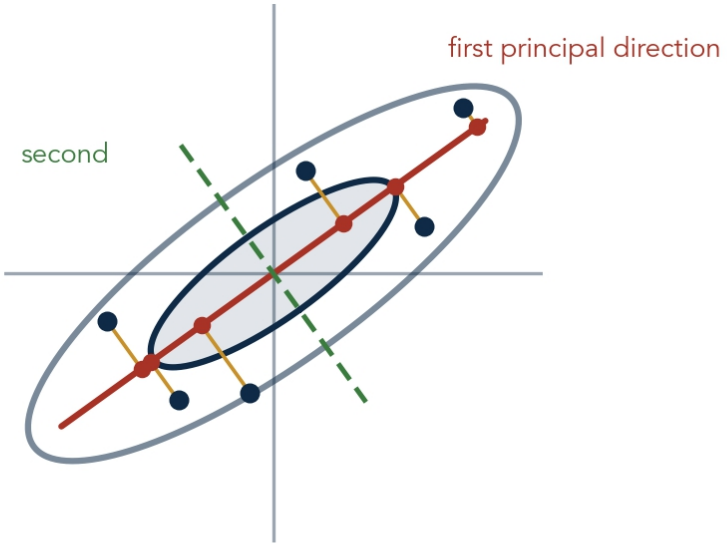
17.1 The problem, stated two ways

Take m data vectors in $\mathbb{R}^n$, centred so that their mean is zero, and stacked as the rows of X. We want to describe each vector by $k < n$ numbers — its coordinates along k orthonormal directions — losing as little as possible.

There are two natural meanings of "losing as little as possible", and the first surprise of the subject is that they have the same answer.

$$\max_{Q^TQ=I_k} \frac{1}{m}\sum_{i=1}^{m} \|Q^T\mathbf{x}_i\|^2 \quad\Longleftrightarrow\quad \min_{Q^TQ=I_k} \frac{1}{m}\sum_{i=1}^{m} \|\mathbf{x}_i - QQ^T\mathbf{x}_i\|^2 \tag{17.1}$$

On the left, keep as much **variance** as possible in the projected coordinates. On the right, make the **reconstruction** from those coordinates as accurate as possible. They are equivalent because of Pythagoras: for each point, $\|\mathbf{x}_i\|^2 = \|QQ^T\mathbf{x}_i\|^2 + \|\mathbf{x}_i - QQ^T\mathbf{x}_i\|^2$, and the left side does not depend on Q. Whatever the projection keeps, the residual loses, and the two always add to the same total.



keeping one number per point means projecting onto a line. The line that keeps the most variance -- and loses the least squared distance -- is the top eigenvector of the covariance.

FIGURE 17.1 Keeping one number per point means projecting onto a line. The red line keeps the most variance of the projected points and, equivalently, leaves the smallest total of squared perpendicular distances (gold).

17.2 The solution

Theorem 17.1 Principal components

Let $C = X^T X/m$ be the covariance of the centred data, with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ and orthonormal eigenvectors $\mathbf{q}_1, \ldots, \mathbf{q}_n$. The best k-dimensional subspace in the sense of Equation 17.1 is spanned by $\mathbf{q}_1, \ldots, \mathbf{q}_k$. The variance it keeps is $\lambda_1 + \cdots + \lambda_k$, and the mean squared reconstruction error is $\lambda_{k+1} + \cdots + \lambda_n$.

Proof

The variance kept by an orthonormal Q is $1/m \sum \| Q^T \mathbf{x}_i \|^2 = \operatorname{tr}(Q^T C Q) = \sum_{j=1}^{k} \mathbf{q}_j'^{\,T} C \mathbf{q}_j'$, where the $\mathbf{q}_j'$ are the columns of Q.

For $k = 1$ this is the Rayleigh quotient, maximised by the top eigenvector with value λ_1 (Theorem 14.2). For larger k, the sum of k such quotients over orthonormal vectors is at most $\lambda_1 + \cdots + \lambda_k$ — Ky Fan's inequality, proved by writing each $\mathbf{q}_j'$ in the eigenbasis and noting that the resulting weights on each eigenvalue lie between 0 and 1 and add up to k.

The eigenvectors attain the bound. The reconstruction error is the total variance $\operatorname{tr}(C) = \lambda_1 + \cdots + \lambda_n$ minus what is kept. ■

The eigenvectors $\mathbf{q}_j$ are the **principal directions**, the coordinates $\mathbf{q}_j \cdot \mathbf{x}$ are the **principal components**, and the eigenvalue λ_j is the variance of the data along direction j. Because the eigenvectors of a symmetric matrix are orthogonal, the principal components of different directions are uncorrelated.

17.3 Computing it with the SVD

Forming the covariance and finding its eigenvectors works, but it squares the condition number exactly as the normal equations did (Section 12.3). The stable route is the SVD of the data matrix itself.

$$X = U \Sigma V^T \implies C = \tfrac{1}{m} X^T X = V \tfrac{\Sigma^2}{m} V^T, \qquad \lambda_j = \tfrac{\sigma_j^2}{m}, \qquad X V_k = U_k \Sigma_k \tag{17.2}$$

The right singular vectors of X are the principal directions, the squared singular values divided by m are the variances, and the principal components of all the data at once are $U_k \Sigma_k$. The best rank-k approximation of the data table, by Eckart–Young, is the

reconstruction from k components. PCA is the truncated SVD of a centred table.

Worked Example 17.1 PCA from a covariance

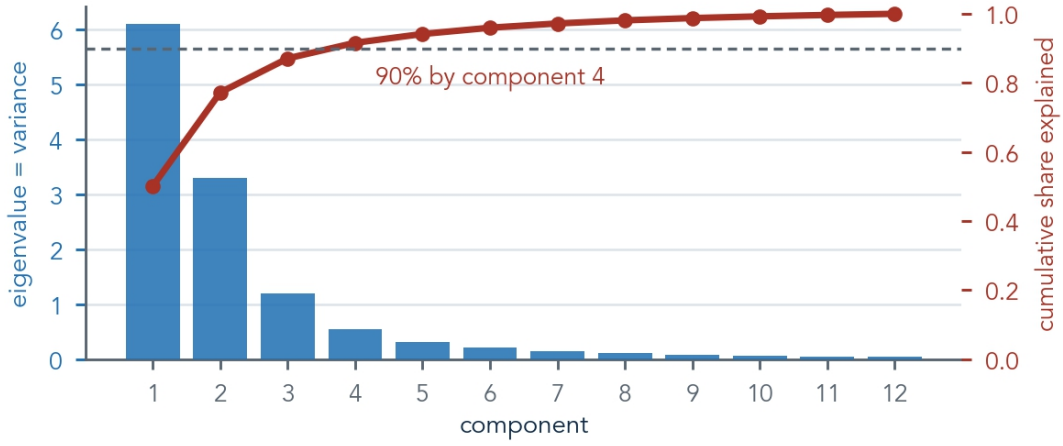
Two standardised measurements have covariance matrix with rows (1, 0.8) and (0.8, 1). Find the principal directions and the share of variance the first one keeps.

The characteristic equation is $(1-\lambda)^2 = 0.64$, so $\lambda = 1.8$ and $\lambda = 0.2$.

For $\lambda = 1.8$ the eigenvector is $(1, 1)/\sqrt{2}$; for 0.2 it is $(1, -1)/\sqrt{2}$.

The total variance is 2, so the first component — essentially the average of the two measurements — keeps $1.8/2 = 90$. The second, their difference, carries the remaining 10%. With a correlation of 0.8, one number per observation retains nine tenths of the variation.

17.4 How many components?



twelve measured quantities, and the eigenvalues of their covariance. Three directions carry most of the variation; the rest is the kind of small, shapeless spread that looks like noise.

FIGURE 17.2 The eigenvalues of the covariance of twelve measured quantities (bars) and the cumulative share of variance explained (line). Three components carry 87% of the variance and four carry 92%; the rest is small and shapeless.

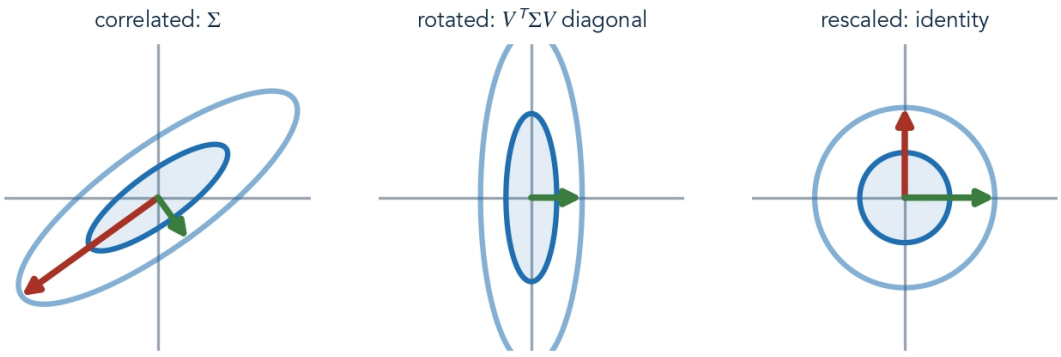
TABLE 17.1 Common rules for choosing k. None is a theorem; each is a reasonable stopping point for a particular purpose.

rule	choose k so that	suited to
variance threshold	cumulative share exceeds 90% or 95%	compression with a stated loss
elbow	the eigenvalues level off	exploratory analysis
noise floor	eigenvalues exceed the level expected from noise	separating signal from noise (Section 20.5)
downstream task	performance on the actual task stops improving	preprocessing for a model
visualisation	$k = 2$ or 3	looking at the data

17.5 Whitening

Rotating onto the principal directions removes correlation; dividing each component by its standard deviation removes scale. The result has identity covariance.

$$\mathbf{z} = \Lambda^{-1/2} V^T \mathbf{x}, \qquad \mathrm{Cov}(\mathbf{z}) = \Lambda^{-1/2} V^T C V \Lambda^{-1/2} = I \tag{17.3}$$



rotating onto the eigenvectors removes the correlation; dividing each axis by its standard deviation removes the scale. The result looks the same in every direction.

FIGURE 17.3 Correlated data (left), rotated onto its principal axes (centre), and rescaled to unit variance in every direction (right). The ellipse becomes a circle.

Whitening is a change of basis that makes the data's own ellipse round, and it is the data-side version of the preconditioning of

Chapter 19. Its danger is the same as the pseudoinverse's: a direction with tiny variance, often pure noise, is divided by a tiny number and amplified, so in practice components below a threshold are dropped before rescaling.

Worked Example 17.2 Whitening two measurements

Continue Example 17.1: the covariance has eigenvalues 1.8 and 0.2, with eigenvectors $(1, 1)/\sqrt{2}$ and $(1, -1)/\sqrt{2}$.

For $\mathbf{x} = (1, 1)$, $V^T\mathbf{x} = (\sqrt{2}, 0)$, and dividing by the standard deviations gives $\mathbf{z} = (\sqrt{2}/\sqrt{1.8}, 0) \approx (1.05, 0)$.

For $\mathbf{x} = (1, -1)$, $V^T\mathbf{x} = (0, \sqrt{2})$ and $\mathbf{z} = (0, \sqrt{2}/\sqrt{0.2}) \approx (0, 3.16)$.

Two inputs of equal length become very different: the direction of small variance is magnified three times as much. If that direction is noise, whitening has amplified the noise.

17.6 What PCA cannot see

- **Scale.** PCA maximises variance, and variance depends on units. A feature measured in millimetres dominates one measured in metres. Standardising each feature first is almost always right.
- **Curvature.** PCA finds flat subspaces. Data lying on a curved surface — a circle, a spiral, a folded sheet — needs more linear components than its true dimension, and non-linear methods exist for exactly this reason.
- **Relevance.** The direction of greatest variance need not be the one that matters. Two classes can differ along a direction of small variance, which PCA would discard first.
- **Outliers.** A single extreme point contributes its squared distance to the variance and can drag the first direction toward itself.

In machine learning. PCA is used to visualise high-dimensional embeddings in two or three dimensions, to reduce the dimension of inputs before fitting a model, to decorrelate features, and to remove noise by reconstructing from the leading components. It also sets a baseline for the dictionary methods of Chapter 4: PCA finds at most n perpendicular directions, ordered by variance, while a sparse dictionary may use many more directions than dimensions, chosen so that each vector needs only a few. The contrast between the two is the contrast between orthogonality and superposition.

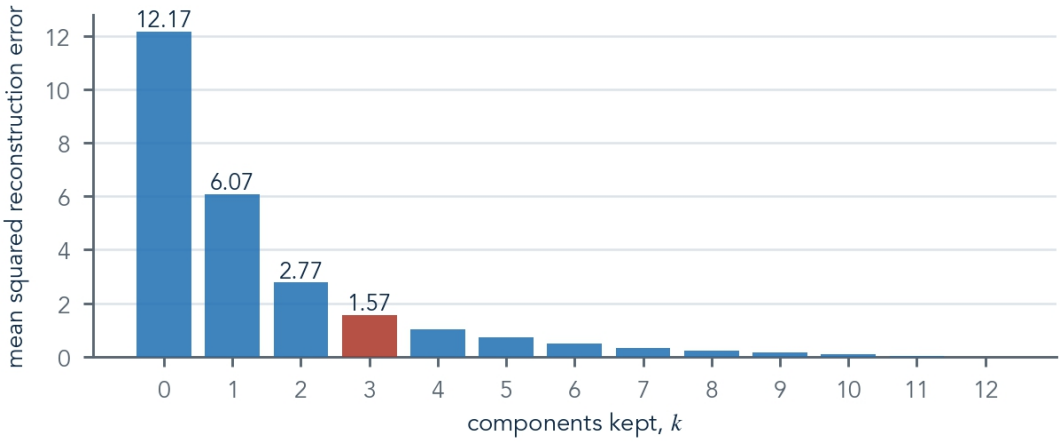
Principal component analysis is not a separate technique. It is the spectral theorem applied to a covariance, computed by the SVD of the data, justified by Eckart–Young, and interpreted through the Rayleigh quotient. A reader who has followed Part IV already knows everything PCA does, and exactly what it cannot do.

Worked Example 17.3 Reconstruction error from the eigenvalues

For the twelve measurements of Figure 17.2, how much is lost by keeping three components?

The discarded eigenvalues are 0.55, 0.31, 0.22, 0.15, 0.11, 0.08, 0.06, 0.05 and 0.04, which sum to 1.57.

By Theorem 17.1 the mean squared reconstruction error is exactly 1.57, which is $1.57/12.17 \approx 12.9$ of the total variance. No other three-dimensional description of these data can do better. Figure 17.4 shows the error for every choice of k.



the error is the sum of the eigenvalues left out. With three components (red) it is 1.57 of a total variance of 12.17, and every later component removes less than the one before.

FIGURE 17.4 Mean squared reconstruction error against the number of principal components kept, for the twelve measurements of Figure 17.2. Each bar is the sum of the eigenvalues not yet kept, and it falls fastest where the eigenvalues are largest.

Key ideas

- Keeping the most variance and minimising the reconstruction error are the same problem, by Pythagoras.
- The principal directions are the top eigenvectors of the covariance, and each eigenvalue is the variance its direction keeps.
- PCA is computed stably as the SVD of the centred data matrix.
- Whitening rotates onto the principal axes and rescales to unit variance, magnifying the directions of small variance.
- PCA is sensitive to units and outliers, finds only flat structure, and keeps variance rather than relevance.

Exercises

1. A covariance matrix has rows (4, 2) and (2, 1). Find its eigenvalues, and explain what the zero eigenvalue says about the data.
2. Show that the principal components $\mathbf{q}_1 \cdot \mathbf{x}$ and $\mathbf{q}_2 \cdot \mathbf{x}$ are uncorrelated.
3. Using the eigenvalues listed for Figure 17.2 (6.1, 3.3, 1.2, 0.55, 0.31, 0.22, 0.15, 0.11, 0.08, 0.06, 0.05, 0.04), how many components are needed to keep 95% of the variance?
4. Why should data be centred before PCA? Describe what the first principal direction tends to find if it is not.
5. Give an example of two-dimensional data with two classes where the first principal direction is useless for separating them.
6. What is the total variance of data whose covariance has rows (2, 0.5) and (0.5, 1)?
7. Why should features measured in different units be standardised before PCA?
8. PCA of three-dimensional data gives eigenvalues 4, 1 and 0.01. How many components would you keep, and what does the third eigenvalue say?

PART

V

INSIDE LEARNING MACHINES

derivatives, optimisation, attention

CHAPTER EIGHTEEN

18 Derivatives Are Linear Maps

Part V turns to how learning machines learn, and learning is differentiation. A model adjusts millions of numbers in the direction that reduces its error, and that direction is a gradient. The central claim of this chapter is that a derivative is not a number, or even a vector, but a linear map — a matrix — and that the chain rule is matrix multiplication. Backpropagation, the algorithm that trains every neural network, is that multiplication performed in a clever order.

18.1 The derivative as the best linear approximation

For a function of one variable, the derivative is the slope of the tangent line: near a point, $f(x + h) \approx f(x) + f'(x)h$. The same idea works in any dimension if "slope" is replaced by "linear map".

Definition 18.1 Gradient

For $f\colon \mathbb{R}^n \to \mathbb{R}$, the gradient $\nabla f(\mathbf{x})$ is the vector of partial derivatives $(\partial f/\partial x_1, \dots, \partial f/\partial x_n)$. It is the vector for which $f(\mathbf{x} + \mathbf{h}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot \mathbf{h}$ for small $\mathbf{h}$.

The change in f is a dot product with the step. Chapter 2 immediately says which step increases f fastest.

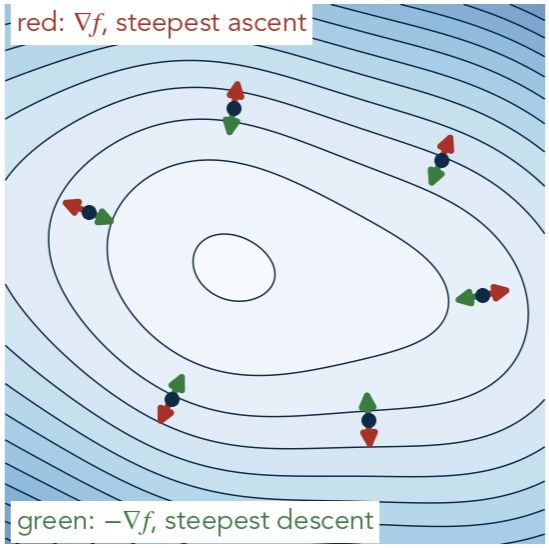
Theorem 18.1 Steepest ascent

Among all unit vectors $\mathbf{u}$, the rate of change $\nabla f \cdot \mathbf{u}$ is largest when $\mathbf{u}$ points along ∇f, where it equals $\|\nabla f\|$. The gradient is perpendicular to the level set of f through the point.

Proof

By Cauchy–Schwarz, $\nabla f \cdot \mathbf{u} \le \|\nabla f\|\,\|\mathbf{u}\| = \|\nabla f\|$, with equality exactly when $\mathbf{u}$ is a positive multiple of ∇f.

Along a level set f is constant, so its rate of change in any tangent direction is zero: $\nabla f \cdot \mathbf{t} = 0$, and the gradient is perpendicular to every tangent. ■



the gradient is a vector of partial derivatives, and geometrically it is the direction of fastest increase, perpendicular to the contour through the point. Training steps the other way.

FIGURE 18.1 Contours of a function of two variables. At each marked point the gradient (red) points uphill, perpendicular to the contour; its negative (green) is the direction of steepest descent, the direction a training step takes.

18.2 Jacobians and the chain rule

A function from $\mathbb{R}^n$ to $\mathbb{R}^m$ has m outputs, each with its own gradient. Stacking those gradients as rows gives the Jacobian of Chapter 8, and the approximation becomes a matrix acting on a vector.

$$F(\mathbf{x}+\mathbf{h}) \approx F(\mathbf{x}) + J_F(\mathbf{x})\,\mathbf{h}, \qquad J_F \in \mathbb{R}^{m\times n}, \qquad (J_F)_{ij} = \frac{\partial F_i}{\partial x_j} \tag{18.1}$$

Theorem 18.2 Chain rule

If $G = g \circ f$, then $J_G(\mathbf{x}) = J_g(f(\mathbf{x}))\, J_f(\mathbf{x})$.

Proof

Near $\mathbf{x}$, $f(\mathbf{x}+\mathbf{h}) \approx f(\mathbf{x}) + J_f\mathbf{h}$.

Near $f(\mathbf{x})$, $g(f(\mathbf{x})+\mathbf{k}) \approx g(f(\mathbf{x})) + J_g\mathbf{k}$.

Substituting $\mathbf{k} = J_f\mathbf{h}$ gives $G(\mathbf{x}+\mathbf{h}) \approx G(\mathbf{x}) + J_gJ_f\mathbf{h}$, and the best linear approximation of a composition is the composition of the best linear approximations — which, by Chapter 6, is the product of their matrices. ■

The chain rule is Theorem 6.1 in disguise. A derivative is a linear map, composing functions composes their derivatives, and composing linear maps multiplies their matrices. Everything about computing gradients in deep networks follows from applying that sentence to a long composition.

18.3 Gradients worth knowing by heart

TABLE 18.1 Gradients of the expressions that appear most often in learning, all with respect to the vector $\mathbf{x}$ or the parameters $\mathbf{c}$. Each can be derived from Definition 18.1 in a few lines.

function	gradient	where it appears
$\mathbf{a}^T\mathbf{x}$	$\mathbf{a}$	a linear model's score
$\mathbf{x}^T\mathbf{x} = \|\mathbf{x}\|^2$	$2\mathbf{x}$	weight decay
$\mathbf{x}^T A\mathbf{x}$	$(A + A^T)\mathbf{x}$, or $2A\mathbf{x}$ if symmetric	quadratic models of a loss
$\|A\mathbf{c} - \mathbf{y}\|^2$	$2A^T(A\mathbf{c} - \mathbf{y})$	least squares
cross-entropy of softmax($\mathbf{z}$) with target $\mathbf{y}$	softmax($\mathbf{z}$) $- \mathbf{y}$	classification
$\mathbf{a}^T W\mathbf{b}$, as a function of the matrix W	$\mathbf{a}\mathbf{b}^T$	the gradient of any linear layer

Worked Example 18.1 A least-squares gradient by hand

For A with rows $(1, 0)$, $(1, 1)$, $(1, 2)$, target $\mathbf{y} = (1, 2, 4)$ and $\mathbf{c} = (0, 1)$, compute the gradient of $L = \|A\mathbf{c} - \mathbf{y}\|^2$.

$A\mathbf{c} = (0, 1, 2)$, so the residual is $A\mathbf{c} - \mathbf{y} = (-1, -1, -2)$.

A^T applied to the residual gives $(-1 - 1 - 2, 0 - 1 - 4) = (-4, -5)$, so $\nabla L = 2(-4, -5) = (-8, -10)$.

Both components are negative: increasing either coefficient reduces the error. The minimum found in Example 11.1, $(0.83, 1.5)$, lies in exactly that direction from $(0, 1)$, and at that minimum the gradient is zero — the normal equations.

The last row of Table 18.1 deserves emphasis. The gradient of a linear layer's output with respect to its weight matrix is an outer product, and outer products have rank one. Summed over a batch, this gives a structural fact about every gradient in a network.

Theorem 18.3 Rank of a layer's gradient

For a layer $\mathbf{z} = W\mathbf{h}$ and a batch of B examples, the gradient of the loss with respect to W is $\sum_{b=1}^{B} \mathbf{z}_b \mathbf{h}_b^T$, and so has rank at most B.

Proof

For one example, $\partial L / \partial W_{ij} = (\partial L / \partial z_i)(\partial z_i / \partial W_{ij}) = \bar{z}_i h_j$, which is the (i, j) entry of $\mathbf{z}\mathbf{h}^T$.

The loss of a batch is a sum over examples, so its gradient is the sum of B such rank-one matrices.

A sum of B matrices of rank one has rank at most B (Chapter 7). ■

A 4096×4096 weight matrix trained with a batch of 32 examples receives, at each step, a gradient of rank at most 32. This is one reason the low-rank structure of Chapter 16 appears naturally in training, and why methods that store gradients in factored form save so much memory.

18.4 Forward and reverse: the order of a product

Consider a loss computed by a chain of three maps, $\mathbf{x} \mapsto \mathbf{u} \mapsto \mathbf{v} \mapsto L$, with $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^k$, $\mathbf{v} \in \mathbb{R}^m$ and a scalar L. By the chain rule the gradient, written as a row, is a product of three Jacobians.

$$\underset{1\times n}{\nabla L^T} = \underset{1\times m}{J_3}\ \underset{m\times k}{J_2}\ \underset{k\times n}{J_1} \tag{18.2}$$

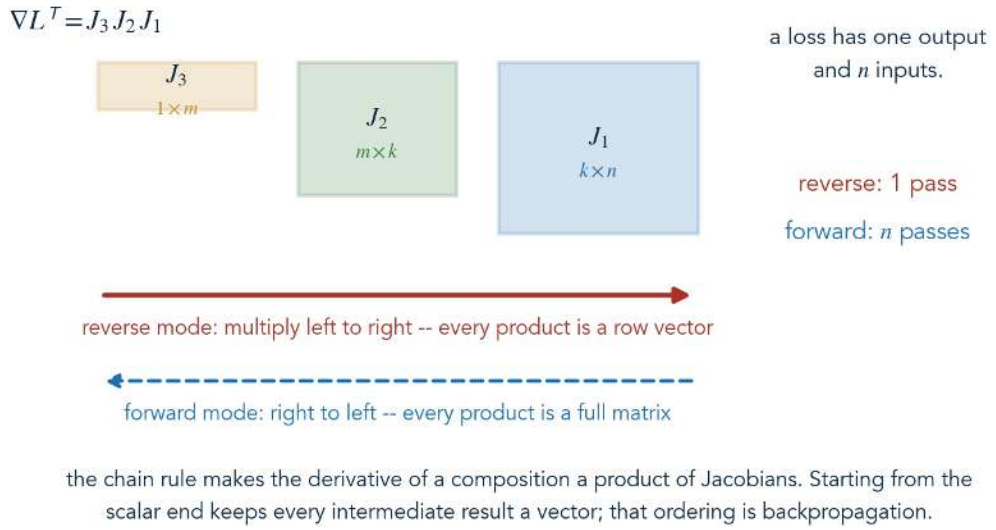


FIGURE 18.2 The gradient of a scalar loss as a product of Jacobians. Multiplying from the left keeps every intermediate result a row vector; multiplying from the right builds full matrices, or needs one pass per input coordinate.

Matrix multiplication is associative, so the answer does not depend on the order, but the cost does. Multiplying right to left — **forward mode** — first forms $J_2 J_1$, an $m \times k$ times $k \times n$ product, a full matrix. Multiplying left to right — **reverse mode** — first forms $J_3 J_2$, a row vector times a matrix, which is just another row vector. Every step of reverse mode is a vector times a matrix.

Theorem 18.4 The cost of a gradient

For a function with n inputs and one output built from elementary operations, reverse-mode differentiation computes the entire gradient at a cost that is a small constant multiple of evaluating the function once, independent of n. Forward mode needs n passes.

Proof

Reverse mode multiplies the Jacobians from the scalar end. The first factor is a $1 \times m$ row, and each subsequent multiplication by a Jacobian produces a row of the next size, a vector–Jacobian product.

Each vector–Jacobian product for an elementary operation costs about as much as the operation itself, so the whole backward pass costs a constant multiple of the forward pass.

Forward mode can instead compute Jacobian–vector products $J\mathbf{e}_j$ one input direction at a time, which needs a pass for each of the n inputs to assemble the gradient. ■

Backpropagation is the observation that a gradient is a product of matrices whose leftmost factor is a row vector, together with the decision to multiply from the left. That one choice of order is why a model with a billion parameters can be trained at all: its gradient costs about as much as a few evaluations of its loss, not a billion.

18.5 Backpropagation through a layer

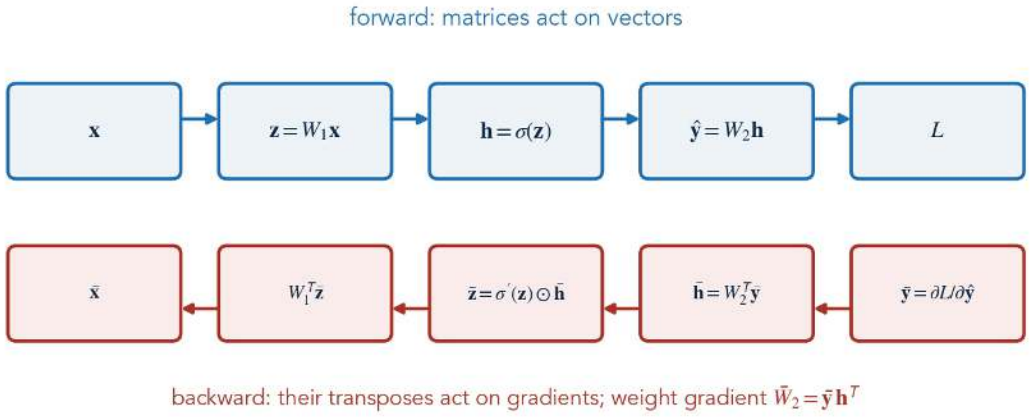


FIGURE 18.3 A two-layer network. The forward pass (blue) applies W_1, a non-linearity σ and W_2. The backward pass (red) applies their transposes to gradient vectors, multiplies by the derivative of σ entry by entry, and forms each weight gradient as an outer product.

$$\bar{\mathbf{h}} = W_2^T \bar{\mathbf{y}}, \qquad \bar{\mathbf{z}} = \sigma'(\mathbf{z}) \odot \bar{\mathbf{h}}, \qquad \bar{\mathbf{x}} = W_1^T \bar{\mathbf{z}}, \qquad \frac{\partial L}{\partial W_2} = \bar{\mathbf{y}}\mathbf{h}^T, \qquad \frac{\partial L}{\partial W_1} = \bar{\mathbf{z}}\mathbf{x}^T \tag{18.3}$$

Here a bar over a quantity means the gradient of the loss with respect to it, and $\odot$ multiplies entry by entry. The transposes appear because a vector–Jacobian product with the linear map W is $\mathbf{z}^T W = (W^T \mathbf{z})^T$: moving a gradient backwards through a matrix is applying its transpose. The four fundamental subspaces of Chapter 7 now have a training interpretation. A component of the gradient $\mathbf{z}$ that lies in the left null space of W_1 is sent to zero by $W_1{}^T$ and never reaches the earlier layers.

Worked Example 18.2 Backpropagation with single numbers

Take $x = 2$, $W_1 = 0.5$, the rectifier, $W_2 = 3$, target $y = 1$ and loss $L = 1/2(\hat{y} - y)^2$.

Forward: $z = 1$, $h = 1$, $\hat{y} = 3$ and $L = 2$.

Backward: $\bar{y} = \hat{y} - y = 2$, so $\partial L/\partial W_2 = \bar{y}h = 2$. Then $\bar{h} = W_2\bar{y} = 6$, and since $z > 0$, $\bar{z} = 6$, so $\partial L/\partial W_1 = \bar{z}x = 12$.

Check the last by hand: for $W_1 > 0$, $L = 1/2(6W_1 - 1)^2$, whose derivative at $W_1 = 0.5$ is $6(6W_1 - 1) = 12$.

18.6 The Hessian

The gradient is itself a function from $\mathbb{R}^n$ to $\mathbb{R}^n$, and its Jacobian is the matrix of second derivatives.

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}, \qquad f(\mathbf{x}+\mathbf{h}) \approx f(\mathbf{x}) + \nabla f^{\mathsf{T}}\mathbf{h} + \tfrac{1}{2}\mathbf{h}^{\mathsf{T}}H\mathbf{h} \tag{18.4}$$

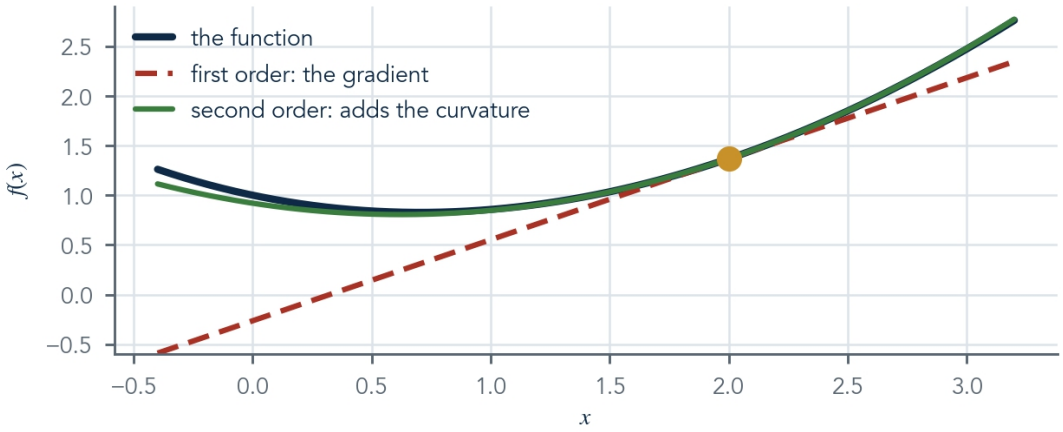
For smooth functions the order of differentiation does not matter, so the Hessian is symmetric. The spectral theorem therefore applies: the Hessian has real eigenvalues and perpendicular eigenvectors, and near any point a smooth loss looks like the quadratic forms of Figure 14.2 — a bowl, a saddle or a trough, decided by the signs of its eigenvalues. Chapter 19 is the study of what those eigenvalues do to training.

Worked Example 18.3 What a gradient step really does

Let $f(x, y) = x^2 + 3y^2$ at the point $(1, 1)$, where $f = 4$ and $\nabla f = (2, 6)$.

A step of size 0.1 against the gradient moves to $(0.8, 0.4)$, where $f = 0.64 + 0.48 = 1.12$.

The first-order prediction was $4 - 0.1 \times \|\nabla f\|^2 = 4 - 4 = 0$, far too optimistic. The Hessian is diagonal with entries 2 and 6, and the second-order term $1/2\mathbf{h}^T H\mathbf{h}$ with $\mathbf{h} = (-0.2, -0.6)$ adds back $1/2(0.08 + 2.16) = 1.12$, exactly. Figure 18.4 shows the two approximations for a function of one variable.



near the gold point all three agree; farther away only the parabola keeps up. A step chosen from the tangent line alone can overshoot, and the second derivative says by how much.

FIGURE 18.4 A function of one variable, its tangent line (the first-order approximation) and its osculating parabola (second order) at one point. The gradient gives the first; the curvature in the Hessian says how far a step can go before the first misleads.

Key ideas

- The gradient is the vector of partial derivatives; it points in the direction of steepest ascent and is perpendicular to level sets.
- A derivative is a linear map, and the Jacobian of a composition is the product of the Jacobians.
- The gradient of a linear layer is an outer product, so a batch gradient has rank at most the batch size.
- Reverse mode multiplies Jacobians from the scalar end, so a full gradient costs about as much as a few evaluations of the loss.
- Backpropagation applies transposed weight matrices to gradient vectors, and the Hessian is the symmetric matrix of second derivatives.

Exercises

1. Compute the gradient of $f(x, y) = x^2 y + 3y$ at $(1, 2)$, and find the direction of steepest descent there.
2. Verify the entry $\nabla(\mathbf{x}^T A\mathbf{x}) = (A + A^T)\mathbf{x}$ in Table 18.1 for a general 2×2 matrix by writing out $\mathbf{x}^T A\mathbf{x}$.
3. For $F(x, y) = (xy, x + y, x^2)$, write down the Jacobian and check its shape against Equation 18.1.
4. A network maps $\mathbb{R}^{1000}$ through layers of width 500 and 100 to a scalar loss. Write the shapes of the three Jacobians and count the entries of the largest intermediate matrix in forward mode and in reverse mode.
5. A layer's weight matrix is 512×512 and the batch size is 8. What is the largest possible rank of its gradient for one step? What does Theorem 18.3 imply after 100 steps of plain gradient descent from zero?
6. Compute the gradient of $f(\mathbf{x}) = \|\mathbf{x}\|^2$ at $(1, -2, 2)$.
7. Find the Jacobian and its determinant for $F(x, y) = (x + y, x - y)$.
8. In Example 18.2, compute the gradient of the loss with respect to the input x.

CHAPTER NINETEEN

19 The Geometry of Optimisation

Training a model is walking downhill on a surface in a space of millions of dimensions. How fast the walk goes, whether it goes at all, and which step sizes are safe are all decided by the shape of that surface near the walker — and near any point, a smooth surface is a quadratic whose shape is a symmetric matrix. This chapter shows that the speed of gradient descent is the condition number of that matrix, that preconditioning is a change of basis that rounds it, and that a family of optimisers adopted for large-scale training in 2025 is steepest descent measured in a norm chosen to respect matrix structure.

19.1 The local model

By Equation 18.4, near a minimum $\mathbf{x}^\ast$ where the gradient vanishes, a smooth loss is approximately a quadratic bowl.

$$L(\mathbf{x}^\ast + \mathbf{e}) \approx L(\mathbf{x}^\ast) + \tfrac{1}{2}\,\mathbf{e}^T H \mathbf{e}, \qquad \nabla L(\mathbf{x}^\ast + \mathbf{e}) \approx H\mathbf{e} \tag{19.1}$$

The Hessian H is symmetric and, at a minimum, positive semidefinite. Its eigenvectors are the principal axes of the bowl and its eigenvalues are the curvatures along them: a large eigenvalue is a steep, narrow direction; a small one is a gentle, wide valley.

19.2 Gradient descent, analysed exactly

Gradient descent moves against the gradient with a step size η. On the quadratic model the error $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^\ast$ obeys a linear recursion, and linear recursions are what eigenvectors are for.

$$\mathbf{e}_{k+1} = \mathbf{e}_k - \eta H \mathbf{e}_k = (I - \eta H)\,\mathbf{e}_k, \qquad \mathbf{e}_k = (I - \eta H)^k\,\mathbf{e}_0 \tag{19.2}$$

Theorem 19.1 Convergence of gradient descent on a quadratic

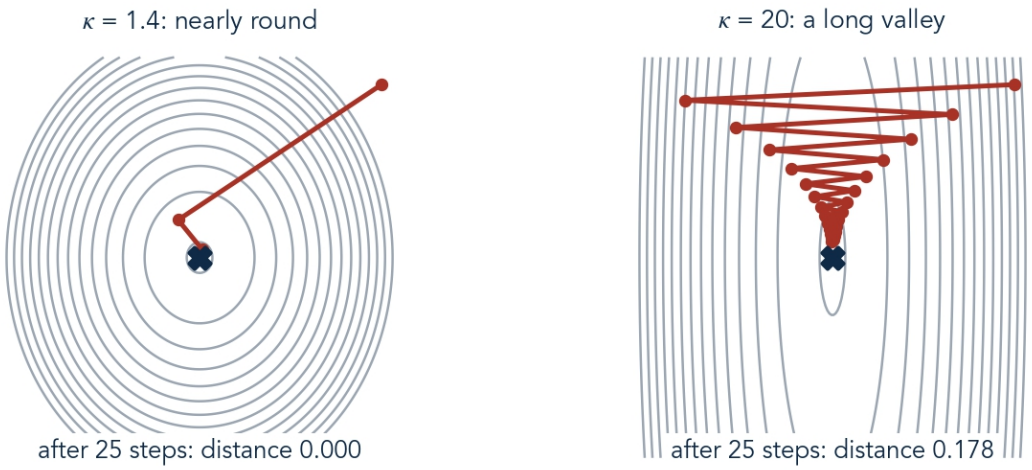
Let H have eigenvalues $0 < \lambda_{\min} \le \cdots \le \lambda_{\max}$. Gradient descent converges if and only if $0 < \eta < 2/\lambda_{\max}$. The best fixed step is $\eta = 2/(\lambda_{\max} + \lambda_{\min})$, and with it the error shrinks at every step by at least the factor $(\kappa - 1)/(\kappa + 1)$, where $\kappa = \lambda_{\max}/\lambda_{\min}$.

Proof

In the eigenbasis of H, the matrix $I - \eta H$ is diagonal with entries $1 - \eta\lambda_i$, so the component of the error along eigenvector i is multiplied by $1 - \eta\lambda_i$ at every step.

Every component shrinks exactly when $|1 - \eta\lambda_i| < 1$ for all i, which is $0 < \eta < 2/\lambda_i$ for all i; the binding constraint is the largest eigenvalue.

The slowest factor is $\max_i |1 - \eta\lambda_i|$, attained at $\lambda_{\min}$ or $\lambda_{\max}$. It is smallest when the two are balanced, $1 - \eta\lambda_{\min} = \eta\lambda_{\max} - 1$, giving $\eta = 2/(\lambda_{\max} + \lambda_{\min})$ and the factor $(\lambda_{\max} - \lambda_{\min})/(\lambda_{\max} + \lambda_{\min}) = (\kappa - 1)/(\kappa + 1)$. ■



the same step rule on two quadratics. The step size is limited by the steep direction and the progress by the shallow one; their ratio is the condition number of the Hessian.

FIGURE 19.1 Twenty-five steps of gradient descent on a nearly round bowl (left) and on a long narrow valley with condition number 20 (right). On the right, the step size that is safe for the steep direction is far too small for the shallow one, and the path zigzags across the valley.

The theorem says precisely what Figure 19.1 shows. The step size is capped by the steepest direction and the progress is limited by the shallowest. With large, $(-1)/(+1) \approx 1 - 2/$, so reducing the error by a factor $1/\varepsilon$ takes about $/2\ln(1/\varepsilon)$ steps: **the number of steps grows in proportion to the condition number.**

Worked Example 19.1 Counting steps

A quadratic loss has Hessian eigenvalues 1 and 0.05. How many steps of gradient descent with the best fixed step reduce the error by a factor of a million?

$= 1/0.05 = 20$, so the contraction factor is $19/21 \approx 0.905$.

We need $(19/21)^k \le 10^{-6}$, so $k \ge \ln(10^6)/\ln(21/19) = 13.82/0.1001 \approx 138$ steps.

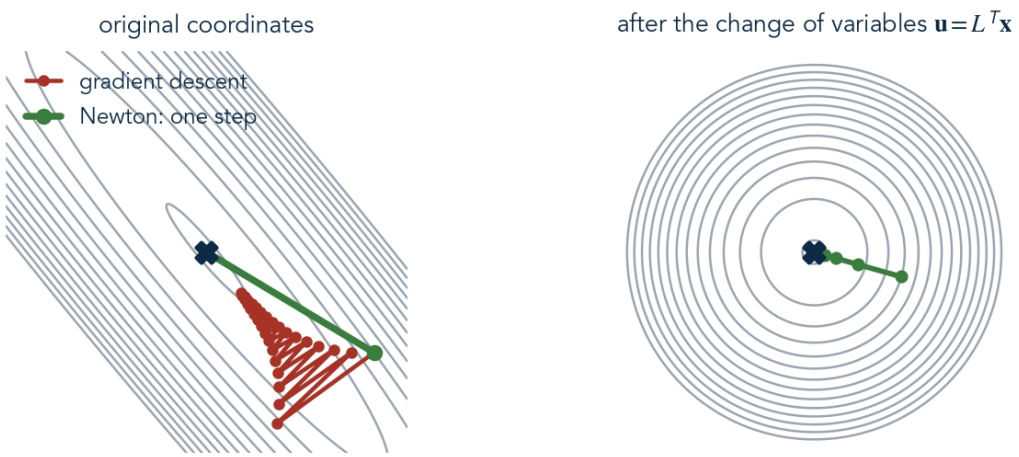
Had the eigenvalues been 1 and 0.5, $= 2$, the factor would be $1/3$, and 13 steps would do. The same algorithm on the same kind of problem needs ten times as many steps because one ellipse is ten times thinner.

The difficulty of optimising a smooth loss is not its dimension, and not the size of its gradients. It is the spread of the eigenvalues of its Hessian. Every technique that makes training faster is, to a first approximation, a way of making that spread smaller or of stepping as though it were smaller.

19.3 Preconditioning and Newton's method

If the problem is that the bowl is elliptical, change coordinates until it is round. Writing $H = LL^T$ (Cholesky) and substituting $\mathbf{u} = L^T\mathbf{x}$ turns the quadratic $1/2\mathbf{x}^T H\mathbf{x}$ into $1/2\|\mathbf{u}\|^2$, whose condition number is one. A gradient step in the new coordinates, translated back, is a step in the direction $H^{-1}\nabla L$.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - P^{-1}\nabla L(\mathbf{x}_k), \qquad P = H \implies \text{Newton's method: exact in one step on a quadratic} \tag{19.3}$$



a preconditioner is a change of basis that makes the curvature the same in every direction.
In the new coordinates the valley is a round bowl and the gradient points straight at the minimum.

FIGURE 19.2 Left: gradient descent zigzags along a tilted valley while a Newton step goes straight to the minimum. Right: the same problem after the change of variables $\mathrm{u} = L^T\mathrm{x}$, where the contours are circles and the ordinary gradient already points at the answer.

Newton's method is the ideal preconditioner and an impractical one: for a model with n parameters the Hessian has n^2 entries and solving with it costs about n^3 operations. Practical optimisers use cheap approximations of P. The most common approximates it by a diagonal matrix of running averages of squared gradients, rescaling each coordinate separately — a preconditioner that can round an ellipse only when its axes happen to line up with the coordinate axes. More structured preconditioners, such as those that approximate P for a weight matrix by a product of one matrix acting on its rows and another acting on its columns, were developed through 2018–2024 and refined during 2024–2025.

19.4 Matrices deserve their own norm

The steepest-descent direction depends on how the length of a step is measured. Theorem 18.1 used the Euclidean length, which treats a weight matrix W as a long list of d^2 unrelated numbers. But W is a map, and what matters about a change ΔW is how much it can

change the map's outputs — which is its largest stretch, the spectral norm $\|\Delta W\|_2 = \sigma_1(\Delta W)$, since $\|\Delta W \mathbf{h}\| \le \|\Delta W\|_2 \, \|\mathbf{h}\|$ for every input.

Theorem 19.2 Steepest descent under the spectral norm

Let $G = U \Sigma V^T$ be the gradient of the loss with respect to a weight matrix, with thin SVD. Among all steps ΔW with $\|\Delta W\|_2 \le 1$, the one that decreases the linear approximation of the loss the most is $\Delta W = -UV^T$, and the decrease it achieves is $\sigma_1 + \cdots + \sigma_r$, the nuclear norm of G.

Proof

The first-order change in the loss is $\langle G, \Delta W \rangle = \operatorname{tr}(G^T \Delta W)$, so we maximise $\operatorname{tr}(G^T \Delta W)$ over $\|\Delta W\|_2 \le 1$ and then take the negative.

Writing $G = \sum \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ gives $\operatorname{tr}(G^T \Delta W) = \sum \sigma_i \mathbf{u}_i^T \Delta W \mathbf{v}_i$. Each term $\mathbf{u}_i^T \Delta W \mathbf{v}_i$ is at most $\|\Delta W\|_2 \le 1$, so the total is at most $\sum \sigma_i$.

The choice $\Delta W = UV^T$ has spectral norm 1 and makes every term $\mathbf{u}_i^T UV^T \mathbf{v}_i = 1$, attaining the bound. ■

The optimal step keeps the singular vectors of the gradient and sets every singular value to one. It moves the weights by the same amount along every direction the gradient points in — strong or weak — instead of mostly along the strongest. For a gradient whose singular values fall steeply, which by Theorem 18.3 is the usual case, this is a dramatic rebalancing, and it is the matrix analogue of rounding an ellipse.

Worked Example 19.2 An orthogonalised step by hand

Take the gradient G with rows $(3, 0)$ and $(4, 5)$, whose SVD was found in Example 15.1: $\mathbf{u}_1 = (1, 3)/\sqrt{10}$, $\mathbf{u}_2 = (3, -1)/\sqrt{10}$, $\mathbf{v}_1 = (1, 1)/\sqrt{2}$, $\mathbf{v}_2 = (1, -1)/\sqrt{2}$.

$UV^T = \mathbf{u}_1\mathbf{v}_1{}^T + \mathbf{u}_2\mathbf{v}_2{}^T$, which is $1/\sqrt{20}$ times the matrix with rows $(4, -2)$ and $(2, 4)$ — about rows $(0.89, -0.45)$ and $(0.45, 0.89)$.

That is a rotation by about $26.6°$. The gradient stretched one direction three times as much as the other; the orthogonalised step moves both by the same amount and keeps only the directions.

19.5 Orthogonalising without an SVD

Computing a full SVD of every weight gradient at every step would be too slow. The factor UV^T — the **polar factor** of G — can be approached instead by an iteration that uses only matrix products.

$$X_0 = \frac{G}{\|G\|_F}, \qquad X_{k+1} = \tfrac{3}{2}X_k - \tfrac{1}{2}X_k X_k^T X_k, \qquad X_k \to UV^T \tag{19.4}$$

Theorem 19.3 The iteration acts on singular values alone

If $X_k = U\mathrm{diag}(s_i)\,V^T$, then $X_{k+1} = U\mathrm{diag}(3/2\,s_i - 1/2\,s_i^3)\,V^T$. Starting from singular values in $(0, 1]$, every one converges to 1, while U and V never change.

Proof

$X_k X_k^T X_k = U\mathrm{diag}(s_i)\, V^T V \mathrm{diag}(s_i)\, U^T U \mathrm{diag}(s_i)\, V^T = U\mathrm{diag}(s_i^3)\, V^T$, so the update applies $p(s) = 3/2 s - 1/2 s^3$ to each singular value.

Dividing by the Frobenius norm makes every $s_i \le 1$. On $(0, 1)$ we have $p(s) > s$ and $p(s) < 1$, and $p(1) = 1$, so each sequence increases toward the fixed point 1.

Near 1 the error $1 - s$ is squared at each step, since $1 - p(s) = 1/2(1 - s)^2(2 + s)$, so convergence is quadratic once close. ■

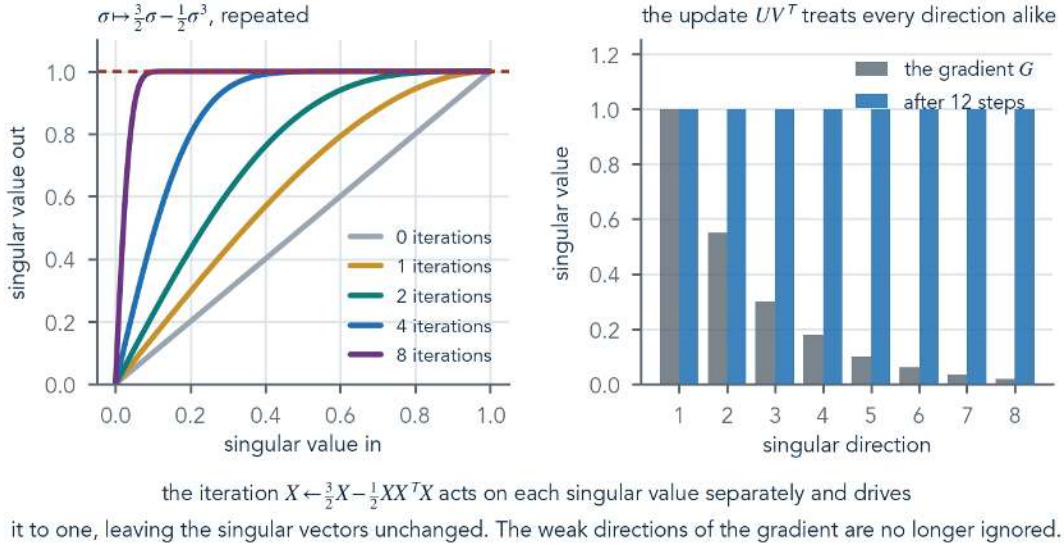


FIGURE 19.3 Left: the map $s \mapsto 3/2 s - 1/2 s^3$ applied repeatedly, pushing every singular value toward one. Right: the singular values of a gradient before and after twelve iterations. The singular vectors are untouched; only the stretches are equalised.

This is the idea behind the Muon family of optimisers, introduced by Jordan and colleagues at the end of 2024: accumulate a momentum matrix for each weight matrix, orthogonalise it with a few iterations of a polynomial of this kind (in practice a tuned higher-degree polynomial that reaches a band near one faster), and step along the result. Liu, Su and colleagues reported in early 2025 that, suitably adjusted, it scaled to the training of large transformer models with substantially better efficiency than the widely used adaptive

methods, and further large-scale adoptions, distributed versions and faster orthogonalisation schemes followed during 2025 and into 2026. The theory of this section explains what the method does, even where the full account of why it works so well is still being written.

TABLE 19.1 Four optimisers read as choices of geometry. Each moves against a gradient transformed by some matrix; they differ in what that matrix knows about the problem.

method	step direction	geometry it assumes	weakness
gradient descent	$-\nabla L$	all directions alike	slow when is large
diagonal adaptive	$-D^{-1}\nabla L$, D diagonal	axes aligned with coordinates	blind to rotated valleys
Newton	$-H^{-1}\nabla L$	the exact local quadratic	n^3 cost
orthogonalised (Muon family)	$-UV^T$ from the momentum's SVD	spectral norm on each matrix	ignores how large each singular value was

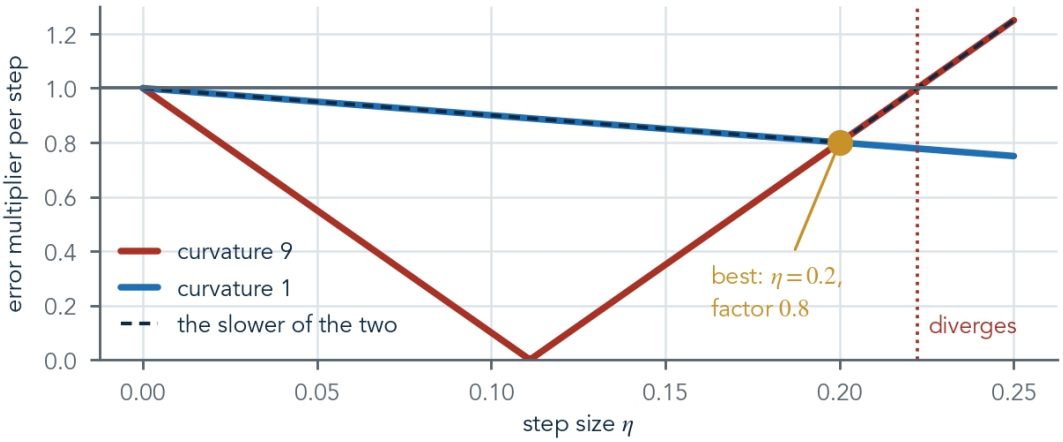
In machine learning. Three observations connect this chapter to practice. Warm-up schedules, which start with small steps, respect the $2/\lambda_{\max}$ limit while the Hessian's largest eigenvalue is still large. Normalisation layers change the effective Hessian and tend to reduce its condition number. And the spectral-norm view of Section 19.4 is the same view that motivates spectral normalisation of layers: a matrix's effect on a network is governed by its stretches, not by its individual entries.

Worked Example 19.3 Balancing two curvatures

A quadratic loss has Hessian eigenvalues 9 and 1.

With step η the error along each eigenvector is multiplied by $1 - 9\eta$ and $1 - \eta$. At $\eta = 0.1$ the sizes are 0.1 and 0.9, so the slow direction sets the rate at 0.9. At $\eta = 0.21$ they are 0.89 and 0.79, and now the steep direction is the slow one.

They balance at $\eta = 2/(9 + 1) = 0.2$, where both are 0.8 — the rate $(-1)/(+1)$ with $= 9$. Figure 19.4 draws both factors across all step sizes.



the steep direction wants a small step and the shallow one a large step. The best compromise is where the two factors meet, and past $\eta = 2/9$ the steep direction grows without bound.

FIGURE 19.4 The contraction factor along each eigenvector, the size of $1 - \eta\lambda$, for curvatures 9 and 1. The larger of the two sets the speed; they balance at $\eta = 0.2$, where both equal 0.8, and the iteration diverges beyond $\eta = 2/9$.

Key ideas

- Near a minimum a loss is a quadratic whose Hessian eigenvalues are its curvatures.
- Gradient descent converges for step sizes below $2/\lambda_{\max}$, and its speed is set by the condition number of the Hessian.
- Preconditioning is a change of basis that rounds the bowl; Newton's method is the exact but expensive version.
- For weight matrices, steepest descent in the spectral norm steps along the polar factor UV^T of the gradient.
- The Newton–Schulz iteration drives every singular value to one using only matrix products, the core of the Muon family of optimisers.

Exercises

1. For a quadratic with Hessian eigenvalues 4 and 1, find the largest safe step size, the best fixed step size, and the contraction factor per step.
2. Explain why gradient descent with $\eta = 2/\lambda_{\max}$ exactly neither converges nor diverges along the steepest direction.
3. Show that one Newton step minimises a quadratic $1/2\mathbf{x}^T H\mathbf{x} - \mathbf{b}^T\mathbf{x}$ exactly from any starting point.
4. Compute UV^T for the gradient with rows $(3, 0)$ and $(0, 1)$, and describe how it differs from the gradient itself.
5. Apply two steps of $s \mapsto 3/2s - 1/2s^3$ to $s = 0.2$ and to $s = 0.9$. Which converges faster, and why is that exactly what an orthogonalised update wants?
6. A quadratic loss has Hessian eigenvalues 10 and 0.1. Find and estimate the number of steps needed to reduce the error a thousandfold with the best fixed step.
7. How many Newton steps does it take to minimise a quadratic exactly?
8. What is the orthogonalised step UV^T for a rank-one gradient $s\,\mathbf{u}\mathbf{v}^T$ with unit vectors $\mathbf{u}$ and $\mathbf{v}$?

CHAPTER TWENTY

20 Inside a Modern Model

This last chapter opens a modern model and reads it with the tools of the book. Nothing new is introduced. The embedding table is Chapter 1, the similarity it supports is Chapter 2, the room it has is Chapter 4, the attention layer is Chapters 2, 6 and 7 at once, its cost is a matrix shape, the limits on what it can compare are a rank, and the structure learned in its weights is visible in their singular values. The purpose is recognition: to show that the working parts of the most capable learning systems yet built are constructions a reader of this book already understands.

20.1 Embeddings: a table of vectors

A model that works with a vocabulary of V symbols begins with an embedding matrix $E \in \mathbb{R}^{V \times d}$, one row per symbol. Looking up a symbol is multiplying its one-hot indicator vector by E^T, which selects a row — so the lookup is a linear map, and the table is learned like any other weight matrix.

$$\mathbf{x}_i = E^T \mathbf{e}_{s_i}, \qquad \text{logits} = E\,\mathbf{h}, \qquad \text{logit}_j = \mathbf{E}_j \cdot \mathbf{h} \tag{20.1}$$

The second equation shows a common design in which the same table is used at the output: the model's final internal vector $\mathbf{h}$ is compared with every symbol's embedding by a dot product, and the scores become probabilities through a softmax. The model's prediction is literally a ranking by agreement, in the sense of Chapter 2, between where its internal state points and where each symbol points. Chapter 4 explains why d in the low thousands is enough room for a vocabulary many times larger: there are astronomically many nearly perpendicular directions to place the symbols along.

20.2 Attention, written with matrices

Chapter 2 described attention for one query. For a whole sequence of n vectors, stack them as the rows of $X \in \mathbb{R}^{n \times d}$ and compute all the queries, keys and values at once with three weight matrices.

$$Q = XW_Q^T, \quad K = XW_K^T, \quad V = XW_V^T, \qquad \text{Attn}(X) = \text{softmax}_{\text{rows}}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{20.2}$$

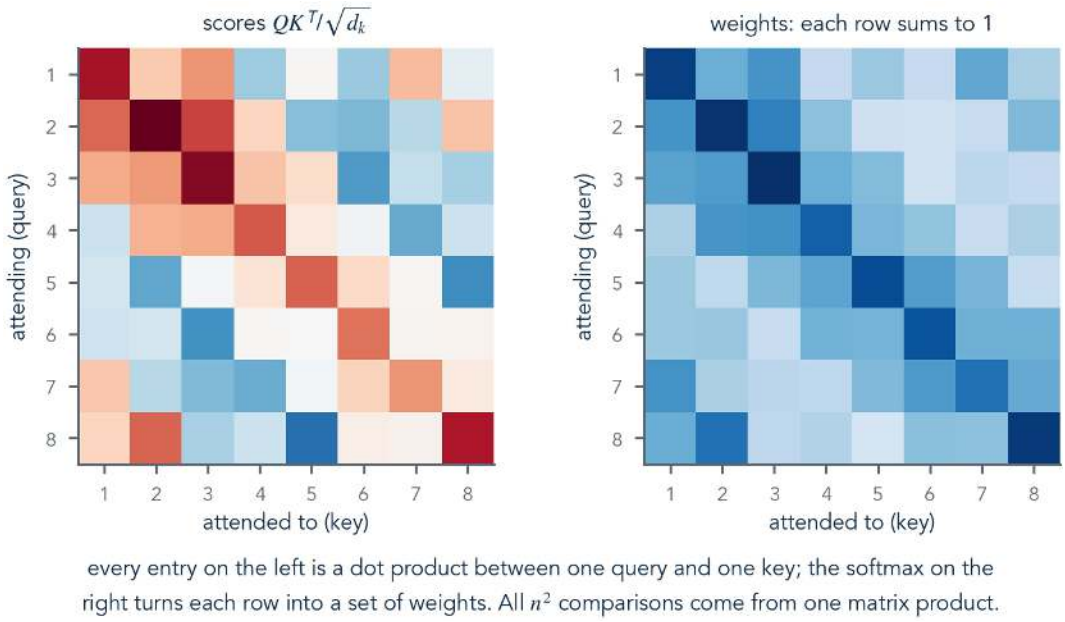


FIGURE 20.1 For a sequence of eight items: the matrix of scaled dot products between every query and every key (left), and the same matrix after the softmax is applied to each row (right). Row i says how much item i draws on each item of the sequence.

Every piece is a construction from earlier chapters. QK^T is an $n \times n$ table of dot products (Chapter 2), computed as one matrix product (Chapter 6). The division by $\sqrt{d_k}$ keeps the scores from growing with dimension (Chapter 4). The softmax turns each row into weights that sum to one, and multiplying by V makes each output a weighted average of value vectors — a linear combination (Chapter 1) with coefficients chosen by agreement. The shapes also reveal the cost: QK^T has n^2 entries, which is why the length of the sequence, not the width of the model, is the dominant expense for long inputs.

20.3 What attention can compare: a rank limit

Look at a single score. It is a quadratic expression in the two input vectors, sandwiching one fixed matrix.

$$s_{ij} = \frac{(W_Q \mathbf{x}_i) \cdot (W_K \mathbf{x}_j)}{\sqrt{d_k}} = \frac{1}{\sqrt{d_k}} \mathbf{x}_i^T \left(W_Q^T W_K \right) \mathbf{x}_j \tag{20.3}$$

Theorem 20.1 Attention scores are a low-rank bilinear form

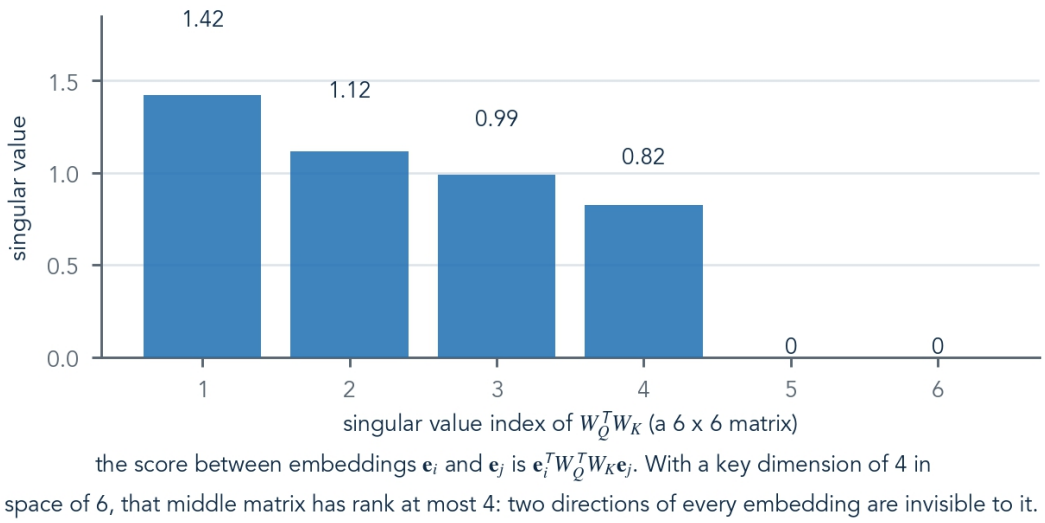
With W_Q, $W_K \in \mathbb{R}^{dk \times d}$, the matrix $B = W_Q{}^T W_K \in \mathbb{R}^{d \times d}$ has rank at most d_k. Consequently there is a subspace of dimension at least $d - d_k$ such that adding any vector from it to a key changes no score.

Proof

B is a product with a factor of size $d_k \times d$, so by Theorem 7.4 its rank is at most d_k.

By rank–nullity (Theorem 7.1) the null space of B has dimension at least $d - d_k$.

If $\mathbf{n}$ is in that null space then $\mathbf{x}_i{}^T B(\mathbf{x}_j + \mathbf{n}) = \mathbf{x}_i{}^T B \mathbf{x}_j$, so the score is unchanged. ■



the score between embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ is $\mathbf{e}_i^T W_Q^T W_K \mathbf{e}_j$. With a key dimension of 4 in a space of 6, that middle matrix has rank at most 4: two directions of every embedding are invisible to it.

FIGURE 20.2 The singular values of $W_Q{}^T W_K$ for a head with key dimension 4 in a space of dimension 6. Only four are nonzero: two directions of every embedding are invisible to the comparison this head makes.

> **A single attention head does not compare two vectors in full. It compares their projections onto a subspace of dimension at most d_k, chosen by training. A layer with several heads is a sum of such low-rank comparisons, each looking at a different subspace — which is why many narrow heads, rather than one wide one, is the standard design.**

Worked Example 20.1 Counting what a head sees

A model has $d = 4096$ and 32 attention heads, each with $d_k = 128$.

Each head's bilinear matrix $W_Q{}^T W_K$ has rank at most 128, so it ignores a subspace of dimension at least $4096 - 128 = 3968$ of every key.

Together the 32 heads have total key dimension $32 \times 128 = 4096 = d$, so, if their subspaces are different, between them they can attend to every direction of the space. The design splits one full-rank comparison into 32 rank-128 comparisons that can each specialise.

20.4 The residual stream as a shared space

In a transformer, each layer does not replace its input. It computes an update and adds it: $\mathbf{x} \leftarrow \mathbf{x} + \text{Attn}(\mathbf{x})$ and then $\mathbf{x} \leftarrow \mathbf{x} + \text{MLP}(\mathbf{x})$. The vector that flows through the model, receiving these additions, is often called the residual stream, and linear algebra suggests a clear way to think about it: a single vector space shared by all layers, into which each layer writes along some directions and from which later layers read along others.

Reading along a direction is a dot product, and writing along a direction is adding a multiple of a vector. That picture is exactly the linear representation hypothesis of Section 4.4, and it is what the sparse-dictionary methods described there were built to test. Their reported successes during 2024 and 2025 — many directions in the residual stream of large models corresponding to recognisable properties, far more directions than the stream has dimensions — are empirical support for a picture whose mathematics is entirely in Part I.

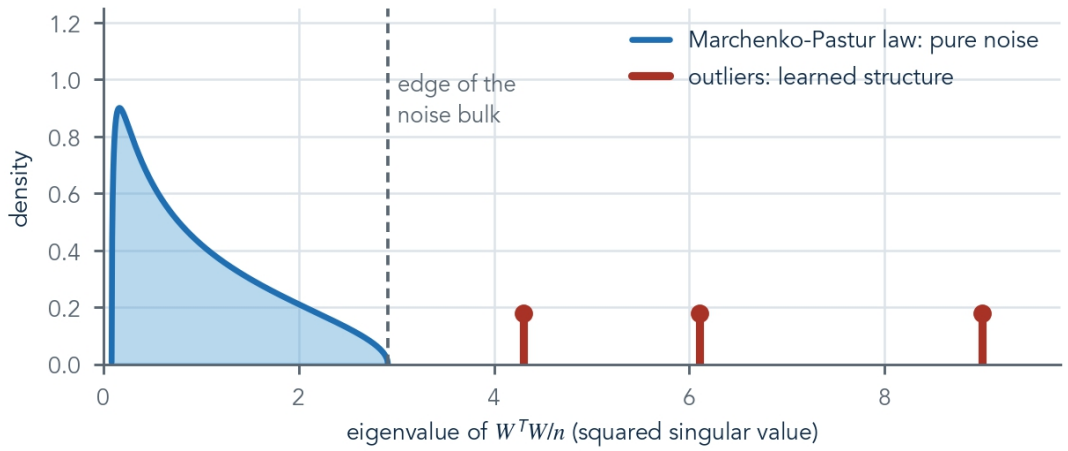
In machine learning. The same reading explains why low-rank adapters (Chapter 16) are so effective at changing a model's

behaviour: an update of rank r to a weight matrix changes what a layer writes into, or reads from, the shared space along only r new directions. It also explains a common diagnostic, the linear probe: fit a least-squares or logistic model (Chapter 11) from the residual stream to some property of the input, and if it succeeds, the property is present, linearly readable, along the direction the probe found.

20.5 The spectra of trained weights

A weight matrix before training is usually filled with independent random numbers. Random matrix theory describes exactly what its singular values look like: for an $n \times p$ matrix of independent entries with variance $1/n$ and $p/n = q$, the squared singular values fill an interval with a predictable density, the Marchenko–Pastur law, and almost nothing falls outside it.

$$\rho(\lambda) = \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{2\pi q \lambda}, \qquad \lambda_\pm = \left(1 \pm \sqrt{q}\right)^2 \tag{20.4}$$



the squared singular values of a large matrix of independent noise fill a predictable bulk with a sharp edge. In a trained weight matrix, the directions that matter appear as values outside it.

FIGURE 20.3 The Marchenko–Pastur density of squared singular values for a matrix of pure noise with $q = 0.5$, and its sharp upper edge. In trained weight matrices, the directions that carry learned structure appear as singular values beyond that edge.

Training changes this picture in a characteristic way. Studies of the spectra of trained networks have reported that the bulk remains broadly noise-like while a number of singular values separate from it and grow — directions along which the matrix has become a structured map rather than a random one. The phenomenon is the matrix version of the scree plot of Figure 17.2: a few strong components standing out from a sea of small ones, and it is one reason low-rank methods, spectral diagnostics and the orthogonalised updates of Chapter 19 all find purchase on trained models.

Worked Example 20.2 Attention between two items

Let two items have embeddings $(1, 0)$ and $(0, 1)$, with $W_Q = W_K = W_V = I$ and $d_k = 2$.

$QK^T = I$, so the scaled scores are 0.707 on the diagonal and 0 off it. The softmax of each row gives the weights $e^{0.707}/(e^{0.707} + 1) \approx 0.67$ and 0.33.

The first output is $0.67(1, 0) + 0.33(0, 1) = (0.67, 0.33)$, and the second is $(0.33, 0.67)$. Each item keeps two thirds of itself and borrows one third of the other — the simplest picture of how attention mixes information along a sequence.

20.6 The whole book in one picture

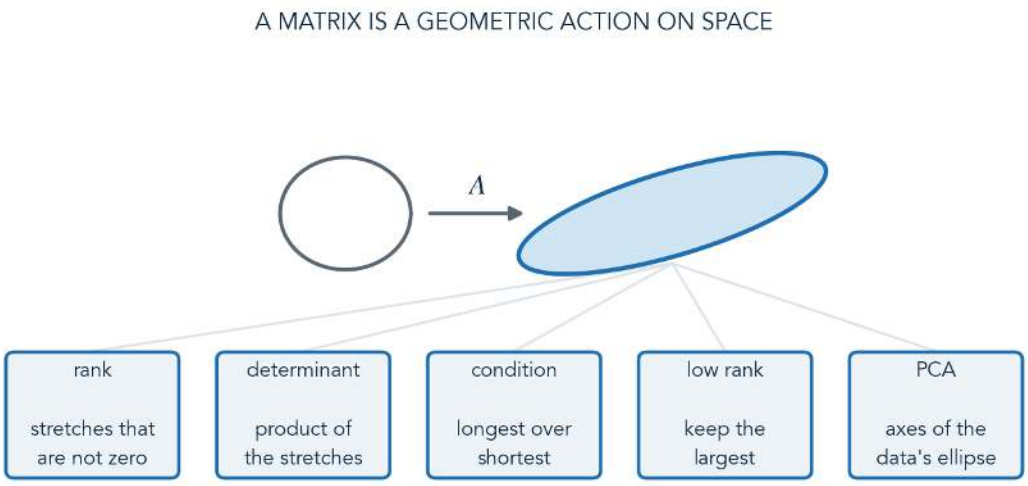


FIGURE 20.4 A circle, the ellipse a matrix turns it into, and five quantities read directly off it: rank, determinant, condition number, low-rank approximation and principal components.

TABLE 20.1 The components of a modern model and the chapters that explain them.

component	what it is, linear-algebraically	chapters
embedding table	a matrix of vectors; lookup is selecting a row	1, 2, 4
linear layer	a map $W\mathbf{x} + \mathbf{b}$ followed by a fold	5, 6
attention	dot products QK^T, a softmax, a weighted average	2, 6, 7
multi-head attention	a sum of low-rank bilinear forms	7, 16, 20
residual stream	a shared space read and written along directions	3, 4
backpropagation	a product of Jacobians taken from the left	6, 18
optimiser	a choice of geometry for the step	12, 14, 19
fine-tuning adapter	a low-rank update BA	15, 16
interpretability probe	least squares from a representation	11, 17

> **A matrix is a geometric action on space, and a learning machine is a long composition of such actions**

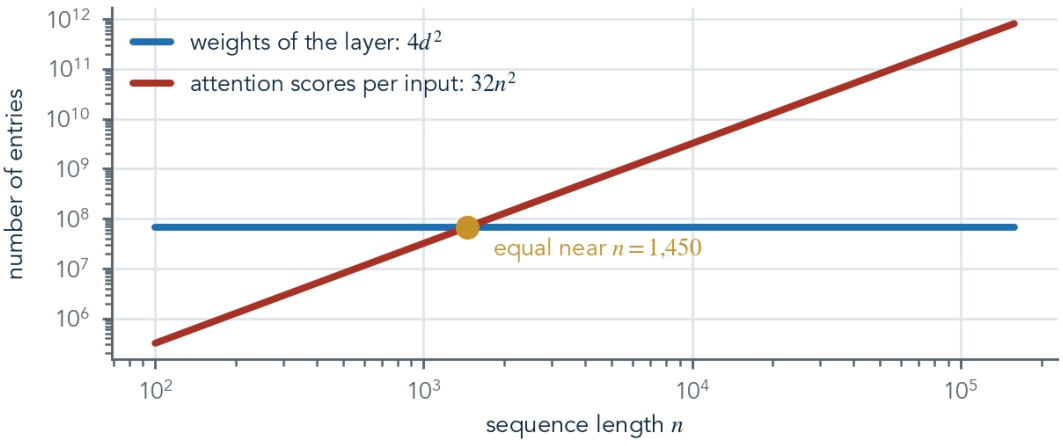
interrupted by simple folds. Every operation in it can be understood by asking what it does to directions — which it stretches, which it destroys, which it leaves on their own line — and the singular value decomposition answers that question for every matrix there is. The mathematics is short. That it is enough is one of the genuinely hopeful facts of our time.

Worked Example 20.3 The parameters of an attention layer

A model has $d = 4096$ and 32 heads with $d_k = 128$, so the heads' query, key and value matrices together have the shape of three 4096×4096 matrices, followed by an output matrix of the same size.

Each has $4096^2 = 16{,}777{,}216$ entries, so the layer has $4 \times 16{,}777{,}216 = 67{,}108{,}864$ weights.

The count does not depend on the length of the sequence at all; the n^2 cost of Section 20.2 is a cost of computation, paid afresh for every input, not a cost of storage. Figure 20.5 compares the two as the input grows.



for $d = 4096$ and 32 heads, the layer stores 67 million weights however long the input. The score matrices are rebuilt for every input and grow with the square of its length.

FIGURE 20.5 The attention layer of Example 20.3. Its stored weights are fixed at about 67 million, while the score matrices it forms grow as the square of the sequence length and overtake the weights near $n = 1{,}450$.

Key ideas

- An embedding table is a matrix of vectors, and output scores are dot products with those vectors.
- Attention stacks queries, keys and values as matrices, and its n^2 cost is visible in the shape of QK^T.
- Each head compares vectors through a bilinear form of rank at most d_k, so it sees only a subspace of each embedding.
- The residual stream is a shared space that layers read from and write to along directions.
- Structure learned in a weight matrix appears as singular values beyond the edge of the random-matrix bulk.

Exercises

1. A vocabulary of 50,000 symbols is embedded in $d = 1024$. How many numbers are in the embedding table, and what is the largest possible rank of that matrix?
2. For a sequence of length 8,000, how many entries does QK^T have? By what factor does that grow if the sequence doubles in length?
3. Show that multiplying W_Q by an invertible $d_k \times d_k$ matrix M and W_K by $(M^T)^{-1}$ leaves every attention score unchanged. What does this say about which part of the query and key weights is actually determined by training?
4. A linear probe finds a unit direction $\mathbf{u}$ such that $\mathbf{u}\cdot\mathbf{x}$ predicts a property well. Explain how adding $t\mathbf{u}$ to $\mathbf{x}$ could be used to test whether the model uses that direction.
5. For $q = 0.5$, compute the edges $\lambda_{\pm}$ of the Marchenko–Pastur law and check them against Figure 20.3.
6. For $n = 1000$ items and $d_k = 64$, what is the largest possible rank of QK^T?
7. How many parameters are in one head's W_Q, W_K and W_V when $d = 512$ and $d_k = 64$?
8. Why is each row of a softmax attention matrix a probability vector?

APPENDIX A · NOTATION AND THE FACTS USED

This appendix collects the notation of the book and every fact the later chapters rely on, each with the chapter where it is proved. It is meant to be consulted rather than read.

Notation

TABLE A.1 Notation used throughout.

symbol	meaning	first used
$\mathbf{v}$, v_i	a vector and its i-th component	Chapter 1
$\mathbb{R}^n$	the space of lists of n real numbers	Chapter 1
$\mathbf{v}^T$	transpose: a column laid on its side	Chapter 1
$\mathbf{u}\cdot\mathbf{v}$, $\mathbf{u}^T\mathbf{v}$	dot product	Chapter 2
$\|\mathbf{v}\|$	Euclidean length	Chapter 1
$\mathbf{e}_i$	the i-th standard basis vector	Chapter 3
A, A_{ij}	a matrix and its entry in row i, column j	Chapter 5
I	the identity matrix	Chapter 6
A^{-1}	inverse	Chapter 6
col, null	column space, null space	Chapter 7
rank	dimension of the column space	Chapter 7
det A	signed volume factor	Chapter 8
λ, Λ	eigenvalue, diagonal matrix of eigenvalues	Chapter 13
σ_i, Σ	singular value, matrix of singular values	Chapters 12, 15
(A)	condition number σ_1/σ_n	Chapter 12
$\|A\|_2$, $\|A\|_F$	spectral norm, Frobenius norm	Chapter 15
A^+	pseudoinverse	Chapter 15
∇f, J_F, H	gradient, Jacobian, Hessian	Chapters 8, 18
$\odot$	entry-by-entry product	Chapter 18

Facts about vectors

- $\mathbf{u}\cdot\mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$, and $|\mathbf{u}\cdot\mathbf{v}| \le \|\mathbf{u}\|\|\mathbf{v}\|$ (Theorems 2.1, 2.2).
- The projection of $\mathbf{v}$ onto $\mathbf{u}$ is $(\mathbf{u}\cdot\mathbf{v}/\mathbf{u}\cdot\mathbf{u})\,\mathbf{u}$, and the residual is perpendicular to $\mathbf{u}$ (Section 2.2).
- In a basis, every vector has unique coordinates; in an orthonormal basis they are dot products (Theorem 3.1, Equation 10.1).
- Random unit vectors in $\mathbb{R}^d$ satisfy $P(|\mathbf{u}\cdot\mathbf{v}| \ge \varepsilon) \le 2e^{-d\varepsilon 2/2}$, and at least $e^{d\varepsilon 2/4}$ pairwise ε-orthogonal unit vectors exist (Theorems 4.1, 4.2).

Facts about matrices

- The columns of A are the images of the standard basis vectors (Definition 5.2).
- $(AB)\mathbf{x} = A(B\mathbf{x})$; products are associative but not commutative; $(AB)^T = B^T A^T$ and $(AB)^{-1} = B^{-1}A^{-1}$ (Chapter 6).
- rank(A) + null(A) = n, and row rank equals column rank (Theorems 7.1, 7.2).
- The row space and null space are orthogonal complements, as are the column space and left null space (Theorem 7.3).
- rank(AB) $\le$ min(rank A, rank B) (Theorem 7.4).
- det(AB) = det Adet B; det $A \ne 0$ exactly when A is invertible (Theorems 8.2, 8.3).
- $Q^T Q = I$ preserves lengths and angles (Theorem 10.1).
- A projection satisfies $P^2 = P = P^T$ and is QQ^T for an orthonormal basis Q (Theorem 10.3).
- Least squares solves $A^T A\mathbf{c} = A^T\mathbf{y}$, stably via $R\mathbf{c} = Q^T\mathbf{y}$ (Theorem 11.1, Equation 11.3).
- $\|\delta\mathbf{x}\|/\|\mathbf{x}\| \le (A)\,\|\delta\mathbf{b}\|/\|\mathbf{b}\|$, and $(A^T A) = (A)^2$ (Theorem 12.1, Equation 12.2).

Facts about eigenvalues and singular values

- The eigenvalues of A are the roots of $\det(A - \lambda I) = 0$; they add to the trace and multiply to the determinant (Section 13.2).
- If A has n independent eigenvectors, $A = V \Lambda V^{-1}$ and $A^k = V \Lambda^k V^{-1}$ (Theorem 13.1).
- A symmetric matrix has real eigenvalues and orthonormal eigenvectors: $S = Q \Lambda Q^T$ (Theorem 14.1).
- The Rayleigh quotient of a symmetric matrix lies between its smallest and largest eigenvalues (Theorem 14.2).
- Every matrix has an SVD $A = U \Sigma V^T$; the singular values are the square roots of the eigenvalues of $A^T A$ (Theorem 15.1).
- The truncated SVD is the best approximation of each rank, with errors σ_{k+1} and $\sqrt{\sum_{i>k} \sigma_i^2}$ (Theorem 16.1).
- The principal directions of centred data are the right singular vectors of the data matrix (Theorem 17.1, Equation 17.2).

Facts about derivatives and optimisation

- The gradient points in the direction of steepest ascent and is perpendicular to level sets (Theorem 18.1).
- The Jacobian of a composition is the product of Jacobians (Theorem 18.2).
- Reverse-mode differentiation computes a full gradient at a constant multiple of the cost of the function (Theorem 18.4).
- Gradient descent on a quadratic converges for $\eta < 2/\lambda_{\max}$, at rate $(-1)/(+1)$ with the best fixed step (Theorem 19.1).
- Steepest descent under the spectral norm steps along $-UV^T$, and the iteration $X \; arrow \; 3/2X - 1/2XX^TX$ drives every singular value to one (Theorems 19.2, 19.3).

APPENDIX B · THE DECOMPOSITIONS AT A GLANCE

Seven factorisations account for nearly all of computational linear algebra. Each writes a matrix as a product of simpler matrices, and each simple factor has a geometric meaning: triangular factors are sequences of shears, orthogonal factors are rotations and reflections, diagonal factors are stretches along axes.

TABLE B.1 The major decompositions: what they say and when they exist.

decomposition	form	exists when	the factors
LU	$PA = LU$	any square matrix, with row swaps P	lower and upper triangular
Cholesky	$S = LL^T$	S symmetric positive definite	lower triangular with positive diagonal
QR	$A = QR$	any matrix	orthonormal columns, upper triangular
eigendecomposition	$A = V\Lambda V^{-1}$	A square with n independent eigenvectors	eigenvectors, eigenvalues
spectral	$S = Q\Lambda Q^T$	S symmetric	orthonormal eigenvectors, real eigenvalues
singular value	$A = U\Sigma V^T$	every matrix	two orthogonal matrices, non-negative stretches
polar	$A = (UV^T)(V\Sigma V^T)$	every square matrix	a rotation times a symmetric stretch

TABLE B.2 What each decomposition is for, what it costs for a dense $n \times n$ matrix, and where it appears in machine learning.

decomposition	main use	cost	in machine learning
LU	solving square systems	$n^3/3$	inside general-purpose solvers
Cholesky	solving with covariances and Hessians	$n^3/6$	Gaussian processes, Newton steps, whitening
QR	least squares, orthonormal bases	about $2n^3/3$	stable regression, orthogonal initialisation
eigendecomposition	powers and long-run behaviour	about $10n^3$, iterative	Markov chains, PageRank, recurrent dynamics
spectral	quadratic forms, definiteness	about $4n^3$, iterative	Hessian analysis, spectral clustering
singular value	rank, norms, best approximation	about $10n^3$, iterative	PCA, low-rank adapters, recommenders, weight spectra
polar	the nearest orthogonal matrix	one SVD, or matrix-product iterations	orthogonalised optimisers (Chapter 19)

The costs are order-of-magnitude guides for dense matrices; sparse or structured matrices, and randomised methods for the leading components (Section 16.4), can be far cheaper. Two habits follow from the second table. Prefer the factorisation that exploits the structure you have — Cholesky for symmetric positive definite matrices, QR rather than the normal equations for least squares. And when only a few leading components are needed, never compute the rest.

Reading a decomposition geometrically

TABLE B.3 The geometric vocabulary of the factors.

factor	geometric action	property
orthogonal Q, U, V	rotate or reflect	preserves lengths and angles; $Q^{-1} = Q^T$
diagonal Σ, Λ	stretch along the coordinate axes	condition number is the ratio of extreme entries
triangular L, U, R	a sequence of shears and axis stretches	determinant is the product of the diagonal
permutation P	reorder the coordinates	an orthogonal matrix with 0/1 entries
symmetric positive definite S	stretch along perpendicular axes	all eigenvalues positive

APPENDIX C · ANSWERS TO THE EXERCISES

Answers are given briefly. Where an exercise asks for a proof, the key step is shown; where it asks for a number, the number is checked when checking is quick.

Chapter 1

1. $\mathbf{u}+\mathbf{v}=(4,1)$, $\mathbf{u}-\mathbf{v}=(-2,3)$, $2\mathbf{u}-\mathbf{v}=(-1,5)$.
2. $(4,2)=2(1,0)+2(1,1)$, and $(4,2)=2(2,1)+0(1,3)$. The point is the same; the coefficients are its address in two different bases.
3. Length $\sqrt{4+1+4}=3$; unit vector $(2,-1,2)/3$.
4. If $x+y+z=0$ for two vectors, it holds for their sum and for any multiple, and $\mathbf{0}$ qualifies. The set with $x+y+z=1$ does not contain $\mathbf{0}$, and the sum of two of its members has coordinate sum 2.
5. Area in square metres takes values in the hundreds while the other coordinates are single digits, so differences in area dominate the distance. Standardise each coordinate by subtracting its mean and dividing by its standard deviation.
6. $\|c\mathbf{v}\|=\sqrt{\sum c^2 v_i^2}=\sqrt{c^2}\sqrt{\sum v_i^2}$, and $\sqrt{c^2}$ is the size of c.
7. The difference is $(1,-2,-2,0)$, so the distance is $\sqrt{1+4+4}=3$.
8. Scaling any vector by the number 0 gives the zero vector, and a vector space is closed under scaling.

Chapter 2

1. $\mathbf{u}\cdot\mathbf{v} = 1$ and both lengths are $\sqrt{2}$, so $\cos\theta = 1/2$ and $\theta = 60°$.
2. $(1, -1, 1)$: its dot products with $(1, 1, 0)$ and $(0, 1, 1)$ are both zero.
3. The projection is $6/3(1, 1, 1) = (2, 2, 2)$; the residual $(1, -1, 0)$ has dot product 0 with $(1, 1, 1)$.
4. $\|\mathbf{u} + \mathbf{v}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 + 2\,\mathbf{u}\cdot\mathbf{v}$, so equality holds exactly when $\mathbf{u}\cdot\mathbf{v} = 0$. In the plane this is Pythagoras' theorem.
5. Multiplying every score by ten concentrates nearly all the weight on the largest score; multiplying by one tenth makes the weights nearly equal. Dividing by $\sqrt{d}$ keeps scores from growing with dimension, so the softmax neither saturates nor flattens as d changes.
6. The dot product is $2 + 2 - 4 = 0$, so the cosine similarity is 0: the vectors are orthogonal.
7. $\mathbf{u}\cdot\mathbf{u} = 1$, so the denominator in Definition 2.2 disappears.
8. The scores double to 2.83, 0 and 1.41, giving weights of about 0.77, 0.05 and 0.19: the softmax sharpens and attention concentrates on the best match.

Chapter 3

1. The span of $(1, 0, 1)$ and $(0, 1, 0)$ is all vectors (a, b, a). So $(1, 1, 1)$ is in it and $(1, 2, 3)$ is not.
2. $-3(1, 2) + 1(3, 5) + 1(0, 1) = (0, 0)$.
3. $(4, 6) = 8/3(1, 2) + 2/3(2, 1)$, so the coordinates are $(8/3, 2/3)$.
4. $x = 2y - z$, so a basis is $(2, 1, 0)$ and $(-1, 0, 1)$; the dimension is 2.
5. $\mathbb{R}^8$ has dimension 8, so more than 8 vectors are always dependent, and fewer than 8 cannot span.
6. Both contain $\mathbf{0}$ and are closed under addition and scaling.
7. The third vector is the sum of the first two, and the first two are independent, so the dimension is 2.
8. $c_1 = (3 - 2)/5 = 0.2$ and $c_2 = (6 + 1)/5 = 1.4$; indeed $0.2(1, 2) + 1.4(2, -1) = (3, -1)$.

Chapter 4

1. The bound is $2e^{-500(0.15)2/2} = 2e^{-5.625} \approx 0.0072$.
2. $0.9^{100} = e^{100\ln 0.9} \approx e^{-10.5} \approx 2.7 \times 10^{-5}$.
3. The squared length has expected value d; the dot product has expected value 0 (and variance d).
4. Exactly perpendicular directions number only 4,096, while Table 4.1 guarantees about 6×10^{17} with $|\cos| < 0.2$; insisting on exact perpendicularity would use a vanishing fraction of the available directions.
5. The lemma only needs each of the $N(N-1)/2$ pairwise differences to keep its length, and a random projection to k dimensions distorts any single fixed vector by more than ε with a probability that depends only on k and ε. A union bound over the pairs gives k of order $\varepsilon^{-2}\log N$, with no reference to the original dimension.
6. At most $2e^{-100\times0.09/2} = 2e^{-4.5} \approx 0.022$.
7. At least $e^{2000\times0.01/4} = e^{5}$, about 148.
8. The dot product is a sum of d independent terms of variance one, so its variance is d and its typical size is $\sqrt{d}$.

Chapter 5

1. Columns $(0, 1)$ and $(1, 0)$; it sends $(3, 1)$ to $(1, 3)$.
2. Columns $(1, 0)$ and $(0, 0)$; it sends every vector $(0, y)$ on the y-axis to $\mathbf{0}$.
3. $T(0, 0) = (1, 0) \neq \mathbf{0}$, so T is not linear. In three coordinates it is the matrix with rows $(1, 0, 1)$, $(0, 1, 0)$, $(0, 0, 1)$ acting on $(x, y, 1)$.
4. By columns, $2(1, 2, 0) + 5(3, -1, 1) = (17, -1, 5)$; by rows, $(1, 3)\cdot(2, 5) = 17$, $(2, -1)\cdot(2, 5) = -1$, $(0, 1)\cdot(2, 5) = 5$.
5. $2048 \times 512 = 1{,}048{,}576$ weights and $2{,}048$ biases. The outputs $W\mathbf{x}$ form a space of dimension at most 512.
6. Columns $(-1, 0)$ and $(0, -1)$: the matrix $-I$.
7. Columns $(1, 1)$ and $(1, -1)$; the image of $(2, 3)$ is $2(1, 1) + 3(1, -1) = (5, -1)$.
8. $T(\mathbf{0}) = T(0\,\mathbf{v}) = 0\,T(\mathbf{v}) = \mathbf{0}$.

Chapter 6

1. AB has columns (2, 0) and (1, 1): stretch, then shear. BA has columns (2, 0) and (2, 1): shear, then stretch.
2. The matrix with columns (0, 0) and (1, 0) sends the y-axis onto the x-axis and the x-axis to zero, so applying it twice sends everything to zero.
3. $(AB)^T$ has rows (3, 5) and (1, 1), and so does B^TA^T.
4. The inverse has rows (1, −1) and (−1, 2), since $ad - bc = 1$; the product with the original is I.
5. At most 64, the width of the narrowest layer (Theorem 7.4).
6. AB has rows (5, 2) and (1, 1), so $(AB)^T$ has rows (5, 1) and (2, 1); B^TA^T gives the same.
7. $R^2 = -I$ is a half-turn, and two half-turns make a full turn, $R^4 = I$.
8. It maps $\mathbb{R}^2$ into a plane inside $\mathbb{R}^3$, so it cannot reach every vector of $\mathbb{R}^3$, and no matrix can undo it on all of $\mathbb{R}^3$.

Chapter 7

1. Rank 1; the column space is spanned by $(1, 2)$; the null space by $(-2, 1, 0)$ and $(-3, 0, 1)$.
2. Column space 6 and left null space 1 (in $\mathbb{R}^7$); row space 6 and null space 4 (in $\mathbb{R}^{10}$).
3. If $A^TA\mathbf{x} = \mathbf{0}$ then $\mathbf{x}^TA^TA\mathbf{x} = \|A\mathbf{x}\|^2 = 0$, so $A\mathbf{x} = \mathbf{0}$; the converse is immediate.
4. All solutions are $(1, 2, 0) + t(1, -1, 1)$. Perpendicularity to $(1, -1, 1)$ requires $3t - 1 = 0$, giving $(4/3, 5/3, 1/3)$.
5. The rank of a 100×3 matrix is at most 3, so rank 4 is impossible; a null space of dimension zero means rank exactly 3.
6. One: every column is a multiple of $\mathbf{u}$.
7. For example, rows $(1, -1, 0)$, $(0, 1, -1)$ and $(1, 0, -1)$: each row is perpendicular to $(1, 1, 1)$ and the rank is 2.
8. Only $\mathbf{0}$; the matrix is invertible and $A\mathbf{x} = \mathbf{b}$ has exactly one solution for every $\mathbf{b}$.

Chapter 8

1. $3 \cdot 2 - 1 \cdot 1 = 5$: area 5, and the positive sign means orientation is preserved.
2. Elimination gives pivots 1, -5 and $11/5$, so the determinant is -11.
3. $\det(Q^T Q) = (\det Q)^2 = \det I = 1$, so $\det Q = \pm 1$: orthogonal maps preserve volume, and only reflections reverse orientation.
4. Scaling both sides of a parallelogram by c scales its area by c^2.
5. The Jacobian has rows $(2u, -2v)$ and $(2v, 2u)$, with determinant $4(u^2 + v^2)$. It is zero at the origin, where the map crushes a neighbourhood of area into a point to second order.
6. Zero: the rows are dependent, since row 1 plus row 3 is twice row 2, so the map flattens space into a plane.
7. $2 \times (-1) \times 4 = -8$.
8. $2^4 \times 3 = 48$.

Chapter 9

1. Subtracting 3 times the first equation from the second gives $-2y = -4$, so $y = 2$ and $x = 1$.
2. L has rows (1, 0), (3, 1) and U has rows (1, 2), (0, 2). Solving $L\mathbf{c} = (5, 19)$ gives $\mathbf{c} = (5, 4)$, and $U\mathbf{x} = \mathbf{c}$ gives $\mathbf{x} = (1, 2)$.
3. Entry (i, j) of the product with $i < j$ is $\sum_k L_{ik}M_{kj}$; a nonzero term would need $k \le i$ and $k \ge j$, which is impossible.
4. At $k = 4$ the second pivot vanishes, and after swapping rows the third pivot is zero too: the columns are dependent and the map flattens space to a plane.
5. Refactoring each time costs about $1000 \times 500^3/3 \approx 4.2 \times 10^{10}$ operations; factoring once and solving 1,000 times costs about $4.2 \times 10^7 + 1000 \times 500^2 \approx 2.9 \times 10^8$ — roughly 140 times less.
6. Subtracting half the first equation from the second gives $2.5y = 2.5$, so $y = 1$ and $x = 1$.
7. The pivots are 1 and 0.0001: the matrix is nearly singular, and solutions will be very sensitive to the data.
8. Each unknown is found by one substitution involving at most n values already known, so the total work is about $n^2/2$.

Chapter 10

1. The dot product of the columns is $(-12 + 12)/25 = 0$ and both have length 1. Coordinates: $(30 + 20)/5 = 10$ and $(-40 + 15)/5 = -5$.
2. $\mathbf{q}_1 = (1, 0, 1)/\sqrt{2}$, $\mathbf{q}_2 = (1, 2, -1)/\sqrt{6}$, $\mathbf{q}_3 = (-1, 1, 1)/\sqrt{3}$.
3. $P = 1/9$ times the matrix with rows $(1, 2, 2)$, $(2, 4, 4)$, $(2, 4, 4)$; $P^2 = \mathbf{u}\mathbf{u}^T\mathbf{u}\mathbf{u}^T/81 = \mathbf{u}\mathbf{u}^T/9 = P$.
4. $(I - P)^2 = I - 2P + P^2 = I - P$, and $I - P$ projects onto the orthogonal complement of the subspace.
5. A projection onto a proper subspace flattens space into fewer dimensions, so every volume becomes zero.
6. The columns are perpendicular unit vectors, so $Q^TQ = I$; the determinant is $-1/2 - 1/2 = -1$, so the matrix is a reflection.
7. The projection is $(1, 2, 0)$, performed by the diagonal matrix with entries 1, 1 and 0.
8. Some remainder $\mathbf{w}_k$ is the zero vector, which cannot be normalised: that vector added no new direction.

Chapter 11

1. The normal equations $4c_0 + 10c_1 = 16$ and $10c_0 + 30c_1 = 47$ give $c_1 = 1.4$ and $c_0 = 0.5$. The residuals $0.1, -0.3, 0.3, -0.1$ sum to zero.
2. The first normal equation says $\sum(c_0 + c_1 x_i - y_i) = 0$; dividing by m gives $c_0 + c_1 \bar{x} = \bar{y}$.
3. A is 5×3, $A^T A$ is 3×3 and $A^T \mathbf{y}$ is 3×1.
4. $c = 4/(2 + \lambda)$: as $\lambda \to 0$ it tends to 2, the mean; as $\lambda \to \infty$ it tends to 0.
5. The normal equations say $A^T(\mathbf{y} - A\mathbf{c}) = \mathbf{0}$, which is the definition of the left null space.
6. $c = \sum x_i y_i / \sum x_i^2 = 29/14 \approx 2.07$.
7. $A^T A$ has rows $(3, 3)$ and $(3, 5)$ and $A^T \mathbf{y} = (2, 3)$, so $c_1 = 0.5$ and $c_0 = 1/6$.
8. $A^T A + \lambda I$ has every eigenvalue at least $\lambda > 0$, so it is positive definite and invertible.

Chapter 12

1. Singular values 10 and 0.1, condition number 100; the circle becomes an ellipse with half-axes 10 and 0.1.
2. $\det A = 10^{-150}$ but $\kappa(A) = 1$: a uniform scaling is perfectly conditioned, and the determinant only reflects the overall scale.
3. About $8 - 6 = 2$ digits.
4. An orthogonal matrix carries the unit sphere to itself, so every singular value is 1.
5. The columns are nearly parallel; the stretch across their difference is about 10^{-3} of the stretch along them, so is of order 10^{3}. Remedies: remove or average one of the features, or use ridge regression.
6. 10^{6}: about six digits are lost.
7. It is the largest singular value divided by the smallest, and the largest is never the smaller of the two.
8. $1.99/0.01 = 199$.

Chapter 13

1. Eigenvalues 5 and 2, with eigenvectors $(1, 1)$ and $(1, -2)$; they add to 7 and multiply to 10.
2. $\det(T - \lambda I)$ is the product of the diagonal entries $t_{ii} - \lambda$.
3. $A^{10} = V\text{diag}(3^{10}, 1)\, V^{-1}$; since $(1, 0) = 1/2(1, 1) + 1/2(1, -1)$, $A^{10}(1, 0) = (29525, 29524)$, almost exactly along $(1, 1)$.
4. $P^2 = P$ gives $\lambda^2 = \lambda$, so λ is 0 or 1. Eigenvectors for 1 are the vectors in the subspace; for 0, the vectors perpendicular to it.
5. Stationary distribution $(5/6, 1/6)$.
6. Eigenvalues 1 and -1, with eigenvectors $(1, 1)$ and $(1, -1)$: the matrix reflects across the diagonal.
7. $A^2\mathbf{v} = 9\mathbf{v}$ and $A^{-1}\mathbf{v} = \mathbf{v}/3$.
8. The matrix is triangular, so the eigenvalues are 0.5 and 0.9. Both are below one, so every sequence decays to $\mathbf{0}$, eventually at the rate 0.9 per step.

Chapter 14

1. Eigenvalues 4 and 2 with eigenvectors $(1, 1)/\sqrt{2}$ and $(1, -1)/\sqrt{2}$, so $S = 4\,\mathbf{q}_1\mathbf{q}_1{}^T + 2\,\mathbf{q}_2\mathbf{q}_2{}^T$.
2. Eigenvalues 3 and -1: indefinite, a saddle.
3. $\mathbf{x}^T A^T A\mathbf{x} = \|A\mathbf{x}\|^2 \geq 0$, and it is zero only if $A\mathbf{x} = \mathbf{0}$, which forces $\mathbf{x} = \mathbf{0}$ when the columns are independent.
4. The form has matrix with rows (1, 2) and (2, 1); the maximum on the unit circle is the largest eigenvalue, 3, at $(1, 1)/\sqrt{2}$.
5. Standard deviations 3 and 1; total variance 10.
6. No. Its eigenvalues are 5 and 0, so it is positive semidefinite but not definite: the form vanishes along $(1, -2)$.
7. $S_{ii} = \mathbf{e}_i{}^T S\mathbf{e}_i > 0$.
8. The eigenvalues are 4 and -2, so the form is $4y_1{}^2 - 2y_2{}^2$: a saddle.

Chapter 15

1. A^TA has eigenvalues 4 and 0, so $\sigma_1 = 2$ and $\sigma_2 = 0$, with $\mathbf{u}_1 = \mathbf{v}_1 = (1, 1)/\sqrt{2}$; indeed $A = 2\,\mathbf{u}_1\mathbf{v}_1{}^T$.
2. For orthogonal Q, $Q^TQ = I$ has all eigenvalues 1. For symmetric positive semidefinite S, $S^TS = S^2$ has eigenvalues λ_i^2, so $\sigma_i = \lambda_i$.
3. $\|A\|_2 = 3\sqrt{5} \approx 6.71$, $\|A\|_F = \sqrt{50} \approx 7.07$, $= 3$.
4. $A^+ = (1, 2, 2)/9$ as a row, so $x = 5/9$.
5. Each of the r rank-one layers needs one singular value, one vector of length m and one of length n.
6. $\sigma_1 = 2$ with $\mathbf{u}_1 = -\mathbf{e}_1$ and $\mathbf{v}_1 = \mathbf{e}_1$; $\sigma_2 = 1$ with $\mathbf{u}_2 = \mathbf{v}_2 = \mathbf{e}_2$.
7. $\|A\|_F^2 = \mathrm{tr}(A^TA)$, which is the sum of the eigenvalues of A^TA: the squared singular values.
8. The projection itself.

Chapter 16

1. Spectral error $\sigma_3 = 2$; Frobenius error $\sqrt{4 + 1 + 0.25} \approx 2.29$; energy kept $125/130.25 \approx 96$.
2. Such a matrix has rank one, so $\sigma_2 = 0$ and the rank-one truncation discards nothing.
3. $20 \times (5000 + 3000 + 1) = 160{,}020$ numbers, about 1.1% of the 15,000,000 entries.
4. $2r/d = 0.1$ gives $r \approx 51$.
5. With exactly k random columns, there is a real chance that their span nearly misses part of the dominant subspace; a few extra columns make that chance vanish rapidly, so the captured subspace contains the dominant one with overwhelming probability.
6. $16/25 = 64$.
7. $2 \times 1000 \times 8 = 16{,}000$ parameters, which is 1.6% of the 1,000,000 in a full update.
8. Noise spreads its energy thinly across many small singular values, while structure concentrates in a few large ones, so discarding the small ones removes mostly noise.

Chapter 17

1. Eigenvalues 5 and 0, with eigenvectors $(2, 1)/\sqrt{5}$ and $(1, -2)/\sqrt{5}$. The zero eigenvalue means the data has no variance at all along $(1, -2)$: every centred observation satisfies $x_1 = 2x_2$, so the two measurements carry one number's worth of information.
2. $\mathrm{Cov}(\mathbf{q}_1 \cdot \mathbf{x}, \mathbf{q}_2 \cdot \mathbf{x}) = \mathbf{q}_1{}^T C \mathbf{q}_2 = \lambda_2 \, \mathbf{q}_1 \cdot \mathbf{q}_2 = 0.$
3. Six: the cumulative shares are 94.2% after five components and 96.0% after six.
4. Without centring, the covariance is replaced by the matrix of raw second moments, and its top eigenvector points largely toward the mean of the data rather than along the direction in which the data varies.
5. Two long, parallel clusters stretched along the *x*-axis and separated slightly along the *y*-axis: the first principal direction is the *x*-axis, along which the classes do not differ at all.
6. The trace, 3.
7. Variance depends on units, so without standardising, the feature with the largest numbers dominates the first direction.
8. Two components keep about 99.8% of the variance; the tiny third eigenvalue says the data lie very close to a plane.

Chapter 18

1. $\nabla f = (2xy,\ x^2 + 3) = (4, 4)$ at $(1, 2)$; the direction of steepest descent is $-(1, 1)/\sqrt{2}$.
2. $\mathbf{x}^T A\mathbf{x} = ax_1^2 + (b + c)x_1x_2 + dx_2^2$, whose gradient $(2ax_1 + (b + c)x_2, (b + c)x_1 + 2dx_2)$ is $(A + A^T)\mathbf{x}$.
3. The Jacobian has rows (y, x), $(1, 1)$, $(2x, 0)$: three outputs and two inputs, so it is 3×2.
4. The Jacobians are 500×1000, 100×500 and 1×100. Forward mode first forms a 100×1000 matrix of 100,000 entries; reverse mode only ever holds a row vector, the largest being the final gradient of 1,000 entries.
5. At most 8 for one step. After k steps from zero the weights are a sum of k updates of rank at most 8, so their rank is at most $\min(8k, 512)$; after 100 steps that bound, 800, no longer restricts anything, although in practice such updates are often much closer to low rank than the bound requires.
6. $2\mathbf{x} = (2, -4, 4)$.
7. Rows $(1, 1)$ and $(1, -1)$, with determinant -2.
8. $\bar{x} = W_1\bar{z} = 0.5 \times 6 = 3$.

Chapter 19

1. The largest safe step is just below $2/4 = 0.5$; the best fixed step is $2/5 = 0.4$; the contraction factor is $(4 - 1)/(4 + 1) = 0.6$ per step.
2. Along the steepest direction the factor is $1 - \eta\lambda_{\max} = -1$, so that component of the error flips sign at every step and keeps its size forever.
3. The gradient is $H\mathbf{x} - \mathbf{b}$, so one Newton step gives $\mathbf{x} - H^{-1}(H\mathbf{x} - \mathbf{b}) = H^{-1}\mathbf{b}$, which is the minimiser, whatever $\mathbf{x}$ was.
4. The gradient is already diagonal, with $U = V = I$ and singular values 3 and 1, so $UV^T = I$. The orthogonalised step moves equally along both directions, where the gradient moved three times as far along the first.
5. From 0.2: 0.296, then 0.431. From 0.9: 0.9855, then 0.9997. Values near one converge quadratically, while small ones grow only by about half each step — which is why practical methods use tuned polynomials that lift small singular values faster. The small singular values are exactly the weak directions an orthogonalised update exists to strengthen.
6. $= 100$, the factor per step is $99/101$, and $\ln 1000/\ln(101/99) \approx 345$ steps.
7. One.
8. $\mathbf{u}\mathbf{v}^T$: the size s is discarded and the direction is kept.

Chapter 20

1. $50{,}000 \times 1024 = 51{,}200{,}000$ numbers, of rank at most 1,024.
2. $8000^2 = 64{,}000{,}000$ entries; doubling the length multiplies that by four.
3. The score becomes $\mathbf{x}_i^T W_Q^T M^T (M^T)^{-1} W_K \mathbf{x}_j = \mathbf{x}_i^T W_Q^T W_K \mathbf{x}_j$, unchanged. Only the product $W_Q^T W_K$, a bilinear form of rank at most d_k, is determined by the scores; the individual factors are not.
4. Add $t\mathbf{u}$ for several values of t and observe whether the model's outputs change in the way the property would predict, and compare with adding a random direction of the same length. A consistent, specific effect is evidence that the model reads the property along $\mathbf{u}$, not merely that it can be read there.
5. $\lambda_{\pm} = (1 \pm \sqrt{0.5})^2$, about 0.086 and 2.914, matching the dashed edge in Figure 20.3.
6. At most 64, since QK^T is the product of an $n \times d_k$ matrix and a $d_k \times n$ matrix.
7. $3 \times 64 \times 512 = 98{,}304$.
8. Its entries are exponentials divided by their sum, so they are positive and add up to one.

APPENDIX D · BIBLIOGRAPHIC NOTES

These notes point to the sources of the results in each part and to the best places to go further. They are selective rather than complete.

General texts

The geometric view of linear algebra that runs through this book owes most to Gilbert Strang, whose *Introduction to Linear Algebra* (sixth edition, 2023) and whose article "The Fundamental Theorem of Linear Algebra" (*American Mathematical Monthly*, 1993) made the four fundamental subspaces the organising picture of the subject; his *Linear Algebra and Learning from Data* (2019) connects it to machine learning. Sheldon Axler's *Linear Algebra Done Right* (fourth edition, 2024) develops the theory without determinants and is the best companion for proofs. For numerical questions, Lloyd N. Trefethen and David Bau's *Numerical Linear Algebra* (1997) is the clearest introduction and Gene Golub and Charles Van Loan's *Matrix Computations* (fourth edition, 2013) the standard reference. Roger Horn and Charles Johnson's *Matrix Analysis* (second edition, 2012) covers norms, eigenvalue inequalities and positive definiteness in depth.

Part I · Vectors

The inequality of Theorem 2.2 is due to Cauchy (1821) for sums and Schwarz (1888) for integrals. The attention mechanism of Sections 2.5 and 20.2 was introduced in its modern form by Vaswani and colleagues, "Attention Is All You Need" (2017). The concentration results of Chapter 4 are treated elegantly in Keith Ball's "An Elementary Introduction to Modern Convex Geometry" (1997), whose cap estimate underlies Theorem 4.1. The Johnson–Lindenstrauss lemma appeared in W. B. Johnson and J. Lindenstrauss (1984); a short proof is given by S. Dasgupta and A. Gupta (2003). Superposition in learned representations was studied by Elhage and colleagues in "Toy Models of Superposition" (2022).

The linear representation hypothesis was given a precise formulation by K. Park, Y. J. Choe and V. Veitch (2024). Sparse dictionaries on model activations were developed by Bricken and colleagues in "Towards Monosemanticity" (2023) and scaled by Templeton and colleagues in "Scaling Monosemanticity" (2024); L. Gao and colleagues, "Scaling and Evaluating Sparse Autoencoders" (2024), introduced dictionaries that keep only the largest coefficients, and B. Bussmann and colleagues (2025) proposed nested, multi-level dictionaries.

Part II · Matrices

The composition view of matrix multiplication goes back to Arthur Cayley's memoir on matrices (1858). The four fundamental subspaces are presented in Strang (1993), cited above. The use of Jacobian determinants in normalising flows was popularised by D. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows" (2015), and triangular-Jacobian designs followed from the same cost argument as Section 8.5.

Part III · Solving and fitting

Least squares was published by Legendre (1805) and developed by Gauss. Trefethen and Bau (1997) give the clearest account of why *QR* is preferred to the normal equations and of the condition number results of Chapter 12. The implicit preference of gradient descent for minimum-norm solutions (Theorem 11.2) is classical for least squares and has been studied extensively for over-parameterised models; ridge regression is due to A. Hoerl and R. Kennard (1970).

Part IV · Eigenvalues and the SVD

The spectral theorem in its modern form is associated with Cauchy, Hilbert and von Neumann. The singular value decomposition was found independently by Beltrami (1873) and Jordan (1874); its approximation property is Eckart and Young, "The Approximation of One Matrix by Another of Lower Rank" (*Psychometrika*, 1936), extended to all unitarily invariant norms by Mirsky (1960). Randomised algorithms for the leading singular vectors are surveyed in N. Halko, P.-G. Martinsson and J. Tropp, "Finding Structure with

Randomness" (*SIAM Review*, 2011). Low-rank factorisation for recommendation is described in Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems" (2009). PageRank was introduced in L. Page, S. Brin, R. Motwani and T. Winograd (1999). Principal component analysis goes back to K. Pearson (1901) and H. Hotelling (1933). Low-rank adapters were introduced by E. Hu and colleagues, "LoRA" (2021), and the weight-decomposed variant by S.-Y. Liu and colleagues, "DoRA" (2024).

Part V · Inside learning machines

Backpropagation as it is used today was set out by D. Rumelhart, G. Hinton and R. Williams (1986); A. Griewank and A. Walther's *Evaluating Derivatives* (second edition, 2008) is the reference on forward and reverse differentiation and the cost result of Theorem 18.4. The convergence analysis of Theorem 19.1 is standard; see Y. Nesterov, *Lectures on Convex Optimization* (2018). Kronecker-factored preconditioning appears in V. Gupta, T. Koren and Y. Singer, "Shampoo" (2018), and in N. Vyas and colleagues, "SOAP" (2024). The view of optimisers as steepest descent under operator norms is developed in J. Bernstein and L. Newhouse, "Old Optimizer, New Norm: An Anthology" (2024). The Muon optimiser was introduced by K. Jordan and colleagues (2024); its scaling to large transformer training was reported by J. Liu, J. Su and colleagues (2025). Spectral normalisation is due to T. Miyato and colleagues (2018). The Marchenko–Pastur law is from V. Marchenko and L. Pastur (1967), and spectral analyses of trained weight matrices include C. Martin and M. Mahoney, "Implicit Self-Regularization in Deep Neural Networks" (*Journal of Machine Learning Research*, 2021).

— ● —

APPENDIX E · GLOSSARY

The terms used in this book, each with a one-line meaning and the chapter where it is introduced or used most.

TABLE E.1 Glossary.

term	meaning	chapter
adapter, low-rank	a trainable update BA of small rank added to a fixed weight matrix	16
affine map	a linear map followed by a shift	5
attention	a weighted average whose weights come from scaled dot products of queries and keys	2, 20
basis	an independent set of vectors that spans a space	3
bilinear form	a score $\mathbf{x}^T B\mathbf{y}$, linear in each argument	20
Cholesky factorisation	$S = LL^T$ for a symmetric positive definite matrix	9, 14
column space	the set of all outputs $A\mathbf{x}$	7
condition number	the largest singular value divided by the smallest	12
cosine similarity	the dot product divided by both lengths	2
covariance matrix	the matrix of average products of centred features	14
determinant	the signed factor by which a map scales volume	8
diagonalisation	writing $A = V\Lambda V^{-1}$ with eigenvectors and eigenvalues	13
dimension	the number of vectors in any basis	3
dot product	the sum of coordinate products of two vectors	2
eigenvalue	the stretch along an eigenvector	13
eigenvector	a direction a matrix does not turn	13
embedding	a learned vector that represents a symbol or item	1, 20
Frobenius norm	the square root of the sum of squared entries	15
gradient	the vector of partial derivatives of a function	18
Gram–Schmidt process	building an orthonormal basis by removing shadows on earlier directions	10
Hessian	the symmetric matrix of second derivatives	18, 19
identity matrix	the matrix that leaves every vector unchanged	6
inverse	the matrix that undoes a map	6
Jacobian	the matrix of first derivatives of a vector-valued function	8, 18
least squares	choosing coefficients to minimise the sum of squared residuals	11

term	meaning	chapter
left null space	the vectors $\mathbf{y}$ with $A^T\mathbf{y} = \mathbf{0}$	7
linear combination	a sum of scaled vectors	1
linear independence	no vector of a set is a combination of the others	3
linear map	a function that respects addition and scaling	5
LU factorisation	elimination recorded as $A = LU$	9
Marchenko–Pastur law	the distribution of squared singular values of a large random matrix	20
norm	the length of a vector	1
normal equations	$A^TA\mathbf{c} = A^T\mathbf{y}$	11
null space	the vectors a matrix sends to zero	7
orthogonal matrix	a square matrix with $Q^TQ = I$	10
orthonormal set	perpendicular vectors of length one	10
outer product	the rank-one matrix $\mathbf{u}\mathbf{v}^T$	6
PageRank	importance defined by the eigenvector of a link matrix for eigenvalue 1	13
pivot	a diagonal entry left by elimination	9
polar factor	the orthogonal matrix UV^T obtained from an SVD	19
positive definite	a symmetric matrix with $\mathbf{x}^TS\mathbf{x} > 0$ for every nonzero $\mathbf{x}$	14
power method	repeated multiplication and normalisation to find the top eigenvector	13
preconditioner	a change of basis that makes curvature more uniform	19
principal component	a coordinate along a top eigenvector of the covariance	17
projection	the nearest point of a subspace, given by a matrix with $P^2 = P = P^T$	2, 10
pseudoinverse	$V\Sigma^+U^T$, the best possible undoing of a matrix	15
QR factorisation	$A = QR$ with orthonormal Q and triangular R	10

term	meaning	chapter
quadratic form	the function $\mathbf{x}^T S\mathbf{x}$	14
rank	the dimension of the column space	7
Rayleigh quotient	$\mathbf{x}^T S\mathbf{x} / \mathbf{x}^T \mathbf{x}$	14
residual stream	the vector that each transformer layer reads and updates	20
ridge regression	least squares with a penalty on the size of the coefficients	11
row space	the span of the rows of a matrix	7
singular value	a half-axis length of the image of the unit sphere	12, 15
singular value decomposition	$A = U\Sigma V^T$	15
span	all linear combinations of a set of vectors	3
spectral norm	the largest singular value of a matrix	15, 19
spectral theorem	a symmetric matrix has real eigenvalues and perpendicular eigenvectors	14
subspace	a set that contains $\mathbf{0}$ and is closed under addition and scaling	3
superposition	storing more features than dimensions along nearly perpendicular directions	4
trace	the sum of the diagonal entries	13
transpose	the matrix with rows and columns exchanged	1, 6
unit vector	a vector of length one	1
whitening	transforming data so that its covariance is the identity	17

APPENDIX F · PROBLEMS THAT CROSS THE CHAPTERS

Each chapter of this book solves its examples with the tools of that chapter. Real problems do not arrive sorted that way. The twelve problems below each need ideas from two or three chapters at once, and each is worked in full on numbers small enough to follow with a pencil. They make a good review before an examination, and a good test of whether the pictures of the earlier chapters have become a way of thinking.

Data and directions

Worked Example F.1 Principal directions of four points (Chapters 2, 13 and 17)

Four centred data points are $(2, 1)$, $(-2, -1)$, $(1, 2)$ and $(-1, -2)$. Find the principal directions and the share of variance along the first.

The sums of x^2, xy and y^2 are 10, 8 and 10, so dividing by the four points gives the covariance with rows $(2.5, 2)$ and $(2, 2.5)$.

A symmetric 2×2 matrix with equal diagonal entries has eigenvectors $(1, 1)/\sqrt{2}$ and $(1, -1)/\sqrt{2}$, with eigenvalues $2.5 + 2 = 4.5$ and $2.5 - 2 = 0.5$. The first direction carries $4.5/5 = 90$ of the variance.

Check with Theorem 17.1: the projections onto $(1, 1)/\sqrt{2}$ are $\pm 3/\sqrt{2}$, whose mean square is 4.5, and keeping one component leaves a mean squared reconstruction error of exactly the other eigenvalue, 0.5.

Worked Example F.2 A line through three points, checked by geometry (Chapters 10 and 11)

Fit $y = c + mx$ to $(0, 1)$, $(1, 2)$ and $(2, 4)$.

The normal equations are $3c + 3m = 7$ and $3c + 5m = 10$. Subtracting gives $2m = 3$, so $m = 1.5$ and $c = 5/6$.

The fitted values are $5/6$, $7/3$ and $23/6$, and the residuals are $1/6$, $-1/3$ and $1/6$. Two checks follow from the geometry of Chapter 11. The residuals sum to zero, so the residual is perpendicular to the column of ones; and $0 \times 1/6 + 1 \times (-1/3) + 2 \times 1/6 = 0$, so it is perpendicular to the column of x values too. Its squared length, $1/36 + 1/9 + 1/36 = 1/6$, is the smallest any line can achieve.

Worked Example F.3 How fast repeated multiplication finds an eigenvalue (Chapters 13 and 14)

Let A have rows $(2, 1)$ and $(1, 2)$, with eigenvalues 3 and 1. Start from $(1, 0)$ and multiply repeatedly.

The first three vectors are $(2, 1)$, $(5, 4)$ and $(14, 13)$; in general the kth is $1/2(3^k + 1, 3^k - 1)$. Their directions approach $(1, 1)$.

Estimate the eigenvalue by the Rayleigh quotient of each vector. Writing the vector in the eigenvector basis gives the exact value $3 - 2/(9^k + 1)$: that is 2.8, then 2.9756, then 2.9973.

The error falls by about a factor of nine per step — the square of the eigenvalue ratio $1/3$. The vector converges at the rate $1/3$, but the Rayleigh quotient, being stationary at an eigenvector (Chapter 14), converges twice as fast in the exponent.

Decompositions at work

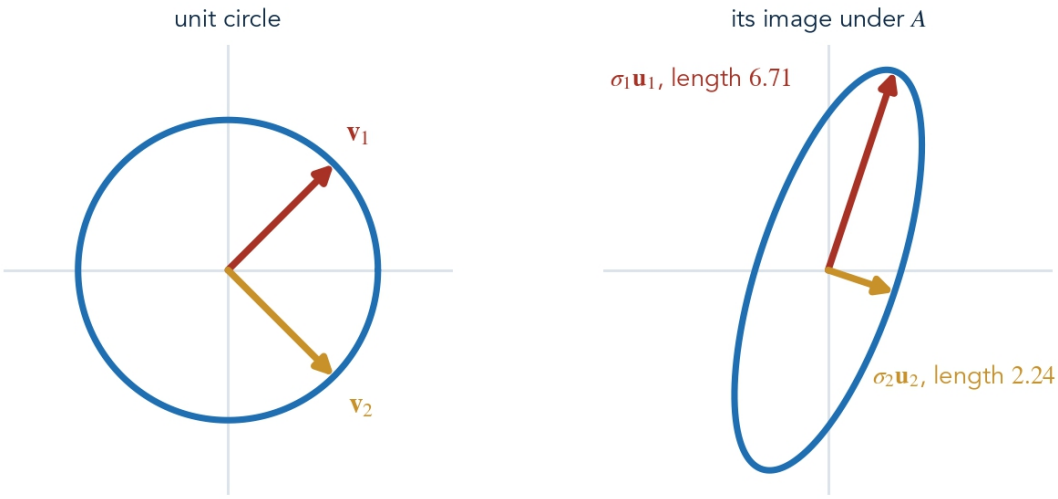
Worked Example F.4 A singular value decomposition by hand (Chapters 14 and 15)

Find the SVD of the matrix A with rows $(3, 0)$ and $(4, 5)$.

A^TA has rows $(25, 20)$ and $(20, 25)$, with eigenvalues 45 and 5 and eigenvectors $\mathbf{v}_1 = (1, 1)/\sqrt{2}$ and $\mathbf{v}_2 = (1, -1)/\sqrt{2}$. The singular values are $\sigma_1 = \sqrt{45} = 3\sqrt{5} \approx 6.71$ and $\sigma_2 = \sqrt{5} \approx 2.24$.

The output directions follow from $\mathbf{u}_i = A\mathbf{v}_i/\sigma_i$: $A\mathbf{v}_1 = (3, 9)/\sqrt{2}$ gives $\mathbf{u}_1 = (1, 3)/\sqrt{10}$, and $A\mathbf{v}_2 = (3, -1)/\sqrt{2}$ gives $\mathbf{u}_2 = (3, -1)/\sqrt{10}$, which are perpendicular, as promised.

Three checks: $\sigma_1\sigma_2 = 15 = \det A$; ${\sigma_1}^2 + {\sigma_2}^2 = 50 = 9 + 16 + 25$, the squared Frobenius norm; and the condition number is $\sigma_1/\sigma_2 = 3$. Figure F.1 shows the decomposition as a picture.



the perpendicular directions $\mathbf{v}_1$ and $\mathbf{v}_2$ are the ones A keeps perpendicular. They become the axes of the ellipse, stretched by the singular values; the ellipse's area is $\pi\sigma_1\sigma_2 = 15\pi$.

FIGURE F.1 The matrix of Example F.4 acting on the unit circle. The input directions $\mathbf{v}_1$ and $\mathbf{v}_2$ land on the axes of the ellipse, with lengths equal to the singular values.

Worked Example F.5 Two ways to project onto a plane (Chapters 7, 10 and 11)

Project $\mathbf{b} = (2, 3, 4)$ onto the plane spanned by $(1, 0, 1)$ and $(0, 1, 1)$.

The first way uses the normal. The vector $\mathbf{n} = (1, 1, -1)$ is perpendicular to both spanning vectors, so it spans the plane's orthogonal complement. Remove the component of $\mathbf{b}$ along it: $\mathbf{b}\cdot\mathbf{n} = 1$ and $\mathbf{n}\cdot\mathbf{n} = 3$, so the projection is $\mathbf{b} - 1/3\mathbf{n} = (5/3, 8/3, 13/3)$, at distance $1/\sqrt{3}$.

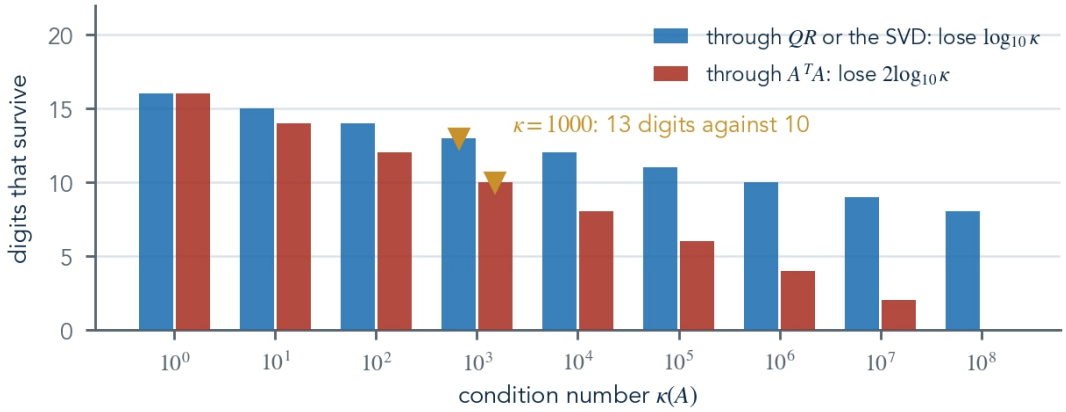
The second way uses the basis: $a(1, 0, 1) + c(0, 1, 1) = (a, c, a + c)$, and matching the first two entries gives $a = 5/3$, $c = 8/3$, whose sum $13/3$ matches the third. Both routes agree, and the first is shorter because the complement has dimension one.

Worked Example F.6 Why the normal equations lose digits (Chapters 11, 12 and 15)

A matrix A has singular values 10 and 0.01, so its condition number is 1000.

The matrix A^TA of the normal equations has singular values 100 and 0.0001, so its condition number is 10^6 — the square.

Arithmetic with sixteen significant digits can therefore lose about three digits solving a least-squares problem through a QR or singular value decomposition, which work with A itself, but about six through the normal equations. Theorem 12.1 turns the arithmetic into a practical rule: form A^TA only when A is well conditioned. Figure F.2 shows the loss for every condition number.



starting from sixteen significant digits, forming A^TA squares the condition number and so doubles the loss. By $\kappa = 10^8$ the normal equations have nothing left to give.

FIGURE F.2 Significant digits that survive a least-squares solution, from a start of sixteen, through an orthogonal factorisation (blue) and through the normal equations (red), as in Example F.6.

Learning and optimisation

Worked Example F.7 One well-chosen gradient step (Chapters 11, 18 and 19)

Minimise $f(\mathbf{w}) = 1/2\|A\mathbf{w} - \mathbf{b}\|^2$ where A is diagonal with entries 1 and 3 and $\mathbf{b} = (1, 3)$, starting from $\mathbf{w} = \mathbf{0}$.

The minimiser is $\mathbf{w}^* = (1, 1)$. The gradient is $A^T(A\mathbf{w} - \mathbf{b})$, which at the origin is $-(1, 9)$, and the Hessian is A^TA, with curvatures 1 and 9.

Example 19.3 found the best fixed step for these curvatures: $\eta = 2/(1 + 9) = 0.2$. One step gives $\mathbf{w} = (0.2, 1.8)$, whose error $(-0.8, 0.8)$ is 0.8 times the starting error $(-1, -1)$ in size along both axes.

The condition number of A^TA, here 9, is what limits the rate $(9 - 1)/(9 + 1) = 0.8$. Rescaling the second coordinate by 3 would make both curvatures equal and finish the problem in one step.

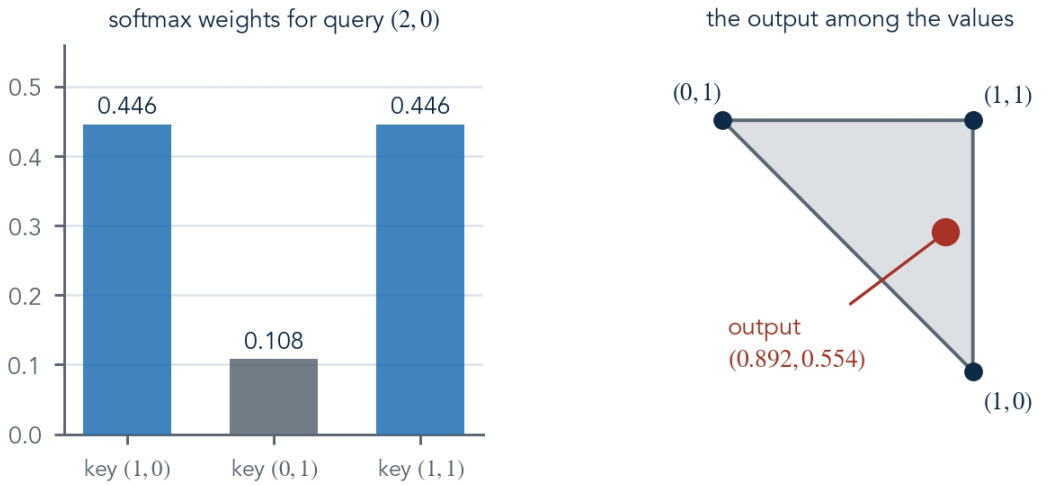
Worked Example F.8 Attention as a weighted average (Chapters 2 and 20)

Three positions have keys (1, 0), (0, 1) and (1, 1) and values (1, 0), (0, 1) and (1, 1). The query is (2, 0) and $d_k = 2$.

The dot products of the query with the keys are 2, 0 and 2; dividing by $\sqrt{2}$ gives 1.414, 0 and 1.414.

The softmax turns these into weights proportional to $e^{1.414} \approx 4.113$, 1 and 4.113, which sum to 9.227: the weights are 0.446, 0.108 and 0.446.

The output is $0.446(1, 0) + 0.108(0, 1) + 0.446(1, 1) = (0.892, 0.554)$. It leans towards the values whose keys point along the query, and it is a convex combination, so it lies inside the triangle of the three values. Figure F.3 shows the weights and where the output lands.



the query agrees equally with the first and third keys, so those values share most of the weight. The output is a weighted average and always lands inside the triangle of values.

FIGURE F.3 The attention step of Example F.8: the three softmax weights (left) and the output among the three values (right).

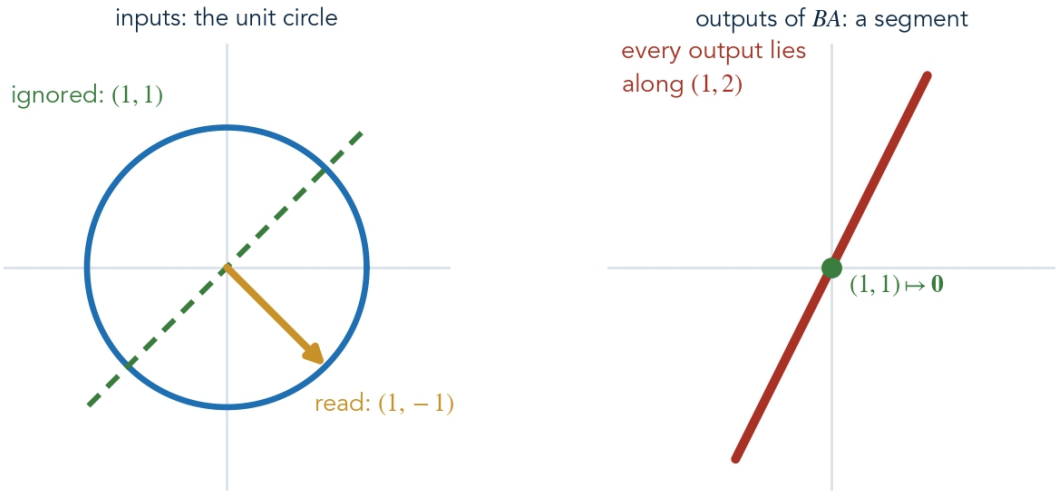
Worked Example F.9 What a low-rank update can change (Chapters 7 and 16)

A weight matrix of size 1000×1000 receives an adapter BA with B of size 1000×8 and A of size 8×1000.

The adapter has $2 \times 1000 \times 8 = 16{,}000$ trainable numbers, 1.6 of the million in the matrix, and its rank is at most 8.

To see what that rank restriction means, shrink to 2×2 with $B = (1, 2)$ as a column and $A = (1, -1)$ as a row. Then BA has rows $(1, -1)$ and $(2, -2)$. Every output of the update lies along $(1, 2)$, and inputs along $(1, 1)$, the null space, are ignored entirely.

An update of rank r can add at most r new input directions and r new output directions to the layer — and the evidence of Chapter 16 is that fine-tuning seldom needs more. Figure F.4 draws the 2×2 case.



the update with rows $(1, -1)$ and $(2, -2)$ reads a single input direction and writes a single output direction. The whole circle collapses onto a segment, and the perpendicular input is sent to zero.

FIGURE F.4 The rank-one update of Example F.9 acting on the unit circle: one input direction is read, the perpendicular one is ignored, and every output lies on one line.

Geometry in many dimensions

Worked Example F.10 The area of a parallelogram in space (Chapters 8 and 14)

Find the area of the parallelogram in $\mathbb{R}^3$ spanned by $(1, 0, 1)$ and $(0, 1, 1)$.

A 2×3 matrix has no determinant, but its Gram matrix does. The dot products give the Gram matrix with rows $(2, 1)$ and $(1, 2)$, whose determinant 3 is the squared area. The area is $\sqrt{3}$.

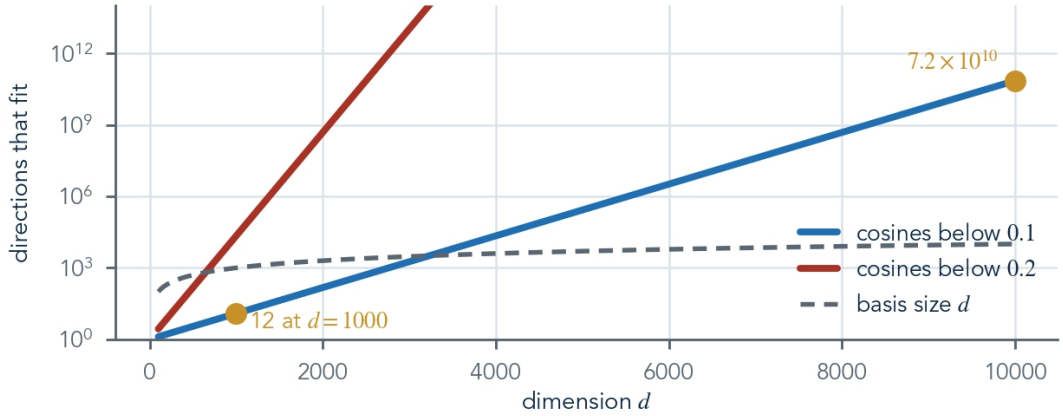
Check: the cross product of the two vectors is $(-1, -1, 1)$, whose length is also $\sqrt{3}$ — and whose direction is the normal $\mathbf{n}$ of Example F.5, up to sign. The Gram matrix is positive definite exactly when the vectors are independent, so a zero area and a singular Gram matrix are the same event.

Worked Example F.11 How many nearly perpendicular directions fit? (Chapters 3 and 4)

Chapter 4 showed that $N = e^{d\varepsilon 2/4}$ unit vectors can be chosen so that every pair has cosine below ε in size.

With $\varepsilon = 0.1$ and $d = 1000$, the exponent is $1000 \times 0.01/4 = 2.5$ and $N = e^{2.5} = 12$ — hardly impressive.

With $d = 10{,}000$ the exponent is 25 and $N \approx 7.2 \times 10^{10}$: seventy-two billion directions, each within about $6°$ of perpendicular to every other, in a space whose basis has only ten thousand vectors. The count is exponential in the dimension while the basis is linear, which is why a representation space can hold far more features than it has coordinates. Figure F.5 compares the two growth rates.



the guaranteed count grows exponentially with the dimension, while a basis grows only in proportion to it. Loosening the tolerance from 0.1 to 0.2 raises the exponent fourfold.

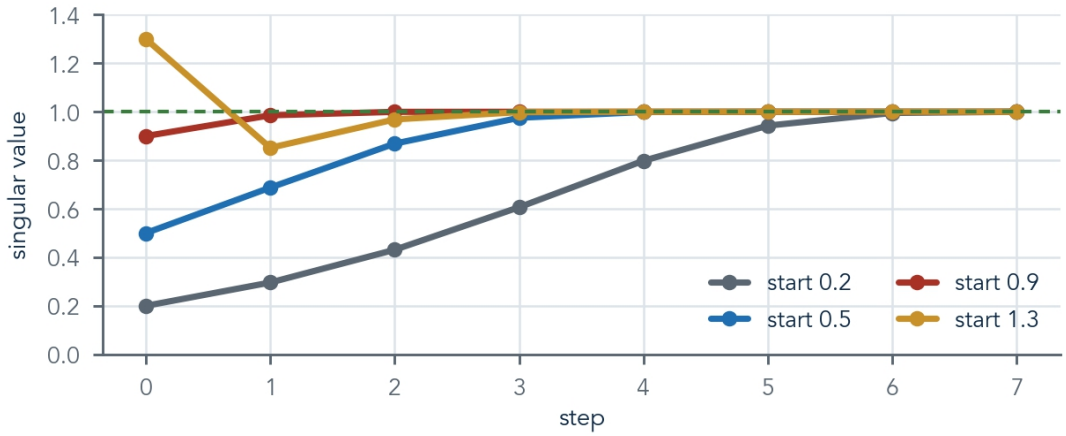
FIGURE F.5 The number of unit vectors with all pairwise cosines below a tolerance, from the bound of Example F.11, compared with the size of a basis.

Worked Example F.12 Orthogonalising a matrix by polishing its singular values (Chapters 15 and 19)

A matrix has singular values 0.5 and 0.9. Apply the cubic map $\sigma \mapsto 1.5\sigma - 0.5\sigma^3$ of Chapter 19 repeatedly.

Starting from 0.5 the values run 0.500, 0.688, 0.869, 0.975, 0.999. Starting from 0.9 they run 0.900, 0.986, 1.000, 1.000.

Both approach 1, and the singular vectors never change, because each step multiplies X by polynomials in X^TX. The limit is the polar factor UV^T: the same rotations as the original matrix, with every stretch replaced by one. The larger value converges faster, since the map's slope at 1 is zero and the error is squared, roughly, once it is small. Figure F.6 follows four starting values.



each step applies $\sigma \mapsto 1.5\sigma - 0.5\sigma^3$ to every singular value at once. Values below one rise, values slightly above one fall, and all of them settle at one: the matrix becomes orthogonal.

FIGURE F.6 Repeated application of the cubic map of Example F.12 to four singular values. All converge to one, the larger ones almost at once.

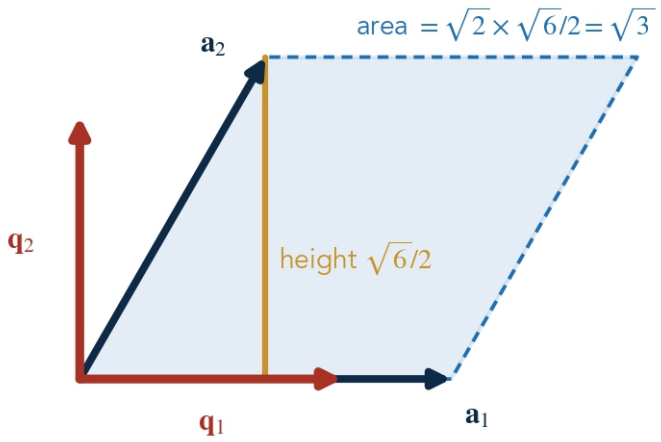
Four more connections

Worked Example F.13 Gram–Schmidt meets area (Chapters 8 and 10)

Orthonormalise $\mathbf{a}_1 = (1, 1, 0)$ and $\mathbf{a}_2 = (1, 0, 1)$, and write $A = QR$.

The first vector normalises to $\mathbf{q}_1 = (1, 1, 0)/\sqrt{2}$. The component of $\mathbf{a}_2$ along it is $\mathbf{a}_2 \cdot \mathbf{q}_1 = 1/\sqrt{2}$; removing it leaves $(1, 0, 1) - (1/2, 1/2, 0) = (1/2, -1/2, 1)$, of length $\sqrt{6}/2$, so $\mathbf{q}_2 = (1, -1, 2)/\sqrt{6}$.

The triangular factor R has rows $(\sqrt{2}, 1/\sqrt{2})$ and $(0, \sqrt{6}/2)$. Its diagonal holds the length of $\mathbf{a}_1$ and the height of $\mathbf{a}_2$ above the line of $\mathbf{a}_1$, so their product $\sqrt{2} \times \sqrt{6}/2 = \sqrt{3}$ is the area of the parallelogram — the same $\sqrt{3}$ that the Gram determinant gave in Example F.10. Figure F.7 draws the construction in the plane of the two vectors.



drawn in the plane of $\mathbf{a}_1$ and $\mathbf{a}_2$. Gram–Schmidt keeps the direction of $\mathbf{a}_1$ and the height of $\mathbf{a}_2$; the diagonal of R records exactly those two lengths, and their product is the area.

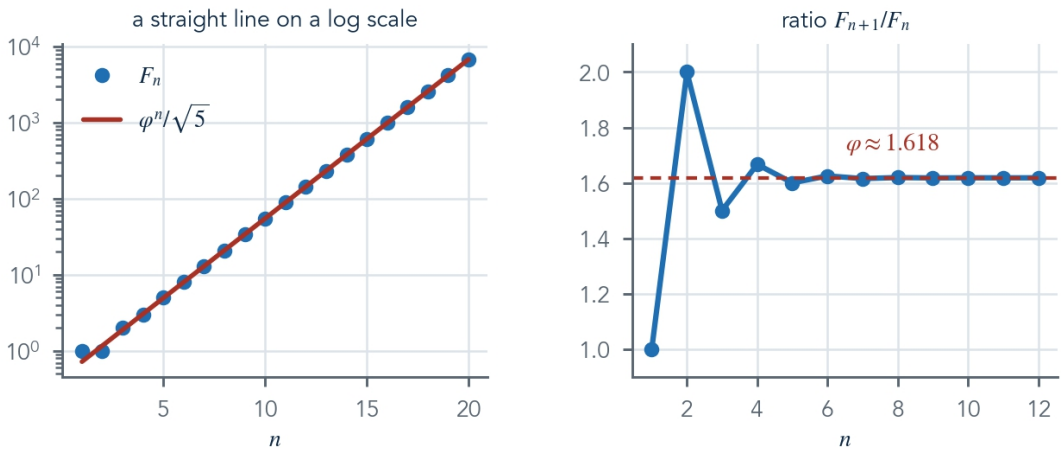
FIGURE F.7 Example F.13 in the plane of a_1 and a_2: the orthonormal pair q_1, q_2 and the two lengths that make up the triangular factor R.

Worked Example F.14 Growth rates from eigenvalues (Chapter 13)

The Fibonacci numbers obey $(F_{n+1},\ F_n) = M(F_n,\ F_{n-1})$, where M has rows $(1, 1)$ and $(1, 0)$.

The characteristic equation is $\lambda^2 - \lambda - 1 = 0$, with roots $\varphi = (1 + \sqrt{5})/2 \approx 1.618$ and $\psi = (1 - \sqrt{5})/2 \approx -0.618$. Expanding the starting vector in the eigenvectors gives the exact formula $F_n = (\varphi^n - \psi^n)/\sqrt{5}$.

Because ψ has size below one, its term dies away: $\varphi^{10}/\sqrt{5} \approx 55.0036$, and $F_{10} = 55$. The ratio of successive terms approaches φ for the same reason repeated multiplication finds the dominant eigenvector in Chapter 13 — every linear recurrence grows at the rate of its largest eigenvalue. Figure F.8 shows both facts.



the sequence grows like the dominant eigenvalue's powers (left), and the ratio of neighbours settles on that eigenvalue within a few steps (right), as the second eigenvalue's influence shrinks away.

FIGURE F.8 The Fibonacci numbers of Example F.14 against $\varphi^n/\sqrt{5}$ (left) and the ratio of successive terms approaching φ (right).

Worked Example F.15 What ridge regularisation does to each direction (Chapters 11, 15 and 17)

Adding a penalty $\lambda\|\mathbf{w}\|^2$ to least squares changes the solution along each right singular vector of the data matrix by the factor $\sigma_i^2/(\sigma_i^2+\lambda)$.

Take singular values 10, 1 and 0.1 with $\lambda = 1$. The factors are $100/101 \approx 0.990$, $1/2 = 0.5$ and $0.01/1.01 \approx 0.0099$.

The strong direction is left almost untouched, the middle one is halved, and the weak one is all but removed. The weak directions are exactly those where Chapter 12 warned that noise is amplified by $1/\sigma_i$, so ridge regression is a smooth version of keeping the top principal components: it turns a sharp cut-off into a gentle dial. Figure F.9 plots the factor for three penalties.

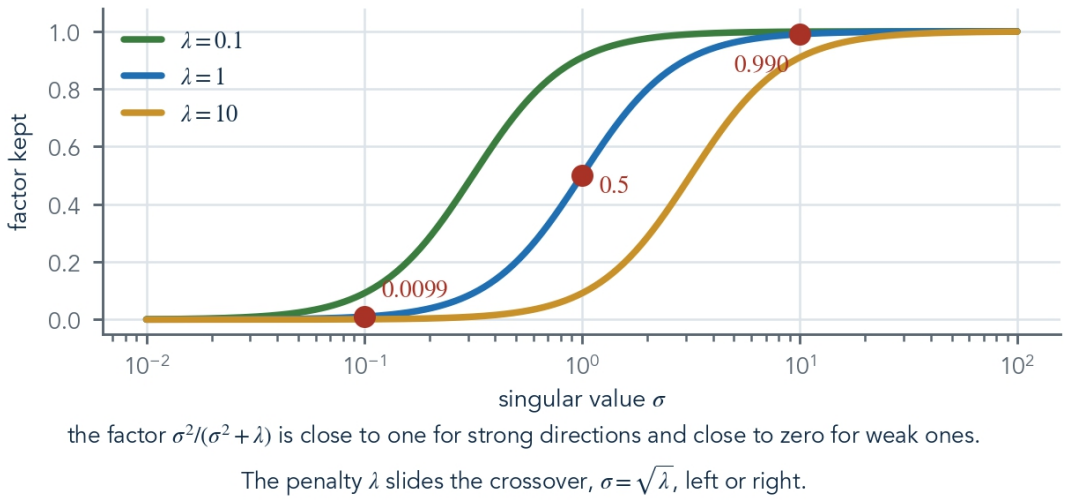


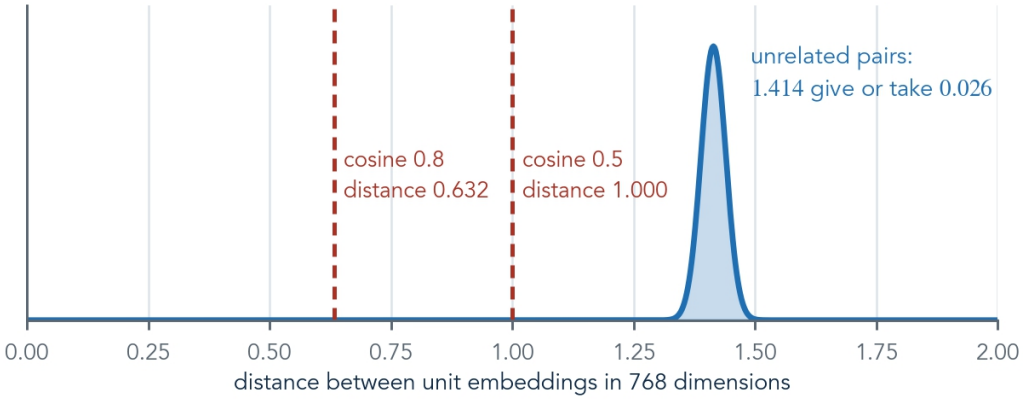
FIGURE F.9 The ridge factor of Example F.15 as a function of the singular value, for three penalties. The three marked points are the singular values 10, 1 and 0.1 with $\lambda = 1$.

Worked Example F.16 Distance and cosine for normalised embeddings (Chapters 2 and 4)

For unit vectors, expanding $\|\mathbf{u} - \mathbf{v}\|^2 = \mathbf{u}\cdot\mathbf{u} - 2\mathbf{u}\cdot\mathbf{v} + \mathbf{v}\cdot\mathbf{v}$ gives $2 - 2\cos\theta$.

So for normalised embeddings, ranking neighbours by distance and ranking them by cosine give the same order. A cosine of 0.8 is a distance of $\sqrt{0.4} \approx 0.632$; a cosine of 0 is a distance of $\sqrt{2} \approx 1.414$.

In $d = 768$ dimensions, unrelated directions have cosines spread by about $1/\sqrt{768} \approx 0.036$ around zero, so their distances cluster tightly at 1.414, give or take about 0.026. That tight cluster is the background against which a genuinely related pair, at cosine 0.5 or more, stands out clearly. Figure F.10 shows the band and two related pairs.



unrelated directions crowd into a narrow band near $\sqrt{2}$. A related pair sits far to the left of it, which is why a nearest-neighbour search in a large embedding space separates signal from background so well.

FIGURE F.10 Distances between unit vectors in 768 dimensions (Example F.16). Unrelated directions fall in a narrow band around $\sqrt{2}$; pairs with cosine 0.5 and 0.8 lie well clear of it.

Which tool answers which question

The table below turns the book around. Instead of listing ideas chapter by chapter, it starts from the questions that arise in practice and points to the idea that answers each.

TABLE F.1 Questions and the tools that answer them.

question	tool	chapter
how similar are two embeddings?	cosine similarity	2
are these vectors redundant?	independence and rank	3, 7
how many nearly independent features fit?	concentration of angles in high dimension	4
what does this matrix do to space?	read its columns as images of the axes	5
what do several steps do together?	matrix multiplication, in order	6
does $A\mathbf{x} = \mathbf{b}$ have a solution?	is $\mathbf{b}$ in the column space?	7
is the map invertible, and does it keep volume?	the determinant	8
how do I solve a square system?	$PA = LU$, then two triangular solves	9
how do I project onto a subspace?	an orthonormal basis Q, then QQ^T	10
more equations than unknowns?	least squares, through QR	11
how far can I trust the answer?	the condition number σ_1/σ_n	12
what happens after many steps?	the largest eigenvalue and its eigenvector	13
is this quadratic shaped like a bowl?	positive definiteness, by eigenvalues or Cholesky	14
what is the geometry of this matrix?	the SVD: rotate, stretch, rotate	15
what is the best rank-k copy?	the truncated SVD	16
how many numbers does an update need?	its rank; $2dr$ for an adapter	16
which directions of the data matter?	eigenvectors of the covariance	17
which way is downhill?	the gradient	18
why is training slow?	the spread of the Hessian's eigenvalues	19
how does attention mix information?	softmax of $QK^T/\sqrt{d_k}$, rank at most d_k	20

Where to go from here

Linear algebra rewards a second reading more than almost any other subject, because each idea makes the others easier. Read Chapter 15 again after Chapter 19, and the singular value decomposition looks

less like a factorisation and more like the natural coordinates of every problem in the book. Read Chapter 4 again after Chapter 20, and the strange geometry of high dimensions stops being a curiosity and becomes the reason modern models can hold so much in so few numbers.

Three habits will carry the subject further than any list of theorems. First, draw the 2×2 case before trusting a general statement; almost every idea in this book is already visible in the plane. Second, ask what a matrix does rather than what it contains: which directions it keeps, which it stretches and which it sends to zero. Third, check a result two ways, as the problems above do — by a formula and by a picture, or by two different decompositions — because agreement between independent routes is the most reliable evidence mathematics offers.

The natural next steps are numerical linear algebra, where the methods of Chapters 9 to 16 are made fast and stable at scale; optimisation, where Chapters 18 and 19 open onto a wide and practical field; and probability in high dimensions, which grows directly from the concentration results of Chapter 4. Each of them is built on the same few pictures you now carry, and each will feel familiar from its first page.

INDEX OF PROPOSITIONS

The claims this book advances on its own account, in order of appearance. Each is meant to be defensible on its own, and each is meant to be worth disputing.

9. **9** A matrix is not a table of numbers that happens to support a multiplication rule. It is a complete description of a geometric action, compressed into mn numbers by the fact that linearity lets the images of n vectors determine everything. The multiplication rule is a consequence, and the next chapter derives it.

10. **10** Every shape mismatch reported by a numerical library is a composition error: an attempt to feed the output of one map into a map expecting a different space. Reading shapes as spaces — "this matrix goes from 768 dimensions to 3,072" — makes such errors impossible to write rather than merely possible to debug.

11. **11** Deep networks are compositions, and the algebra of composition — associativity, non-commutativity, the reversal of order under transposes and inverses, the rank cap of a narrow factor — governs them completely between the non-linear folds. Understanding a network architecture is largely a matter of reading its sequence of shapes as a sequence of spaces.

12. **12** Every matrix, whatever its size or entries, does the same thing: it throws away a subspace of its input, maps the perpendicular remainder one-to-one onto a subspace of its output, and leaves the rest of the output unreachable. The four fundamental subspaces are not four topics; they are one picture of what a matrix does, drawn in two spaces.

13. **13** The determinant is the answer to one question — by how much, and with what orientation, does this map scale volume — and every one of its properties is that question asked about a composition, an inverse, a transpose or a shear. Treated as a formula it is a burden; treated as a volume it is almost free.

14. **14** "Solve, don't invert" is the most practically valuable rule in numerical linear algebra. An expression like $A^{-1}\mathbf{b}$ or $(X^TX)^{-1}X^T\mathbf{y}$ describes a result, not a method; the method is always a factorisation followed by triangular solves.

15. **15** Projection is the geometric form of "the best approximation from a restricted set", and the condition that characterises it — the error is perpendicular to everything you were allowed to use — is the same condition in regression, in principal component analysis, in the Fourier series and in the Kalman filter. Recognising the condition is recognising that a problem is a projection.

16. **16** An over-parameterised model trained by gradient descent does not pick an arbitrary member of its infinite family of perfect fits. For linear least squares it provably picks the shortest one, and that implicit preference for small solutions — invisible in the objective, supplied by the method — is part of why models with more parameters than data can still predict well.

17. **17** The determinant says whether a matrix flattens space; the condition number says how close it comes. A matrix can have a determinant of 10^{-100} and be perfectly conditioned, or a determinant of one and be useless. For every numerical question the second number is the one to ask for.

18. **18** Nearly every practical technique for making training faster or more stable — feature scaling, normalisation layers, careful initialisation, preconditioning, orthogonalised updates — can be understood as an intervention on the condition

number of some matrix. When a method mysteriously helps, the first question to ask is which ellipse it made rounder.

19. **19** Eigenvectors turn the long-run behaviour of any repeated linear process into arithmetic. Whether a signal survives a hundred layers, whether a chain settles down, which page is most important and which direction of data carries most variance are all the same question: what does repeated application of this matrix favour, and by how much?

20. **20** For a symmetric matrix the eigenvectors and the axes of the ellipse coincide. That coincidence is what makes symmetric matrices easy: their action is a rotation to perpendicular axes, a stretch along each, and the rotation back — no shearing, no hidden turning, nothing that cannot be drawn.

21. **21** Every linear map, whatever its entries, is a rotation (or reflection), an axis-aligned stretch, and a rotation. There is no other kind of linear behaviour. Shears, projections and every "strange" matrix are this sequence with particular angles and stretches, and anything a matrix does can be described by naming its two rotations and its list of stretches.

22. **22** Low rank is the mathematical form of "a few underlying factors explain most of what we see". When a matrix of observations has rapidly falling singular values, the observations are the combined result of a small number of hidden patterns, and the truncated SVD recovers those patterns optimally. Much of what is called structure in data is, precisely, fast singular-value decay.

23. **23** Principal component analysis is not a separate technique. It is the spectral theorem applied to a covariance, computed by the SVD of the data, justified by Eckart–Young, and interpreted through the Rayleigh quotient. A reader who has followed Part IV already knows everything PCA does, and exactly what it cannot do.

24. **24** The chain rule is Theorem 6.1 in disguise. A derivative is a linear map, composing functions composes their derivatives, and composing linear maps multiplies their matrices. Everything about computing gradients in deep networks follows from applying that sentence to a long composition.

25. **25** Backpropagation is the observation that a gradient is a product of matrices whose leftmost factor is a row vector, together with the decision to multiply from the left. That one choice of order is why a model with a billion parameters can be trained at all: its gradient costs about as much as a few evaluations of its loss, not a billion.

26. **26** The difficulty of optimising a smooth loss is not its dimension, and not the size of its gradients. It is the spread of the eigenvalues of its Hessian. Every technique that makes training faster is, to a first approximation, a way of making that spread smaller or of stepping as though it were smaller.

27. **27** A single attention head does not compare two vectors in full. It compares their projections onto a subspace of dimension at most d_k, chosen by training. A layer with several heads is a sum of such low-rank comparisons, each looking at a

different subspace — which is why many narrow heads, rather than one wide one, is the standard design.

28. **28** A matrix is a geometric action on space, and a learning machine is a long composition of such actions interrupted by simple folds. Every operation in it can be understood by asking what it does to directions — which it stretches, which it destroys, which it leaves on their own line — and the singular value decomposition answers that question for every matrix there is. The mathematics is short. That it is enough is one of the genuinely hopeful facts of our time.

How the propositions fit together

Read in sequence, the propositions make a single argument in five steps, one for each part of the book.

The first step is that a vector is best understood as a direction with a length, and that in many dimensions directions are abundant: far more nearly perpendicular directions fit into a space than its dimension suggests. That abundance is what allows a learned representation to hold many features at once.

The second step is that a matrix is an action on space, not a table of numbers. Once multiplication is read as composition and rank as the number of directions that survive, the rules of matrix algebra become statements about pictures, and most of them can be checked by drawing.

The third step is that solving, projecting and fitting are one operation seen from three sides. Elimination, orthogonal bases and least squares all answer the question of which point of a subspace lies closest to a target, and the condition number measures how far that answer can be trusted.

The fourth step is that every matrix has natural coordinates. Eigenvectors provide them for repeated action and for symmetric forms; singular vectors provide them for everything else. In those coordinates a matrix is only a set of stretches, which is why compression, principal components and low-rank adaptation all come down to keeping the largest few.

The fifth step is that learning is geometry in motion. A gradient is a direction, curvature is a matrix, and the speed of training is set by the spread of that matrix's eigenvalues. Methods that make training

faster are, one way or another, methods that make that spread smaller.

None of these steps is new on its own; each rests on work summarised in Appendix D. What this book adds is the claim that they belong together, and that a reader who holds all five pictures at once can reason about a new method in machine learning before reading a single equation of it. That claim, like the others, is offered to be tested.

ABOUT THE AUTHOR

George Chu (Xingxiong Zhu) was born in April 1975. He holds a Master of Engineering degree in Software Engineering from Peking University and is a Senior Member of the IEEE.

As an author, he has published more than 30 books. This one reflects a conviction he has formed across them: that the mathematics under modern machine learning is short, geometric and beautiful, and that a student who can see what a matrix does to space is in a different position from one who has only learned to multiply it.

His recent books include *Applied Machine Learning Workbook*, *Prompt Engineering Essentials for Computer Science Students*, *Introduction to Quantum Computing for Engineers*, *Python Programming: From Scratch to Large Model Applications*, and *The Origin of the Universe: From Quantum Fluctuation to Cosmic Structure*.

FIRST EDITION · 2026