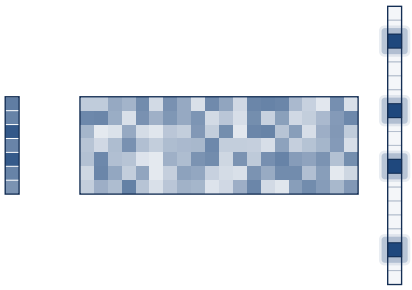
High-Dimensional Sensing and Learning

A Course in Compressed Sensing, Sparsity, and Low-Rank Recovery



Taka Khoo

Thayer School of Engineering
Dartmouth College

High-Dimensional Sensing and Learning: A Course in Compressed Sensing, Sparsity, and Low-Rank Recovery.

First edition, 2026. Published by the author in Hanover, New Hampshire, as a companion to a graduate course at the Thayer School of Engineering, Dartmouth College.

This book is based on the lectures of Professor Peter Chin, and is offered as a tribute to his teaching. The mathematics, the framing, and the demonstrations are his; the exposition, the worked solutions, and any remaining errors are the author's. The dedication and preface tell that story.

Suggested citation.
T. Khoo, *High-Dimensional Sensing and Learning: A Course in Compressed Sensing, Sparsity, and Low-Rank Recovery*, 1st ed. Hanover, NH: published by the author, 2026. A specific result may be cited as, for example, "Khoo, *High-Dimensional Sensing and Learning*, Ch. 8."

A complete, self-contained development of compressed sensing, from a single linear measurement model to working systems for imaging, recommendation, and recognition. Every definition is stated, every symbol named, every proof written out in full, and every problem solved step by step, with each algorithm traced once by hand on real numbers.

Subjects: compressed sensing; sparse recovery; low-rank matrix recovery; convex optimization; high-dimensional statistics; signal processing.

Typeset by the author in LaTeX with the Latin Modern type family; diagrams drawn in Ti*k*Z.

For Professor Peter Chin,

who convinced a room full of engineers
that a few good measurements can outweigh all the data in the world.

Preface

This book began as a pile of scribe notes. Three mornings a week, Professor Peter Chin built up the theory of compressed sensing from a single equation, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, to a working understanding of how a few well-chosen measurements can recover a signal that classical sampling theory says is hopelessly underdetermined. I took notes on every lecture, worked every problem set, and wrote up the review materials. What you are holding is all of that, gathered into one place, filled in, and finished.

The goal here is completeness in the most literal sense. Every definition the course used is stated and every symbol in it is named. Every theorem is given with its exact hypotheses, and every proof is written out to the last algebraic step, including the steps that get skipped at the board because they are "obvious." Often those are the steps a reader actually needs. Every algorithm comes with pseudocode and at least one numerical run traced by hand, so that you can see the method work on real numbers before trusting it on a computer. And every problem from the nine problem sets is restated in full and solved completely, with the reasoning that connects it back to the lecture that motivated it.

I made a few deliberate choices about organization. The course unfolded as a sequence of lectures, but the ideas group naturally into themes, and the chapters follow the themes rather than the calendar. Part I sets up the language of the subject. Parts II and III develop the core theory of sparse recovery and the algorithms that carry it out. Part IV answers the question that hangs over the whole subject: why does sensing with a *random* matrix work, and how many measurements does it take? Part V leaves vectors behind for matrices and develops low-rank recovery, robust PCA, dictionary learning, and nonnegative factorization. Part VI collects the extensions and applications. After that come the problem sets, a set of practice examinations, and a group of reference appendices. A reader who wants the path the course actually took can follow the lecture map in the next section, which lists, chapter by chapter, exactly which lectures and problem sets each part draws on.

A word on the look of the book. The mathematics is meant to breathe. I have kept colored boxes to a minimum and used them only for the single canonical statement of a major result, for a step-by-step recipe, or for a pointer to the problem sets. Everything else is plain prose and displayed mathematics, with enough white space that you can write in the margins. This is a book to be printed, marked up, and argued with.

The notation throughout matches the lectures. Vectors and matrices are bold, $\boldsymbol{x}$ and $\boldsymbol{A}$; the measurement model is always $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ with $\boldsymbol{x} \in \mathbb{C}^N$ unknown, $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ known, and $\boldsymbol{y} \in \mathbb{C}^m$ observed; and the recurring tension of the whole subject is that m is much smaller than N. The single most important idea, repeated at three levels of generality, is that a hard combinatorial problem about counting nonzeros becomes a tractable convex problem about summing magnitudes, and that under the right conditions the two have the same answer. If a reader takes away only that one sentence, the book has done most of its job.

Acknowledgments

The mathematics, the framing, the stories, and the demonstrations in this book are Professor Chin's. He has a gift for making a hard theorem feel inevitable, and for reminding a room full of engineers why any of it matters. Whatever is clear here, he made clear first; whatever errors remain are mine, introduced in the writing up. I am grateful to the teaching assistants, Minh Bui and Riya Yadav, who ran the problem-set sessions and debugged a great deal of Python, and to my classmates, who asked the questions in lecture that turned into the warnings and asides scattered through these pages.

This book is a thank-you. Professor Chin, this is the course you taught us, written down so that none of it gets lost.

Taka Khoo
Hanover, New Hampshire

How to Use This Book

This is a book you can read front to back, or open to any single result and find it self-contained. This short section explains how it is laid out, how the pieces depend on one another, and how to trace any statement back to the lecture or problem set it came from.

The shape of the book

The material is organized into eight parts.

Part I, Foundations.
The measurement model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, why classical sampling is not the end of the story, and all the linear algebra and convexity the rest of the book relies on.

Part II, The Sparse Recovery Problem.
What sparsity is, why exact recovery is even possible, and the two theoretical pillars that guarantee it: spark and the null space property. This part also introduces the convex relaxation from ℓ_0 to ℓ_1 that the whole subject turns on.

Part III, Algorithms for Sparse Recovery.
How to actually compute the answer: greedy pursuit methods on one side, proximal and iterative thresholding methods on the other.

Part IV, Why Random Sensing Works.
The restricted isometry property, coherence, and the central theorem that a random matrix with $m \approx s \log(N/s)$ rows lets you recover any s-sparse signal. This is the theoretical heart of the course, proved in full.

Part V, Low-Rank and Factorization Models.
Everything from Parts II–IV, redone with matrices in place of vectors and rank in place of sparsity: matrix completion, robust PCA, dictionary learning, and nonnegative matrix factorization.

Part VI, Extensions and Applications.
Joint sparsity across many signals, sparse representation-based classification (the application that ties the theory to a working face-recognition system), compressed sensing in practice (single-pixel cameras and accelerated MRI), the bridge to deep learning, and a closing synthesis that casts the whole book as one template.

Part VII, The Problem Sets.
All nine problem sets, each problem restated in full and solved completely, cross-referenced to the chapters and to the Python notebooks.

Part VIII, Practice Examinations.
A bank of fully worked problems sorted by type, and complete practice exams with solutions.

The appendices collect the linear-algebra and probability facts used in the main text, a complete treatment of the singular value decomposition, a timeline of the field, a glossary of notation, and an annotated guide to further reading.

Three kinds of problems

Almost every question in this subject is one of three kinds, and it pays to classify a problem in the first ten seconds.

1. **Compute** something concrete: a spark, a mutual coherence, a singular value decomposition, one step of orthogonal matching pursuit, a soft-threshold, a measurement count m. These reward careful arithmetic. Show every step.
2. **Prove** an equivalence or an inequality: that the null space property is equivalent to recovery, that spark controls uniqueness, the Cauchy–Schwarz inequality, the fact that ℓ_0 is not a norm. These reward structure. State what you assume, state what you want, then go by contradiction or by construction.
3. **Explain** a design choice: why a structured random matrix instead of a Gaussian one, why the nuclear norm, why nonnegativity produces parts. These reward one clear sentence and one supporting fact.

The worked examples in every chapter, and the entire problem bank in Part VIII, are sorted with these three categories in mind.

Conventions

- **Numbering.** Definitions, theorems, lemmas, propositions, corollaries, remarks, and examples share a single counter within each chapter. So Definition 4.2 and Theorem 4.3 are consecutive, and you can find any result by its number alone. Equations, figures, and tables are numbered separately, also by chapter.
- **Cross-references** are live links in the digital version. "See Chapter 1" and similar pointers will take you to the exact place.
- **Colored boxes are rare and meaningful.** A box with a dark border holds the single canonical statement of a major result. A green box is a step-by-step recipe. A purple box points to the problem sets and the notebooks. A red left-rule flags a subtlety or a common mistake. Everything else is ordinary prose and displayed mathematics.
- **The notebooks.** The course was half theory and half code. Where a result was implemented numerically, the text points to the accompanying problem-set notebook and describes what it computes, so the computational and the mathematical sides stay connected.
- **Vectors and matrices** are bold: $\boldsymbol{x}$, $\boldsymbol{A}$. Sets and operators are named in the notation guide that follows this section, and an exhaustive symbol glossary appears in Appendix G.

The lecture map

The chapters follow themes, not the lecture calendar. The table below records, for every chapter and problem-set chapter, exactly which lectures of ENGS 109 and which problem sets it draws on, so that any statement can be traced to its source.

Chapter	Lectures	Problem sets
1. The Compressed Sensing Revolution	L1	PS1
2. Mathematical Preliminaries	L2, L3, L6, L11X	PS4
3. Sparsity and the ℓ_0 Problem	L2, L2X, L3, L4, L5	PS1
4. Uniqueness: Spark and the NSP	L4, L5, L8, L9	PS2
5. Convex Relaxation and ℓ_1	L4, L5, L6	PS2
6. Greedy Pursuit Algorithms	L6, L7, L8, L8X, L11X	PS3, PS4
7. Proximal and Iterative Solvers	L26, L27	PS6
8. Coherence and the RIP	L9, L10, L11, L12	PS4
9. Random Matrices and the RIP	L12, L13, L16–L19	PS5
10. Structured Random Matrices	L14, L15, L16	PS5, PS6
11. Low-Rank Recovery, Completion	L20, L21	PS8
12. Robust PCA	L22, L23	—
13. Dictionary Learning	L23, L24	PS8
14. Nonnegative Matrix Factorization	L25	PS9
15. Joint Sparsity and MMV	L26	—
16. Sparse Representation Classification	L8, L17X, L19, L20X	PS7
17. Compressed Sensing in Practice	L1, L14, L26	—
18. Sparse Models and Deep Learning	L26	—
19. One Template, Many Problems	L26	—
Practice Examinations	L25X, final review	all

An "X" on a lecture number marks an X-hour session, which in this course was usually a problem-set walkthrough or an application. Those sessions are folded into the relevant chapters and into the problem-set solutions.

Dependencies and reading paths

The fastest path to the central theorem of the subject is Chapter 1 → Chapter 2 → Chapter 3 → Chapter 4 → Chapter 8 → Chapter 9. A reader who cares most about algorithms can go Chapter 3 → Chapter 5 → Chapter 6 → Chapter 7. The matrix chapters of Part V depend on the singular value decomposition, which is developed from scratch in Chapter 11 and summarized in Appendix D. The application chapters of Part VI can be read any time after Chapter 5.

Contents

Notation and Conventions

The notation below is used consistently throughout the book. A fuller glossary, including every named constant and operator, appears in Appendix G.

Scalars, vectors, matrices	
$x,\ \lambda,\ \alpha$	scalars (lightface)
$\boldsymbol{x},\ \boldsymbol{y},\ \boldsymbol{v}$	vectors (boldface lowercase); $\boldsymbol{x} \in \mathbb{C}^N$ unless stated
$\boldsymbol{A},\ \boldsymbol{F},\ \boldsymbol{\Psi}$	matrices (boldface uppercase)
$x_i,\ A_{ij}$	entries of $\boldsymbol{x}$ and $\boldsymbol{A}$
$\boldsymbol{a}_j$	the j-th column of $\boldsymbol{A}$ (an "atom")
$\boldsymbol{A}^\top,\ \boldsymbol{A}^*$	transpose and conjugate (Hermitian) transpose
$\boldsymbol{A}^\dagger$	Moore–Penrose pseudoinverse
$\boldsymbol{I},\ \mathbf{1},\ \mathbf{0}$	identity matrix, all-ones vector, zero vector
The measurement model	
$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$	$\boldsymbol{x} \in \mathbb{C}^N$ unknown signal, $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ sensing matrix, $\boldsymbol{y} \in \mathbb{C}^m$ measurements
N	ambient dimension (length of the signal)
m	number of measurements; the regime of interest is $m \ll N$
s	sparsity level (number of nonzeros)
$S,\ T$	index sets (supports); $\boldsymbol{x}_S$, $\boldsymbol{A}_S$ restrict to S
Norms and inner products	
$\|\boldsymbol{x}\|_p$	ℓ_p norm, $\left(\sum_i \|x_i\|^p\right)^{1/p}$
$\|\boldsymbol{x}\|_0$	number of nonzero entries (the ℓ_0 "norm")
$\langle \boldsymbol{x},\ \boldsymbol{y}\rangle$	inner product, $\sum_i \overline{x_i}\, y_i$
$\|\boldsymbol{A}\|_{p\to q}$	induced operator norm
$\|\boldsymbol{A}\|_F,\ \|\boldsymbol{A}\|_*$	Frobenius norm, nuclear norm
Sets and operators	
$\mathbb{R},\ \mathbb{C},\ \mathbb{Z},\ \mathbb{N}$	real, complex, integer, natural numbers
$\operatorname{supp}(\boldsymbol{x})$	support, $\{i : x_i \neq 0\}$
$\operatorname{spark}(\boldsymbol{A})$	smallest number of linearly dependent columns
$\operatorname{rank},\ \operatorname{tr},\ \operatorname{diag}$	rank, trace, diagonal
$\mathcal{N}(\boldsymbol{A}),\ \mathcal{R}(\boldsymbol{A})$	null space and range (column space)
$\operatorname{sgn},\ \operatorname{prox},\ \mathcal{S},\ \mathcal{H}$	sign, proximal operator, soft- and hard-threshold
$\mathbb{E},\ \mathbb{P},\ \operatorname{Var}$	expectation, probability, variance
The canonical problems	
(P_0)	$\min\|\boldsymbol{x}\|_0$ s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$
(P_1)	$\min\|\boldsymbol{x}\|_1$ s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ (basis pursuit)
$(\mathrm{P}_{1,\varepsilon})$	$\min\|\boldsymbol{x}\|_1$ s.t. $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \le \varepsilon$
$(\mathrm{PM}_0),\ (\mathrm{PM}_1)$	the matrix analogues: rank and nuclear-norm minimization

Conventions. Indices run from 1 unless stated otherwise. Logarithms are natural unless a base is written. "With high probability" means with probability tending to 1 as the dimension grows, at a rate made explicit each time. The symbol $\lozenge$ closes an example and $\blacksquare$ closes a proof.

PART I

Foundations

CHAPTER 1

The Compressed Sensing Revolution

Information is the resolution of uncertainty.

Claude Shannon

Every digital device you own is built on a quiet assumption: to capture a signal faithfully, you must sample it densely. A camera with more pixels sees more. A microphone that samples faster hears more. The classical theory of sampling, due to Nyquist and Shannon, makes this precise and sets a hard floor on how fast you must sample to avoid losing information. For half a century that floor was taken as a law of nature.

This book is about the discovery, in the mid-2000s, that the floor is an illusion for a very large and very useful class of signals. If a signal is *sparse*, meaning it has only a few degrees of freedom hidden inside a high-dimensional representation, then it can be reconstructed exactly from far fewer measurements than the classical theory demands, and the reconstruction can be carried out by a tractable algorithm. The name for this circle of ideas is *compressed sensing*. This first chapter tells the story at a high level: a puzzle that contains the whole idea in miniature, the classical barrier that compressed sensing breaks, the central equation that organizes the entire course, the structural assumption (sparsity) that makes the magic possible, and the applications and history that gave the field its urgency. Everything stated informally here is made precise in the chapters that follow.

1.1 A Puzzle to Begin

We start with a puzzle, because the puzzle is the entire subject compressed into one question.

The Hundred Bags Puzzle

You have 100 bags, each containing 100 coins. In 99 of the bags every coin weighs exactly 10 g. In one defective bag every coin weighs 11 g, one gram too heavy. You have a single digital scale that reports an exact total weight (not a balance that compares two pans). **What is the minimum number of weighings needed to identify the defective bag?**

The instinct of most people is to weigh bags one at a time, or to split the bags into halves and binary-search, which would take about $\log_2 100 \approx 7$ weighings. Both instincts are beaten badly. The answer is **one**.

Recipe

Strategy. Number the bags 1 through 100. From bag k, take exactly k coins. So you take 1 coin from bag 1, 2 coins from bag 2, and so on up to 100 coins from bag 100. Put all of these selected coins together on the scale in a *single* weighing.

Let us compute what the scale reads. The total number of coins placed on the scale is

$$\sum_{k=1}^{100} k = \frac{100 \cdot 101}{2} = 5050.$$

If every coin weighed the nominal 10 g, the scale would read

$$W_{\text{nominal}} = 5050 \times 10\,\text{g} = 50\,500\,\text{g}.$$

Now suppose bag j is the defective one. The j coins that we drew from bag j each weigh one gram too much, so they contribute an *excess* of exactly $j \times 1\,\text{g} = j$ grams over the nominal. Every other coin is nominal. Therefore the scale actually reads

$$W_{\text{actual}} = 50\,500\,\text{g} + j \text{ grams}.$$

The excess weight, which we read directly off the scale, *is* the index of the defective bag:

$$\boxed{j = W_{\text{actual}} - 50\,500\,\text{g}.}$$

If the scale reads 50 537 g, then $j = 37$ and bag 37 is defective. One weighing, done.

1.1.1 Why this is the whole course in miniature

The trick was to encode the unknown (which bag) into a single cleverly designed measurement. Let us write that encoding in the language we will use for the rest of the book. Define a vector $\boldsymbol{x} \in \mathbb{R}^{100}$ whose entries record the *excess weight per coin* in each bag:

$$x_k = \begin{cases} 1 & \text{if bag } k \text{ is defective,} \\ 0 & \text{otherwise.} \end{cases}$$

This vector has exactly one nonzero entry. We call such a vector 1-*sparse*. Now define the measurement vector $\boldsymbol{a} = (1, 2, 3, \ldots, 100)^{\top} \in \mathbb{R}^{100}$, the recipe "take k coins from bag k." The single number the scale reports (the excess) is the inner product

$$y = \boldsymbol{a}^{\top}\boldsymbol{x} = \sum_{k=1}^{100} k\, x_k = j.$$

We made $m = 1$ measurement of an unknown vector living in $N = 100$ dimensions, and we recovered it completely. The classical view says that to determine 100 unknowns you need 100 equations. That view is correct *in general* and wrong *here*, because our unknown was not a generic vector in $\mathbb{R}^{100}$. It had structure: only one nonzero entry. The measurement was designed to exploit exactly that structure (every coordinate of $\boldsymbol{a}$ is distinct, so the value of the inner product reveals which coordinate is active).

Remark 1.1. The leap from this puzzle to the general theory is the leap from 1-sparse to s-sparse, and from a hand-crafted measurement vector to a small number of *random* measurement vectors. The central theorem of the subject, Chapter 9, will say that an s-sparse vector in $\mathbb{R}^N$ can be recovered from about $m = O\big(s \log(N/s)\big)$ random linear measurements. For $s = 1$ and $N = 100$ this is roughly $\log 100 \approx 5$ random measurements, already far below $N = 100$. Our deterministic one-weighing design does even better because it uses the exact structure of the problem. The art of compressed sensing is to get this kind of efficiency *without* knowing in advance which coordinates are active.

> Structure is everything. If the "signal" were pure white noise, with each entry an independent random number and no pattern at all, no clever measurement could compress it; you really would need N numbers to pin down N unrelated unknowns. Compressed sensing is not a violation of information theory. It is the statement that real signals (images, audio, medical scans, ratings matrices) are nowhere near random; they carry a small number of meaningful degrees of freedom, and a good measurement system can target those directly.

1.2 The Classical Barrier: Shannon and Nyquist

To appreciate what compressed sensing overturns, we state the classical sampling theorem it appears to contradict. A continuous-time signal $x(t)$ is *bandlimited* to a maximum frequency $f_{\max}$ if its Fourier transform $\hat{x}(f)$ vanishes for all $|f| > f_{\max}$; the signal contains no oscillations faster than $f_{\max}$ cycles per second.

Theorem 1.2 (Shannon–Nyquist Sampling Theorem). Let $x(t)$ be a continuous-time signal bandlimited to $f_{\max}$. Then x is completely determined by its samples taken at a uniform rate f_s, and can be reconstructed exactly from them, if and only if

$$f_s \;\geq\; 2f_{\max}. \tag{1.1}$$

The threshold rate $f_s = 2f_{\max}$ is the *Nyquist rate*. When Eq. (1.1) holds, reconstruction is given by the sinc interpolation formula

$$x(t) \;=\; \sum_{n=-\infty}^{\infty} x\!\left(\frac{n}{f_s}\right) \operatorname{sinc}\!\big(f_s t - n\big), \qquad \operatorname{sinc}(u) = \frac{\sin(\pi u)}{\pi u}. \tag{1.2}$$

The theorem is a cornerstone of digital signal processing, and it is sharp for arbitrary bandlimited signals: sample any slower than $2f_{\max}$ and distinct signals collapse onto the same samples (the phenomenon called aliasing), so exact reconstruction becomes impossible in general.

Example 1.3 (CD-quality audio). Human hearing extends to about $f_{\max} = 20\,\text{kHz}$. The Nyquist rate is therefore $f_s \geq 40\,\text{kHz}$. Compact-disc audio samples at $f_s = 44.1\,\text{kHz}$, set slightly above the Nyquist rate so that a practical (not perfectly sharp) anti-aliasing filter can roll off the inaudible content above 20 kHz before sampling. ◊

> Read the hypothesis of Theorem 1.2 carefully. It asks only that the signal be *bandlimited*. It says nothing about any further structure. This is both the strength of the theorem (it works for *every* bandlimited signal) and the opening that compressed sensing exploits (most real signals have far more structure than mere bandlimitedness, and that extra structure is left completely unused by classical sampling).

1.3 The Empirical Miracle

Before any theory, it helps to see that the thing actually works. The following three demonstrations were shown in the first lecture, and each one looks, at first, impossible.

1. **A one-dimensional signal from under 10% of its samples.** Take a signal that is a sum of a few sinusoids. Sample it at fewer than 10% of the number of samples the Nyquist rate would require, and choose those sample *positions at random*. Feed the samples to a compressed sensing recovery algorithm (the ℓ_1 minimization of Chapter 5). The reconstruction lands exactly on the original. Plotted together, the recovered curve and the true curve are visually identical.
2. **A photograph with 75% of its pixels deleted.** Take an ordinary photograph (in the lecture, a picture of Dartmouth's Baker Library), throw away three quarters of the pixels at random, and reconstruct. The result is a clean, recognizable image.
3. **The same photograph with 95% of its pixels deleted.** Keep only one pixel in twenty. The reconstruction is no longer perfect, but the building, the tower, and the skyline are all clearly there.

Figure 1.1 is a schematic of the first demonstration. The reason these work is that the signals are *compressible*: a natural image is nearly sparse in a wavelet basis, and a few-tone audio clip is exactly sparse in the Fourier basis. The rest of this book is the precise statement and proof of when, and why, this kind of recovery from radical undersampling succeeds.

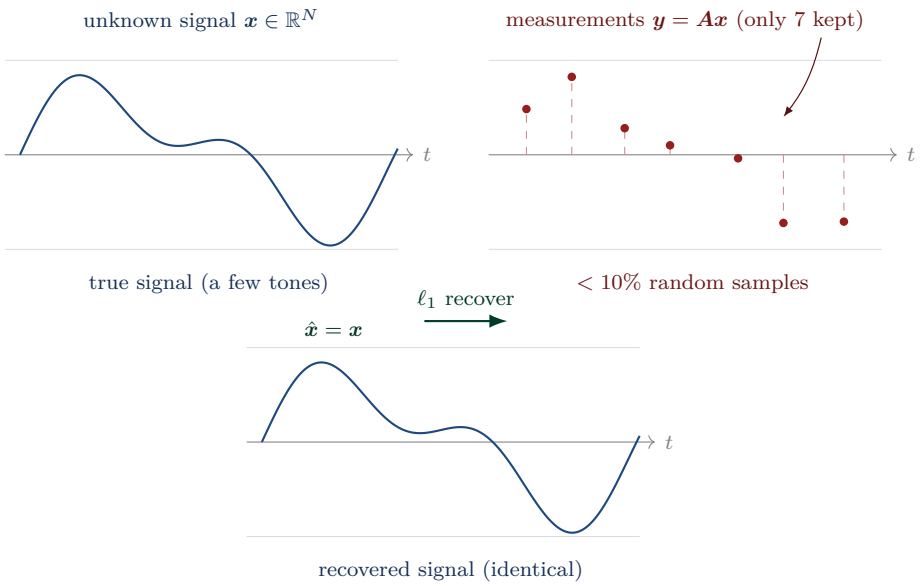


Figure 1.1. The compressed sensing miracle in one dimension. The unknown signal $\boldsymbol{x}$ has only a few Fourier tones; it is sampled at random positions far below the Nyquist rate to give the measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ (here just 7 samples), and ℓ_1 minimization recovers it exactly, so the reconstruction $\hat{\boldsymbol{x}}$ lands on the original.

1.4 The Central Equation

From here on, all of the demonstrations above are instances of a single linear model. It is worth stating once, prominently, because it organizes the entire book.

The Core Measurement Model

Throughout this course we study the linear system

$$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x}, \tag{1.3}$$

where

- $\boldsymbol{x} \in \mathbb{C}^N$ is the **unknown signal** we wish to recover;
- $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ is the **sensing (measurement) matrix**, known to us and often something we get to design;
- $\boldsymbol{y} \in \mathbb{C}^m$ is the vector of **measurements** we observe.

The central question is: given $\boldsymbol{y}$ and $\boldsymbol{A}$, when can we recover $\boldsymbol{x}$, and by what algorithm?

Each measurement y_i is the inner product of the signal with one row of $\boldsymbol{A}$:

$$y_i = \sum_{j=1}^{N} A_{ij}\, x_j = \left\langle \overline{\boldsymbol{a}^{(i)}},\, \boldsymbol{x} \right\rangle,$$

where $\boldsymbol{a}^{(i)}$ is the i-th row. So "taking a measurement" means "correlating the unknown signal against a known test vector." In the coins puzzle, $\boldsymbol{A}$ was the single row $(1, 2, \ldots, 100)$. In an MRI machine, the rows of $\boldsymbol{A}$ are Fourier modes. In the single-pixel camera of Section 1.8, the rows are random black-and-white patterns. The answer to the central question turns entirely on the relationship between m and N, and on what we know about $\boldsymbol{x}$.

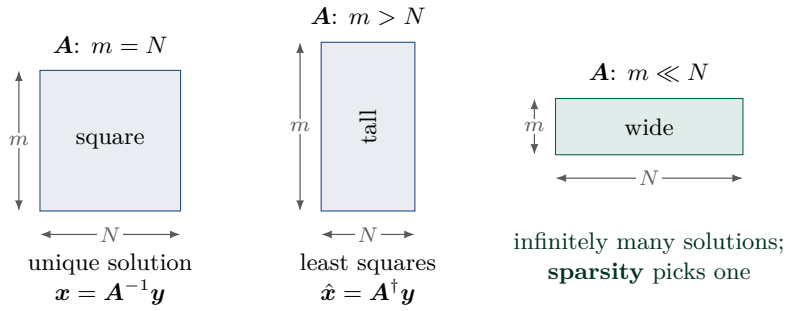


Figure 1.2. The three regimes of $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, drawn to scale: the height of each block is the number of measurements m, the width is the number of unknowns N. Classical linear algebra handles the square ($m = N$) and tall ($m > N$) cases. The wide case $m \ll N$ is underdetermined and is the subject of compressed sensing.

1.5 Three Regimes

The shape of $\boldsymbol{A}$ (how its number of rows m compares to its number of columns N) splits the problem into three regimes, drawn in Fig. 1.2. The first two are classical. The third is the home of this book.

1.5.1 Case $m = N$: the square invertible system

When $\boldsymbol{A}$ is square and invertible, the system has exactly one solution,

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}.$$

This is the world of an introductory linear algebra course: as many equations as unknowns, and (generically) a unique answer.

Example 1.4. With $N = m = 2$, $\boldsymbol{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ and $\boldsymbol{y} = \begin{pmatrix} 5 \\ 11 \end{pmatrix}$, we have $\det(\boldsymbol{A}) = 4 - 6 = -2 \neq 0$, so

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y} = \frac{1}{-2}\begin{pmatrix} 4 & -2 \\ -3 & 1 \end{pmatrix}\begin{pmatrix} 5 \\ 11 \end{pmatrix} = \frac{1}{-2}\begin{pmatrix} 20 - 22 \\ -15 + 11 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}. \qquad \lozenge$$

1.5.2 Case $m > N$: the overdetermined system

With more equations than unknowns, the system is overdetermined and usually has *no* exact solution, because the measured $\boldsymbol{y}$ need not lie in the N-dimensional column space of $\boldsymbol{A}$ inside the larger space $\mathbb{C}^m$. The standard remedy is to ask for the $\boldsymbol{x}$ that comes closest, in the sense of least squares: minimize $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$. When $\boldsymbol{A}$ has full column rank, the minimizer is unique and given by the *pseudoinverse*

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^*\boldsymbol{A})^{-1}\boldsymbol{A}^*\boldsymbol{y} \;=\; \boldsymbol{A}^{\dagger}\boldsymbol{y}. \tag{1.4}$$

This is the workhorse of regression and curve fitting. We derive Eq. (1.4) carefully in Chapter 2.

Example 1.5 (Fitting a line). Fit $y = \beta_0 + \beta_1 t$ to the three points $(0, 1), (1, 2.5), (2, 4.2)$. Stacking the model rows,

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} 1 \\ 2.5 \\ 4.2 \end{pmatrix}, \qquad \boldsymbol{A}^{\top}\boldsymbol{A} = \begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix}, \qquad \boldsymbol{A}^{\top}\boldsymbol{y} = \begin{pmatrix} 7.7 \\ 10.9 \end{pmatrix}.$$

Solving the 2×2 system,

$$\hat{\boldsymbol{x}} = \begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix}^{-1}\begin{pmatrix} 7.7 \\ 10.9 \end{pmatrix} = \frac{1}{6}\begin{pmatrix} 5 & -3 \\ -3 & 3 \end{pmatrix}\begin{pmatrix} 7.7 \\ 10.9 \end{pmatrix} = \begin{pmatrix} 0.97 \\ 1.60 \end{pmatrix},$$

so the best-fit line is $y \approx 0.97 + 1.60\,t$. ◊

1.5.3 Case $m \ll N$: the underdetermined system

Now the case that classical theory gives up on. With far fewer measurements than unknowns, the system $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ has *infinitely many* solutions. If $\boldsymbol{x}_0$ is any one solution, then for every vector $\boldsymbol{z}$ in the null space $\mathcal{N}(\boldsymbol{A}) = \{\boldsymbol{z} : \boldsymbol{A}\boldsymbol{z} = \boldsymbol{0}\}$, the vector $\boldsymbol{x}_0 + \boldsymbol{z}$ is also a solution. The null space has dimension at least $N - m$, which is large, so the solution set is a vast affine subspace.

> With no further information, the underdetermined problem is genuinely unsolvable: nothing in the measurements distinguishes $\boldsymbol{x}_0$ from $\boldsymbol{x}_0 + \boldsymbol{z}$. Recovering a unique answer requires an extra assumption that singles one solution out. The assumption that works, and that the whole book develops, is **sparsity**.

1.6 Sparsity: The Escape

The structural assumption that rescues the underdetermined problem is that the signal has only a few nonzero entries.

Definition 1.6 (Sparsity). A vector $\boldsymbol{x} \in \mathbb{C}^N$ is **s-sparse** if it has at most s nonzero entries:

$$\|\boldsymbol{x}\|_0 \;:=\; |\{i : x_i \neq 0\}| \;\leq\; s.$$

The quantity $\|\boldsymbol{x}\|_0$ is the number of nonzeros, often called the ℓ_0 "norm" (the quotation marks are deliberate, and explained in Chapter 3). The set of all s-sparse vectors is written $\Sigma_s := \left\{\boldsymbol{x} \in \mathbb{C}^N : \|\boldsymbol{x}\|_0 \leq s\right\}$.

Sparsity is the mathematical form of a very old principle.

Remark 1.7 *(Occam's razor).* William of Ockham's rule, that among competing explanations consistent with the evidence one should prefer the simplest, becomes a precise optimization once we agree that "simplest" means "fewest nonzeros." Among all $\boldsymbol{x}$ that satisfy the measurements $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, seek the sparsest:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \tag{1.5}$$

Problem Eq. (1.5) is called (P_0). It is the right thing to want. Unfortunately, as stated it is NP-hard, because $\|\cdot\|_0$ is a combinatorial object. The single most important theme of the course is that the convex relaxation of Eq. (1.5), obtained by replacing $\|\boldsymbol{x}\|_0$ with $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$, very often has exactly the same solution, while being solvable by a linear program. We develop this in Chapter 5.

The headline result, to be proved in Chapter 9, is that sparsity collapses the number of measurements you need from N to something close to the true number of degrees of freedom.

The Headline (informal)

If $\boldsymbol{x} \in \mathbb{C}^N$ is s-sparse and $\boldsymbol{A}$ is a suitable random matrix with

$$m \;\approx\; C\, s\, \log(N/s)$$

rows, then $\boldsymbol{x}$ is the unique sparsest solution of $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ and can be recovered exactly by ℓ_1 minimization, with overwhelming probability. The number of measurements scales with the sparsity s, not with the ambient dimension N.

A useful way to keep this honest is to count degrees of freedom. A generic vector in $\mathbb{R}^N$ has N degrees of freedom, so naively you need N numbers to pin it down. An s-sparse vector really has only about $2s$ degrees of freedom: s values, plus s locations telling you where those values sit. The theory is the precise sense in which $m \approx s\log(N/s)$ cleverly chosen measurements suffice to recover those $2s$ degrees of freedom. Keep this accounting in mind; it is the sanity check behind every measurement bound in the book.

1.7 Applications

The field did not grow out of a puzzle. It grew because several important measurement problems are expensive, slow, or physically constrained in exactly the way compressed sensing addresses. Three applications recur throughout the course.

1.7.1 Accelerating MRI

Magnetic resonance imaging acquires its data in the Fourier domain (radiologists call it k-space). A full scan samples a dense grid of Fourier coefficients, which is slow: the patient must hold still for many minutes, and scanner time is expensive. But MR images are compressible (nearly sparse in a wavelet or a finite-difference basis), so one can acquire only a fraction of the Fourier coefficients, at randomly chosen frequencies, and reconstruct by ℓ_1-based recovery. The model is

$$\boldsymbol{y} = \boldsymbol{P}_\Omega \boldsymbol{F} \boldsymbol{x}, \tag{1.6}$$

where $\boldsymbol{x} \in \mathbb{C}^N$ is the (vectorized) image, $\boldsymbol{F}$ is the discrete Fourier transform, $\Omega \subset \{1, \dots, N\}$ with $|\Omega| = m \ll N$ is the set of sampled frequencies, and $\boldsymbol{P}_\Omega$ selects those rows. Acquiring only 20 to 30 percent of k-space can cut scan times by factors of four to eight, and compressed sensing MRI is now FDA-cleared and shipping in commercial scanners. We return to the structured-Fourier sensing of Eq. (1.6) in Chapter 10.

1.7.2 The Netflix Problem and Matrix Completion

In 2006 Netflix released a dataset of about 480,000 users rating 17,770 movies and offered $1,000,000 to anyone who could improve its recommendations by 10%. The ratings form a matrix $\boldsymbol{M}$ with more than eight billion entries, of which only about 1% are observed; most users have rated only a handful of movies. The question is whether the missing entries can be predicted from the few that are known.

They can, because the matrix is approximately *low rank*. Tastes are not arbitrary; a modest number of latent factors (genre, era, tone, a taste for a given actor) explains most of the variation, so $\boldsymbol{M} \approx \boldsymbol{U}\boldsymbol{V}^\top$ with $\boldsymbol{U}$ and $\boldsymbol{V}$ having only r columns each, r small. Low rank is the matrix version of sparsity: a low-rank matrix has only a few nonzero singular values. This analogy is exact and powerful, and it carries the entire vector theory over to matrices. The convex surrogate for rank is the nuclear norm $\|\boldsymbol{M}\|_* = \sum_i \sigma_i(\boldsymbol{M})$, the ℓ_1 norm of the singular values. We build matrix completion from the ground up in Chapter 11.

1.7.3 Image and Video Compression

Ordinary compression already exploits sparsity, just after the fact. JPEG represents each block of an image in the discrete cosine basis and keeps only the few large coefficients; MPEG does the same for video and adds motion prediction.

Definition 1.8 (Compression ratio). The **compression ratio** of a scheme is

$$\mathrm{CR} = \frac{\text{uncompressed size}}{\text{compressed size}}.$$

A ratio of 20:1 means the stored file is one twentieth the size of the original, so 95% of the raw data was discarded with little perceptual loss.

Typical ratios are around 20:1 for JPEG images and 40:1 for MPEG video. This invites an awkward question. If a camera is going to keep only 5% of the data, why did it measure all 100% in the first place, only to throw most of it away? Compressed sensing is the answer that you do not have to: design the measurement system to acquire the informative part of the signal *directly*, and skip the acquire-everything-then-discard detour.

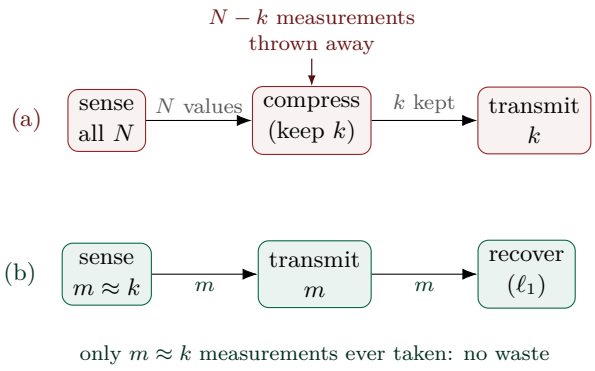


Figure 1.3. (a) The classical pipeline senses all N values, then compresses, then discards the $N-k$ it just measured. (b) Compressed sensing acquires only $m \approx k$ measurements up front, carries that same small amount through, and recovers the signal at the other end, so nothing is wasted.

1.8 A Short History

Compressed sensing as a mathematical field was born in 2005 and 2006 in the work of Emmanuel Candès, Justin Romberg, Terence Tao, and David Donoho.

Remark 1.9 *(How it started).* The often-told origin story is that Candès and Tao met at their children's daycare. Candès mentioned that ℓ_1 minimization seemed, in his numerical experiments, to recover sparse signals from random measurements *exactly*, and that he could not prove it. Tao was initially skeptical, then took the problem home and worked out the key ideas. The resulting papers, including "Robust Uncertainty Principles" (joint with Justin Romberg) [14] and "Near-Optimal Signal Recovery from Random Projections" [12], together with Donoho's "Compressed Sensing" [20], established the field. The three results, stated loosely, are that an s-sparse signal can be recovered exactly from $m = O(s \log(N/s))$ random linear measurements, that the recovery is tractable because it reduces to a linear program, and that the whole thing is robust to noise and to signals that are only approximately sparse.

1.8.1 The single-pixel camera

The clearest hardware embodiment of $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is the single-pixel camera built at Rice University [23]. Instead of an array of N pixel sensors, it uses one photodiode and a digital micromirror device that imposes a programmable black-and-white pattern on the scene. Each measurement is a single number: the inner product of the scene with the current random pattern. After $m \ll N$ such measurements, a compressed sensing algorithm reconstructs the full N-pixel image. The rows of $\boldsymbol{A}$ are the random patterns, and the readings y_i are the photodiode outputs. The point is not that one should photograph the world one inner product at a time, but that in wavelengths where pixel arrays are ruinously expensive (parts of the infrared, for instance) a single good detector plus compressed sensing can be the practical design.

Remark 1.10 *(Compression and intelligence).* There is a thread connecting this subject to modern machine learning. To predict the next pixel, sample, or word, a model must have captured the structure of its data, and capturing structure is exactly what makes data compressible. The compressed sensing philosophy, that one should exploit structure (sparsity, low rank, smoothness) to represent and recover signals from minimal data, is the same instinct that drives representation learning. We will see this concretely when sparse coding reappears as a neural network in Chapter 13.

What's Ahead

The plan of the book follows the logic of the problem. To recover a sparse $\boldsymbol{x}$ from $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ we must answer three questions in turn, and then generalize.

1. **When is the answer unique?** Even before algorithms, we need to know that the sparsest solution is the *only* sparse solution. Chapter 4 answers this with the spark of a matrix and the null space property.
2. **How do we compute it?** Chapters 5 to 7 develop the two families of algorithms, convex (ℓ_1 minimization) and greedy (matching pursuit and its relatives), and the iterative solvers that run them.
3. **Which matrices $\boldsymbol{A}$ work, and how small can m be?** Chapters 8 to 10 develop coherence, the restricted isometry property, and the random-matrix theory that proves $m \approx s \log(N/s)$ suffices.

After that, Part V repeats the entire story with matrices and rank in place of vectors and sparsity, giving matrix completion, robust PCA, dictionary learning, and nonnegative matrix factorization. Part VI covers joint sparsity and the sparse-representation classifier. The problem sets and practice exams put all of it to work.

Notes and References

The sampling theorem is due to Nyquist [41] and Shannon [46]. The founding compressed sensing papers are [12, 14, 20]; a very readable overview is Candès and Wakin [13]. The two standard textbooks for the whole subject are Foucart and Rauhut [28] and Elad [25], both of which the course drew on. Compressed sensing MRI is from Lustig, Donoho, and Pauly [36]; the single-pixel camera is from Duarte et al. [23]; matrix completion and the Netflix story are taken up with full references in Chapter 11.

Exercises

Exercise 1.1. In the hundred bags puzzle, suppose instead that an unknown number of bags may be defective (each defective coin is still one gram heavy). Explain why the single-weighing scheme "take k coins from bag k" no longer identifies the defective set, by exhibiting two different sets of defective bags that produce the same total weight. How does this failure correspond to the signal $\boldsymbol{x}$ no longer being 1-sparse?

Exercise 1.2. A signal is bandlimited to $f_{\max} = 4\,\text{kHz}$ (telephone speech). What is the Nyquist rate? Standard telephony samples at 8 kHz; how much headroom is that over the Nyquist rate, and what is it for?

Exercise 1.3. Let $\boldsymbol{A} \in \mathbb{R}^{m \times N}$ with $m < N$ and suppose $\boldsymbol{x}_0$ solves $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$. Show that the full solution set is $\{\boldsymbol{x}_0 + \boldsymbol{z} : \boldsymbol{z} \in \mathcal{N}(\boldsymbol{A})\}$, and that $\dim \mathcal{N}(\boldsymbol{A}) \geq N - m$. Conclude that without further assumptions the underdetermined problem has infinitely many solutions.

Exercise 1.4. Compute the compression ratio implied by "keep the largest 5% of the discrete cosine coefficients." If a raw image is 12 MB, roughly how large is the compressed file, ignoring overhead? Relate the 5% figure to the sparsity level s relative to N.

Exercise 1.5. Verify the least-squares line fit in Section 1.5 by directly minimizing $\sum_i (\beta_0 + \beta_1 t_i - y_i)^2$ over β_0, β_1: take partial derivatives, set them to zero, and confirm you recover the normal equations $\boldsymbol{A}^\top \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}^\top \boldsymbol{y}$ and the same numerical answer.

CHAPTER 2

Mathematical Preliminaries

All of this course is linear algebra in disguise.

a recurring remark in lecture

Everything that follows is built from a small set of tools: vectors and matrices, inner products and norms, the singular value decomposition, the pseudoinverse, and a little convexity and calculus. This chapter collects those tools and works every one of them on concrete numbers. None of it is deep, but fluency matters: a later proof stalls far more often on a forgotten norm inequality or a transposed pseudoinverse than on a genuinely hard idea. Read this chapter with a pencil, redo each worked computation, and come back to it whenever a derivation later in the book moves too fast. A condensed version of these facts, with no derivations, lives in Appendix B for quick reference, and the singular value decomposition gets its own focused treatment in Appendix D.

2.1 Vectors, Matrices, and Two Views of Multiplication

A **vector** $\boldsymbol{x} \in \mathbb{R}^n$ is an ordered list of n numbers, written as a column. A **matrix** $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ has m rows and n columns, with entry A_{ij} in row i and column j. A matrix is two things at once: a grid of numbers, and a *linear map* $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$ that sends $\mathbb{R}^n$ to $\mathbb{R}^m$. In this course $\boldsymbol{A}$ is the measurement operator: feed it the unknown signal and it returns what the sensor records.

The **transpose** $\boldsymbol{A}^\top$ swaps rows and columns, $(\boldsymbol{A}^\top)_{ij} = A_{ji}$, and reverses the order of products, $(\boldsymbol{A}\boldsymbol{B})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top$. For complex matrices the **conjugate transpose** (Hermitian adjoint) $\boldsymbol{A}^*$ transposes and conjugates, $(\boldsymbol{A}^*)_{ij} = \overline{A_{ji}}$; for real matrices $\boldsymbol{A}^* = \boldsymbol{A}^\top$. The Fourier matrix of Chapter 10 is complex, so there the distinction matters.

> Before any arithmetic, write the shape of every object and of every product you intend to form. The product $\boldsymbol{A}\boldsymbol{B}$ is defined only when the inner dimensions match: $(m \times n)(n \times p) = (m \times p)$. The expression $\boldsymbol{a}^\top \boldsymbol{b}$ is $(1 \times n)(n \times 1)$, a scalar (the inner product); the expression $\boldsymbol{a}\boldsymbol{b}^\top$ is $(n \times 1)(1 \times n)$, an $n \times n$ rank-one matrix (the outer product). Confusing these two is the most common dimension error in the course.

There are two equally correct pictures of a product $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$, and both are used constantly.

Two views of AB

(1) Entry / row-times-column. Entry C_{ij} is the inner product of row i of $\boldsymbol{A}$ with column j of $\boldsymbol{B}$:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}.$$

(2) Sum of rank-one outer products. If $\boldsymbol{a}_k$ is the k-th column of $\boldsymbol{A}$ and $\boldsymbol{b}_k^\top$ the k-th row of $\boldsymbol{B}$, then

$$\boldsymbol{A}\boldsymbol{B} = \sum_{k=1}^{n} \boldsymbol{a}_k \, \boldsymbol{b}_k^\top.$$

Each term is a rank-one $m \times p$ matrix. This second view is how the SVD, K-SVD, and every low-rank approximation in the book are understood.

Example 2.1 (A product computed both ways). Let $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ and $\boldsymbol{B} = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$. The product is 2×2.

By the entry view,

$$C_{11} = 1 \cdot 7 + 2 \cdot 9 + 3 \cdot 11 = 58, \quad C_{12} = 1 \cdot 8 + 2 \cdot 10 + 3 \cdot 12 = 64,$$
$$C_{21} = 4 \cdot 7 + 5 \cdot 9 + 6 \cdot 11 = 139, \quad C_{22} = 4 \cdot 8 + 5 \cdot 10 + 6 \cdot 12 = 154,$$

so $\boldsymbol{C} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$. By the outer-product view,

$$\begin{bmatrix} 1 \\ 4 \end{bmatrix} \begin{bmatrix} 7 & 8 \end{bmatrix} + \begin{bmatrix} 2 \\ 5 \end{bmatrix} \begin{bmatrix} 9 & 10 \end{bmatrix} + \begin{bmatrix} 3 \\ 6 \end{bmatrix} \begin{bmatrix} 11 & 12 \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 28 & 32 \end{bmatrix} + \begin{bmatrix} 18 & 20 \\ 45 & 50 \end{bmatrix} + \begin{bmatrix} 33 & 36 \\ 66 & 72 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix},$$

and the two views agree, as they must. ◊

The single most useful mental model in the course is that $\boldsymbol{A}\boldsymbol{x}$ is a *linear combination of the columns of* $\boldsymbol{A}$, weighted by the entries of $\boldsymbol{x}$: if $\boldsymbol{A} = [\boldsymbol{a}_1 \mid \cdots \mid \boldsymbol{a}_n]$ then $\boldsymbol{A}\boldsymbol{x} = \sum_j x_j \boldsymbol{a}_j$.

Example 2.2 (Matrix-vector product as a column combination).

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \\ 1 \end{bmatrix} = 4 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 5 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix}.$$

A sparse $\boldsymbol{x}$ picks out a *few* columns and weights them. This is exactly why we will call the columns of $\boldsymbol{A}$ *atoms*, and why the "support" of $\boldsymbol{x}$ means "which atoms are used." ◊

2.2 Inner Products, Orthogonality, and Orthonormal Bases

Definition 2.3 (Inner product, orthogonality). The standard inner product of $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ is $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^\top \boldsymbol{v} = \sum_{i=1}^n u_i v_i$. It is symmetric, linear in each argument, and satisfies $\langle \boldsymbol{v}, \boldsymbol{v} \rangle = \|\boldsymbol{v}\|_2^2 \geq 0$. Over $\mathbb{C}^n$ the inner product carries a conjugate, $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^* \boldsymbol{v} = \sum_{i=1}^n \overline{u_i}\, v_i$, which keeps $\langle \boldsymbol{v}, \boldsymbol{v} \rangle = \|\boldsymbol{v}\|_2^2$ real and nonnegative; it is then conjugate-symmetric and linear in the second argument. We use the real form by default and the complex form only for the Fourier vectors of Chapters 8 and 10. Two vectors are **orthogonal** when $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$. The angle between them satisfies

$$\cos\theta = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\|_2 \, \|\boldsymbol{v}\|_2}.$$

A set $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n\}$ is **orthonormal** if $\langle \boldsymbol{q}_i, \boldsymbol{q}_j \rangle = \delta_{ij}$. Stacking such vectors as columns gives an **orthogonal matrix** $\boldsymbol{Q}$ with $\boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}$, hence $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^\top$ and $\|\boldsymbol{Q}\boldsymbol{x}\|_2 = \|\boldsymbol{x}\|_2$.

The inner product measures alignment: large and positive when two vectors point the same way, large and negative when opposite, zero when perpendicular. Every "correlate the residual against the columns" step in the greedy algorithms of Chapter 6 is just a batch of inner products, looking for the atom that aligns best with what is left to explain. The quantity $|\cos\theta|$ between two unit-norm columns is the mutual coherence of Chapter 8.

Example 2.4 (Inner product and angle). For $\boldsymbol{u} = (1, 2, 3)$ and $\boldsymbol{v} = (4, 5, 6)$ we have $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 4 + 10 + 18 = 32$, $\|\boldsymbol{u}\|_2 = \sqrt{14}$, $\|\boldsymbol{v}\|_2 = \sqrt{77}$, so $\cos\theta = 32/\sqrt{14 \cdot 77} = 32/\sqrt{1078} \approx 0.975$ and $\theta \approx 12.9°$: the vectors nearly align. ◊

Remark 2.5 *(Orthonormal bases make norms free).* If $\boldsymbol{\Psi}$ is orthonormal and $\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{\alpha}$, then $\|\boldsymbol{x}\|_2 = \|\boldsymbol{\alpha}\|_2$. Changing to an orthonormal basis does not distort ℓ_2 geometry, which is why "sparse in a basis" is a clean notion and why the Parseval identity holds for the Fourier transform. We lean on this throughout Chapter 10.

2.3 Vector Norms

Definition 2.6 (Norm). A **norm** on $\mathbb{R}^n$ (or $\mathbb{C}^n$) is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ satisfying three axioms:

1. *Positive definiteness:* $\|\boldsymbol{x}\| \geq 0$, with $\|\boldsymbol{x}\| = 0$ if and only if $\boldsymbol{x} = \boldsymbol{0}$.
2. *Absolute homogeneity:* $\|\alpha \boldsymbol{x}\| = |\alpha| \|\boldsymbol{x}\|$ for every scalar α.
3. *Triangle inequality:* $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$.

Definition 2.7 (The ℓ_p norms). For $1 \leq p < \infty$,

$$\|\boldsymbol{x}\|_p = \Big(\sum_{i=1}^n |x_i|^p \Big)^{1/p}, \qquad \|\boldsymbol{x}\|_\infty = \max_i |x_i|.$$

The three that appear everywhere are the taxicab norm $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$, the Euclidean norm $\|\boldsymbol{x}\|_2 = \sqrt{\sum_i x_i^2}$, and the max norm $\|\boldsymbol{x}\|_\infty$. The ℓ_0 **"norm"** $\|\boldsymbol{x}\|_0 = |\{i : x_i \neq 0\}|$ counts nonzeros; it is not a true norm.

The subscript p controls how harshly the norm penalizes spread-out energy. As p grows the norm cares more about the single largest entry and less about the many small ones, until at $p = \infty$ only the biggest entry matters. As p shrinks toward 0 the norm cares less about magnitude and more about how *many* entries are nonzero, until at $p = 0$ it is pure counting. The whole recovery problem lives in this tension: we want ℓ_0 but settle for ℓ_1, the smallest $p \geq 1$ (hence convex) that still prefers sparse vectors. Figure 2.1 shows the unit balls.

Example 2.8 (All four norms of $\boldsymbol{x} = (3, -4, 0, 1)$).

$$\begin{aligned} \|\boldsymbol{x}\|_0 &= 3 \quad \text{(three nonzeros; the 0 does not count)}, \\ \|\boldsymbol{x}\|_1 &= 3 + 4 + 0 + 1 = 8, \\ \|\boldsymbol{x}\|_2 &= \sqrt{9 + 16 + 0 + 1} = \sqrt{26} \approx 5.099, \\ \|\boldsymbol{x}\|_\infty &= \max\{3, 4, 0, 1\} = 4. \end{aligned}$$

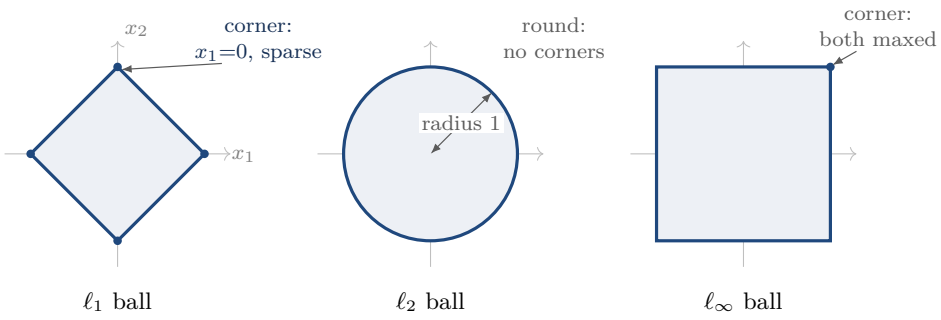


Figure 2.1. Unit balls $\{\boldsymbol{x} : \|\boldsymbol{x}\|_p \le 1\}$ in $\mathbb{R}^2$. The ℓ_1 ball has sharp corners on the axes (marked), where one coordinate is exactly zero; this is the geometric reason ℓ_1 minimization favors sparse solutions (Chapter 5). The ℓ_2 ball is round and prefers no direction, while the corners of the ℓ_∞ ball are the least sparse points of all, with both coordinates at their maximum.

The ordering $\|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_2 \le \|\boldsymbol{x}\|_1$ ($4 \le 5.099 \le 8$) is general: a larger p gives a smaller or equal norm. Keep it as a sanity check. ◊

Remark 2.9 *(Why ℓ_0 is not a norm).* Homogeneity demands $\|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\|$. Take $\boldsymbol{x} = (1, 0, 2)$, so $\|\boldsymbol{x}\|_0 = 2$, and $\alpha = 5$. Then $5\boldsymbol{x} = (5, 0, 10)$ still has two nonzeros, so $\|5\boldsymbol{x}\|_0 = 2$, whereas homogeneity would require $|5| \cdot 2 = 10$. Scaling never changes how many entries are nonzero, so ℓ_0 ignores magnitude entirely. It is properly called a quasi-norm, or just "the sparsity." We keep the norm notation $\|\cdot\|_0$ because it is standard, with the quotation marks as a reminder.

2.4 Two Fundamental Inequalities

Two inequalities underlie almost every estimate in the book. We state and prove both, because the proofs themselves are stock exam problems (they appear in Problem Set 4, treated in Chapter 23).

Theorem 2.10 (Cauchy–Schwarz). For all $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$,

$$|\langle \boldsymbol{u}, \boldsymbol{v} \rangle| \;\le\; \|\boldsymbol{u}\|_2 \, \|\boldsymbol{v}\|_2,$$

with equality if and only if $\boldsymbol{u}$ and $\boldsymbol{v}$ are linearly dependent.

Proof. If $\boldsymbol{v} = \boldsymbol{0}$ both sides are 0. Otherwise, for every real t the squared norm $\|\boldsymbol{u} - t\boldsymbol{v}\|_2^2 \ge 0$ expands to the quadratic

$$q(t) = \|\boldsymbol{u}\|_2^2 - 2t\,\langle \boldsymbol{u}, \boldsymbol{v} \rangle + t^2 \|\boldsymbol{v}\|_2^2 \;\ge\; 0.$$

A quadratic $at^2 + bt + c$ with $a > 0$ stays nonnegative for all t exactly when its discriminant is nonpositive, $b^2 - 4ac \le 0$. Here $a = \|\boldsymbol{v}\|_2^2$, $b = -2\,\langle \boldsymbol{u}, \boldsymbol{v} \rangle$, $c = \|\boldsymbol{u}\|_2^2$, so

$$4\,\langle \boldsymbol{u}, \boldsymbol{v} \rangle^2 - 4\|\boldsymbol{u}\|_2^2 \|\boldsymbol{v}\|_2^2 \le 0 \;\Longrightarrow\; |\langle \boldsymbol{u}, \boldsymbol{v} \rangle| \le \|\boldsymbol{u}\|_2 \|\boldsymbol{v}\|_2.$$

Equality holds when the discriminant vanishes, that is when $q(t) = 0$ has a real root $t_\star$, meaning $\boldsymbol{u} = t_\star \boldsymbol{v}$. ■

Remark 2.11 *(The geometric one-liner).* Once the angle formula $\cos\theta = \langle \boldsymbol{u}, \boldsymbol{v} \rangle / (\|\boldsymbol{u}\|_2 \|\boldsymbol{v}\|_2)$ is in hand, Cauchy–Schwarz is just $|\cos\theta| \le 1$. The proof above is the honest version that does not assume the angle formula, which is itself a consequence of the inequality.

Theorem 2.12 (Triangle and reverse triangle inequalities). For all $\boldsymbol{u}, \boldsymbol{v}$,

$$\|\boldsymbol{u} + \boldsymbol{v}\|_2 \le \|\boldsymbol{u}\|_2 + \|\boldsymbol{v}\|_2 \qquad \text{and} \qquad \big|\, \|\boldsymbol{u}\|_2 - \|\boldsymbol{v}\|_2 \,\big| \le \|\boldsymbol{u} - \boldsymbol{v}\|_2.$$

Proof. Expand and apply Cauchy–Schwarz:

$$\|\boldsymbol{u} + \boldsymbol{v}\|_2^2 = \|\boldsymbol{u}\|_2^2 + 2\,\langle \boldsymbol{u}, \boldsymbol{v} \rangle + \|\boldsymbol{v}\|_2^2 \le \|\boldsymbol{u}\|_2^2 + 2\|\boldsymbol{u}\|_2\|\boldsymbol{v}\|_2 + \|\boldsymbol{v}\|_2^2 = \big(\|\boldsymbol{u}\|_2 + \|\boldsymbol{v}\|_2\big)^2.$$

Taking square roots gives the triangle inequality. For the reverse form, write $\|\boldsymbol{u}\|_2 = \|(\boldsymbol{u}-\boldsymbol{v})+\boldsymbol{v}\|_2 \le \|\boldsymbol{u}-\boldsymbol{v}\|_2 + \|\boldsymbol{v}\|_2$, so $\|\boldsymbol{u}\|_2 - \|\boldsymbol{v}\|_2 \le \|\boldsymbol{u}-\boldsymbol{v}\|_2$; swapping $\boldsymbol{u}$ and $\boldsymbol{v}$ gives the other side, and together they give the absolute value. ■

2.5 Matrix Norms

We will need five matrix norms. Three are defined through the singular values $\sigma_1 \ge \sigma_2 \ge \cdots \ge 0$ of Section 2.7; two are defined directly from the entries.

The matrix norms used in this book

For $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with singular values σ_i:

$$\begin{aligned}
\text{Frobenius:} \quad & \|\boldsymbol{A}\|_F = \sqrt{\textstyle\sum_{i,j} A_{ij}^2} = \sqrt{\textstyle\sum_i \sigma_i^2} = \sqrt{\operatorname{tr}(\boldsymbol{A}^\top \boldsymbol{A})}, \\
\text{Spectral } (2\to 2): \quad & \|\boldsymbol{A}\|_{2\to 2} = \sigma_1 = \sqrt{\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A})}, \\
\text{Nuclear:} \quad & \|\boldsymbol{A}\|_* = \textstyle\sum_i \sigma_i, \\
\text{Induced } 1\to 1: \quad & \|\boldsymbol{A}\|_{1\to 1} = \max_j \textstyle\sum_i |A_{ij}| \ \ \text{(max absolute \textit{column} sum)}, \\
\text{Induced } \infty\to\infty: \quad & \|\boldsymbol{A}\|_{\infty\to\infty} = \max_i \textstyle\sum_j |A_{ij}| \ \ \text{(max absolute \textit{row} sum)}.
\end{aligned}$$

The Frobenius norm treats the matrix as one long vector of its mn entries; it is the matrix ℓ_2. The spectral norm is the largest factor by which $\boldsymbol{A}$ can stretch a unit vector, the worst-case gain, and it is the Lipschitz constant of $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$. The nuclear norm sums all the stretching factors; it is the matrix ℓ_1 and the convex surrogate for rank in Chapter 11. An **induced** (operator) norm measures the worst-case output size per unit input size, $\|\boldsymbol{A}\|_{p\to q} = \max_{\boldsymbol{x}\neq \boldsymbol{0}} \|\boldsymbol{A}\boldsymbol{x}\|_q / \|\boldsymbol{x}\|_p$; the $1\to 1$ and $\infty\to\infty$ cases have the clean closed forms above.

> The induced $1\to 1$ norm sums down *columns*; the induced $\infty\to\infty$ norm sums across *rows*. The reason is that $\|\boldsymbol{A}\boldsymbol{x}\|_1$ over $\|\boldsymbol{x}\|_1 \le 1$ is maximized by putting all the weight on the column of largest ℓ_1 length, while $\|\boldsymbol{A}\boldsymbol{x}\|_\infty$ is governed by the row that best matches the sign pattern of $\boldsymbol{x}$. Swapping the two is a frequent slip.

Example 2.13 (Four norms of a 2×2 matrix). Let $\boldsymbol{A} = \begin{bmatrix} 1 & -2 \\ 3 & -4 \end{bmatrix}$. Directly, $\|\boldsymbol{A}\|_F = \sqrt{1+4+9+16} = \sqrt{30} \approx 5.477$, $\|\boldsymbol{A}\|_{1\to 1} = \max(1+3,\, 2+4) = 6$, and $\|\boldsymbol{A}\|_{\infty\to\infty} = \max(1+2,\, 3+4) = 7$. For the spectral and nuclear norms, form $\boldsymbol{A}^\top \boldsymbol{A} = \begin{bmatrix} 10 & -14 \\ -14 & 20 \end{bmatrix}$, whose eigenvalues solve $\lambda^2 - 30\lambda + 4 = 0$, giving $\lambda \approx 29.87$ and 0.134. Hence $\sigma_1 \approx 5.465$ and $\sigma_2 \approx 0.366$, so $\|\boldsymbol{A}\|_{2\to 2} \approx 5.465$ and $\|\boldsymbol{A}\|_* \approx 5.83$. Two sanity checks pass: $\sigma_1^2 + \sigma_2^2 \approx 30 = \|\boldsymbol{A}\|_F^2$, and $\sigma_1\sigma_2 \approx 2.0 = |\det \boldsymbol{A}| = |-4+6| = 2$. ◊

2.6 Eigenvalues, the Spectral Theorem, and PSD Matrices

Definition 2.14 (Eigenvalues and eigenvectors). A nonzero $\boldsymbol{v}$ is an **eigenvector** of a square matrix $\boldsymbol{A}$ with **eigenvalue** λ if $\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v}$. Equivalently $(\boldsymbol{A} - \lambda \boldsymbol{I})\boldsymbol{v} = \boldsymbol{0}$ has a nonzero solution, which forces $\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$ (the characteristic equation). The trace and determinant recover the eigenvalues' sum and product: $\operatorname{tr}(\boldsymbol{A}) = \sum_i \lambda_i$, $\det(\boldsymbol{A}) = \prod_i \lambda_i$.

An eigenvector is a direction the matrix does not rotate, only stretches (or flips); the eigenvalue is the stretch factor. The case that matters most for us is symmetric, because the Gram matrix $\boldsymbol{A}^\top \boldsymbol{A}$ is always symmetric.

Theorem 2.15 (Spectral theorem, symmetric case). A real symmetric matrix $\boldsymbol{S} = \boldsymbol{S}^\top \in \mathbb{R}^{n\times n}$ has n real eigenvalues and an orthonormal basis of eigenvectors. Equivalently $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$ with $\boldsymbol{Q}$ orthogonal and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$.

Example 2.16 (A symmetric 2×2 decomposition). Let $\boldsymbol{S} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$. The characteristic equation $(2-\lambda)^2 - 1 = 0$ gives $\lambda = 3$ and $\lambda = 1$. For $\lambda = 3$, $v_2 = v_1$, normalized $\boldsymbol{q}_1 = \frac{1}{\sqrt{2}}(1,1)^\top$; for $\lambda = 1$, $v_2 = -v_1$, normalized $\boldsymbol{q}_2 = \frac{1}{\sqrt{2}}(1,-1)^\top$. They are orthonormal, confirming the spectral theorem, and $\boldsymbol{S} = \boldsymbol{Q}\,\mathrm{diag}(3,1)\boldsymbol{Q}^\top$ with $\boldsymbol{Q} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. $\lozenge$

Example 2.17 (A symmetric 3×3 decomposition). Let $\boldsymbol{S} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$. Expanding along the first row,

$$\det(\boldsymbol{S} - \lambda \boldsymbol{I}) = (2-\lambda)\big[(2-\lambda)^2 - 1\big] - 1\big[(2-\lambda)\big] = (2-\lambda)\big[(2-\lambda)^2 - 2\big].$$

So $\lambda = 2$ and $(2-\lambda)^2 = 2$, giving eigenvalues $2+\sqrt{2} \approx 3.414$, 2, and $2-\sqrt{2} \approx 0.586$. Checks: the sum is $6 = \mathrm{tr}(\boldsymbol{S})$, and the product is $(2+\sqrt{2})(2)(2-\sqrt{2}) = 2(4-2) = 4 = \det(\boldsymbol{S})$. The eigenvector for $\lambda = 2$ is $(1,0,-1)^\top$, and the other two are $(1, \pm\sqrt{2}, 1)^\top$, all mutually orthogonal. $\lozenge$

Definition 2.18 (Positive semidefinite). A symmetric $\boldsymbol{M}$ is **positive semidefinite** ($\boldsymbol{M} \succeq 0$) if $\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$, equivalently all eigenvalues are ≥ 0; it is **positive definite** ($\succ 0$) if strict for $\boldsymbol{x} \neq \boldsymbol{0}$. The Gram matrix $\boldsymbol{A}^\top \boldsymbol{A} \succeq 0$ always, which is why the singular values $\sigma_i = \sqrt{\lambda_i(\boldsymbol{A}^\top \boldsymbol{A})}$ are real and nonnegative.

A quick test: a symmetric $\begin{bmatrix} a & b \\ b & d \end{bmatrix}$ is positive semidefinite if and only if $a \geq 0$, $d \geq 0$, and $ad - b^2 \geq 0$. Because $\boldsymbol{A}^\top \boldsymbol{A}$ must pass this test, a negative determinant in a Gram matrix you just computed means an arithmetic slip.

Example 2.19 (Definite, and indefinite). $\boldsymbol{M} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ has $a = 2 > 0$ and $\det = 3 > 0$, so it is positive definite (eigenvalues $3, 1$). But $\boldsymbol{N} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ has $\det = 1 - 4 = -3 < 0$, so it is indefinite; concretely $\boldsymbol{x} = (1,-1)^\top$ gives $\boldsymbol{x}^\top \boldsymbol{N}\boldsymbol{x} = -2 < 0$. $\lozenge$

The **trace** $\mathrm{tr}(\boldsymbol{A}) = \sum_i A_{ii}$ is linear and obeys the *cyclic* property $\mathrm{tr}(\boldsymbol{ABC}) = \mathrm{tr}(\boldsymbol{BCA}) = \mathrm{tr}(\boldsymbol{CAB})$ (rotate the factors, do not reverse them). Two consequences used constantly are $\mathrm{tr}(\boldsymbol{A}^\top \boldsymbol{A}) = \|\boldsymbol{A}\|_F^2$ and the Frobenius inner product $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_F = \mathrm{tr}(\boldsymbol{A}^\top \boldsymbol{B}) = \sum_{i,j} A_{ij} B_{ij}$.

2.7 The Singular Value Decomposition

The singular value decomposition is the master factorization of the course. It defines the spectral and nuclear norms, drives PCA (Chapter 12), matrix completion (Chapter 11), and K-SVD (Chapter 13), and gives the cleanest description of the pseudoinverse.

Theorem 2.20 (Singular value decomposition). Every $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ factors as

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$

with $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ orthogonal and $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ diagonal with entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, the singular values. Equivalently

$$\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i \, \boldsymbol{u}_i \boldsymbol{v}_i^\top, \qquad r = \mathrm{rank}(\boldsymbol{A}),$$

a sum of r rank-one matrices. The right singular vectors $\boldsymbol{v}_i$ are the eigenvectors of $\boldsymbol{A}^\top \boldsymbol{A}$, the values are $\sigma_i = \sqrt{\lambda_i(\boldsymbol{A}^\top \boldsymbol{A})}$, and the left singular vectors are $\boldsymbol{u}_i = \sigma_i^{-1} \boldsymbol{A}\boldsymbol{v}_i$.

A proof of existence, via the spectral theorem applied to $\boldsymbol{A}^\top \boldsymbol{A}$, appears in Appendix D; here we develop fluency by computing.

Recipe

Computing an SVD by hand.

1. Form the smaller Gram matrix, $\boldsymbol{A}^\top\boldsymbol{A}$ ($n\times n$) or $\boldsymbol{A}\boldsymbol{A}^\top$ ($m\times m$).
2. Find its eigenvalues λ_i and orthonormal eigenvectors. The $\sigma_i=\sqrt{\lambda_i}$, sorted descending, are the singular values.
3. If you used $\boldsymbol{A}^\top\boldsymbol{A}$, its eigenvectors are the right singular vectors $\boldsymbol{v}_i$; get the left ones from $\boldsymbol{u}_i=\sigma_i^{-1}\boldsymbol{A}\boldsymbol{v}_i$.
4. Assemble $\boldsymbol{U},\boldsymbol{\Sigma},\boldsymbol{V}$, and check $\sum_i\sigma_i^2=\|\boldsymbol{A}\|_F^2$ and, for square $\boldsymbol{A}$, $\prod_i\sigma_i=|\det\boldsymbol{A}|$.

Example 2.21 (A full 2×2 SVD). Let $\boldsymbol{A}=\begin{bmatrix}3&0\\4&5\end{bmatrix}$. Form $\boldsymbol{A}^\top\boldsymbol{A}=\begin{bmatrix}25&20\\20&25\end{bmatrix}$; its eigenvalues come from $(25-\lambda)^2=400$, so $\lambda=45$ and $\lambda=5$, giving $\sigma_1=\sqrt{45}=3\sqrt{5}\approx6.708$ and $\sigma_2=\sqrt{5}\approx2.236$ (and indeed $45+5=50=\|\boldsymbol{A}\|_F^2$). The right singular vectors are $\boldsymbol{v}_1=\frac{1}{\sqrt{2}}(1,1)^\top$ and $\boldsymbol{v}_2=\frac{1}{\sqrt{2}}(1,-1)^\top$. The left ones are

$$\boldsymbol{u}_1=\frac{1}{3\sqrt{5}}\boldsymbol{A}\,\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}=\frac{1}{\sqrt{10}}\begin{bmatrix}1\\3\end{bmatrix},\qquad \boldsymbol{u}_2=\frac{1}{\sqrt{5}}\boldsymbol{A}\,\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}=\frac{1}{\sqrt{10}}\begin{bmatrix}3\\-1\end{bmatrix}.$$

Therefore

$$\boldsymbol{A}=\underbrace{\frac{1}{\sqrt{10}}\begin{bmatrix}1&3\\3&-1\end{bmatrix}}_{\boldsymbol{U}}\underbrace{\begin{bmatrix}3\sqrt{5}&0\\0&\sqrt{5}\end{bmatrix}}_{\boldsymbol{\Sigma}}\underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}}_{\boldsymbol{V}^\top},$$

with $\|\boldsymbol{A}\|_{2\to2}=3\sqrt{5}$ and $\|\boldsymbol{A}\|_*=4\sqrt{5}\approx8.944$. A last check: $\sigma_1\sigma_2=15=|\det\boldsymbol{A}|=15$. ◊

Theorem 2.22 (Eckart–Young). The best rank-k approximation of $\boldsymbol{A}$ in both the Frobenius and the spectral norm is the truncated SVD $\boldsymbol{A}_k=\sum_{i=1}^{k}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^\top$. The errors are exactly

$$\|\boldsymbol{A}-\boldsymbol{A}_k\|_{2\to2}=\sigma_{k+1},\qquad \|\boldsymbol{A}-\boldsymbol{A}_k\|_F=\sqrt{\textstyle\sum_{i>k}\sigma_i^2}.$$

This is why "low-rank approximation" and "truncate the SVD" mean the same thing, and it is the engine of PCA, image compression, and the K-SVD atom update. We prove the rank-one case in Appendix D.

2.8 Rank, the Four Subspaces, and the Pseudoinverse

The **rank** of $\boldsymbol{A}\in\mathbb{R}^{m\times n}$ is the number of linearly independent columns, equal to the number of independent rows, equal to the number of nonzero singular values. It organizes four subspaces.

Definition 2.23 (The four fundamental subspaces). For $\boldsymbol{A}\in\mathbb{R}^{m\times n}$ of rank r:

- the **column space** (range) $\mathcal{R}(\boldsymbol{A})\subseteq\mathbb{R}^m$, dimension r;
- the **null space** $\mathcal{N}(\boldsymbol{A})=\{\boldsymbol{x}:\boldsymbol{A}\boldsymbol{x}=\boldsymbol{0}\}\subseteq\mathbb{R}^n$, dimension $n-r$;
- the **row space** $\mathcal{R}(\boldsymbol{A}^\top)\subseteq\mathbb{R}^n$, dimension r;
- the **left null space** $\mathcal{N}(\boldsymbol{A}^\top)\subseteq\mathbb{R}^m$, dimension $m-r$.

The rank–nullity theorem is $\operatorname{rank}(\boldsymbol{A})+\dim\mathcal{N}(\boldsymbol{A})=n$.

Why the null space is the whole story

An underdetermined system $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y}$ has a whole affine family of solutions: if $\boldsymbol{x}_0$ is one, then $\boldsymbol{x}_0+\boldsymbol{v}$ is another for every $\boldsymbol{v}\in\mathcal{N}(\boldsymbol{A})$. So two candidate signals are confusable exactly when they differ by a null-space vector. The question "can a sparse signal be mistaken for another sparse signal?" becomes "does $\mathcal{N}(\boldsymbol{A})$ contain a vector that is itself (nearly) sparse?" Spark, coherence, the null space property, and the restricted isometry property are four different ways of guaranteeing that the null space *avoids* sparse vectors. This single observation is the spine of Chapters 4 and 8.

Definition 2.24 (Moore–Penrose pseudoinverse). For a **tall, full-column-rank** $\boldsymbol{A}$ ($m > n$, rank n), the least-squares solution of $\boldsymbol{A}\boldsymbol{x} \approx \boldsymbol{y}$ is $\boldsymbol{x} = \boldsymbol{A}^{\dagger}\boldsymbol{y}$ with the *left* inverse

$$\boldsymbol{A}^{\dagger} = (\boldsymbol{A}^{\top}\boldsymbol{A})^{-1}\boldsymbol{A}^{\top}, \qquad \boldsymbol{A}^{\dagger}\boldsymbol{A} = \boldsymbol{I}.$$

For a **wide, full-row-rank** $\boldsymbol{A}$ ($m < n$, rank m), the minimum-norm solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ is $\boldsymbol{x} = \boldsymbol{A}^{\dagger}\boldsymbol{y}$ with the *right* inverse

$$\boldsymbol{A}^{\dagger} = \boldsymbol{A}^{\top}(\boldsymbol{A}\boldsymbol{A}^{\top})^{-1}, \qquad \boldsymbol{A}\boldsymbol{A}^{\dagger} = \boldsymbol{I}.$$

In general $\boldsymbol{A}^{\dagger} = \boldsymbol{V}\boldsymbol{\Sigma}^{\dagger}\boldsymbol{U}^{\top}$, inverting the nonzero singular values.

To choose the right formula, count rows versus columns. A tall matrix over-determines the system, so the pseudoinverse gives the least-squares fit and $\boldsymbol{A}^{\top}\boldsymbol{A}$ is the small invertible Gram matrix. A wide matrix under-determines it, so the pseudoinverse picks the minimum-ℓ_2-norm exact solution and $\boldsymbol{A}\boldsymbol{A}^{\top}$ is the small invertible one. Using the wrong side is a classic error. The orthogonal matching pursuit of Chapter 6 uses the tall version on its selected columns at every step.

Example 2.25 (Pseudoinverse of a tall matrix). Let $\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$ (3×2, rank 2). Then $\boldsymbol{A}^{\top}\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$, with $\det = 3$ and $(\boldsymbol{A}^{\top}\boldsymbol{A})^{-1} = \frac{1}{3}\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$, so

$$\boldsymbol{A}^{\dagger} = (\boldsymbol{A}^{\top}\boldsymbol{A})^{-1}\boldsymbol{A}^{\top} = \frac{1}{3}\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & 1 \end{bmatrix}.$$

Check: $\boldsymbol{A}^{\dagger}\boldsymbol{A} = \frac{1}{3}\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} = \boldsymbol{I}$. ◊

Example 2.26 (Pseudoinverse of a wide matrix). Let $\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$ (2×3, full row rank). Use $\boldsymbol{A}^{\dagger} = \boldsymbol{A}^{\top}(\boldsymbol{A}\boldsymbol{A}^{\top})^{-1}$ with $\boldsymbol{A}\boldsymbol{A}^{\top} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$:

$$\boldsymbol{A}^{\dagger} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \frac{1}{3}\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 2 & -1 \\ 1 & 1 \\ -1 & 2 \end{bmatrix}.$$

Check $\boldsymbol{A}\boldsymbol{A}^{\dagger} = \boldsymbol{I}_2$. The minimum-norm solution of $\boldsymbol{A}\boldsymbol{x} = (1,1)^{\top}$ is $\boldsymbol{A}^{\dagger}(1,1)^{\top} = \frac{1}{3}(1,2,1)^{\top}$, and indeed $\boldsymbol{A}\cdot\frac{1}{3}(1,2,1)^{\top} = (1,1)^{\top}$. ◊

2.9 Convexity

Convexity is the property that makes the ℓ_1 relaxation tractable, so we will use it without comment in later chapters.

Definition 2.27 (Convex set and convex function). A set C is **convex** if for all $\boldsymbol{x}, \boldsymbol{y} \in C$ and $\theta \in [0,1]$, the point $\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}$ lies in C. A function f with convex domain is **convex** if

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \le \theta f(\boldsymbol{x}) + (1-\theta)f(\boldsymbol{y}) \qquad \text{for all } \theta \in [0,1],$$

that is, every chord lies on or above the graph (Fig. 2.2). If f is twice differentiable, convexity is equivalent to $\nabla^2 f \succeq 0$.

Proposition 2.28 (Every norm is convex). Any norm $\|\cdot\|$ is a convex function.

Proof. For $\theta \in [0,1]$, homogeneity and the triangle inequality give

$$\|\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}\| \le \|\theta\boldsymbol{x}\| + \|(1-\theta)\boldsymbol{y}\| = \theta\|\boldsymbol{x}\| + (1-\theta)\|\boldsymbol{y}\|,$$

which is exactly the convexity inequality. ■

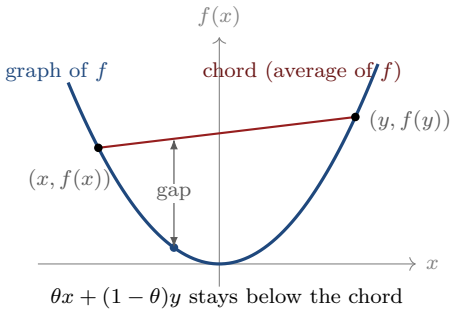


Figure 2.2. A convex function: the chord between any two points on the graph lies on or above the graph. At any intermediate point the graph sits below the chord (the "gap"), which is exactly the convexity inequality $f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y)$.

Remark 2.29 *(Why convexity is the whole point).* If f is convex, every local minimum is a global minimum, so a solver that finds *any* local optimum has found the answer; if f is strictly convex the minimizer is unique. This is precisely why the relaxation (P_1) is useful: it minimizes a convex function over a convex (affine) set. The original problem (P_0) is not convex (its feasible structure is a union of coordinate subspaces), which is the source of its NP-hardness in Chapter 3. Sums of convex functions and compositions with affine maps stay convex, so the LASSO objective $\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$ is convex.

The ℓ_1 norm is convex but not differentiable at 0, so we need a generalized derivative.

Definition 2.30 (Subgradient). A **subgradient** of a convex f at $\boldsymbol{x}$ is any $\boldsymbol{g}$ with $f(\boldsymbol{z}) \ge f(\boldsymbol{x}) + \langle \boldsymbol{g}, \boldsymbol{z} - \boldsymbol{x}\rangle$ for all $\boldsymbol{z}$; the set of all subgradients is the **subdifferential** $\partial f(\boldsymbol{x})$. For the scalar absolute value,

$$\partial |x| = \begin{cases} \{+1\} & x > 0, \\ [-1, +1] & x = 0, \\ \{-1\} & x < 0. \end{cases}$$

Example 2.31 (A subgradient of $\|\cdot\|_1$). At $\boldsymbol{x} = (3, 0, -2)^\top$, coordinate by coordinate: $x_1 = 3 > 0$ forces $g_1 = 1$; $x_2 = 0$ allows any $g_2 \in [-1, 1]$, say 0; $x_3 = -2 < 0$ forces $g_3 = -1$. So $\boldsymbol{g} = (1, 0, -1)^\top \in \partial\|\boldsymbol{x}\|_1$. The nonzero coordinates pin the subgradient to $\operatorname{sgn}(x_i)$; only the zero coordinates have freedom, and that freedom at the zeros is exactly what lets the LASSO hold many coordinates at *precisely* zero. We use this in Chapter 7 to derive soft thresholding. ◊

2.10 Gradients and Matrix Calculus

Two gradient formulas appear in nearly every algorithm derivation in the second half of the book.

The two gradients to know cold

$$\nabla_{\boldsymbol{x}} \ \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2 = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}),$$
$$\nabla_{\boldsymbol{x}} \ \boldsymbol{x}^\top\boldsymbol{Q}\boldsymbol{x} = (\boldsymbol{Q}+\boldsymbol{Q}^\top)\boldsymbol{x} = 2\boldsymbol{Q}\boldsymbol{x} \quad (\boldsymbol{Q} \text{ symmetric}).$$

Example 2.32 (Deriving the least-squares gradient). Write $\|\boldsymbol{z}\|_2^2 = \boldsymbol{z}^\top\boldsymbol{z}$ and expand:

$$g(\boldsymbol{x}) = \tfrac{1}{2}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y})^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}) = \tfrac{1}{2}\big(\boldsymbol{x}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{y}^\top\boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top\boldsymbol{y}\big).$$

The first term, with $\boldsymbol{Q} = \boldsymbol{A}^\top\boldsymbol{A}$ symmetric, has gradient $\tfrac{1}{2}\cdot 2\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x}$; the second term is linear with gradient $-\boldsymbol{A}^\top\boldsymbol{y}$; the constant drops. Hence

$$\nabla g(\boldsymbol{x}) = \boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{y} = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}).$$

Setting this to zero gives the **normal equations** $\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^\top\boldsymbol{y}$, whose solution is the tall pseudoinverse $\boldsymbol{x} = \boldsymbol{A}^\dagger\boldsymbol{y}$ of Definition 2.24. This one derivation is the engine behind least squares, the OMP update, ridge regression, and alternating least squares. ◊

Remark 2.33 *(The universal gradient-step template).* Every proximal or projected-gradient algorithm in Chapter 7 has the form

$$\boldsymbol{x}^{k+1} = (\text{threshold or project})\Big(\boldsymbol{x}^k - \tfrac{1}{L}\,\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y})\Big).$$

The gradient $\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y})$ is always present; only the nonlinear map changes (soft thresholding for ISTA, hard thresholding for IHT, the identity for plain gradient descent). Learn the template once and the three algorithms come together.

Notes and References

This material is standard and can be found in any graduate linear algebra or numerical methods text. The treatment here follows the course's own conventions so that the notation matches the rest of the book. The convex-analysis facts (every local minimum global, subgradients) are developed at length in Boyd and Vandenberghe [7]; the matrix-analysis facts, including the SVD and the Eckart–Young low-rank approximation theorem [24], are in any standard reference and are revisited in Appendix D. Vershynin [51] is the modern reference for the probabilistic tools that join these in Chapter 9.

Exercises

Exercise 2.1. For $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ and the same $\boldsymbol{B}$ as in Example 2.1, compute $\boldsymbol{BA}$ and verify it is 3×3, hence not equal to $\boldsymbol{AB}$. Then confirm the cyclic trace identity $\operatorname{tr}(\boldsymbol{AB}) = \operatorname{tr}(\boldsymbol{BA})$ numerically.

Exercise 2.2. Compute all four norms $(\ell_0, \ell_1, \ell_2, \ell_\infty)$ of $\boldsymbol{x} = (0, 0, -5, 12, 0)$ and verify the ordering $\|\boldsymbol{x}\|_\infty \le \|\boldsymbol{x}\|_2 \le \|\boldsymbol{x}\|_1$. Why does the ℓ_2 norm come out to a whole number here?

Exercise 2.3. Prove the Cauchy–Schwarz inequality in the complex case: for $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{C}^n$ with $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_i \overline{u_i} v_i$, show $|\langle \boldsymbol{u}, \boldsymbol{v} \rangle| \le \|\boldsymbol{u}\|_2 \|\boldsymbol{v}\|_2$. *Hint: apply the real argument to $\|\boldsymbol{u} - te^{i\phi}\boldsymbol{v}\|_2^2$ for a suitable phase ϕ.*

Exercise 2.4. Compute the full SVD of $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$ by hand (work with the 2×2 Gram matrix), and read off $\|\boldsymbol{A}\|_{2\to 2}$, $\|\boldsymbol{A}\|_*$, and $\|\boldsymbol{A}\|_F$. Verify the ordering $\|\boldsymbol{A}\|_{2\to 2} \le \|\boldsymbol{A}\|_F \le \|\boldsymbol{A}\|_*$.

Exercise 2.5. Let $\boldsymbol{A}$ be wide and full row rank. Show that $\boldsymbol{A}^\dagger = \boldsymbol{A}^\top (\boldsymbol{A}\boldsymbol{A}^\top)^{-1}$ indeed satisfies $\boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{I}$, and that for any solution $\boldsymbol{x}$ of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, the particular solution $\boldsymbol{A}^\dagger \boldsymbol{y}$ has the smallest ℓ_2 norm. *Hint: write $\boldsymbol{x} = \boldsymbol{A}^\dagger \boldsymbol{y} + \boldsymbol{z}$ with $\boldsymbol{z} \in \mathcal{N}(\boldsymbol{A})$ and expand $\|\boldsymbol{x}\|_2^2$.*

Exercise 2.6. Show that the LASSO objective $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{x}\|_1$ is convex for every $\lambda \ge 0$, using only that norms are convex and that sums and affine compositions preserve convexity. Is it strictly convex? Discuss how the answer depends on $\mathcal{N}(\boldsymbol{A})$.

PART II

The Sparse Recovery Problem

CHAPTER 3

Sparsity and the ℓ_0 Problem

Entities should not be multiplied beyond necessity.

William of Ockham

Chapter 1 promised that sparsity is what rescues the underdetermined problem $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. This chapter makes the promise precise and then runs into the obstacle that organizes the next several chapters. We define sparsity and support carefully, learn the two ways to read the measurement model, and write down the problem we would *like* to solve: find the sparsest vector consistent with the measurements. We then show two things about that problem. If an oracle whispered the support to us, recovery would be a one-line least-squares computation. But finding the support ourselves, by the obvious method of trying every possibility, would take centuries on a small problem, and we prove that the sparsest-solution problem is NP-hard in the worst case. That hardness is the wall the rest of the book is built to get around.

3.1 Sparsity, Support, and the Set of Sparse Vectors

We work over $\mathbb{C}^N$, since the measurement matrices in applications like MRI are complex. Everything specializes to $\mathbb{R}^N$ without change.

Definition 3.1 (Support and the ℓ_0 count). The **support** of $\boldsymbol{x} \in \mathbb{C}^N$ is the set of indices where it is nonzero,

$$\operatorname{supp}(\boldsymbol{x}) := \{j \in [N] : x_j \neq 0\}, \qquad [N] := \{1, \dots, N\},$$

and the ℓ_0 **count** is its cardinality,

$$\|\boldsymbol{x}\|_0 := |\operatorname{supp}(\boldsymbol{x})| = |\{j : x_j \neq 0\}|.$$

A vector is s**-sparse** if $\|\boldsymbol{x}\|_0 \leq s$. As noted in Chapter 2, $\|\cdot\|_0$ is not a norm (it violates homogeneity), which is why the subscript-zero notation carries an implicit set of quotation marks.

The support answers *which* entries are active; the ℓ_0 count answers *how many*. The magnitudes of the nonzero entries are irrelevant to both.

Example 3.2 (Support and sparsity). Let $\boldsymbol{x} = (0, 0, 5, 0, -2i, 0, 0, 7)^\top \in \mathbb{C}^8$. Then $\operatorname{supp}(\boldsymbol{x}) = \{3, 5, 8\}$ and $\|\boldsymbol{x}\|_0 = 3$, so $\boldsymbol{x}$ is 3-sparse. The values $5, -2i, 7$ do not matter for sparsity; only their positions do. ◊

The set of all s-sparse vectors has an important geometric shape, and it is *not* a subspace.

Definition 3.3 (The model set Σ_s). Write $\Sigma_s := \left\{\boldsymbol{x} \in \mathbb{C}^N : \|\boldsymbol{x}\|_0 \leq s\right\}$ for the set of all s-sparse vectors. It is a **union of subspaces**, one for each possible support:

$$\Sigma_s = \bigcup_{\substack{S \subseteq [N] \\ |S| \leq s}} \left\{\boldsymbol{x} \in \mathbb{C}^N : \operatorname{supp}(\boldsymbol{x}) \subseteq S\right\}.$$

There are $\binom{N}{s}$ such coordinate subspaces, each of dimension s.

> Σ_s is not closed under addition: if $\boldsymbol{x}$ and $\boldsymbol{x}'$ are each s-sparse but on different supports, their sum can have up to $2s$ nonzeros. This single fact, that two s-sparse vectors can differ in as many as $2s$ positions, is the reason the uniqueness threshold in Chapter 4 is $2s$ rather than s. Keep it in mind.

The number of subspaces, $\binom{N}{s}$, is the combinatorial monster at the heart of the chapter. For $N = 100$ and $s = 5$ it is already $\binom{100}{5} = 75{,}287{,}520$. The whole difficulty of sparse recovery is that the active subspace is one of these tens of millions, and we do not know which.

3.2 Two Ways to Read $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$

The product $\boldsymbol{A}\boldsymbol{x}$ can be read by rows or by columns, and both readings recur throughout the book.

3.2.1 The analysis view: measurements are inner products

Stack $\boldsymbol{A}$ by its rows, writing the i-th row as a (conjugated) vector $\boldsymbol{\gamma}_i^*$. Then each measurement is an inner product of the signal against a fixed test vector:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \begin{pmatrix} \boldsymbol{\gamma}_1^* \boldsymbol{x} \\ \vdots \\ \boldsymbol{\gamma}_m^* \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \langle \boldsymbol{\gamma}_1, \boldsymbol{x} \rangle \\ \vdots \\ \langle \boldsymbol{\gamma}_m, \boldsymbol{x} \rangle \end{pmatrix}.$$

Reading $y_i = \langle \boldsymbol{\gamma}_i, \boldsymbol{x} \rangle$, each row of $\boldsymbol{A}$ is a *measurement template*: the i-th number records how strongly the signal correlates with the pattern $\boldsymbol{\gamma}_i$. This is the **analysis** view, and it is the right way to think about *designing* a sensor. In MRI the rows are complex sinusoids, so each y_i is the image's Fourier coefficient at one frequency.

3.2.2 The synthesis view: measurements are a combination of atoms

Stack $\boldsymbol{A}$ by its columns $\boldsymbol{a}_1, \dots, \boldsymbol{a}_N$. Then

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \sum_{j=1}^{N} x_j \, \boldsymbol{a}_j,$$

so $\boldsymbol{y}$ is *built* as a weighted combination of the columns, with the signal entries as weights. This is the **synthesis** view, and it is the right way to think about *recovery*. Each column $\boldsymbol{a}_j$ is an **atom** (or dictionary element), and recovering $\boldsymbol{x}$ means discovering which atoms were used and in what amounts. When $\boldsymbol{x}$ is s-sparse, only s atoms participate; this is the structure every recovery algorithm exploits.

Don't multiply, combine

The most useful habit in this subject is to read $\boldsymbol{A}\boldsymbol{z}$ not as "matrix times vector" but as a *linear combination of the columns of* $\boldsymbol{A}$. A sparse $\boldsymbol{z}$ then selects a few atoms and weights them, and the support of $\boldsymbol{z}$ is precisely "which atoms are in play."

Example 3.4 (The synthesis view on a sparse signal). Let $\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 2 & 1 & 3 \\ 0 & 1 & 1 & 2 & 1 \end{pmatrix}$ and $\boldsymbol{x} = (0, 0, 4, 0, -1)^\top$, which is 2-sparse on $S = \{3, 5\}$. Only two atoms participate:

$$\boldsymbol{y} = 4\boldsymbol{a}_3 - \boldsymbol{a}_5 = 4\begin{pmatrix} 2 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \end{pmatrix}.$$

Recovery is the inverse question: given $\boldsymbol{y} = (5, 3)^\top$ and $\boldsymbol{A}$, find that only atoms 3 and 5 are active, with weights 4 and -1. ◊

Remark 3.5 *(Dictionaries and the $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$ factorization).* In the synthesis view the columns form a *dictionary*. When $N > m$ the dictionary is *overcomplete*, and a signal can have many representations, of which we want the sparsest.

Classical dictionaries include the identity (atoms are spikes, good for time-sparse signals), the discrete Fourier or cosine matrix (atoms are sinusoids, good for tonal signals), and wavelets (atoms localized in time and frequency, used in JPEG 2000). The general picture factors the sensing matrix as $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$, where $\boldsymbol{\Phi}$ does the physical measuring and $\boldsymbol{\Psi}$ is the basis in which the signal is sparse. For recovery to work, $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ must be *incoherent*; we make this precise in Chapter 10, and dictionary learning, the problem of choosing $\boldsymbol{\Psi}$ from data, is the subject of Chapter 13.

3.3 The Sparsest-Solution Problem (P_0)

We can now state the problem we would like to solve. Among all vectors consistent with the measurements, take the one with the fewest nonzeros.

The ℓ_0 recovery problem

$$(P_0): \qquad \min_{\boldsymbol{z}\in\mathbb{C}^N} \|\boldsymbol{z}\|_0 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}.$$

This is the formal version of Occam's razor: among competing explanations of the data, prefer the one that invokes the fewest active atoms. If the true signal $\boldsymbol{x}$ is s-sparse and the matrix is good enough (the precise condition is Chapter 4), then $\boldsymbol{x}$ is the unique solution of (P_0), and solving (P_0) recovers it exactly.

Remark 3.6 *(From analog to information, in one step).* Classical signal processing runs analog → digital → information: sample at the Nyquist rate, store the samples, then extract what you need. Compressed sensing collapses the first two stages. The measurement $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ and the sparsity prior together take the physical signal directly to the recovered information, with (P_0) as the extraction step, never visiting a full Nyquist-rate representation.

3.4 Oracle Recovery: When the Support Is Known

Before confronting the difficulty of (P_0), it pays to see how easy recovery becomes once the hard part is removed. Suppose an oracle hands us the true support $S = \operatorname{supp}(\boldsymbol{x})$. We will see that recovery is then a single least-squares solve.

The reason is that sparsity kills most of the columns. If $\operatorname{supp}(\boldsymbol{z}) = S$, then in the synthesis sum every atom outside S is multiplied by zero:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{z} = \sum_{j=1}^{N} z_j \boldsymbol{a}_j = \sum_{j\in S} z_j \boldsymbol{a}_j.$$

Only the columns indexed by S survive. Collecting them defines a small submatrix.

Definition 3.7 (Column submatrix). For $S \subseteq [N]$, the **column submatrix** $\boldsymbol{A}_S \in \mathbb{C}^{m\times|S|}$ keeps exactly the columns of $\boldsymbol{A}$ indexed by S, and $\boldsymbol{z}_S \in \mathbb{C}^{|S|}$ keeps the corresponding entries of $\boldsymbol{z}$. With this notation the measurement equation collapses to

$$\boldsymbol{y} = \boldsymbol{A}_S\, \boldsymbol{z}_S,$$

a system with only $|S| \le s$ unknowns.

In the compressed sensing regime we have $s \le m \ll N$, so $\boldsymbol{A}_S$ is *tall* (m rows, at most s columns). If its columns are linearly independent, $\boldsymbol{A}_S$ has full column rank, $\boldsymbol{A}_S^*\boldsymbol{A}_S$ is invertible, and the least-squares machinery of Chapter 2 recovers the nonzero entries exactly:

$$\boldsymbol{z}_S = \boldsymbol{A}_S^{\dagger}\,\boldsymbol{y}, \qquad \boldsymbol{A}_S^{\dagger} = (\boldsymbol{A}_S^*\boldsymbol{A}_S)^{-1}\boldsymbol{A}_S^*.$$

Oracle recovery

Given $\boldsymbol{y}$, $\boldsymbol{A}$, and the support S with $|S| = s$:

1. Extract the submatrix $\boldsymbol{A}_S \in \mathbb{C}^{m\times s}$.
2. Form the pseudoinverse $\boldsymbol{A}_S^\dagger = (\boldsymbol{A}_S^*\boldsymbol{A}_S)^{-1}\boldsymbol{A}_S^*$.
3. Recover the active entries $\hat{\boldsymbol{x}}_S = \boldsymbol{A}_S^\dagger \boldsymbol{y}$.
4. Place them back: $\hat{x}_j = (\hat{\boldsymbol{x}}_S)_k$ if j is the k-th element of S, and $\hat{x}_j = 0$ otherwise.

Output $\hat{\boldsymbol{x}} = \boldsymbol{x}$, exact recovery.

Example 3.8 (A full oracle-recovery computation). Let $N = 5$, $m = 3$, $s = 2$, with

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 1 & 2 & 1 \\ 0 & 1 & 1 & 0 & 2 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}, \qquad \boldsymbol{x} = (0,0,4,0,-1)^\top, \quad S = \{3,5\}.$$

The measurements are $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = 4\boldsymbol{a}_3 - \boldsymbol{a}_5 = 4(1,1,0)^\top - (1,2,1)^\top = (3,2,-1)^\top$. Extract

$$\boldsymbol{A}_S = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 0 & 1 \end{pmatrix}, \qquad \boldsymbol{A}_S^\top \boldsymbol{A}_S = \begin{pmatrix} 2 & 3 \\ 3 & 6 \end{pmatrix}, \quad (\boldsymbol{A}_S^\top \boldsymbol{A}_S)^{-1} = \frac{1}{3}\begin{pmatrix} 6 & -3 \\ -3 & 2 \end{pmatrix}.$$

Then

$$\boldsymbol{A}_S^\dagger = (\boldsymbol{A}_S^\top \boldsymbol{A}_S)^{-1}\boldsymbol{A}_S^\top = \begin{pmatrix} 1 & 0 & -1 \\ -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \end{pmatrix}, \qquad \hat{\boldsymbol{x}}_S = \boldsymbol{A}_S^\dagger \boldsymbol{y} = \begin{pmatrix} 3-(-1) \\ -1+\frac{2}{3}+\frac{-2}{3} \end{pmatrix} = \begin{pmatrix} 4 \\ -1 \end{pmatrix}.$$

Reassembling, $\hat{\boldsymbol{x}} = (0,0,4,0,-1)^\top = \boldsymbol{x}$, exact recovery. ◊

> The oracle did all the real work. Once S is known, recovery is a polynomial-time least-squares solve with a closed form. The entire difficulty of compressed sensing is that we are *not* given S, and finding it is equivalent to solving (P_0). Every practical algorithm in Chapters 5 and 6 is a strategy for discovering the support without an exhaustive search.

3.5 The Cost of Brute Force

The obvious way to solve (P_0) is to ask the oracle's question ourselves: try every support, run oracle recovery on it, and keep the sparsest consistent answer. The following back-of-the-envelope estimate shows why this is hopeless even on a small problem.

Take $N = 1000$ and $s = 10$. The number of candidate supports of size s is

$$\binom{1000}{10} = \frac{1000 \cdot 999 \cdots 991}{10!} \approx 2.63 \times 10^{23}.$$

For each candidate we extract a 10-column submatrix and solve a 10×10 least-squares problem, which on modern hardware costs on the order of 10^{-10} seconds. The total time is then

$$\underbrace{2.63 \times 10^{23}}_{\text{supports}} \times \underbrace{10^{-10}\,\text{s}}_{\text{per solve}} \approx 2.6 \times 10^{13}\,\text{s}.$$

Using the engineer's approximation $1\,\text{year} \approx 3.16 \times 10^7\,\text{s}$ (close to $\pi \times 10^7$),

$$\frac{2.6 \times 10^{13}\,\text{s}}{3.16 \times 10^7\,\text{s/yr}} \approx 8 \times 10^5 \text{ years}.$$

> Nearly a million years, for a signal of length 1000 with only 10 nonzeros. Real applications run at $N = 10^5$ to 10^6 and $s = 10^2$ to 10^3, where the count $\binom{N}{s}$ is astronomically larger. Brute-force search over supports is not slow; it is impossible. We need either a tractable algorithm or a proof that none exists. The next section delivers the second, which forces us toward the first.

3.6 (P_0) Is NP-Hard

The brute-force estimate shows that the obvious algorithm is too slow. It leaves open whether some clever algorithm could solve (P_0) quickly. We now rule that out (modulo the most famous open problem in computer science) by proving that (P_0) is NP-hard.

3.6.1 The complexity landscape

Recall the rough geography. The class P contains decision problems solvable in polynomial time. The class NP contains problems whose "yes" answers can be *verified* in polynomial time given a certificate, and P $\subseteq$ NP. A problem is **NP-hard** if every problem in NP reduces to it in polynomial time (so it is at least as hard as everything in NP), and **NP-complete** if it is both in NP and NP-hard. Figure 3.1 shows the standard picture. To prove (P_0) is NP-hard, we take a known NP-complete problem and reduce it to (P_0) in polynomial time.

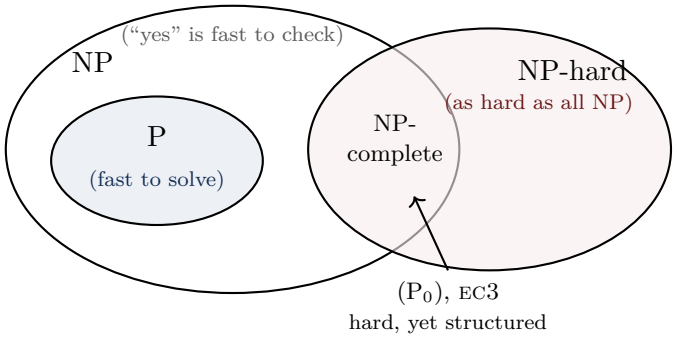


Figure 3.1. The complexity landscape, drawn assuming P $\neq$ NP. (P_0) and the exact-cover problem EC3 sit in the NP-complete lens, the intersection of NP and NP-hard: their "yes" answers are easy to verify but (unless P = NP) hard to find.

3.6.2 Exact Cover by 3-Sets

The problem we reduce from is **Exact Cover by 3-Sets** (EC3, also written X3C), a classic NP-complete problem.

Definition 3.9 (Exact Cover by 3-Sets). **Input:** an integer m divisible by 3, and a collection $J = \{c_1, \dots, c_N\}$ of 3-element subsets of $[m]$. **Question:** is there a subcollection $\{c_j\}_{j \in S}$ that is an **exact cover** of $[m]$, meaning the chosen triples are pairwise disjoint and their union is all of $[m]$?

$$\bigcup_{j \in S} c_j = [m], \qquad c_j \cap c_{j'} = \emptyset \ \text{ for } j \neq j',\ j, j' \in S.$$

An exact cover partitions $[m]$ into disjoint triples, so it uses exactly $m/3$ of them; this is why m must be divisible by 3.

Example 3.10 (Exact cover, and its absence). With $m = 9$ and $J = \big\{\{1,4,9\},\{1,3,5\},\{4,5,6\}\big\}$, the union of all three triples is $\{1,3,4,5,6,9\}$, which already misses $\{2,7,8\}$, so no exact cover exists. By contrast $\big\{\{1,2,3\},\{4,5,6\},\{7,8,9\}\big\}$ is an exact cover of [9]: three disjoint triples whose union is everything. The difficulty of EC3 is that a given J usually does not contain such a clean partition, and deciding whether *some* subcollection works requires searching an exponentially large space of subcollections. ◊

3.6.3 The reduction

We now turn any EC3 instance into a (P_0) instance in polynomial time, so that solving (P_0) would decide EC3.

Build the matrix. Given (m, J) with $|J| = N$, build $\boldsymbol{A} \in \{0,1\}^{m\times N}$ whose j-th column is the indicator vector of the triple c_j:

$$(\boldsymbol{a}_j)_k = \mathbf{1}(k \in c_j) = \begin{cases} 1 & k \in c_j, \\ 0 & k \notin c_j. \end{cases}$$

Each column has exactly three ones (it is a "3-hot" vector). For instance, with $m = 9$, the triple $c_j = \{4, 7, 8\}$ gives the column $(0, 0, 0, 1, 0, 0, 1, 1, 0)^\top$. Building $\boldsymbol{A}$ touches each of the mN cells once, and $N \le \binom{m}{3} = O(m^3)$, so the construction is $O(m^4)$, comfortably polynomial.

Choose the right-hand side. Set $\boldsymbol{y} = \mathbf{1}_m = (1, 1, \ldots, 1)^\top$. The key observation is that, for a $\{0,1\}$-valued $\boldsymbol{z}$, the equation $\boldsymbol{Az} = \mathbf{1}_m$ says exactly that the chosen triples form an exact cover. Indeed, reading $\boldsymbol{Az}$ in the synthesis view,

$$(\boldsymbol{Az})_k = \sum_{j:z_j=1} \mathbf{1}(k \in c_j) = \#\{j : z_j = 1,\ k \in c_j\},$$

the number of chosen triples that contain element k. For this count to equal 1 for every k, each element of $[m]$ must be covered exactly once, which is the definition of an exact cover.

Example 3.11 (The reduction on a tiny instance). Let $m = 6$ and $J = \big\{\{1,2,3\}, \{4,5,6\}, \{2,3,4\}\big\}$, so

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \qquad \boldsymbol{y} = \mathbf{1}_6.$$

The choice $\boldsymbol{z} = (1, 1, 0)^\top$ gives $\boldsymbol{Az} = \boldsymbol{a}_1 + \boldsymbol{a}_2 = \mathbf{1}_6$, and the triples $\{1,2,3\}, \{4,5,6\}$ are an exact cover. The choice $\boldsymbol{z} = (1, 0, 1)^\top$ gives $\boldsymbol{Az} = (1, 2, 2, 1, 0, 0)^\top \neq \mathbf{1}_6$: elements $2, 3$ are covered twice and $5, 6$ not at all. ◊

Analyze the sparsest solution. To avoid integrality technicalities we analyze a slightly relaxed feasibility, asking for $\boldsymbol{Az}$ merely *close* to $\mathbf{1}_m$: minimize $\|\boldsymbol{z}\|_0$ subject to $\|\boldsymbol{Az} - \mathbf{1}_m\|_2 < \eta$ for a fixed $\eta < 1$. Two bounds pin down the optimal sparsity.

Lower bound, from feasibility. If any coordinate of $\boldsymbol{Az}$ were zero, say $(\boldsymbol{Az})_{k_0} = 0$, then that coordinate of $\boldsymbol{Az} - \mathbf{1}_m$ equals -1, so $\|\boldsymbol{Az} - \mathbf{1}_m\|_2^2 \ge (-1)^2 = 1$, contradicting $\eta < 1$. Hence *every* coordinate of $\boldsymbol{Az}$ is nonzero, that is, $\|\boldsymbol{Az}\|_0 = m$.

Upper bound, from structure. Each column has exactly three ones, so the support of $\boldsymbol{Az}$ is contained in the union of the triples used:

$$\operatorname{supp}(\boldsymbol{Az}) \subseteq \bigcup_{j:z_j\neq 0} c_j, \qquad \|\boldsymbol{Az}\|_0 \le \sum_{j:z_j\neq 0} |c_j| = 3\,\|\boldsymbol{z}\|_0,$$

with equality exactly when the chosen triples are pairwise disjoint.

Combine. Setting the two together, $m = \|\boldsymbol{Az}\|_0 \le 3\|\boldsymbol{z}\|_0$, so

$$\boxed{\|\boldsymbol{z}\|_0 \ \ge\ \frac{m}{3}}$$

for every feasible $\boldsymbol{z}$, and in particular for the sparsest solution $\boldsymbol{x}^\#$.

3.6.4 The two cases

The optimal value $\|\boldsymbol{x}^\#\|_0$ is either exactly $m/3$ or strictly larger, and the two cases correspond to the two answers of EC3.

Case $\|\boldsymbol{x}^\#\|_0 = m/3$. Then $\boldsymbol{x}^\#$ uses $m/3$ columns, contributing $3 \cdot (m/3) = m$ ones in total, and these must land one per row to make $\mathbf{1}_m$. So no two chosen triples share an element: they are pairwise disjoint, and since $m/3$ disjoint triples have union of size m, they cover all of $[m]$. The chosen triples form an exact cover.

Case $\|\boldsymbol{x}^\#\|_0 > m/3$. We argue by contraposition. If J did contain an exact cover S with $|S| = m/3$, the indicator $\widetilde{\boldsymbol{z}}$ of S would satisfy $\boldsymbol{A}\widetilde{\boldsymbol{z}} = \mathbf{1}_m$ (hence be feasible) with $\|\widetilde{\boldsymbol{z}}\|_0 = m/3 < \|\boldsymbol{x}^\#\|_0$, contradicting the optimality of $\boldsymbol{x}^\#$. So J has no exact cover.

(P_0) is NP-hard

For the indicator construction above,

$$J \text{ has an exact cover of } [m] \quad \Longleftrightarrow \quad \text{the optimal value of } (P_0) \text{ on } (\boldsymbol{A}, \mathbf{1}_m) \text{ equals } m/3.$$

A polynomial-time algorithm for (P_0) would therefore decide EC3 in polynomial time. Since EC3 is NP-complete, (P_0) is NP-hard: no polynomial-time algorithm can solve it in the worst case unless P = NP.

Remark 3.12 *(A tiling picture).* Think of $[m]$ as the cells of a floor and each triple in J as a three-tile mosaic piece you may place. An exact cover is a gapless, overlap-free tiling of the floor using pieces from your box. The reduction says that the fewest pieces needed to put a tile on every cell is at least $m/3$, and equals $m/3$ exactly when a perfect tiling exists. Solving (P_0) on this instance is solving the tiling problem, and tiling is hard.

3.7 The Escape: Relax to (P_1)

We have arrived at a genuine impasse. The problem we want to solve, (P_0), is exactly right and exactly intractable. We cannot expect a fast, always-correct algorithm for it unless P = NP. The way forward is to change the objective. Replace the discontinuous, nonconvex count $\|\cdot\|_0$ by the convex norm $\|\cdot\|_1$:

$$(P_1): \qquad \min_{\boldsymbol{z}\in\mathbb{C}^N} \|\boldsymbol{z}\|_1 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}.$$

As Chapter 5 shows, (P_1) is a linear program, solvable in polynomial time. The miracle of compressed sensing, which occupies the next several chapters, is that under mild conditions on $\boldsymbol{A}$ the convex relaxation (P_1) returns *exactly* the sparse solution that the intractable (P_0) would. We trade an NP-hard combinatorial search for a tractable convex program and, when the geometry cooperates, lose nothing.

Two questions must be answered to make that promise real, and they structure the chapters ahead. First, *when is the sparse solution even unique*, so that there is a single right answer for any algorithm to find? That is Chapter 4. Second, *when does the relaxation* (P_1) *actually coincide with* (P_0)? That is Chapter 5 and, through the restricted isometry property, Chapters 8 and 9.

In the Problem Sets

Problem Set 1 (Chapter 20) is the computational counterpart of this chapter. You implement brute-force (P_0) on small instances, watch it slow to a crawl as N and s grow exactly as the $\binom{N}{s}$ count predicts, and verify oracle recovery against the pseudoinverse formula of Example 3.8. The accompanying notebook produces the residual-versus-sparsity and progressive-recovery plots.

Notes and References

The union-of-subspaces view of Σ_s and the oracle/least-squares perspective are standard; see Foucart and Rauhut [28] and Elad [25]. The NP-hardness of sparse approximation is due to Natarajan [39]; the reduction presented here is from Exact Cover by 3-Sets, a classic NP-complete problem. The analysis/synthesis terminology and the $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$ factorization are developed throughout Elad's book [25]. The relaxation of (P_0) to (P_1) is the founding move of the field [14, 20].

Exercises

Exercise 3.1. Show that Σ_s is closed under scalar multiplication but not under addition, by exhibiting two 2-sparse vectors in $\mathbb{R}^4$ whose sum is 4-sparse. How does this relate to the $2s$ threshold mentioned after Definition 3.3?

Exercise 3.2. Let $\boldsymbol{x} = (0, 7, 0, 0, -3, 0)^\top$ with $\boldsymbol{A} \in \mathbb{R}^{4\times 6}$ of your choosing (full rank on the support). Carry out oracle recovery as in Example 3.8: form $\boldsymbol{A}_S$, compute $\boldsymbol{A}_S^\dagger$, and confirm $\boldsymbol{A}_S^\dagger \boldsymbol{A}\boldsymbol{x}$ returns the nonzero entries.

Exercise 3.3. Repeat the brute-force timing estimate of Section 3.5 for $N = 10^5$ and $s = 20$, assuming 10^{-9} seconds per candidate. Compare your answer to the age of the universe (about 1.4×10^{10} years).

Exercise 3.4. In the EC3 reduction, verify on Example 3.11 that $\|\boldsymbol{A}\boldsymbol{z}\|_0 \le 3\|\boldsymbol{z}\|_0$ for both $\boldsymbol{z} = (1, 1, 0)^\top$ and $\boldsymbol{z} = (1, 0, 1)^\top$, and identify exactly when equality holds in terms of triple disjointness.

Exercise 3.5. Explain why the relaxed feasibility threshold in the reduction is taken to be $\eta < 1$ rather than $\eta = 0$. Where in the lower-bound argument is the strict inequality $\eta < 1$ used?

CHAPTER 4

Uniqueness: Spark and the Null Space Property

Before you ask how to find the answer, make sure there is only one.

Chapter 3 ended at an impasse: the sparsest-solution problem (P_0) is exactly what we want and exactly intractable, and its convex surrogate (P_1) is what we can actually compute. Before chasing algorithms, this chapter settles a prior question that both problems depend on. *When is the sparse solution unique?* If two different s-sparse vectors produced the same measurements, no algorithm could tell them apart, and recovery would be meaningless. We answer the question twice. First we characterize exactly when (P_0) has a unique s-sparse solution, through a four-part equivalence and the matrix invariant called the spark. Then we characterize exactly when the tractable (P_1) recovers the true signal, through the null space property. The two characterizations are parallel theorems, one for the ideal problem and one for the practical one, and together they are the theoretical backbone of the subject.

4.1 When Is Sparse Recovery Unique?

Fix a sensing matrix $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ with $m < N$ and a sparsity level s. The measurements of an s-sparse signal $\boldsymbol{x}$ are $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. Recovery is well posed only if $\boldsymbol{x}$ is the *only* s-sparse vector with these measurements. Geometrically, recall from Chapter 3 that the solution set $\{\boldsymbol{z} : \boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}\}$ is an affine subspace, and the s-sparse vectors form the union of subspaces Σ_s. Uniqueness means the affine solution set pierces Σ_s at exactly one point, as in Fig. 4.1.

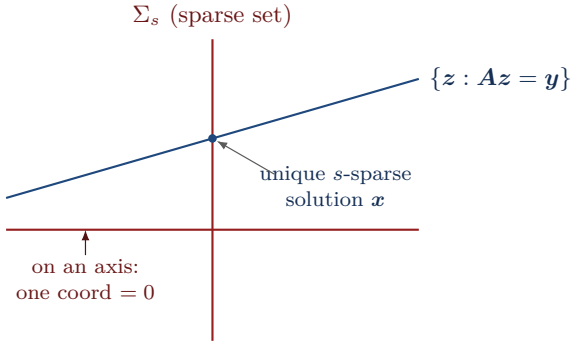


Figure 4.1. Unique sparse recovery: the affine solution set (the diagonal line) meets the union of sparse subspaces Σ_s (drawn here as the two coordinate axes, the 1-sparse vectors) at the single point $\boldsymbol{x}$. A point on an axis has one coordinate equal to zero, so it is sparse; uniqueness means only one such point is consistent with the measurements.

The chapter makes this precise in two steps. The spark theorem (Section 4.4) says exactly when the intersection is a single point, which is the uniqueness condition for (P_0). The null space property (Sections 4.5 and 4.6) says exactly when the tractable program (P_1) returns that point.

4.2 The Equivalence Theorem for ℓ_0 Uniqueness

We first characterize (P_0) uniqueness through four equivalent conditions on $\boldsymbol{A}$. The proof is short and worth knowing in full, because every later uniqueness statement is a refinement of it.

Theorem 4.1 (Equivalent conditions for unique s-sparse recovery). Fix s with $2s \le N$ and $\boldsymbol{A} \in \mathbb{C}^{m\times N}$. The following are equivalent.

(i) For every s-sparse $\boldsymbol{x}$, the only s-sparse $\boldsymbol{z}$ with $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$ is $\boldsymbol{z} = \boldsymbol{x}$. (The matrix separates s-sparse signals.)

(ii) The kernel of $\boldsymbol{A}$ contains no nonzero $2s$-sparse vector: $\mathcal{N}(\boldsymbol{A}) \cap \Sigma_{2s} = \{\boldsymbol{0}\}$. Equivalently, every nonzero $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A})$ has $\|\boldsymbol{v}\|_0 > 2s$.

(iii) For every $S \subseteq [N]$ with $|S| \le 2s$, the column submatrix $\boldsymbol{A}_S$ is injective ($\mathcal{N}(\boldsymbol{A}_S) = \{\boldsymbol{0}\}$).

(iv) Every set of $2s$ columns of $\boldsymbol{A}$ is linearly independent.

The four statements are four angles on one condition: a *recovery* statement (i), a *kernel* statement (ii), a *submatrix* statement (iii), and a *geometric* statement (iv). We prove (i)⇔(ii) in full and then observe that (ii), (iii), (iv) are restatements of each other.

Proof of (ii)⇒(i). Assume the kernel has no nonzero $2s$-sparse vector. Let $\boldsymbol{x}, \boldsymbol{z}$ be s-sparse with $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z}$. Then $\boldsymbol{A}(\boldsymbol{x} - \boldsymbol{z}) = \boldsymbol{0}$, so $\boldsymbol{x} - \boldsymbol{z} \in \mathcal{N}(\boldsymbol{A})$. Because the support of a difference lies in the union of the supports,

$$\|\boldsymbol{x} - \boldsymbol{z}\|_0 \le \|\boldsymbol{x}\|_0 + \|\boldsymbol{z}\|_0 \le s + s = 2s,$$

so $\boldsymbol{x} - \boldsymbol{z}$ is a $2s$-sparse kernel vector. By hypothesis it must be $\boldsymbol{0}$, hence $\boldsymbol{x} = \boldsymbol{z}$. ∎

Proof of (i)⇒(ii). We prove the contrapositive. Suppose some nonzero $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A})$ has $\|\boldsymbol{v}\|_0 \le 2s$. Split its support into two disjoint halves of size at most s, writing $\boldsymbol{v} = \boldsymbol{\alpha} + \boldsymbol{\beta}$ with $\operatorname{supp}(\boldsymbol{\alpha}) \cap \operatorname{supp}(\boldsymbol{\beta}) = \emptyset$ and $\|\boldsymbol{\alpha}\|_0, \|\boldsymbol{\beta}\|_0 \le s$. From $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{0}$,

$$\boldsymbol{A}\boldsymbol{\alpha} = -\boldsymbol{A}\boldsymbol{\beta} = \boldsymbol{A}(-\boldsymbol{\beta}),$$

so $\boldsymbol{\alpha}$ and $-\boldsymbol{\beta}$ are two s-sparse vectors with the same measurement. By (i) they are equal, $\boldsymbol{\alpha} = -\boldsymbol{\beta}$. But their supports are disjoint, so equality forces both to vanish, giving $\boldsymbol{v} = \boldsymbol{\alpha} + \boldsymbol{\beta} = \boldsymbol{0}$, a contradiction. Hence no nonzero $2s$-sparse kernel vector exists. ∎

For the remaining equivalences, note that a nonzero $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A})$ supported on a set T is precisely a nontrivial linear dependence among the columns indexed by T. So "no nonzero $2s$-sparse kernel vector" (ii) is the same as "no linearly dependent set of $\le 2s$ columns," which is (iv); and "$\boldsymbol{A}_S$ injective for all $|S| \le 2s$" (iii) says the same thing submatrix by submatrix. This closes the chain (i)⇔(ii)⇔(iii)⇔(iv).

Example 4.2 (Splitting a $2s$-sparse vector). The split used in the proof is concrete. With $s = 2$ and $\boldsymbol{v} = (1, 0, 3, 0, -1, 4, 0, 0)^\top$ (which is 4-sparse), take $\boldsymbol{\alpha} = (1, 0, 3, 0, 0, 0, 0, 0)^\top$ and $\boldsymbol{\beta} = (0, 0, 0, 0, -1, 4, 0, 0)^\top$. Both are 2-sparse, their supports $\{1, 3\}$ and $\{5, 6\}$ are disjoint, and their sum is $\boldsymbol{v}$. ◊

4.3 A First Lower Bound: $m \ge 2s$

The equivalence theorem immediately yields the most basic measurement bound in the subject.

Corollary 4.3 (Two measurements per nonzero). If $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ admits unique s-sparse recovery (any of the equivalent conditions of Theorem 4.1), then $m \ge 2s$.

Proof. By (iv), some $2s$ columns are linearly independent, so $\operatorname{rank}(\boldsymbol{A}) \geq 2s$. Since $\boldsymbol{A}$ has m rows, $\operatorname{rank}(\boldsymbol{A}) \leq m$, hence $m \geq 2s$. ∎

So an s-sparse signal needs at least $2s$ measurements: one might say one measurement to locate each nonzero and one to weigh it. For $N = 1000$ and $s = 10$ this is only $m = 20$, a fiftyfold reduction from N. The bound is necessary but not sufficient: a matrix with exactly $m = 2s$ rows can still fail (iv) if two of its columns happen to be parallel. Sufficiency requires the stronger properties of Chapters 8 and 9, and for random matrices the right count turns out to be $m = O(s \log(N/s))$, only a logarithmic factor above this floor.

4.4 The Spark of a Matrix

Statement (iv) of Theorem 4.1 asks about the smallest number of dependent columns, which deserves a name.

Definition 4.4 (Spark). The **spark** of $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ is the smallest number of linearly dependent columns:

$$\operatorname{spark}(\boldsymbol{A}) := \min \{ \|\boldsymbol{z}\|_0 : \boldsymbol{z} \neq \boldsymbol{0},\ \boldsymbol{A}\boldsymbol{z} = \boldsymbol{0} \}.$$

Equivalently, it is the size of the smallest linearly dependent subset of columns.

The name, due to Donoho and Elad [21], marks the ignition point at which $\boldsymbol{A}$ first develops a dependence among its columns. Spark and rank measure opposite extremes:

$$\operatorname{rank}(\boldsymbol{A}) = \text{largest number of } \textit{independent} \text{ columns}, \qquad \operatorname{spark}(\boldsymbol{A}) = \text{smallest number of } \textit{dependent} \text{ columns}.$$

For a matrix in general position (a random Gaussian with $N > m$), any m columns are independent and any $m+1$ are dependent by dimension counting, so $\operatorname{rank}(\boldsymbol{A}) = m$ and $\operatorname{spark}(\boldsymbol{A}) = m + 1$, the largest spark possible.

Restating Theorem 4.1(iv) in this language gives the uniqueness theorem in its cleanest form.

Spark characterizes ℓ_0 uniqueness

(P_0) recovers every s-sparse signal uniquely if and only if

$$\operatorname{spark}(\boldsymbol{A}) \ \geq \ 2s + 1.$$

The reason is immediate: every $2s$ columns are independent exactly when the smallest dependent set has size at least $2s + 1$.

Example 4.5 (Reading the spark condition). To recover 3-sparse signals we need $\operatorname{spark}(\boldsymbol{A}) \geq 2(3) + 1 = 7$: every 6 columns of $\boldsymbol{A}$ must be independent, so any linear dependence involves at least 7 columns. This forces $m \geq 6 = 2s$, the bound of Corollary 4.3, now in its sharpest form. ◊

Computing the spark exactly is itself combinatorial: it would require checking subsets of columns for dependence. But a single, cheaply computed scalar gives a usable lower bound on it.

Proposition 4.6 (Coherence bound on spark). Let $\boldsymbol{A}$ have ℓ_2-normalized columns and mutual coherence $\mu(\boldsymbol{A}) = \max_{i \neq j} |\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle|$. Then

$$\operatorname{spark}(\boldsymbol{A}) \ \geq \ 1 + \frac{1}{\mu(\boldsymbol{A})}.$$

Consequently, (P_0) recovers every s-sparse signal whenever $s < \tfrac{1}{2}\big(1 + 1/\mu(\boldsymbol{A})\big)$.

We prove this bound, via a Gershgorin argument, in Chapter 8, where coherence is the central object. For now it is the practical handle on spark: a matrix with small coherence has large spark and therefore recovers signals of proportionally larger sparsity. This is exactly the uniqueness threshold that governs Problem Set 1.

4.5 From ℓ_0 to ℓ_1: The Null Space Property

The spark theorem certifies that the *ideal* program (P_0) has a unique solution. But (P_0) is intractable, and we intend to solve (P_1) instead. We now ask the corresponding question for the tractable program: when does (P_1) return the true sparse signal? The answer is a different, and strictly stronger, condition on the null space.

First a piece of notation used throughout the rest of the book.

Definition 4.7 (Restriction to a support set). For $\boldsymbol{v} \in \mathbb{C}^N$ and $S \subseteq [N]$, write $\boldsymbol{v}_S \in \mathbb{C}^N$ for the vector that agrees with $\boldsymbol{v}$ on S and is zero off S, and $\boldsymbol{v}_{S^c}$ for the complementary piece. Then $\boldsymbol{v} = \boldsymbol{v}_S + \boldsymbol{v}_{S^c}$, and because the two have disjoint supports,

$$\|\boldsymbol{v}\|_1 = \|\boldsymbol{v}_S\|_1 + \|\boldsymbol{v}_{S^c}\|_1.$$

Definition 4.8 (Null Space Property). A matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ has the **null space property** (NSP) of order s if for every nonzero $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A})$ and every $S \subseteq [N]$ with $|S| \le s$,

$$\|\boldsymbol{v}_S\|_1 \;<\; \|\boldsymbol{v}_{S^c}\|_1.$$

Adding $\|\boldsymbol{v}_S\|_1$ to both sides gives the equivalent form $\|\boldsymbol{v}_S\|_1 < \frac{1}{2}\|\boldsymbol{v}\|_1$: *no null vector concentrates even half of its ℓ_1 mass on any set of size s*. The intuition is exactly the failure mode of (P_1). If recovery failed, some feasible $\boldsymbol{z} \neq \boldsymbol{x}$ would have $\|\boldsymbol{z}\|_1 \le \|\boldsymbol{x}\|_1$, and then $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{z}$ would be a nonzero null vector whose ℓ_1 mass is concentrated on the support of $\boldsymbol{x}$. NSP forbids precisely that: every null vector is spread out, never piled onto a small set.

Proposition 4.9 (NSP is stronger than the spark condition). If $\boldsymbol{A}$ has NSP of order s, then $\mathcal{N}(\boldsymbol{A}) \cap \Sigma_s = \{\boldsymbol{0}\}$.

Proof. Let $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A})$ be s-sparse and take $S = \operatorname{supp}(\boldsymbol{v})$, so $|S| \le s$ and $\boldsymbol{v}_{S^c} = \boldsymbol{0}$. NSP gives $\|\boldsymbol{v}_S\|_1 < \|\boldsymbol{v}_{S^c}\|_1 = 0$, forcing $\boldsymbol{v}_S = \boldsymbol{0}$, hence $\boldsymbol{v} = \boldsymbol{0}$. ∎

So NSP of order s implies no s-sparse null vector, which is the (P_0) uniqueness condition at level s. NSP says more than non-existence: it controls the relative ℓ_1 masses, and that extra control is exactly what the convex program needs.

> NSP is delicate. The strict inequality matters, and it can fail by a hair. The matrix $\boldsymbol{A} = [1, 1, 1] \in \mathbb{R}^{1\times 3}$ has the null vector $\boldsymbol{v} = (1, -1, 0)^\top$, for which $\|\boldsymbol{v}_{\{1\}}\|_1 = 1 = \|\boldsymbol{v}_{\{1\}^c}\|_1$. Equality, not strict inequality, so NSP of order 1 fails. We will see in Example 4.13 that (P_1) correspondingly fails to recover 1-sparse signals from this $\boldsymbol{A}$.

4.6 The Null Space Property Theorem

The central theorem of this chapter says that NSP is not merely sufficient for (P_1) recovery; it is exactly equivalent to it.

Theorem 4.10 (NSP characterizes ℓ_1 recovery). Let $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ with $m < N$. The following are equivalent.

(A) **Recovery:** for every s-sparse $\boldsymbol{x}$, the unique solution of (P_1) with $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is $\boldsymbol{x}$ itself.
(B) **NSP:** $\boldsymbol{A}$ has the null space property of order s.

The two directions use different ideas. The implication (A)⇒(B) picks a clever pair of vectors and reads off NSP from uniqueness. The implication (B)⇒(A) takes an arbitrary competitor, forms a null vector, and closes with the triangle inequality.

Proof of (A)⇒(B). Assume (P_1) uniquely recovers every s-sparse signal. Let $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A})$ be nonzero and let $|S| \le s$. Split $\boldsymbol{v} = \boldsymbol{v}_S + \boldsymbol{v}_{S^c}$. From $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{0}$,

$$\boldsymbol{A}\boldsymbol{v}_S = -\boldsymbol{A}\boldsymbol{v}_{S^c} = \boldsymbol{A}(-\boldsymbol{v}_{S^c}) =: \boldsymbol{y}.$$

The vector $\boldsymbol{v}_S$ is s-sparse (supported on S), so by assumption it is the *unique* (P_1) minimizer for the measurement $\boldsymbol{y}$. The vector $-\boldsymbol{v}_{S^c}$ is feasible for the same problem, and in the generic case it differs from $\boldsymbol{v}_S$ (the degenerate cases $\boldsymbol{v}_S = \boldsymbol{0}$ and $\boldsymbol{v}_{S^c} = \boldsymbol{0}$ make the inequality trivial or are excluded by uniqueness at $\boldsymbol{x} = \boldsymbol{0}$). Uniqueness of the minimizer therefore gives the *strict* inequality

$$\|\boldsymbol{v}_S\|_1 \;<\; \|-\boldsymbol{v}_{S^c}\|_1 = \|\boldsymbol{v}_{S^c}\|_1.$$

Since $\boldsymbol{v}$ and S were arbitrary, $\boldsymbol{A}$ has NSP of order s. ■

Remark 4.11 *(Where the strictness comes from).* NSP demands a *strict* inequality, and the proof shows exactly where it is born: in the word "unique." If we only knew $\boldsymbol{x}$ was *a* minimizer, allowing ties, we would get $\|\boldsymbol{v}_S\|_1 \le \|\boldsymbol{v}_{S^c}\|_1$. Uniqueness of recovery is what upgrades $\le$ to $<$.

Proof of (B)⇒(A). Assume NSP of order s. Let $\boldsymbol{x}$ be s-sparse with support $S^\star = \mathrm{supp}(\boldsymbol{x})$, so $\boldsymbol{x}_{S^\star} = \boldsymbol{x}$ and $\boldsymbol{x}_{(S^\star)^c} = \boldsymbol{0}$. Let $\boldsymbol{z} \neq \boldsymbol{x}$ be any feasible point, $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. We must show $\|\boldsymbol{z}\|_1 > \|\boldsymbol{x}\|_1$.

Step 1 (a null vector). Set $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{z}$. Then $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{0}$ and $\boldsymbol{v} \neq \boldsymbol{0}$, so $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A}) \setminus \{\boldsymbol{0}\}$.

Step 2 (apply NSP at $S^\star$). Since $|S^\star| \le s$,

$$\|\boldsymbol{v}_{S^\star}\|_1 \;<\; \|\boldsymbol{v}_{(S^\star)^c}\|_1.$$

Step 3 (rewrite both sides). On $S^\star$, since $\boldsymbol{x}$ is supported there, $\boldsymbol{v}_{S^\star} = \boldsymbol{x} - \boldsymbol{z}_{S^\star}$. Off $S^\star$, since $\boldsymbol{x}$ vanishes there, $\boldsymbol{v}_{(S^\star)^c} = -\boldsymbol{z}_{(S^\star)^c}$. The inequality becomes

$$\|\boldsymbol{x} - \boldsymbol{z}_{S^\star}\|_1 \;<\; \|\boldsymbol{z}_{(S^\star)^c}\|_1.$$

Step 4 (triangle inequality). Add and subtract $\boldsymbol{z}_{S^\star}$ inside $\|\boldsymbol{x}\|_1$:

$$\|\boldsymbol{x}\|_1 = \|(\boldsymbol{x} - \boldsymbol{z}_{S^\star}) + \boldsymbol{z}_{S^\star}\|_1 \le \|\boldsymbol{x} - \boldsymbol{z}_{S^\star}\|_1 + \|\boldsymbol{z}_{S^\star}\|_1 < \|\boldsymbol{z}_{(S^\star)^c}\|_1 + \|\boldsymbol{z}_{S^\star}\|_1.$$

Step 5 (recombine $\boldsymbol{z}$). The two pieces $\boldsymbol{z}_{S^\star}$ and $\boldsymbol{z}_{(S^\star)^c}$ have disjoint supports, so their ℓ_1 norms add to $\|\boldsymbol{z}\|_1$. Therefore $\|\boldsymbol{x}\|_1 < \|\boldsymbol{z}\|_1$. As $\boldsymbol{z}$ was an arbitrary feasible competitor, $\boldsymbol{x}$ is the unique (P_1) minimizer. ■

Remark 4.12 *(The whole hard direction in two lines).* Apply NSP to $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{z}$ at $S = \mathrm{supp}(\boldsymbol{x})$ to get $\|\boldsymbol{x} - \boldsymbol{z}_S\|_1 < \|\boldsymbol{z}_{S^c}\|_1$, then $\|\boldsymbol{x}\|_1 \le \|\boldsymbol{x} - \boldsymbol{z}_S\|_1 + \|\boldsymbol{z}_S\|_1 < \|\boldsymbol{z}_{S^c}\|_1 + \|\boldsymbol{z}_S\|_1 = \|\boldsymbol{z}\|_1$. The entire content is bookkeeping with restrictions and one use of the triangle inequality.

The payoff

An s-sparse signal is recovered from its measurements by the polynomial-time program (P_1) if and only if $\boldsymbol{A}$ has the null space property of order s. The NP-hard (P_0) and the tractable (P_1) return the same answer precisely when NSP holds. This equivalence is why compressed sensing is a working engineering field and not only a theory.

4.7 Spark and NSP Side by Side

The two main theorems of the chapter are parallel: one governs the ideal combinatorial program, the other the practical convex one. Table 4.1 sets them next to each other.

	ℓ_0 **recovery (ideal)**	ℓ_1 **recovery (practical)**
Program	(P_0), NP-hard	(P_1), a linear program
Exact condition on $\boldsymbol{A}$	$\operatorname{spark}(\boldsymbol{A}) \geq 2s+1$	NSP of order s
Null-space meaning	no $2s$-sparse null vector	no null vector concentrates ℓ_1 mass on a set of size s
Measurements implied	$m \geq 2s$	$m \geq 2s$
How to verify	$\binom{N}{2s}$ subset checks	one inequality per null vector (uncheckable directly)

Table 4.1. The spark theorem and the NSP theorem are the same statement told for two programs. NSP is the strictly stronger condition, since it implies the spark condition (Proposition 4.9).

4.8 Worked Example: When Recovery Fails

The theorem is sharp, and a failure of NSP produces a genuine failure of recovery.

Example 4.13 (NSP fails, and (P_1) fails with it). Take $\boldsymbol{A} = [1,1,1] \in \mathbb{R}^{1\times 3}$, so $\mathcal{N}(\boldsymbol{A}) = \{\boldsymbol{v} : v_1+v_2+v_3 = 0\}$. The null vector $\boldsymbol{v} = (1,-1,0)^\top$ gives, at $S=\{1\}$,

$$\|\boldsymbol{v}_{\{1\}}\|_1 = 1, \qquad \|\boldsymbol{v}_{\{1\}^c}\|_1 = |-1| + |0| = 1,$$

so $\|\boldsymbol{v}_S\|_1 = \|\boldsymbol{v}_{S^c}\|_1$ and NSP of order 1 fails (only an equality, never the required strict inequality). The theorem predicts that (P_1) cannot recover 1-sparse signals. Indeed, take $\boldsymbol{x} = (1,0,0)^\top$, with $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = 1$. The program

$$\min_{\boldsymbol{z}\in\mathbb{R}^3} \|\boldsymbol{z}\|_1 \quad \text{s.t.} \quad z_1+z_2+z_3 = 1$$

has $\|\boldsymbol{z}\|_1 \geq 1$ for every feasible $\boldsymbol{z}$, with equality achieved by $(1,0,0)^\top$, $(0,1,0)^\top$, $(0,0,1)^\top$, and $(\frac{1}{2},\frac{1}{2},0)^\top$ alike. There are infinitely many minimizers tied at ℓ_1-norm 1, so recovery is not unique, exactly as the failure of NSP requires. $\lozenge$

Example 4.14 (A matrix that does recover). By contrast, a normalized random Gaussian $\boldsymbol{A} \in \mathbb{R}^{2\times 4}$ has, with high probability, small mutual coherence, and Proposition 4.6 together with the coherence-implies-NSP bound of Chapter 8 then gives NSP of order 1. For such a matrix every 1-sparse signal is recovered uniquely by (P_1). Two measurements can do no more: recovering *every* 2-sparse signal would need $m \geq 2s = 4$ by Corollary 4.3, and a 2×4 matrix has $\operatorname{spark}(\boldsymbol{A}) \leq m+1 = 3 < 2s+1$, so some 2-sparse signals are unavoidably ambiguous. Random matrices are the gold standard precisely because they satisfy these null-space conditions automatically once m is large enough. $\lozenge$

4.9 The Catch: NSP Cannot Be Checked Directly

Theorem 4.10 is a clean characterization, but it has a serious practical defect: NSP cannot be verified by brute force. Two infinities stand in the way. The null space is a continuous subspace of dimension $N-m$, so the inequality must hold for *uncountably many* null vectors $\boldsymbol{v}$. And it must hold for every support of size up to s, of which there are up to $\binom{N}{s}$, already about 8.8×10^9 for $N=256$ and $s=5$. There is no finite computation that confirms NSP for a given matrix.

NSP is therefore a *certificate*, valuable for telling us what to aim for, but not a *test*. The way forward, taken up in Chapters 8 and 9, is to find conditions that are easy to check or easy to guarantee and that *imply* NSP:

- **Coherence.** A single scalar $\mu(\boldsymbol{A})$, computable in $O(mN^2)$ time. If $\mu(\boldsymbol{A}) < \frac{1}{2s-1}$, then $\boldsymbol{A}$ has NSP of order s.
- **Restricted isometry.** A geometric condition that random matrices satisfy automatically once $m \gtrsim s\log(N/s)$. If the order-$2s$ restricted isometry constant satisfies $\delta_{2s} < \sqrt{2}-1$, then $\boldsymbol{A}$ has NSP of order s.

These two routes, coherence and the restricted isometry property, are how a sensing matrix is actually certified, and they occupy the next part of the book.

In the Problem Sets

Problem Set 2 (Chapter 21) is where these conditions become concrete. You compute mutual coherence for several matrices, use it to predict the maximum recoverable sparsity through the spark bound of Proposition 4.6, and watch (P_1) (solved as a linear program) succeed or fail exactly as the coherence threshold predicts. The accompanying notebook runs the recovery experiments on both well-conditioned and deliberately coherent matrices.

Notes and References

The spark and the bound $\operatorname{spark}(\boldsymbol{A}) \geq 1+1/\mu$ are due to Donoho and Elad [21]; the equivalence with unique ℓ_0 recovery is developed there and in Gribonval and Nielsen [31]. The null space property and the equivalence with ℓ_1 recovery are standard; a complete treatment, including the approximately-sparse and noisy versions that strengthen Definition 4.8, is in Foucart and Rauhut [28]. The course writes NUP for the same property (using NUP as shorthand for the Null Space Property); the literature more often writes NSP, which we follow. The robust and stable variants, with the ℓ_2/ℓ_1 form $\|\boldsymbol{v}_S\|_2 \leq C\|\boldsymbol{v}_{S^c}\|_1/\sqrt{s}$, are what eventually yield error bounds for compressible (not exactly sparse) signals; we reach them through the restricted isometry property in Chapter 8.

Exercises

Exercise 4.1. Let $\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$. Find a nonzero 2-sparse vector in $\mathcal{N}(\boldsymbol{A})$, and conclude that $\operatorname{spark}(\boldsymbol{A}) = 2$. What is the largest s for which (P_0) uniquely recovers all s-sparse signals from this $\boldsymbol{A}$? (Even $s = 1$ fails; explain why.)

Exercise 4.2. Show that for any $\boldsymbol{A} \in \mathbb{C}^{m\times N}$, $\operatorname{spark}(\boldsymbol{A}) \leq \operatorname{rank}(\boldsymbol{A}) + 1$. When is the inequality an equality? Relate your answer to matrices "in general position."

Exercise 4.3. Prove the equivalence of the two forms of NSP: for a fixed null vector $\boldsymbol{v}$ and set S, $\|\boldsymbol{v}_S\|_1 < \|\boldsymbol{v}_{S^c}\|_1$ if and only if $\|\boldsymbol{v}_S\|_1 < \frac{1}{2}\|\boldsymbol{v}\|_1$.

Exercise 4.4. Carefully justify the "generic case" in the proof of Theorem 4.10(A)⇒(B): show that if $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A})$ is nonzero and $\boldsymbol{A}$ admits unique (P_1) recovery of all s-sparse signals, then the degenerate case $\boldsymbol{v}_{S^c} = \boldsymbol{0}$ (i.e. $\boldsymbol{v}$ itself s-sparse) cannot occur. *Hint: such a $\boldsymbol{v}$ would be an s-sparse null vector; relate it to recovery of $\boldsymbol{x} = \boldsymbol{0}$.*

Exercise 4.5. Let $\boldsymbol{A} = [1,\, 1/2,\, 2] \in \mathbb{R}^{1\times 3}$. Find a null vector that violates NSP of order 1, and exhibit a 1-sparse signal that (P_1) fails to recover uniquely from $\boldsymbol{A}$. Compare with Example 4.13.

CHAPTER 5

Convex Relaxation and ℓ_1 Minimization

Among all explanations that fit, take the one of least total weight.

We left Chapter 3 with a tractable substitute for the intractable (P_0): replace the count $\|\cdot\|_0$ with the convex norm $\|\cdot\|_1$, giving

$$(\mathrm{P}_1): \qquad \min_{\boldsymbol{z}} \|\boldsymbol{z}\|_1 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}.$$

This chapter is about why that substitution is the right one. We first see *geometrically* why minimizing the ℓ_1 norm lands on sparse solutions while the ℓ_2 norm does not. We then show that (P_1) is a linear program, so it inherits decades of fast, reliable solvers. We prove that any unique (P_1) solution is automatically sparse, with at most m nonzeros, even though we never asked for sparsity. Finally we collect the noise-tolerant cousins of (P_1), the basis pursuit family, that are what one actually runs on real data. Chapter 4 told us *when* (P_1) recovers the truth; this chapter is about *what* (P_1) is and *how* to solve it.

5.1 Why ℓ_1 Promotes Sparsity

The cleanest explanation is geometric, and it lives already in two dimensions. The constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ is an affine subspace, a line in the plane. Picture it as a fixed track. To minimize a norm subject to staying on the track, we inflate the norm ball outward from the origin until it first touches the track; the point of first contact is the minimizer. The shape of the ball decides where that contact happens.

Figure 5.1 tells the whole story. The ℓ_2 ball is a circle; a circle has no preferred direction, so it generically meets a tilted line at a point off both axes, and the ℓ_2-minimal solution has every coordinate nonzero. The ℓ_1 ball is a diamond whose four corners sit exactly on the coordinate axes. A growing diamond almost always touches a tilted line at one of those corners first, and a corner is a point where one coordinate vanishes. So the ℓ_1-minimal solution is sparse.

Remark 5.1 *(The cross-polytope in high dimension).* In $\mathbb{R}^N$ the ℓ_1 ball is the *cross-polytope*, the convex hull of the $2N$ signed standard basis vectors $\pm\boldsymbol{e}_1, \ldots, \pm\boldsymbol{e}_N$. All of its vertices lie on the coordinate axes, and its low-dimensional faces are aligned with small sets of coordinates. A generic affine constraint first meets this polytope at a low-dimensional face, which is exactly a vector with few nonzeros. The ℓ_2 ball, a perfectly round sphere, has no such structure, which is why ℓ_2 minimization (the minimum-energy solution, the pseudoinverse of Chapter 2) returns dense answers.

The picture also explains why ℓ_1 is the *best* convex choice. Table 5.1 runs through the ℓ_q "balls." For $q > 1$ the ball bulges outward and the contact point moves off the axes, so the solution is dense. For $q < 1$ the ball caves inward and the corners sharpen, so the solution is sparse, but the ball is no longer convex and the optimization becomes hard. Exactly at $q = 1$ the ball is the largest convex shape whose corners still lie on the axes: convex, hence tractable, and corner-bearing, hence sparsity-seeking. The ℓ_1 norm is the tightest convex relaxation of the ℓ_0 count.

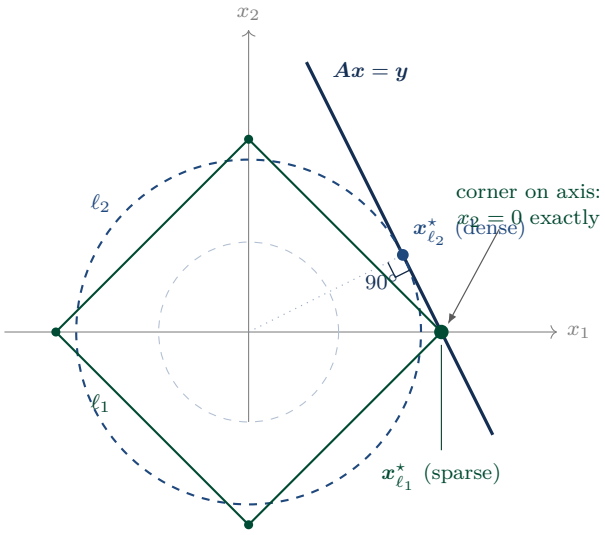


Figure 5.1. Inflating the norm ball to the constraint line. The round ℓ_2 ball touches the tilted line at a point with both coordinates nonzero (a dense answer): its shortest reach is perpendicular to the line, which generically lands off the axes. The ℓ_1 diamond, whose corners sit on the axes, touches first at a corner, where one coordinate is exactly zero (a sparse answer).

q	convex?	sparse solution?	verdict
$q > 1$	yes	no (dense)	tractable, wrong answer
$q = 1$	yes	**yes**	**tractable, right answer**
$0 < q < 1$	no	yes	right answer, nonconvex and hard
$q = 0$	no	yes	right answer, NP-hard

Table 5.1. The ℓ_q spectrum. The norm ℓ_1 is the unique sweet spot: convex (so a local optimum is global, Chapter 2) yet still drawn to the sparse corners.

5.2 (P_1) Is a Linear Program

The geometry is suggestive; the computational payoff is concrete. The objective $\|\boldsymbol{z}\|_1 = \sum_j |z_j|$ is not linear because of the absolute values, but a standard trick removes them and turns (P_1) into a linear program. We give it for real $\boldsymbol{z}$; the complex case splits each entry into four pieces instead of two and is otherwise identical.

Definition 5.2 (Positive and negative parts). For $\boldsymbol{z} \in \mathbb{R}^N$ define $\boldsymbol{z}^+, \boldsymbol{z}^- \in \mathbb{R}^N$ entrywise by $z_j^+ = \max(z_j, 0)$ and $z_j^- = \max(-z_j, 0)$. Both are nonnegative, and

$$\boldsymbol{z} = \boldsymbol{z}^+ - \boldsymbol{z}^-, \qquad |z_j| = z_j^+ + z_j^-.$$

Example 5.3 (The split on a concrete vector). For $\boldsymbol{z} = (1, -5, 2)^\top$, the parts are $\boldsymbol{z}^+ = (1, 0, 2)^\top$ and $\boldsymbol{z}^- = (0, 5, 0)^\top$. Then $\boldsymbol{z}^+ - \boldsymbol{z}^- = (1, -5, 2)^\top = \boldsymbol{z}$, and $\|\boldsymbol{z}\|_1 = 1 + 5 + 2 = 8 = (1{+}0) + (0{+}5) + (2{+}0)$. ◊

With the split, the objective becomes linear, $\|\boldsymbol{z}\|_1 = \sum_j (z_j^+ + z_j^-) = \mathbf{1}^\top \boldsymbol{z}^+ + \mathbf{1}^\top \boldsymbol{z}^-$, and the constraint becomes

$$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{A}\boldsymbol{z}^+ - \boldsymbol{A}\boldsymbol{z}^- = \begin{pmatrix} \boldsymbol{A} & -\boldsymbol{A} \end{pmatrix} \begin{pmatrix} \boldsymbol{z}^+ \\ \boldsymbol{z}^- \end{pmatrix} = \boldsymbol{y}.$$

Collecting these gives a linear program in standard form.

(P_1) as a linear program

$$\underset{\boldsymbol{z}^+,\boldsymbol{z}^-\in\mathbb{R}^N}{\text{minimize}}\ \mathbf{1}^\top \boldsymbol{z}^+ + \mathbf{1}^\top \boldsymbol{z}^- \quad \text{subject to} \quad \begin{pmatrix}\boldsymbol{A} & -\boldsymbol{A}\end{pmatrix}\begin{pmatrix}\boldsymbol{z}^+\\ \boldsymbol{z}^-\end{pmatrix} = \boldsymbol{y}, \quad \boldsymbol{z}^+ \geq \mathbf{0},\ \boldsymbol{z}^- \geq \mathbf{0}.$$

This is a linear program in $2N$ variables with m equality constraints. Its optimum yields $\boldsymbol{z} = \boldsymbol{z}^+ - \boldsymbol{z}^-$, the ($\mathrm{P}_1$) solution.

Linear programs of this form are solved routinely by the simplex method (Dantzig, 1947), which walks along the edges of the feasible polytope from vertex to vertex, and by interior-point methods (Karmarkar, 1984), which cut through the interior along a central path. Any standard linear-programming routine handles a problem of this size in a single call, in any scientific computing environment. This is the entire practical content of "solve (P_1)."

Example 5.4 (A tiny end-to-end LP). Let $\boldsymbol{A} = (1, -1, 1) \in \mathbb{R}^{1\times 3}$ and $\boldsymbol{y} = 2$. The split LP minimizes $\sum_j (z_j^+ + z_j^-)$ subject to $z_1^+ - z_1^- - z_2^+ + z_2^- + z_3^+ - z_3^- = 2$ and nonnegativity. The optimum sets $z_1^+ = 2$ with all other variables zero, giving $\boldsymbol{z} = (2, 0, 0)^\top$ and $\|\boldsymbol{z}\|_1 = 2$; the alternative $\boldsymbol{z} = (0, 0, 2)^\top$ is equally optimal. Both are 1-sparse, so the linear program has found a sparsest solution. ◊

5.2.1 A short tour of the machinery

It is worth recalling, from Chapter 2, why this reformulation is such good news. A function is convex if every chord lies on or above its graph, and a set is convex if it contains every segment between its points. The crucial consequence is that for a convex objective over a convex feasible set, *every local minimum is a global minimum*: there are no false valleys to trap a solver. A linear program is the simplest convex problem, a linear objective over a polytope (the intersection of finitely many half-spaces). Its minimum is always attained at a vertex of the polytope, and the simplex method exploits this by hopping between neighbouring vertices, each step lowering the objective, until no neighbour is lower. Because (P_1) is a linear program, any solver that finds a local optimum has found *the* answer. This is the entire reason the relaxation is useful: (P_0) is nonconvex (its feasible structure is a union of coordinate subspaces, riddled with local minima, hence the NP-hardness of Chapter 3), while (P_1) is convex and solved in polynomial time.

5.3 ℓ_1 Solutions Are Automatically Sparse

We motivated ℓ_1 minimization as a sparsity surrogate by a picture. The picture can be made into a theorem: whenever (P_1) has a unique solution, that solution is provably sparse, with no more nonzeros than there are measurements. The key step is a statement about the columns sitting under the support.

Theorem 5.5 (Support columns are independent). If (P_1) has a unique solution $\boldsymbol{x}^\star$, then the columns of $\boldsymbol{A}$ indexed by its support, $\{\boldsymbol{a}_j : j \in \operatorname{supp}(\boldsymbol{x}^\star)\}$, are linearly independent.

Proof. We prove the contrapositive: if the support columns are dependent, the solution is not unique. Let $S = \operatorname{supp}(\boldsymbol{x}^\star)$ and suppose $\{\boldsymbol{a}_j : j \in S\}$ is linearly dependent. Then there is a nonzero $\boldsymbol{v} \in \mathbb{R}^N$ with $\operatorname{supp}(\boldsymbol{v}) \subseteq S$ and $\boldsymbol{A}\boldsymbol{v} = \mathbf{0}$.

For small ε, consider $\boldsymbol{x}^\star + \varepsilon\boldsymbol{v}$. It is feasible, since $\boldsymbol{A}(\boldsymbol{x}^\star + \varepsilon\boldsymbol{v}) = \boldsymbol{A}\boldsymbol{x}^\star + \varepsilon\boldsymbol{A}\boldsymbol{v} = \boldsymbol{y}$. Because $\operatorname{supp}(\boldsymbol{v}) \subseteq S$, the perturbation only moves coordinates where $\boldsymbol{x}^\star$ is already nonzero, so for $|\varepsilon|$ small enough no sign flips and, writing $s_j = \operatorname{sgn}(x_j^\star)$,

$$\|\boldsymbol{x}^\star + \varepsilon\boldsymbol{v}\|_1 = \sum_{j\in S} |x_j^\star + \varepsilon v_j| = \|\boldsymbol{x}^\star\|_1 + \varepsilon \sum_{j\in S} s_j v_j.$$

If $\sum_{j\in S} s_j v_j \neq 0$, choose the sign of ε to make the second term negative; then $\|\boldsymbol{x}^\star + \varepsilon\boldsymbol{v}\|_1 < \|\boldsymbol{x}^\star\|_1$, contradicting optimality. If $\sum_{j\in S} s_j v_j = 0$, then both $\boldsymbol{x}^\star + \varepsilon\boldsymbol{v}$ and $\boldsymbol{x}^\star - \varepsilon\boldsymbol{v}$ are feasible with the *same* ℓ_1 norm as $\boldsymbol{x}^\star$, giving a second minimizer and contradicting uniqueness. Either way uniqueness fails, which is the contrapositive of the claim. ■

The consequence is the promised sparsity bound, and its proof is two lines.

Corollary 5.6 (ℓ_1 minimizers are sparse). If (P_1) has a unique solution $\boldsymbol{x}^\star$, then

$$\|\boldsymbol{x}^\star\|_0 \;\le\; \operatorname{rank}(\boldsymbol{A}) \;\le\; m \;\ll\; N.$$

Proof. By Theorem 5.5 the $\|\boldsymbol{x}^\star\|_0$ support columns are linearly independent, and a set of independent columns of $\boldsymbol{A}$ has at most $\operatorname{rank}(\boldsymbol{A})$ members. Finally $\operatorname{rank}(\boldsymbol{A}) \le m$ because $\boldsymbol{A}$ has m rows. ∎

> This is the punchline of the chapter. We minimized the ℓ_1 norm, never the count of nonzeros, and yet the solution has at most m nonzeros, hence is sparse because $m \ll N$. Sparsity was not imposed; it fell out of the geometry of the ℓ_1 ball, whose axis-aligned corners force the optimum onto a low-dimensional face. The convex relaxation does not merely tolerate sparse answers, it produces them.

5.4 Basis Pursuit and Its Variants

The equality-constrained program (P_1) is the idealized core, and under the name **basis pursuit** it predates the modern theory.

Definition 5.7 (Basis pursuit). Given $\boldsymbol{A}$ and $\boldsymbol{y}$, the **basis pursuit** problem is

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y},$$

identical to (P_1). The name, coined by Chen, Donoho, and Saunders [16], reflects the goal of *pursuing* a representation of $\boldsymbol{y}$ using as few columns (basis elements) of $\boldsymbol{A}$ as possible.

Real measurements carry noise, and the exact constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ is then too rigid: it may have no feasible sparse point, or it may force the solution to fit the noise. Three relaxations soften it, and all three are convex.

Definition 5.8 (The basis pursuit family). With a noise budget $\eta > 0$, a regularization weight $\lambda > 0$, or a sparsity budget $\tau > 0$:

$$\begin{aligned}
&\text{QCBP} && \min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \;\text{ s.t. }\; \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \le \eta, \\
&\text{BPDN} && \min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \\
&\text{LASSO} && \min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \;\text{ s.t. }\; \|\boldsymbol{x}\|_1 \le \tau.
\end{aligned}$$

Quadratically constrained basis pursuit (QCBP) keeps the sparsity objective but asks only that $\boldsymbol{A}\boldsymbol{x}$ explain $\boldsymbol{y}$ to within the noise level η; setting $\eta = 0$ recovers basis pursuit. Basis pursuit denoising (BPDN) drops the constraint entirely and adds the two goals into one unconstrained objective, with λ trading data fidelity against sparsity: large λ buys sparser, less faithful solutions, small λ the reverse. The LASSO (Tibshirani, 1996 [47]) swaps the roles of objective and constraint relative to BPDN, minimizing the residual subject to a hard cap τ on the total magnitude.

The weight λ is not an abstraction: it sets exactly how aggressively the solver trades fidelity for sparsity, and in the cleanest case the trade is visible in closed form. When $\boldsymbol{A}$ has orthonormal columns the BPDN objective decouples across coordinates and its solution is entrywise soft thresholding, $\hat{x}_j = \mathcal{S}_\lambda(\hat{y}_j)$ with $\hat{\boldsymbol{y}} = \boldsymbol{A}^\top\boldsymbol{y}$ (Chapter 7). Sweeping λ then traces the whole *regularization path*.

Example 5.9 (Sweeping the BPDN weight). Take $\boldsymbol{A} = \boldsymbol{I}_2$ and $\boldsymbol{y} = (4, 0.5)^\top$, so BPDN reads $\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$ and the minimizer is $\hat{\boldsymbol{x}} = \mathcal{S}_\lambda(\boldsymbol{y})$. Increasing λ kills the small coordinate first, then the large one, and the residual grows to pay for it:

λ	$\hat{\boldsymbol{x}}$	$\|\hat{\boldsymbol{x}}\|_0$	$\|\hat{\boldsymbol{x}}\|_1$	$\frac{1}{2}\|\hat{\boldsymbol{x}}-\boldsymbol{y}\|_2^2$
0	$(4,\ 0.5)$	2	4.5	0
1	$(3,\ 0)$	1	3	0.63
2	$(2,\ 0)$	1	2	2.13
4	$(0,\ 0)$	0	0	8.13

Each coordinate $\hat{x}_j(\lambda) = \max(|y_j| - \lambda, 0)\ \mathrm{sgn}(y_j)$ shrinks linearly and hits zero at $\lambda = |y_j|$: the small entry 0.5 dies at $\lambda = 0.5$, the large entry 4 at $\lambda = 4$ (Fig. 5.2). Large λ buys sparsity at the cost of fit, exactly the QCBP-BPDN-LASSO trade in numbers. ◊

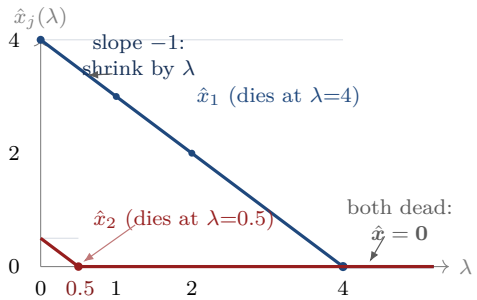


Figure 5.2. The regularization path for Example 5.9. Each recovered coordinate decreases linearly in λ (slope -1) and vanishes once λ exceeds its data value; the small coordinate leaves first, at $\lambda = 0.5$, the large one at $\lambda = 4$. The marked dots are the table's sample rows. This piecewise-linear path is the orthogonal-case form of the LASSO path.

Remark 5.10 *(They are the same problem in three disguises).* For matched parameters, QCBP, BPDN, and the LASSO trace out the *same* solution path, a fact that is Lagrangian duality at work: BPDN is the Lagrangian relaxation of QCBP, with λ playing the role of the multiplier on the residual constraint, and the LASSO is the constrained form whose budget τ corresponds monotonically to λ. Increasing the noise budget η matches increasing the penalty λ and decreasing the magnitude budget τ. So a solver for any one of them solves all three by sweeping its tuning parameter; which form to use is a matter of convenience, not of capability.

Remark 5.11 *(The same objective in machine learning).* The BPDN objective $\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2+\lambda\|\boldsymbol{x}\|_1$ is the ℓ_1-regularized least-squares loss that appears throughout statistics and machine learning, where $\boldsymbol{A}$ is a feature matrix, $\boldsymbol{x}$ a vector of model weights, and the ℓ_1 penalty performs automatic feature selection by driving most weights to exactly zero. The same mathematics that recovers a sparse signal trains a sparse, interpretable model, which is why Chapter 7 on proximal solvers is as much a machine learning chapter as a compressed sensing one.

Notes and References

Basis pursuit and the linear-programming reformulation are from Chen, Donoho, and Saunders [16]; the LASSO is Tibshirani's [47]. The geometry of the ℓ_1 ball as the engine of sparsity is developed in Candès and Wakin [13] and at length in Foucart and Rauhut [28]. The convex-optimization background, the simplex and interior-point methods, and Lagrangian duality behind the equivalence of the basis pursuit family are standard; see Boyd and Vandenberghe [7]. The support-independence theorem and the $\|\boldsymbol{x}^\star\|_0 \le m$ bound are classical facts about the vertices of a linear program; Theorem 5.5 is the statement that a basic feasible solution uses linearly independent columns.

Exercises

Exercise 5.1. Carry out the positive/negative split for $\boldsymbol{z} = (0, -3, 4, -1)^\top$: write $\boldsymbol{z}^+$, $\boldsymbol{z}^-$, verify $\boldsymbol{z} = \boldsymbol{z}^+ - \boldsymbol{z}^-$ and $\|\boldsymbol{z}\|_1 = \mathbf{1}^\top \boldsymbol{z}^+ + \mathbf{1}^\top \boldsymbol{z}^-$.

Exercise 5.2. For $\boldsymbol{A} = (1, 1) \in \mathbb{R}^{1\times 2}$ and $\boldsymbol{y} = 1$, write out the (P_1) linear program in the four variables $z_1^{\pm}, z_2^{\pm}$. Show every feasible point has ℓ_1 objective at least 1, and identify all minimizers. Is the (P_1) solution unique? Relate your answer to NSP of order 1 from Chapter 4.

Exercise 5.3. In two dimensions, take the constraint line $x_1 + 2x_2 = 2$. Find the ℓ_2-minimal and the ℓ_1-minimal points on this line by hand, and confirm that the ℓ_1 minimizer is sparse while the ℓ_2 minimizer is not. Sketch the two norm balls touching the line.

Exercise 5.4. Prove the easy half of Corollary 5.6 directly for a generic full-rank $\boldsymbol{A} \in \mathbb{R}^{m\times N}$: explain why a vertex of the feasible polytope of the (P_1) linear program has at most m nonzero coordinates.

Exercise 5.5. Show that BPDN with weight λ and the LASSO with budget τ have the same minimizer for a suitable correspondence $\lambda \leftrightarrow \tau$. *Hint: write the KKT conditions for each and match the multiplier on the LASSO constraint to λ.*

In the Problem Sets

Problem Set 2 (Chapter 21) implements exactly this chapter. You build the $[\boldsymbol{A} \mid -\boldsymbol{A}]$ linear program, solve it with a standard LP routine, and compare ℓ_1 recovery against the ℓ_2 (pseudoinverse) solution on the same measurements, watching the former return the sparse signal while the latter spreads energy across every coordinate. The accompanying notebook produces the time-sparse and frequency-sparse recovery plots.

PART III

Algorithms for Sparse Recovery

CHAPTER 6

Greedy Pursuit Algorithms

Greed, for lack of a better word, is good.

Gordon Gekko, Wall Street

Chapter 5 solved sparse recovery by convex optimization: turn (P_1) into a linear program and hand it to a solver. There is a second route, older and often much faster, that never forms a linear program at all. The *greedy* algorithms build up the support one or a few atoms at a time, each step picking the column that best explains what is left of the measurement. They trade some of the theoretical robustness of ℓ_1 minimization for speed and simplicity, and on the right kind of signal they recover the truth in exactly as many steps as there are nonzeros. This chapter develops the family in order of increasing sophistication: matching pursuit, its orthogonalized cousin, and the backtracking methods (subspace pursuit and CoSaMP) that can undo a bad early choice. Throughout, the engine is the synthesis identity $\boldsymbol{A}\boldsymbol{x} = \sum_j x_j \boldsymbol{a}_j$ from Chapter 3: recovery means finding which atoms are used and with what weights.

6.1 The Right Scan: Correlation

Every greedy method begins the same way: correlate the current residual against every column and keep the largest. This is not a heuristic; it is the exact solution of the one-sparse problem, and the derivation explains why correlation is the right thing to compute. Assume throughout that the columns are normalized, $\|\boldsymbol{a}_j\|_2 = 1$.

Suppose for a moment that $\boldsymbol{x}$ has a single nonzero entry, at an unknown index j. We seek the index and value that best explain $\boldsymbol{y}$:

$$\min_{j,\, x_j} \|x_j \boldsymbol{a}_j - \boldsymbol{y}\|_2^2.$$

For a fixed j, set the derivative in x_j to zero. Using $\|\boldsymbol{a}_j\|_2 = 1$,

$$\frac{d}{dx_j}\|x_j \boldsymbol{a}_j - \boldsymbol{y}\|_2^2 = 2x_j - 2\langle \boldsymbol{a}_j,\, \boldsymbol{y}\rangle = 0 \implies x_j^\star = \langle \boldsymbol{a}_j,\, \boldsymbol{y}\rangle .$$

Substituting back, the residual energy at the optimal coefficient is

$$\|x_j^\star \boldsymbol{a}_j - \boldsymbol{y}\|_2^2 = \|\boldsymbol{y}\|_2^2 - |\langle \boldsymbol{a}_j,\, \boldsymbol{y}\rangle|^2.$$

Minimizing the left side over j means *maximizing* the correlation magnitude on the right.

The atom-selection rule

The best single atom to explain a measurement is the one most correlated with it:

$$j^\star = \arg\max_{j \in [N]} |\langle \boldsymbol{a}_j,\, \boldsymbol{y}\rangle|, \qquad \text{with optimal coefficient } x_{j^\star}^\star = \langle \boldsymbol{a}_{j^\star},\, \boldsymbol{y}\rangle .$$

When the columns are nearly orthogonal, $\langle \boldsymbol{a}_j,\, \boldsymbol{y}\rangle \approx x_j$, so the largest correlation pinpoints the largest coefficient. Every algorithm in this chapter applies this rule to the current *residual* rather than to $\boldsymbol{y}$ itself, peeling off one atom's worth of signal at a time.

6.2 Matching Pursuit

The simplest greedy method, matching pursuit (MP), was introduced by Mallat and Zhang [37] as a signal-compression tool. It applies the selection rule, subtracts the chosen atom's contribution, and repeats.

Algorithm 1 Matching Pursuit (MP)

Require: sensing matrix $\boldsymbol{A}$ (unit-norm columns), measurement $\boldsymbol{y}$, tolerance ε

$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{0}, \quad \boldsymbol{r} \leftarrow \boldsymbol{y}$

repeat

  $n^\star \leftarrow \arg\max_n |\langle \boldsymbol{a}_n, \boldsymbol{r}\rangle|$ ▷ scan for the best atom

  $\alpha \leftarrow \langle \boldsymbol{a}_{n^\star}, \boldsymbol{r}\rangle$

  $\hat{x}_{n^\star} \leftarrow \hat{x}_{n^\star} + \alpha$ ▷ add a scalar to one coordinate

  $\boldsymbol{r} \leftarrow \boldsymbol{r} - \alpha\, \boldsymbol{a}_{n^\star}$ ▷ remove that atom's contribution

until $\|\boldsymbol{r}\|_2 < \varepsilon$ or $\|\boldsymbol{r}\|_2$ stops decreasing

return $\hat{\boldsymbol{x}}$

Each step removes exactly the component of the residual lying along the chosen atom. Because $\boldsymbol{a}_{n^\star}$ is unit-norm, the update is a vector projection, and the residual shrinks by a precise amount.

Proposition 6.1 (MP makes monotone progress). With $\alpha = \langle \boldsymbol{a}_{n^\star}, \boldsymbol{r}\rangle$ and the new residual $\boldsymbol{r}' = \boldsymbol{r} - \alpha \boldsymbol{a}_{n^\star}$,

$$\|\boldsymbol{r}'\|_2^2 = \|\boldsymbol{r}\|_2^2 - |\alpha|^2.$$

Proof. The vector $\boldsymbol{r}'$ is orthogonal to $\boldsymbol{a}_{n^\star}$, since $\langle \boldsymbol{a}_{n^\star}, \boldsymbol{r}'\rangle = \langle \boldsymbol{a}_{n^\star}, \boldsymbol{r}\rangle - \alpha\|\boldsymbol{a}_{n^\star}\|_2^2 = \alpha - \alpha = 0$. By the Pythagorean theorem applied to $\boldsymbol{r} = \boldsymbol{r}' + \alpha \boldsymbol{a}_{n^\star}$, $\|\boldsymbol{r}\|_2^2 = \|\boldsymbol{r}'\|_2^2 + |\alpha|^2$. ■

So the residual strictly decreases unless every column is orthogonal to it, in which case MP has already explained all of $\boldsymbol{y}$ that lies in the span of the chosen atoms. When the columns are orthogonal to begin with, this happens after exactly s steps and recovery is perfect.

Example 6.2 (MP on orthogonal columns: perfect recovery). Take $\boldsymbol{A} = \boldsymbol{I}_5$, so the atoms are the standard basis vectors, and $\boldsymbol{x} = (0, 7, 0, 0, -3)^\top$, giving $\boldsymbol{y} = \boldsymbol{x}$. The correlations are $(0, 7, 0, 0, -3)$; the largest magnitude is at index 2, so MP sets $\hat{x}_2 = 7$ and the residual becomes $(0, 0, 0, 0, -3)^\top$. The next scan gives largest magnitude at index 5, so $\hat{x}_5 = -3$ and the residual is $\boldsymbol{0}$. MP stops with $\hat{\boldsymbol{x}} = \boldsymbol{x}$, recovered in $s = 2$ steps. ♢

When the columns are *not* orthogonal, MP's one-coordinate-at-a-time update cannot correct itself, and the same atom may be revisited many times.

Example 6.3 (MP on non-orthogonal columns: the defect). Let $m = 3$, $N = 4$, with three orthonormal atoms and one diagonal atom:

$$\boldsymbol{a}_1 = (1, 0, 0)^\top,\ \boldsymbol{a}_2 = (0, 1, 0)^\top,\ \boldsymbol{a}_3 = (0, 0, 1)^\top,\ \boldsymbol{a}_4 = \tfrac{1}{\sqrt{3}}(1, 1, 1)^\top,$$

and true signal $\boldsymbol{x} = (0, 7, 0, -3)^\top$ on support $\{2, 4\}$, so $\boldsymbol{y} = 7\boldsymbol{a}_2 - 3\boldsymbol{a}_4 = (-\sqrt{3},\ 7 - \sqrt{3},\ -\sqrt{3})^\top$. The first scan correlates $\boldsymbol{y}$ against each atom:

$$\langle \boldsymbol{a}_1, \boldsymbol{y}\rangle = -\sqrt{3}, \quad \langle \boldsymbol{a}_2, \boldsymbol{y}\rangle = 7 - \sqrt{3} \approx 5.27, \quad \langle \boldsymbol{a}_3, \boldsymbol{y}\rangle = -\sqrt{3}, \quad \langle \boldsymbol{a}_4, \boldsymbol{y}\rangle = \tfrac{7}{\sqrt{3}} - 3 \approx 1.04.$$

The winner is index 2, so $\hat{x}_2 = 7 - \sqrt{3} \approx 5.27$, already wrong: the true value is 7. MP undershoots because part of coordinate 2 is also explained by $\boldsymbol{a}_4$, which it has not yet selected. The residual is $(-\sqrt{3}, 0, -\sqrt{3})^\top$, the next scan picks index 4 with coefficient -2, and the residual norm drops to $\sqrt{2}$ but does not vanish. MP then oscillates between indices 2 and 4, correcting each by ever smaller amounts and converging to the truth only in the limit. ♢

The defect is structural: once MP moves on, it never revisits and *corrects* the coefficient on an atom in light of later choices. The fix is to re-fit all chosen coefficients jointly at every step, which is orthogonal matching pursuit.

6.3 Orthogonal Projection

The joint re-fit is an orthogonal projection, the tool that distinguishes OMP from MP and reappears in every backtracking method. Given a support set S and the submatrix $\boldsymbol{A}_S$, the best approximation of $\boldsymbol{y}$ from the span of those columns is

$$\boldsymbol{y}_p = \operatorname*{arg\,min}_{\boldsymbol{v} \in \operatorname{span}\{\boldsymbol{A}_S\}} \|\boldsymbol{v} - \boldsymbol{y}\|_2.$$

Figure 6.1 shows the geometry: $\boldsymbol{y}_p$ is the shadow of $\boldsymbol{y}$ on the subspace, and the residual $\boldsymbol{y}_r = \boldsymbol{y} - \boldsymbol{y}_p$ stands perpendicular to it.

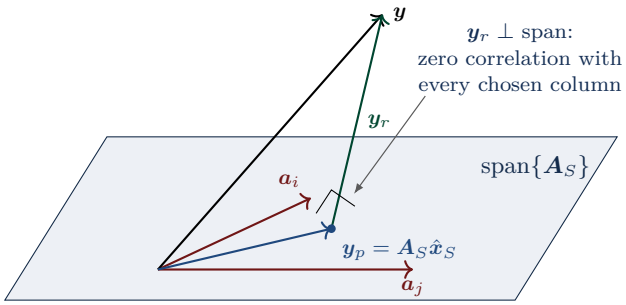


Figure 6.1. Orthogonal projection of $\boldsymbol{y}$ onto the span of the selected columns. The shadow $\boldsymbol{y}_p$ is the best the chosen atoms can do; the residual $\boldsymbol{y}_r = \boldsymbol{y} - \boldsymbol{y}_p$ stands perpendicular to that span, so it has zero correlation with every column already chosen, which forces the next greedy scan to pick a genuinely new atom.

Proposition 6.4 (Projection formula and orthogonality). If $\boldsymbol{A}_S$ has full column rank, the best coefficients and the projection are

$$\hat{\boldsymbol{x}}_S = \boldsymbol{A}_S^{\dagger}\boldsymbol{y} = (\boldsymbol{A}_S^*\boldsymbol{A}_S)^{-1}\boldsymbol{A}_S^*\boldsymbol{y}, \qquad \boldsymbol{y}_p = \boldsymbol{A}_S\hat{\boldsymbol{x}}_S,$$

and the residual $\boldsymbol{y}_r = \boldsymbol{y} - \boldsymbol{y}_p$ satisfies $\boldsymbol{A}_S^*\boldsymbol{y}_r = \boldsymbol{0}$.

Proof. The coefficient formula is the least-squares solution of Chapter 2. For orthogonality, substitute:

$$\boldsymbol{A}_S^*\boldsymbol{y}_r = \boldsymbol{A}_S^*\boldsymbol{y} - \boldsymbol{A}_S^*\boldsymbol{A}_S(\boldsymbol{A}_S^*\boldsymbol{A}_S)^{-1}\boldsymbol{A}_S^*\boldsymbol{y} = \boldsymbol{A}_S^*\boldsymbol{y} - \boldsymbol{A}_S^*\boldsymbol{y} = \boldsymbol{0}. \qquad \blacksquare$$

The orthogonality $\boldsymbol{A}_S^*\boldsymbol{y}_r = \boldsymbol{0}$ is the crucial property: after a joint re-fit, the residual has *zero* correlation with every already-selected column, so the next scan is forced to choose a genuinely new atom.

6.4 Orthogonal Matching Pursuit

Orthogonal matching pursuit (OMP), due to Pati, Rezaiifar, and Krishnaprasad [43], is matching pursuit with the scalar update replaced by the projection of Proposition 6.4.

The only change from MP is line 5: instead of nudging one coordinate, OMP solves a least-squares problem over the entire current support. By Proposition 6.4, the residual is then orthogonal to every selected column, which has two consequences. First, the next scan cannot re-select an old index, since its correlation is exactly zero, so OMP picks a fresh atom every iteration. Second, OMP terminates in at most $\min(m, N)$ iterations, because it can add at most that many independent directions. MP enjoys neither guarantee.

Example 6.5 (OMP on the same signal: exact recovery). Return to the matrix and signal of Example 6.3, where MP only oscillated. The first iteration selects index 2 (same scan) and, with a one-column support, re-fits $\hat{x}_2 = 7 - \sqrt{3}$ and residual $(-\sqrt{3}, 0, -\sqrt{3})^{\top}$, identical to MP so far. The second scan selects index 4, giving $S = \{2, 4\}$ and

$$\boldsymbol{A}_S = \begin{pmatrix} 0 & 1/\sqrt{3} \\ 1 & 1/\sqrt{3} \\ 0 & 1/\sqrt{3} \end{pmatrix}, \qquad \boldsymbol{A}_S^*\boldsymbol{A}_S = \begin{pmatrix} 1 & 1/\sqrt{3} \\ 1/\sqrt{3} & 1 \end{pmatrix}, \quad (\boldsymbol{A}_S^*\boldsymbol{A}_S)^{-1} = \tfrac{3}{2}\begin{pmatrix} 1 & -1/\sqrt{3} \\ -1/\sqrt{3} & 1 \end{pmatrix}.$$

Algorithm 2 Orthogonal Matching Pursuit (OMP)

Require: $\boldsymbol{A}$ (unit-norm columns), $\boldsymbol{y}$, tolerance ε

$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{0}, \quad \boldsymbol{r} \leftarrow \boldsymbol{y}, \quad S \leftarrow \emptyset$
repeat
  $n^\star \leftarrow \arg\max_n |\langle \boldsymbol{a}_n, \boldsymbol{r} \rangle|$ ▷ same scan as MP
  $S \leftarrow S \cup \{n^\star\}$ ▷ augment the support
  $\hat{\boldsymbol{x}}_S \leftarrow \boldsymbol{A}_S^\dagger \boldsymbol{y}, \ \hat{\boldsymbol{x}}_{S^c} \leftarrow \boldsymbol{0}$ ▷ re-fit *all* chosen coefficients
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}$
until $\|\boldsymbol{r}\|_2 < \varepsilon$
return $\hat{\boldsymbol{x}}$

With $\boldsymbol{A}_S^* \boldsymbol{y} = (\,7 - \sqrt{3}, \ \tfrac{7}{\sqrt{3}} - 3\,)^\top$, the re-fit is

$$\hat{\boldsymbol{x}}_S = (\boldsymbol{A}_S^* \boldsymbol{A}_S)^{-1} \boldsymbol{A}_S^* \boldsymbol{y} = \tfrac{3}{2} \begin{pmatrix} 1 & -1/\sqrt{3} \\ -1/\sqrt{3} & 1 \end{pmatrix} \begin{pmatrix} 7 - \sqrt{3} \\ \tfrac{7}{\sqrt{3}} - 3 \end{pmatrix} = \begin{pmatrix} 7 \\ -3 \end{pmatrix}.$$

The top entry is $\tfrac{3}{2}\big[(7-\sqrt{3}) - \tfrac{1}{\sqrt{3}}(\tfrac{7}{\sqrt{3}} - 3)\big] = \tfrac{3}{2}\big[7 - \sqrt{3} - \tfrac{7}{3} + \sqrt{3}\big] = \tfrac{3}{2} \cdot \tfrac{14}{3} = 7$, and the bottom entry is $\tfrac{3}{2}(-2) = -3$. So $\hat{\boldsymbol{x}} = (0, 7, 0, -3)^\top = \boldsymbol{x}$ and the residual vanishes: OMP recovers exactly in $s = 2$ iterations, fixing the very atom that MP could only approach. ◊

Remark 6.6 *(Hiring without firing).* A useful picture casts $\boldsymbol{y}$ as a company's needs, the atoms as job applicants, and the support as the team you hire. OMP interviews sequentially: each round it hires the applicant who best fits the unmet needs, then re-balances everyone's workload (the projection) and reassesses what remains (the residual). Its weakness is that it can never *fire*. An atom selected early, perhaps by bad luck, stays on the support forever, even if later evidence shows it was a poor choice. The backtracking algorithms of Section 6.6 are built to fire.

6.5 When OMP Succeeds

OMP recovers exactly when, at every step, some true atom outscores every false one on correlation with the residual. Tropp [48] made this precise.

Theorem 6.7 (Exact recovery condition). Let $\boldsymbol{x}$ be s-sparse with support $S^\star$ and $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. If $\boldsymbol{A}_{S^\star}$ has full column rank and the *exact recovery condition*

$$\max_{j \notin S^\star} \|\boldsymbol{A}_{S^\star}^\dagger \boldsymbol{a}_j\|_1 \ < \ 1$$

holds, then OMP recovers $\boldsymbol{x}$ in exactly s iterations.

The condition says that no outside atom $\boldsymbol{a}_j$ can be written as too strong a combination of the true atoms; the true support "dominates" the correlation scan at every step. Like the null space property of Chapter 4, the exact recovery condition is hard to check directly. It is implied by a simple bound on the mutual coherence, which is the subject of Chapter 8.

Coherence guarantee for OMP (preview)

If the mutual coherence satisfies $\mu(\boldsymbol{A}) < \frac{1}{2s-1}$, then both the exact recovery condition and the spark condition $\operatorname{spark}(\boldsymbol{A}) \geq 2s + 1$ hold, so OMP *and* (P_1) recover every s-sparse signal. We prove this in Chapter 8.

OMP's strict greediness is also its failure mode. If at some step every remaining true atom is outscored by a false one, OMP hires the false atom and can never undo it: it ends with a support that misses a true index and keeps a spurious one. On a signal whose nonzeros are all equal in magnitude, where the first scan faces a near tie, this is a real risk, and it motivates the next family of methods.

6.6 Backtracking: Subspace Pursuit and CoSaMP

The backtracking algorithms keep the projection idea but add the ability to discard atoms. They select a whole batch each iteration, re-fit, and then *prune* back to the s best, so an early mistake can be corrected later. The pruning step is a single clean operator.

Definition 6.8 (Hard-thresholding operator). For $\boldsymbol{v} \in \mathbb{C}^N$ and $1 \le s \le N$, the **hard-thresholding operator** $\mathcal{H}_s(\boldsymbol{v})$ keeps the s entries of $\boldsymbol{v}$ largest in magnitude and sets the rest to zero (ties broken by any fixed rule).

Example 6.9 (Hard thresholding). For $\boldsymbol{v} = (1,\, 0.1,\, 4,\, 0.04,\, -6)^\top$, the magnitudes are $(1, 0.1, 4, 0.04, 6)$, so $\mathcal{H}_1(\boldsymbol{v}) = (0,0,0,0,-6)^\top$, $\mathcal{H}_2(\boldsymbol{v}) = (0,0,4,0,-6)^\top$, and $\mathcal{H}_3(\boldsymbol{v}) = (1,0,4,0,-6)^\top$. ◊

The operator is exactly best s-sparse approximation: for any $p \ge 1$, $\mathcal{H}_s(\boldsymbol{v}) = \arg\min_{\|\boldsymbol{w}\|_0 \le s} \|\boldsymbol{v} - \boldsymbol{w}\|_p$, since discarding a small entry costs less than discarding a large one. This is the same principle that lets JPEG drop all but the few largest transform coefficients of an image with no visible loss.

Subspace pursuit (SP), due to Dai and Milenkovic [17], and CoSaMP, due to Needell and Tropp [40], share one structure and differ in a single constant.

Algorithm 3 Subspace Pursuit (SP) and CoSaMP

Require: $\boldsymbol{A}$, $\boldsymbol{y}$, sparsity s, tolerance ε; batch size $b = s$ for SP, $b = 2s$ for CoSaMP

$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{0}$, $\boldsymbol{r} \leftarrow \boldsymbol{y}$, $S \leftarrow \emptyset$
repeat
  $T \leftarrow \operatorname{supp}\big(\mathcal{H}_b(\boldsymbol{A}^*\boldsymbol{r})\big)$ ▷ top b correlations
  $\widetilde{S} \leftarrow S \cup T$ ▷ grow the support
  $\boldsymbol{z}_{\widetilde{S}} \leftarrow \boldsymbol{A}_{\widetilde{S}}^{\dagger}\boldsymbol{y}$, $\boldsymbol{z}_{\widetilde{S}^c} \leftarrow \boldsymbol{0}$ ▷ re-fit everybody
  $\hat{\boldsymbol{x}} \leftarrow \mathcal{H}_s(\boldsymbol{z})$ ▷ prune to the best s: this fires atoms
  $S \leftarrow \operatorname{supp}(\hat{\boldsymbol{x}})$, $\boldsymbol{r} \leftarrow \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}$
until $\|\boldsymbol{r}\|_2 < \varepsilon$ or the support stabilizes
return $\hat{\boldsymbol{x}}$

The three lines that matter are *grow*, *re-fit*, *prune*. Growing adds a batch of candidates to the current support; re-fitting gives every candidate an honest least-squares coefficient; pruning keeps only the s largest, so any atom whose re-fit coefficient came out small is dropped. That prune is the firing step OMP lacks: an index present after one iteration can be gone after the next.

Example 6.10 (Backtracking repairs a bad pick). Suppose $N = 10$ with true support $\{1,2,3\}$ and target sparsity $s = 3$. If the first correlation scan happens to rank a spurious index 5 among its top three, SP grows to $\{1,5,6\}$, re-fits, and at first keeps them. But the residual still carries the unexplained atoms $\boldsymbol{a}_2, \boldsymbol{a}_3$, so the next scan surfaces indices 2 and 3. Growing to $\{1,2,3,5,6\}$ and re-fitting lets the genuine atoms claim their coefficients while the spurious $5, 6$ collapse toward zero; the prune $\mathcal{H}_3$ then keeps exactly $\{1,2,3\}$ and discards the impostors. The bad early pick is undone, which OMP could never do. ◊

The difference between the two algorithms is the batch size b. SP grows by the top s correlations, so its working support has size up to $2s$; CoSaMP grows by the top $2s$, working support up to $3s$. The wider net gives each true atom more chances to survive a single scan, which buys CoSaMP cleaner convergence guarantees at a modestly higher per-iteration cost. A close relative, stagewise OMP (StOMP) [22], replaces the fixed batch size with a correlation threshold and admits however many atoms clear it, adapting the batch to the data.

6.7 Choosing an Algorithm

The methods are not strictly ordered; the best choice depends on the signal and the matrix. A useful contrast is between a *decaying* signal, whose nonzero magnitudes fall off in a clear ranking, and a *flat* signal, whose nonzeros are all about equal.

On a decaying signal, OMP shines: its sequential scan picks the largest atom first, then the next, matching the signal's own ranking, and it rarely makes a tie-breaking mistake. On a flat signal the first scan is a near tie among the true atoms, and OMP's commit-and-never-revise rule can lock in an error; the batch selection and pruning of SP or CoSaMP are more robust there. When the signal is unknown, noisy, or the cost of a wrong support is high, ℓ_1 minimization of Chapter 5 is the safe default, since it solves a global convex problem with no greedy commitments to regret. Table 6.1 summarizes.

Method	Update	Best suited to
MP	one scalar coordinate per step	very large dictionaries, cheap steps
OMP	least-squares re-fit on a growing support	decaying signals; the standard choice
SP / CoSaMP	grow, re-fit, prune (backtracking)	flat signals; known sparsity s
ℓ_1 (basis pursuit)	global convex program	noisy or unknown signals; robustness

Table 6.1. The recovery toolbox. Greedy methods trade robustness for speed; OMP is the default greedy choice, the backtracking methods handle flat signals, and ℓ_1 is the conservative option.

Remark 6.11 *(Phase transitions).* All of these methods exhibit a sharp *phase transition*. Plotting the undersampling ratio $\delta = m/N$ against the sparsity ratio $\rho = s/m$ and shading by empirical recovery probability reveals a thin curve separating a region of near-certain success from one of near-certain failure (Fig. 6.2). For Gaussian matrices and ℓ_1 recovery, Donoho and Tanner computed this curve analytically, and it agrees with experiment to several digits. The practical reading is the one we have met before: success sets in once $m \gtrsim s \log(N/s)$, the bound proved in Chapter 9.

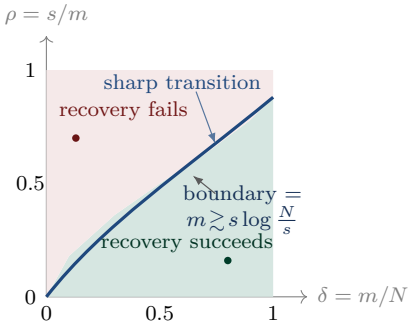


Figure 6.2. The Donoho–Tanner phase transition for ℓ_1 recovery from a Gaussian matrix. In the plane of undersampling $\delta = m/N$ against relative sparsity $\rho = s/m$, a sharp curve separates near-certain recovery (below, green dot) from near-certain failure (above, red dot). The boundary is the geometric form of the $m \gtrsim s \log(N/s)$ law; all the methods of this chapter share a transition of this shape.

Notes and References

Matching pursuit is from Mallat and Zhang [37]; orthogonal matching pursuit from Pati, Rezaiifar, and Krishnaprasad [43], with the exact-recovery analysis and the condition of Theorem 6.7 due to Tropp [48] and, for random matrices, Tropp and Gilbert [49]. Subspace pursuit is Dai and Milenkovic [17]; CoSaMP is Needell and Tropp [40], which also proves the stable recovery guarantees that make the backtracking methods competitive with ℓ_1 minimization. The hard-thresholding operator and best s-term approximation are standard; the same operator drives the iterative hard-thresholding algorithm of Chapter 7. The phase-transition picture is from Donoho and Tanner. Foucart and Rauhut [28] give a unified account of all of these algorithms.

Exercises

Exercise 6.1. Verify Proposition 6.1 numerically on the first step of Example 6.3: compute α, the new residual, and confirm $\|\boldsymbol{r}'\|_2^2 = \|\boldsymbol{y}\|_2^2 - |\alpha|^2$.

Exercise 6.2. Run two iterations of MP by hand on $\boldsymbol{A} = \boldsymbol{I}_4$ with $\boldsymbol{x} = (5, 0, -2, 0)^\top$ and confirm exact recovery. Why does MP succeed here without the OMP re-fit?

Exercise 6.3. Carry out the OMP re-fit of Example 6.5 starting instead from the wrong first pick $S = \{4\}$ (pretend the scan chose index 4). Does OMP still recover $\boldsymbol{x}$ once it later adds index 2? Explain in terms of Proposition 6.4.

Exercise 6.4. Compute $\mathcal{H}_2(\boldsymbol{v})$ and $\mathcal{H}_3(\boldsymbol{v})$ for $\boldsymbol{v} = (-3, 2, 2, -1, 4)^\top$, and resolve the tie between the two entries of magnitude 2 by your chosen rule. Confirm $\mathcal{H}_2(\boldsymbol{v})$ is the best 2-sparse approximation in ℓ_2.

Exercise 6.5. Explain, using the grow/re-fit/prune structure of Algorithm 3, why a true atom that is outscored on one iteration's correlation scan can still be recovered by SP on a later iteration, whereas OMP that misses it once never recovers it.

In the Problem Sets

Problem Set 3 (Chapter 22) is a head-to-head implementation of OMP, subspace pursuit, and CoSaMP on the same measurements. You trace each algorithm's support as it evolves, compare exact-recovery rates as the sparsity s grows toward the measurement count, and watch the backtracking methods rescue cases where OMP locks in a wrong atom. The accompanying notebook records the per-algorithm recovery probability and runtime as the problem size varies.

CHAPTER 7

Proximal and Iterative Solvers

When you cannot differentiate, optimize a little instead.

Chapter 5 reduced (P_1) to a linear program, and Chapter 6 built combinatorial solvers. There is a third family, the one most used in practice on large problems, that keeps the convexity of the ℓ_1 formulation but solves it by simple iteration: each step takes a gradient step on the smooth data-fidelity term and then applies a cheap nonlinear cleanup. The cleanup is a *proximal operator*, and in the cases we need it is nothing more than thresholding. This chapter develops the proximal operator from scratch, computes the three instances that matter, and assembles them into the three workhorse algorithms of sparse recovery: iterative hard thresholding, ISTA, and its accelerated form FISTA. All three are one template with one piece swapped, which is the unifying idea of the chapter.

7.1 The Obstacle: A Kink With No Gradient

The problem we want to solve is the unconstrained ℓ_1-regularized least squares of Chapter 5, the basis pursuit denoising objective

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} F(\boldsymbol{x}), \qquad F(\boldsymbol{x}) = \underbrace{\tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2}_{g(\boldsymbol{x})} + \underbrace{\lambda\|\boldsymbol{x}\|_1}_{h(\boldsymbol{x})}. \tag{7.1}$$

It splits into a smooth convex part g and a nonsmooth convex part h. The smooth part is friendly: its gradient is $\nabla g(\boldsymbol{x}) = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y})$, the workhorse gradient of Chapter 2. The trouble is h. The ℓ_1 norm has a kink at every coordinate axis, and at a kink there is no gradient, so we cannot simply run gradient descent on F. Plain gradient descent stalls exactly where the answer wants to live, on the coordinate planes where entries are zero, and those zeros are the whole point of sparse recovery.

The way around the kink is to handle the smooth and nonsmooth parts differently: take an ordinary gradient step on g, then apply a special operator for h that needs no gradient. That operator is the proximal operator.

7.2 The Proximal Operator

Definition 7.1 (Proximal operator). For a convex function $f : \mathbb{R}^N \to \mathbb{R}\cup\{+\infty\}$ and a parameter $\lambda > 0$, the **proximal operator** of f at a point $\boldsymbol{v}$ is

$$\operatorname{prox}_{\lambda f}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}\in\mathbb{R}^N} \left[f(\boldsymbol{x}) + \frac{1}{2\lambda}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \right]. \tag{7.2}$$

The two terms compete. The first pulls toward the minimizer of f; the second is a quadratic leash penalizing departure from the anchor $\boldsymbol{v}$. The prox returns the best compromise: reduce f, but do not stray far from where you started. The parameter λ sets the leash length. Small λ makes the penalty severe and the prox barely moves; large λ slackens the leash and lets the prox travel to make f small. This is the same role λ plays as a step size, and the resemblance is not an accident: the prox is a *generalized gradient step*. An ordinary step returns $\boldsymbol{v} - \lambda\nabla f(\boldsymbol{v})$, moving downhill on f;

the prox also moves to reduce f, but by solving a small regularized problem rather than following a gradient, which is exactly why it works where no gradient exists.

Proposition 7.2 (The prox is well defined). If f is convex, proper, and lower semicontinuous and $\lambda > 0$, then Eq. (7.2) has a unique minimizer, so $\operatorname{prox}_{\lambda f}$ is a single-valued map.

Proof. Write $\Phi(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{2\lambda}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2$. The quadratic term is $\frac{1}{\lambda}$-strongly convex (its Hessian is $\frac{1}{\lambda}\boldsymbol{I}$), and adding the convex f preserves strong convexity, so Φ is $\frac{1}{\lambda}$-strongly convex. Strong convexity forces coercivity, $\Phi(\boldsymbol{x}) \to +\infty$ as $\|\boldsymbol{x}\| \to \infty$, so a proper lower semicontinuous Φ attains its infimum on a large enough ball. For uniqueness, if $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ both attained the minimum value $\Phi^\star$, strong convexity at the midpoint would give $\Phi\big(\frac{\boldsymbol{x}_1+\boldsymbol{x}_2}{2}\big) \leq \Phi^\star - \frac{1}{8\lambda}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2 < \Phi^\star$, a value below the minimum, which is impossible. ∎

This is exactly what plain gradient descent lacked at the kinks. Gradient descent needs a gradient; the prox needs only the ability to solve Eq. (7.2), which the added quadratic guarantees has one and only one answer. We now compute that answer in the three cases the subject needs.

7.2.1 Three proximal operators

The zero function gives the identity. With $f = 0$ there is nothing to minimize but the leash, and the point nearest $\boldsymbol{v}$ is $\boldsymbol{v}$ itself:

$$\operatorname{prox}_{\lambda \cdot 0}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 = \boldsymbol{v}.$$

With no objective pulling it away, the prox stays home. This is the degenerate case that recovers ordinary gradient descent.

An indicator gives a projection. For a closed convex set C, let ι_C be its *indicator*, equal to 0 on C and $+\infty$ off it. Then

$$\operatorname{prox}_{\lambda \iota_C}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x} \in C} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 = \operatorname{proj}_C(\boldsymbol{v}),$$

the Euclidean projection onto C: outside C the cost is infinite, so the minimizer lies in C, and there the leash alone is minimized by the closest point. This identifies projected gradient descent as a proximal method, and it is the link to iterative hard thresholding in Section 7.3, where C is the set of sparse vectors.

The ℓ_1 norm gives soft thresholding. The case the chapter is built around: $f = \|\cdot\|_1$. Both terms of Eq. (7.2) are sums over coordinates,

$$\|\boldsymbol{x}\|_1 + \frac{1}{2\lambda}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 = \sum_{i=1}^{N} \Big[|x_i| + \frac{1}{2\lambda}(x_i - v_i)^2 \Big],$$

so the prox acts one coordinate at a time. Fix a coordinate and minimize $\phi(x) = |x| + \frac{1}{2\lambda}(x - v)^2$, strictly convex with a unique minimizer. Away from the kink, $\phi'(x) = \operatorname{sgn}(x) + \frac{1}{\lambda}(x - v)$. If $v > \lambda$, the guess $x > 0$ gives $x = v - \lambda > 0$, consistent. If $v < -\lambda$, the guess $x < 0$ gives $x = v + \lambda < 0$, consistent. Otherwise neither stationary point is consistent and the minimizer sits at the kink $x = 0$; the subgradient test confirms it, since $0 \in \partial\phi(0) = [-1, 1] - \frac{v}{\lambda}$ exactly when $|v| \leq \lambda$. Collecting the cases,

$$\big[\operatorname{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{v})\big]_i = \begin{cases} v_i - \lambda, & v_i > \lambda, \\ 0, & |v_i| \leq \lambda, \\ v_i + \lambda, & v_i < -\lambda, \end{cases} \quad = \operatorname{sgn}(v_i)\,\max(|v_i| - \lambda,\, 0).$$

Soft thresholding is the ℓ_1 prox

For $\boldsymbol{v} \in \mathbb{R}^N$ and $\lambda > 0$, $\operatorname{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{v}) = \mathcal{S}_\lambda(\boldsymbol{v})$, where

$$\big[\mathcal{S}_\lambda(\boldsymbol{v})\big]_i = \operatorname{sgn}(v_i)\,\max(|v_i| - \lambda,\, 0).$$

Soft thresholding does two things at once: it *kills* every coordinate of magnitude at most λ, producing sparsity, and it *shrinks* every survivor toward zero by exactly λ. It costs $O(N)$ with no linear system to solve.

Example 7.3 (Soft thresholding by hand). With $\lambda = 1$ and $\boldsymbol{v} = (3, -0.5, 2, -4)^\top$, threshold each entry: $3 \mapsto 2$, $-0.5 \mapsto 0$, $2 \mapsto 1$, $-4 \mapsto -3$, so $\mathcal{S}_1(\boldsymbol{v}) = (2, 0, 1, -3)^\top$. The small entry is killed, the survivors shrink by one. ◊

7.2.2 Soft versus hard thresholding

The two thresholding operators differ exactly as ℓ_1 differs from ℓ_0. Hard thresholding $\mathcal{H}_s$ from Chapter 6 keeps the s largest entries *untouched* and zeros the rest; it is the projection onto the (nonconvex) set of s-sparse vectors. Soft thresholding $\mathcal{S}_\lambda$ keeps the entries above a magnitude threshold but *shrinks* them; it is the prox of the convex ℓ_1 norm. Figure 7.1 shows the two transfer functions: hard thresholding is the identity above the threshold with a jump, soft thresholding is continuous but biased downward by λ.

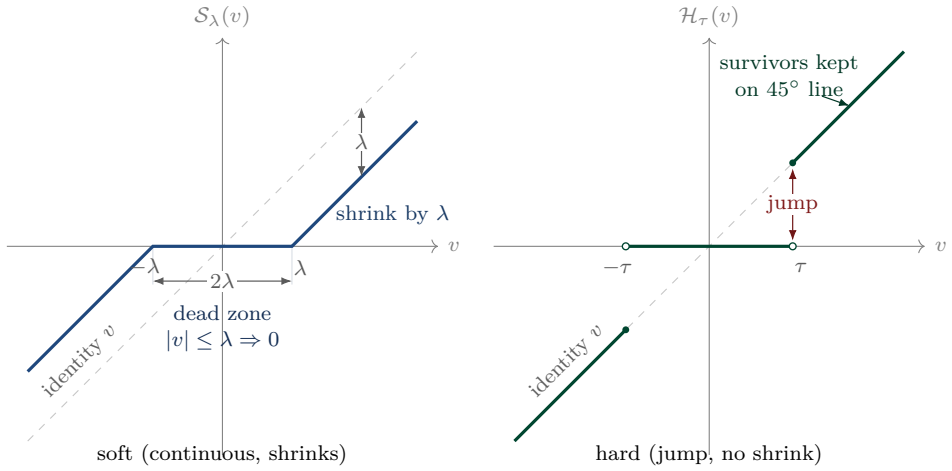


Figure 7.1. Thresholding transfer functions, both compared to the dashed 45° identity. Soft thresholding (the ℓ_1 prox, used by ISTA) is continuous: it kills the dead zone $|v| \le \lambda$ and pulls every survivor toward zero by exactly λ, so its arms run parallel to the identity but below it. Hard thresholding (projection onto the sparse set, used by IHT) leaves survivors on the identity but jumps discontinuously at the threshold τ.

Algorithm 4 Iterative Hard Thresholding (IHT)

Require: $\boldsymbol{A}$, $\boldsymbol{y}$, sparsity s, step size μ (with $\mu \le 1/\|\boldsymbol{A}\|_{2\to 2}^2$)
1: $\boldsymbol{x} \leftarrow \boldsymbol{0}$
2: **repeat**
3: $\quad \boldsymbol{x} \leftarrow \mathcal{H}_s\big(\boldsymbol{x} - \mu\, \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})\big)$ ▷ gradient step, then project onto Σ_s
4: **until** converged
5: **return** $\boldsymbol{x}$

7.3 Iterative Hard Thresholding

The simplest iterative method attacks the ℓ_0-constrained problem $\min\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ subject to $\|\boldsymbol{x}\|_0 \le s$ directly. It alternates a gradient step on the data-fidelity term with a projection back onto the sparse set, and that projection is hard thresholding.

The single line is the gradient-step template of Chapter 2 with hard thresholding as the cleanup: descend on $\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2$, then keep only the s largest entries. Because $\mathcal{H}_s$ is the projection onto the sparse set Σ_s, IHT is projected gradient descent for the sparsity constraint. The step-size condition $\mu \le 1/\|\boldsymbol{A}\|_{2\to 2}^2$ keeps the gradient step from overshooting, and under a restricted isometry assumption (Chapter 8) IHT converges to the true s-sparse signal at a geometric rate.

Example 7.4 (One IHT step). Let $\boldsymbol{A} = \boldsymbol{I}_4$, so $\|\boldsymbol{A}\|_{2\to 2} = 1$ and $\mu = 1$ is allowed, and let $\boldsymbol{y} = (0, 3, 0, -2)^\top$ with target sparsity $s = 2$. Starting from $\boldsymbol{x} = \boldsymbol{0}$, the gradient step is $\boldsymbol{0} - 1 \cdot (\boldsymbol{0} - \boldsymbol{y}) = \boldsymbol{y} = (0, 3, 0, -2)^\top$, and $\mathcal{H}_2$ keeps the two largest entries, which are already only two, giving $\boldsymbol{x} = (0, 3, 0, -2)^\top = \boldsymbol{y}$. With $\boldsymbol{A} = \boldsymbol{I}$ recovery is immediate; the interest is that the same two lines recover sparse signals from genuinely underdetermined $\boldsymbol{A}$, one threshold at a time. ◊

Algorithm 5 ISTA (Iterative Shrinkage-Thresholding Algorithm)

Require: $\boldsymbol{A}$, $\boldsymbol{y}$, penalty λ, $L = \|\boldsymbol{A}\|_{2\to 2}^2$

$\boldsymbol{x} \leftarrow \boldsymbol{0}$

repeat

  $\boldsymbol{x} \leftarrow \mathcal{S}_{\lambda/L}\big(\boldsymbol{x} - \frac{1}{L}\,\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})\big)$ ▷ gradient step, then soft-threshold

until converged

return $\boldsymbol{x}$

7.4 ISTA: Iterative Soft Thresholding

For the convex problem Eq. (7.1), the same template uses the ℓ_1 prox, soft thresholding, in place of hard thresholding. The resulting algorithm is the iterative shrinkage-thresholding algorithm (ISTA) of Daubechies, Defrise, and De Mol [18]. It is best derived by majorization-minimization, which explains both the form of the update and the step size.

The smooth part $g(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ has a Lipschitz gradient with constant $L = \|\boldsymbol{A}\|_{2\to 2}^2$, meaning $\|\nabla g(\boldsymbol{x}) - \nabla g(\boldsymbol{x}')\|_2 \le L\|\boldsymbol{x} - \boldsymbol{x}'\|_2$. This yields the quadratic upper bound, valid for all $\boldsymbol{x}$ and any anchor $\boldsymbol{x}^k$,

$$g(\boldsymbol{x}) \;\le\; g(\boldsymbol{x}^k) + \big\langle \nabla g(\boldsymbol{x}^k),\, \boldsymbol{x} - \boldsymbol{x}^k \big\rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}^k\|_2^2.$$

Replacing g in Eq. (7.1) by this majorant and minimizing the result gives the next iterate. Completing the square turns the majorant plus $\lambda\|\boldsymbol{x}\|_1$ into

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}} \; \frac{L}{2}\Big\|\boldsymbol{x} - \big(\boldsymbol{x}^k - \tfrac{1}{L}\nabla g(\boldsymbol{x}^k)\big)\Big\|_2^2 + \lambda\|\boldsymbol{x}\|_1 = \operatorname{prox}_{(\lambda/L)\|\cdot\|_1}\Big(\boldsymbol{x}^k - \tfrac{1}{L}\nabla g(\boldsymbol{x}^k)\Big),$$

which by the keybox above is soft thresholding of a gradient step.

ISTA is identical in shape to IHT; only the cleanup differs, soft thresholding for the convex ℓ_1 problem versus hard thresholding for the ℓ_0 constraint. Because the majorant decreases the objective at every step, ISTA is a descent method, and it converges at the rate $F(\boldsymbol{x}^k) - F^\star = O(1/k)$.

One template, three algorithms

Every method of this chapter has the form

$$\boldsymbol{x}^{k+1} = \big(\text{cleanup}\big)\Big(\boldsymbol{x}^k - \tfrac{1}{L}\,\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y})\Big),$$

where the gradient step is always the same and only the cleanup changes: the identity for plain gradient descent, $\mathcal{H}_s$ for IHT, $\mathcal{S}_{\lambda/L}$ for ISTA. Learn the template once and the algorithms follow.

7.5 FISTA: Adding Momentum

ISTA's $O(1/k)$ rate is slow: halving the error takes twice as many iterations. Beck and Teboulle [4] accelerated it to $O(1/k^2)$ with Nesterov momentum, at essentially no extra cost per iteration. The fast algorithm (FISTA) performs the soft-thresholding step not at the current iterate but at an extrapolated point that carries momentum from the previous step.

The momentum term $\frac{t-1}{t^+}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k)$ nudges the next gradient step in the direction the iterates have been moving, much as a heavy ball rolling downhill overshoots a local wiggle. The growing sequence t tunes how much momentum to carry. The effect is a provable improvement from $O(1/k)$ to $O(1/k^2)$, so reaching a target accuracy takes the square root of the iterations, a large saving on the ill-conditioned problems typical of compressed sensing. Table 7.1 collects the three iterative methods.

The difference between the two rates is dramatic over a run (Fig. 7.2). To reach an error of 10^{-4}, ISTA's $O(1/k)$ rate needs on the order of 10^4 iterations, while FISTA's $O(1/k^2)$ needs only about 10^2, a hundredfold saving for the

Algorithm 6 FISTA (Fast ISTA)

Require: $\boldsymbol{A}$, $\boldsymbol{y}$, penalty λ, $L = \|\boldsymbol{A}\|_{2\to 2}^2$

$\boldsymbol{x}^0 \leftarrow \boldsymbol{0}$, $\boldsymbol{w} \leftarrow \boldsymbol{x}^0$, $t \leftarrow 1$

for $k = 0, 1, 2, \ldots$ until converged **do**

  $\boldsymbol{x}^{k+1} \leftarrow \mathcal{S}_{\lambda/L}\big(\boldsymbol{w} - \frac{1}{L}\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{w} - \boldsymbol{y})\big)$ ▷ ISTA step at the extrapolated point

  $t^+ \leftarrow \frac{1}{2}\big(1 + \sqrt{1 + 4t^2}\big)$

  $\boldsymbol{w} \leftarrow \boldsymbol{x}^{k+1} + \dfrac{t-1}{t^+}\big(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\big)$ ▷ extrapolate with momentum

  $t \leftarrow t^+$

end for

return $\boldsymbol{x}$

Method	Cleanup after the gradient step	Solves	Rate
IHT	$\mathcal{H}_s$ (hard threshold)	ℓ_0-constrained	geometric (under RIP)
ISTA	$\mathcal{S}_{\lambda/L}$ (soft threshold)	ℓ_1 (LASSO)	$O(1/k)$
FISTA	$\mathcal{S}_{\lambda/L}$ at an extrapolated point	ℓ_1 (LASSO)	$O(1/k^2)$

Table 7.1. The iterative thresholding family. All share the gradient-step template; the cleanup and, for FISTA, the momentum are the only differences.

same per-iteration cost. The gap widens as the target accuracy tightens, which is why FISTA is the default first-order solver.

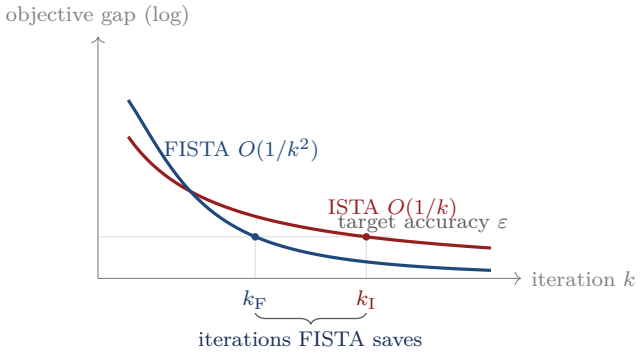


Figure 7.2. ISTA versus FISTA. Both pay the same per-iteration cost, but FISTA's momentum drives the objective gap down as $1/k^2$ against ISTA's $1/k$. To reach the same target accuracy ε (dashed line) FISTA needs k_{F} iterations where ISTA needs k_{I}, roughly the square root as many: about 10^2 versus 10^4 for $\varepsilon = 10^{-4}$.

7.6 Sparsity in the Gradient: Total Variation

Not every signal is sparse in its own coordinates, but many are sparse in their *differences*. A piecewise-constant image has few edges, so its gradient is sparse even though the image itself is not. Total-variation (TV) minimization exploits exactly this, replacing $\|\boldsymbol{x}\|_1$ with the ℓ_1 norm of the discrete gradient,

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\,\mathrm{TV}(\boldsymbol{x}), \qquad \mathrm{TV}(\boldsymbol{x}) = \sum_i |(\nabla \boldsymbol{x})_i|.$$

Penalizing the gradient's ℓ_1 norm promotes images that are flat over regions and change only at a few sharp edges, which is why TV denoising removes noise while preserving boundaries where ordinary smoothing would blur them.

Example 7.5 (A dense signal with a sparse gradient). Take the piecewise-constant signal $\boldsymbol{x} = (3,3,3,7,7,2,2,2)^\top$. As a vector it is *dense*, with eight nonzero entries, so $\|\boldsymbol{x}\|_0 = 8$ and ordinary ℓ_1 recovery sees no sparsity to exploit. Its discrete gradient (successive differences) is

$$\nabla \boldsymbol{x} = (0,0,4,0,-5,0,0)^\top,$$

with only *two* nonzeros, at the two jumps. So $\|\nabla \boldsymbol{x}\|_0 = 2$ and $\mathrm{TV}(\boldsymbol{x}) = |4| + |-5| = 9$. The signal is sparse in its gradient even though it is dense in its samples (Fig. 7.3). Minimizing TV instead of $\|\boldsymbol{x}\|_1$ is then exactly compressed sensing applied to $\nabla \boldsymbol{x}$, which is why a piecewise-constant image recovers from very few measurements. ◊

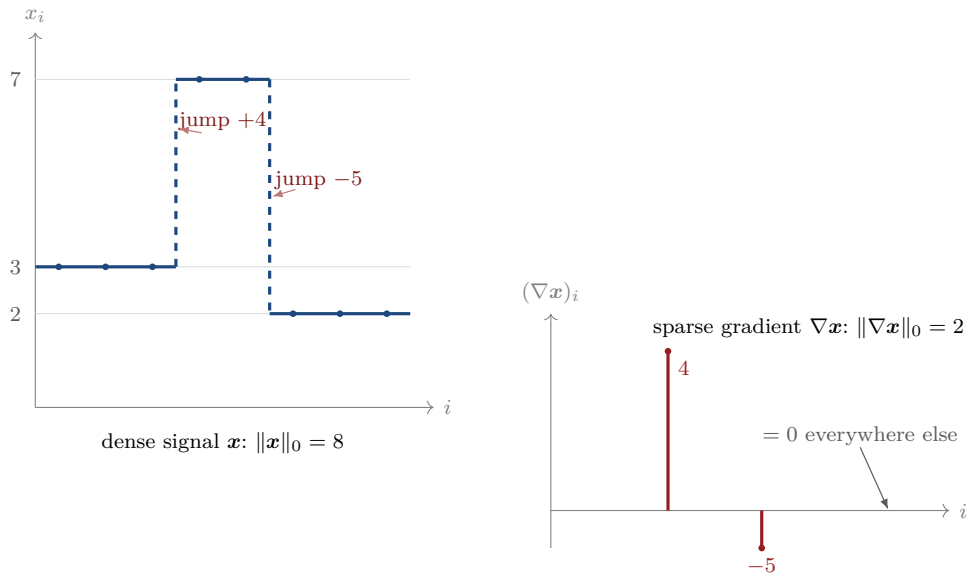


Figure 7.3. The total-variation idea. The staircase signal (left) has eight nonzero samples ($\|\boldsymbol{x}\|_0 = 8$, dense) yet changes value at only two edges; its gradient (right) is zero everywhere except those two jumps ($\|\nabla \boldsymbol{x}\|_0 = 2$, sparse), with the same heights $+4$ and -5. TV minimization recovers the signal by exploiting that gradient sparsity.

The same proximal machinery solves it: the only change is that the prox is now taken with respect to the gradient operator, computed by its own short inner iteration. TV recovery is the standard model behind compressed sensing MRI, where the structured Fourier sensing of Chapter 10 meets an edge-sparse image.

7.7 Splitting Methods: GPSR, ALM, and ADMM

ISTA and FISTA are not the only fast solvers, and for some problems they are not the best. This section collects the other members of the modern first-order family the course used, all built on the same idea of *splitting* a hard problem into easy pieces. They solve the very same LASSO/basis-pursuit objectives; what differs is how the split is arranged.

7.7.1 Three equivalent problems, one Lagrangian

Recall the basis pursuit family of Chapter 5: the noise-constrained program $\min \|\boldsymbol{x}\|_1$ s.t. $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \le \varepsilon$, the penalized BPDN $\min \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{x}\|_1$, and the budget-constrained LASSO. By Lagrangian duality these trace the same solution path: BPDN is the Lagrangian relaxation of the constrained forms, with λ the multiplier, and for matched parameters the three minimizers coincide (a Rockafellar-style equivalence). A solver for one solves all three by sweeping

its parameter. The splitting methods below all attack the penalized form, because its unconstrained objective is the easiest to iterate.

7.7.2 Gradient projection (GPSR)

Gradient projection for sparse reconstruction (GPSR) removes the ℓ_1 kink by the positive–negative split of Chapter 5: write $\boldsymbol{x} = \boldsymbol{u} - \boldsymbol{v}$ with $\boldsymbol{u}, \boldsymbol{v} \geq \boldsymbol{0}$, so $\|\boldsymbol{x}\|_1 = \mathbf{1}^\top \boldsymbol{u} + \mathbf{1}^\top \boldsymbol{v}$ and the BPDN objective becomes a *smooth bound-constrained quadratic*,

$$\min_{\boldsymbol{u},\boldsymbol{v} \geq \boldsymbol{0}} \ \tfrac{1}{2}\|\boldsymbol{A}(\boldsymbol{u} - \boldsymbol{v}) - \boldsymbol{y}\|_2^2 + \lambda \mathbf{1}^\top (\boldsymbol{u} + \boldsymbol{v}).$$

Now there is no nonsmooth term, only a nonnegativity constraint, so plain gradient descent with a projection onto the orthant works: take a gradient step, then clip every negative entry to zero. The projection onto $\{\boldsymbol{u} \geq \boldsymbol{0}\}$ is just $\max(\boldsymbol{u}, 0)$ entrywise, as cheap as soft thresholding. GPSR is competitive when the measurement operator $\boldsymbol{A}$ is dense and a good Barzilai–Borwein step size is used.

7.7.3 The augmented Lagrangian (ALM)

For the *constrained* program $\min \|\boldsymbol{x}\|_1$ s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, the **augmented Lagrangian method** adds both a multiplier and a quadratic penalty to the constraint,

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{\lambda}) = \|\boldsymbol{x}\|_1 + \langle \boldsymbol{\lambda},\, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle + \tfrac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2,$$

and alternates minimizing over $\boldsymbol{x}$ (a prox-type step) with dual ascent $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \rho(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})$. The quadratic penalty makes the inner problem well conditioned even when the constraint is nearly degenerate, and growing ρ drives the constraint to hold exactly. This is the *same* augmented Lagrangian that solves robust PCA in Chapter 12; there the unknown is a matrix and the prox steps are soft and singular-value thresholding, but the skeleton is identical.

7.7.4 ADMM: alternating direction multipliers

The workhorse is the **alternating direction method of multipliers** (ADMM), which splits the objective into two blocks tied by an equality. Write BPDN as

$$\min_{\boldsymbol{x},\boldsymbol{z}} \ \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{z}\|_1 \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{z},$$

separating the smooth data term (in $\boldsymbol{x}$) from the nonsmooth penalty (in $\boldsymbol{z}$). The augmented Lagrangian in scaled form has dual variable $\boldsymbol{u}$ and penalty ρ, and ADMM cycles three closed-form updates (Algorithm 7): a ridge solve for $\boldsymbol{x}$, a soft threshold for $\boldsymbol{z}$, and a dual ascent for $\boldsymbol{u}$.

Algorithm 7 ADMM for the LASSO

Require: $\boldsymbol{A}$, $\boldsymbol{y}$, penalty λ, ADMM parameter $\rho > 0$

$\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{u} \leftarrow \boldsymbol{0}$; precompute $\boldsymbol{M} = (\boldsymbol{A}^\top \boldsymbol{A} + \rho \boldsymbol{I})^{-1}$

repeat

  $\boldsymbol{x} \leftarrow \boldsymbol{M}\big(\boldsymbol{A}^\top \boldsymbol{y} + \rho(\boldsymbol{z} - \boldsymbol{u})\big)$ ▷ ridge solve (smooth block)

  $\boldsymbol{z} \leftarrow \mathcal{S}_{\lambda/\rho}(\boldsymbol{x} + \boldsymbol{u})$ ▷ soft threshold (nonsmooth block)

  $\boldsymbol{u} \leftarrow \boldsymbol{u} + \boldsymbol{x} - \boldsymbol{z}$ ▷ dual ascent

until $\|\boldsymbol{x} - \boldsymbol{z}\|_2$ small

return $\boldsymbol{z}$

Why ADMM is the workhorse

ADMM decouples the two things that make sparse recovery hard. The data term wants a least-squares fit (a linear solve); the penalty wants sparsity (a threshold). Splitting them with the constraint $\boldsymbol{x} = \boldsymbol{z}$ lets each do its own job in closed form, and the dual variable $\boldsymbol{u}$ is the referee that nudges the two copies into agreement. The matrix $\boldsymbol{M} = (\boldsymbol{A}^\top \boldsymbol{A} + \rho \boldsymbol{I})^{-1}$ is factored once and reused, so every iteration is a back-substitution plus a soft threshold.

Example 7.6 (One ADMM iteration by hand). Take $\boldsymbol{A} = \boldsymbol{I}_2$, $\boldsymbol{y} = (3, 0.5)^\top$, $\lambda = 1$, $\rho = 1$, starting from $\boldsymbol{x} = \boldsymbol{z} = \boldsymbol{u} = \boldsymbol{0}$. Then $\boldsymbol{M} = (\boldsymbol{I} + \boldsymbol{I})^{-1} = \frac{1}{2}\boldsymbol{I}$.

$$\boldsymbol{x} \leftarrow \tfrac{1}{2}\big(\boldsymbol{y} + (\boldsymbol{z} - \boldsymbol{u})\big) = \tfrac{1}{2}(3, 0.5)^\top = (1.5, 0.25)^\top,$$

$$\boldsymbol{z} \leftarrow \mathcal{S}_1\big((1.5, 0.25)^\top\big) = (0.5, 0)^\top, \qquad \boldsymbol{u} \leftarrow \boldsymbol{0} + (1.5, 0.25)^\top - (0.5, 0)^\top = (1.0, 0.25)^\top.$$

The small second coordinate is already killed. Iterating, $\boldsymbol{z}$ climbs to the true LASSO solution $\mathcal{S}_1(\boldsymbol{y}) = (2, 0)^\top$ (the orthogonal case of Chapter 5), which ADMM reaches in a handful of steps. ◊

7.7.5 The elastic net

A last variant fixes a real weakness of the LASSO. When two columns of $\boldsymbol{A}$ are highly correlated, the LASSO arbitrarily picks one and zeroes the other, and its minimizer can even be non-unique (Chapter 5). The **elastic net** adds a small ridge term,

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda_1\|\boldsymbol{x}\|_1 + \lambda_2\|\boldsymbol{x}\|_2^2,$$

combining the ℓ_1 penalty (which keeps the solution sparse) with an ℓ_2 penalty (which makes the objective *strictly* convex, so the minimizer is always unique, and which encourages correlated columns to enter or leave together, the "grouping effect"). It is the method of choice when features come in correlated clusters, and its prox is a soft threshold followed by a shrink, $\mathcal{S}_{\lambda_1}(\cdot)/(1 + 2\lambda_2)$.

Method	Mechanism	Best when
ISTA / FISTA	gradient step then soft threshold	large, structured $\boldsymbol{A}$ (fast transform)
GPSR	sign split, gradient projection onto the orthant	dense $\boldsymbol{A}$, good step-size rule
ALM	multiplier + quadratic penalty on the constraint	exact-constraint (noiseless) recovery
ADMM	split data and penalty, alternate + dual ascent	general; the robust default
Elastic net	$\ell_1 + \ell_2$ penalty	correlated columns; uniqueness wanted

Table 7.2. The first-order solver family. All minimize the same recovery objective; they differ in how they split the smooth fit from the nonsmooth penalty.

Notes and References

The proximal operator and the calculus around it are surveyed by Parikh and Boyd; the existence and uniqueness argument is standard convex analysis. ISTA in this form is Daubechies, Defrise, and De Mol [18], building on a long history of forward-backward splitting; the soft-thresholding-as-ℓ_1-prox identity goes back to the wavelet shrinkage of Donoho and Johnstone. FISTA and its $O(1/k^2)$ analysis are Beck and Teboulle [4], adapting Nesterov's accelerated gradient method. Iterative hard thresholding is Blumensath and Davies [6], with convergence under the restricted isometry property of Chapter 8. Total-variation regularization originates with Rudin, Osher, and Fatemi; its role in compressed sensing MRI is in Lustig, Donoho, and Pauly [36].

Exercises

Exercise 7.1. Compute $\mathcal{S}_{0.5}(\boldsymbol{v})$ for $\boldsymbol{v} = (1, -0.3, 0.5, -2, 0.2)^\top$. Which entries are killed, and by how much is each survivor shrunk?

Exercise 7.2. Show directly from Definition 7.1 that $\operatorname{prox}_{\lambda \iota_C} = \operatorname{proj}_C$ for the box $C = \{\boldsymbol{x} : |x_i| \le 1\}$, and give the resulting coordinatewise formula (clipping to $[-1, 1]$).

Exercise 7.3. Verify the majorization step of ISTA: for $g(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ with $L = \|\boldsymbol{A}\|_{2\to 2}^2$, prove $g(\boldsymbol{x}) \le g(\boldsymbol{x}^k) + \left\langle \nabla g(\boldsymbol{x}^k), \boldsymbol{x} - \boldsymbol{x}^k \right\rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}^k\|_2^2$. *Hint: expand both sides and use* $\|\boldsymbol{A}\boldsymbol{u}\|_2^2 \le L\|\boldsymbol{u}\|_2^2$.

Exercise 7.4. Run one IHT step from Example 7.4 but with target sparsity $s = 1$. What does $\mathcal{H}_1$ return, and how many iterations would IHT need to settle? Contrast with the $s = 2$ case.

Exercise 7.5. Explain why FISTA's momentum coefficient $\frac{t-1}{t^+}$ starts at 0 on the first iteration (so the first FISTA step equals an ISTA step) and approaches 1 as k grows. What would happen if the coefficient were held fixed at 1?

In the Problem Sets

Problem Set 6 (Chapter 25) pits these iterative solvers against the greedy methods of Chapter 6 and the linear program of Chapter 5 on image-recovery tasks. You implement ISTA, FISTA, and IHT from the single template, tune the step size from $\|\boldsymbol{A}\|_{2\to 2}^2$, and watch FISTA reach a target reconstruction quality in far fewer iterations than ISTA. The accompanying notebook reports reconstruction quality against the number of measurements for each solver.

PART IV

Why Random Sensing Works

CHAPTER 8

Coherence and the Restricted Isometry Property

A good sensing matrix is one that almost preserves length on the things you care about.

Chapter 4 left a gap. The null space property is the exact condition for ℓ_1 recovery, but it cannot be checked: it asks an inequality of uncountably many null vectors. This chapter closes the gap with two checkable, sufficient conditions that imply the null space property and therefore guarantee recovery. The first is the *restricted isometry property* (RIP), the geometric requirement that the sensing matrix nearly preserve the length of every sparse vector. We prove the central theorem of the subject: if the matrix has a small enough restricted isometry constant, then ℓ_1 minimization recovers every sparse signal exactly. The second condition is *mutual coherence*, a single number computable from one matrix multiplication, which bounds the restricted isometry constant from above. The two together give a clean chain from a quantity you can compute to the recovery you want. The chapter ends by showing why coherence alone is not enough, which is what forces the random-matrix theory of Chapter 9.

8.1 The Restricted Isometry Property

An *isometry* preserves length exactly: $\|\boldsymbol{A}\boldsymbol{x}\|_2 = \|\boldsymbol{x}\|_2$ for all $\boldsymbol{x}$. A wide matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ with $m < N$ can never be an isometry, since it has a nontrivial null space. But recovery does not need length preserved on *all* of $\mathbb{C}^N$, only on the sparse vectors. The restricted isometry property asks for approximate length preservation, restricted to sparse vectors.

Definition 8.1 (Restricted isometry property). A matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ satisfies the **restricted isometry property** of order s if there is a constant $\delta \in [0, 1)$ such that

$$(1-\delta)\,\|\boldsymbol{x}\|_2^2 \;\le\; \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \;\le\; (1+\delta)\,\|\boldsymbol{x}\|_2^2 \qquad \text{for every } s\text{-sparse } \boldsymbol{x}. \tag{8.1}$$

The smallest such δ is the **restricted isometry constant** $\delta_s(\boldsymbol{A})$.

Reading Eq. (8.1), the matrix stretches or shrinks the length of any s-sparse vector by at most a factor $\sqrt{1\pm\delta_s}$. When δ_s is small, every group of s columns behaves almost like an orthonormal set, so distinct sparse signals stay distinct and far apart after measurement. The constant has a clean operator-norm form that makes it, in principle, computable.

> A picture for the restricted isometry property: think of $\boldsymbol{A}$ as a slightly warped ruler. A general ruler that loses dimensions ($m < N$ marks for N lengths) must misread *some* objects badly. The restricted isometry property says the warping is gentle on *sparse* objects: every s-sparse vector's length is read to within a few percent ($\sqrt{1\pm\delta_s}$). A warped ruler that is still honest about thin objects is exactly what compressed sensing needs, because it never confuses two different sparse signals, and that is all recovery requires.

Lemma 8.2 (Operator-norm form of the restricted isometry constant). $\delta_s(\boldsymbol{A}) = \max_{|S|\le s} \|\boldsymbol{A}_S^*\boldsymbol{A}_S - \boldsymbol{I}_{|S|}\|_{2\to 2}$.

So δ_s measures how far the Gram matrix of any s selected columns is from the identity. When $\boldsymbol{A}_S^*\boldsymbol{A}_S \approx \boldsymbol{I}$ the chosen columns are nearly orthonormal, which is exactly approximate isometry on their span. We use this form twice below.

Example 8.3 (A restricted isometry constant by hand). Let $\boldsymbol{A} = \left(\begin{smallmatrix} 1 & 0 & 1/\sqrt{2} \\ 0 & 1 & 1/\sqrt{2} \end{smallmatrix}\right)$, with unit-norm columns. For the pair $S = \{1, 3\}$,

$$\boldsymbol{A}_S^*\boldsymbol{A}_S - \boldsymbol{I}_2 = \begin{pmatrix} 0 & 1/\sqrt{2} \\ 1/\sqrt{2} & 0 \end{pmatrix},$$

a Hermitian matrix with eigenvalues $\pm 1/\sqrt{2}$, so its spectral norm is $1/\sqrt{2}$. The pair $\{2, 3\}$ gives the same, and $\{1, 2\}$ gives 0. Hence $\delta_2(\boldsymbol{A}) = 1/\sqrt{2} \approx 0.707$. This is large; the third column sits at 45° to each of the first two, so this matrix is a poor sensor, as the recovery theorem below will confirm. ♢

8.2 The Main Recovery Theorem

The restricted isometry property is useful because a small constant implies the null space property, and hence, by Theorem 4.10, exact ℓ_1 recovery. This is the theorem the whole theory has been building toward.

Theorem 8.4 (RIP implies recovery). If $\boldsymbol{A}$ satisfies the restricted isometry property of order $2s$ with

$$\delta_{2s}(\boldsymbol{A}) \;<\; \tfrac{1}{3},$$

then $\boldsymbol{A}$ has the null space property of order s, and consequently ℓ_1 minimization (P_1) recovers every s-sparse signal exactly.

Remark 8.5 *(The sharp constant).* The threshold $\frac{1}{3}$ comes from the clean argument below. With more careful bookkeeping the constant can be improved to the celebrated value $\delta_{2s} < \sqrt{2} - 1 \approx 0.414$ of Candès [9], and refinements push it further. The exact constant does not matter for the theory; what matters is that *some* fixed threshold on δ_{2s} guarantees recovery, because Chapter 9 will show random matrices meet any such threshold with only $m \approx s\log(N/s)$ rows.

The proof rests on three lemmas, which we isolate first.

8.2.1 Three lemmas

Lemma 8.6 (Norm equivalence on sparse vectors). For every s-sparse $\boldsymbol{v}$, $\dfrac{\|\boldsymbol{v}\|_1}{\sqrt{s}} \le \|\boldsymbol{v}\|_2 \le \sqrt{s}\,\|\boldsymbol{v}\|_\infty$.

Proof. Let $T = \operatorname{supp}(\boldsymbol{v})$, $|T| \le s$. For the right inequality, each entry has magnitude at most $\|\boldsymbol{v}\|_\infty$, so $\|\boldsymbol{v}\|_2^2 = \sum_{j\in T}|v_j|^2 \le |T|\|\boldsymbol{v}\|_\infty^2 \le s\|\boldsymbol{v}\|_\infty^2$. For the left, let $\boldsymbol{\sigma}$ be the vector that equals $v_j/|v_j|$ on T and 0 off it, so $\langle \boldsymbol{\sigma}, \boldsymbol{v}\rangle = \sum_{j\in T}|v_j| = \|\boldsymbol{v}\|_1$ and $\|\boldsymbol{\sigma}\|_2 = \sqrt{|T|} \le \sqrt{s}$. By Cauchy–Schwarz (Theorem 2.10), $\|\boldsymbol{v}\|_1 = \langle \boldsymbol{\sigma}, \boldsymbol{v}\rangle \le \|\boldsymbol{\sigma}\|_2\|\boldsymbol{v}\|_2 \le \sqrt{s}\,\|\boldsymbol{v}\|_2$. ■

Lemma 8.7 (Disjoint-support inner-product bound). If $\boldsymbol{u}, \boldsymbol{v}$ have disjoint supports whose union has size at most s, then

$$|\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{A}\boldsymbol{v}\rangle| \;\le\; \delta_s\,\|\boldsymbol{u}\|_2\,\|\boldsymbol{v}\|_2.$$

Proof. Normalize so $\|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1$. Both $\boldsymbol{u} \pm \boldsymbol{v}$ are s-sparse with $\|\boldsymbol{u} \pm \boldsymbol{v}\|_2^2 = 2$ (disjoint supports), so the restricted isometry property gives $2(1 - \delta_s) \le \|\boldsymbol{A}(\boldsymbol{u} \pm \boldsymbol{v})\|_2^2 \le 2(1 + \delta_s)$. The polarization identity $\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{A}\boldsymbol{v}\rangle = \frac{1}{4}\big(\|\boldsymbol{A}(\boldsymbol{u}+\boldsymbol{v})\|_2^2 - \|\boldsymbol{A}(\boldsymbol{u}-\boldsymbol{v})\|_2^2\big)$ then bounds the cross term by $\frac{1}{4}\big(2(1+\delta_s) - 2(1-\delta_s)\big) = \delta_s$. Rescaling restores the factor $\|\boldsymbol{u}\|_2\|\boldsymbol{v}\|_2$. ■

Lemma 8.2 is the third lemma; we use it only to read δ_s as a Gram-matrix distance. With these in hand the proof is a chain of inequalities ending in a self-referential bound.

8.2.2 Proof of the recovery theorem

Proof of Theorem 8.4. We must show $\|\boldsymbol{v}_{S_0}\|_1 < \|\boldsymbol{v}_{S_0^c}\|_1$ for every nonzero null vector $\boldsymbol{v}$ and the support S_0 of its s largest entries; this is the null space property, and it suffices to take S_0 the top-s block, where the inequality is hardest. Sort the remaining coordinates into blocks $S_1, S_2, \dots$ of size s by decreasing magnitude, so $S_0^c = S_1 \cup S_2 \cup \cdots$.

Step 1 (the null relation). Since $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{0}$ and $\boldsymbol{v} = \boldsymbol{v}_{S_0} + \sum_{k\ge 1} \boldsymbol{v}_{S_k}$,

$$\boldsymbol{A}\boldsymbol{v}_{S_0} = -\sum_{k\ge 1} \boldsymbol{A}\boldsymbol{v}_{S_k}.$$

Step 2 (lower bound by RIP). As $\boldsymbol{v}_{S_0}$ is s-sparse, $\|\boldsymbol{v}_{S_0}\|_2^2 \le \frac{1}{1-\delta_{2s}} \|\boldsymbol{A}\boldsymbol{v}_{S_0}\|_2^2$.

Step 3 (expand and bound the cross terms). Writing $\|\boldsymbol{A}\boldsymbol{v}_{S_0}\|_2^2 = \langle \boldsymbol{A}\boldsymbol{v}_{S_0},\, \boldsymbol{A}\boldsymbol{v}_{S_0}\rangle$ and substituting Step 1 into the second slot,

$$\|\boldsymbol{A}\boldsymbol{v}_{S_0}\|_2^2 = \Big| \sum_{k\ge 1} \langle \boldsymbol{A}\boldsymbol{v}_{S_0},\, \boldsymbol{A}\boldsymbol{v}_{S_k}\rangle \Big| \le \sum_{k\ge 1} |\langle \boldsymbol{A}\boldsymbol{v}_{S_0},\, \boldsymbol{A}\boldsymbol{v}_{S_k}\rangle|.$$

Each S_0 and S_k are disjoint with union of size $2s$, so Lemma 8.7 (at order $2s$) bounds each term by $\delta_{2s}\|\boldsymbol{v}_{S_0}\|_2 \|\boldsymbol{v}_{S_k}\|_2$. Combining with Step 2 and cancelling one factor of $\|\boldsymbol{v}_{S_0}\|_2$,

$$\|\boldsymbol{v}_{S_0}\|_2 \;\le\; \frac{\delta_{2s}}{1-\delta_{2s}} \sum_{k\ge 1} \|\boldsymbol{v}_{S_k}\|_2. \tag{8.2}$$

Step 4 (transfer ℓ_2 to ℓ_1). The entries of block S_k are no larger than every entry of the previous block S_{k-1}, so each satisfies $|v_j| \le \|\boldsymbol{v}_{S_{k-1}}\|_1/s$, giving $\|\boldsymbol{v}_{S_k}\|_\infty \le \|\boldsymbol{v}_{S_{k-1}}\|_1/s$. By the right half of Lemma 8.6, $\|\boldsymbol{v}_{S_k}\|_2 \le \sqrt{s}\,\|\boldsymbol{v}_{S_k}\|_\infty \le \|\boldsymbol{v}_{S_{k-1}}\|_1/\sqrt{s}$. Summing over $k \ge 1$ telescopes the blocks back into the whole vector:

$$\sum_{k\ge 1} \|\boldsymbol{v}_{S_k}\|_2 \le \frac{1}{\sqrt{s}} \sum_{k\ge 1} \|\boldsymbol{v}_{S_{k-1}}\|_1 = \frac{\|\boldsymbol{v}_{S_0}\|_1 + \|\boldsymbol{v}_{S_0^c}\|_1}{\sqrt{s}}.$$

Step 5 (the master inequality). Apply the left half of Lemma 8.6 to the left side of Eq. (8.2), $\|\boldsymbol{v}_{S_0}\|_1/\sqrt{s} \le \|\boldsymbol{v}_{S_0}\|_2$, and chain with Step 4. The factors of $\sqrt{s}$ cancel and leave

$$\|\boldsymbol{v}_{S_0}\|_1 \;\le\; \frac{\delta_{2s}}{1-\delta_{2s}} \big(\|\boldsymbol{v}_{S_0}\|_1 + \|\boldsymbol{v}_{S_0^c}\|_1\big).$$

Step 6 (solve it). Write $\alpha = \delta_{2s}/(1-\delta_{2s})$. Moving the $\|\boldsymbol{v}_{S_0}\|_1$ term to the left, $(1-\alpha)\|\boldsymbol{v}_{S_0}\|_1 \le \alpha\|\boldsymbol{v}_{S_0^c}\|_1$, so the strict null space inequality $\|\boldsymbol{v}_{S_0}\|_1 < \|\boldsymbol{v}_{S_0^c}\|_1$ holds as soon as $\alpha/(1-\alpha) < 1$, that is $\alpha < \frac{1}{2}$. Unfolding α,

$$\frac{\delta_{2s}}{1-\delta_{2s}} < \tfrac{1}{2} \iff 2\delta_{2s} < 1-\delta_{2s} \iff \delta_{2s} < \tfrac{1}{3}.$$

So $\delta_{2s} < \frac{1}{3}$ gives the null space property of order s, and Theorem 4.10 converts it to exact ℓ_1 recovery of every s-sparse signal. ∎

What the proof bought us

A single, checkable, geometric number δ_{2s} now controls everything: if it is below $\frac{1}{3}$, the intractable (P_0) and the tractable (P_1) have the same solution for every s-sparse signal. The remaining question is which matrices have a small δ_{2s}, and with how few rows. The rest of the chapter gives the computable but pessimistic answer (coherence); Chapter 9 gives the optimal one (random matrices).

8.3 Stable and Robust Recovery

Theorem 8.4 recovers an *exactly* sparse signal from *noiseless* measurements. Real signals are neither. A measured signal carries noise, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$ with $\|\boldsymbol{e}\|_2 \le \eta$, and a real image or spectrum is only *compressible*, with a few large coefficients and a tail of small ones rather than a hard cutoff. The restricted isometry property is strong enough to handle both, and that is what makes it the workhorse condition.

Recover by the noise-aware program of Section 5.4, quadratically constrained basis pursuit, $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{z}} \|\boldsymbol{z}\|_1$ subject to $\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{y}\|_2 \le \eta$. Its error is controlled by two quantities: the noise level η and the **best s-term approximation error**

$$\sigma_s(\boldsymbol{x})_1 = \min_{\|\boldsymbol{z}\|_0 \le s} \|\boldsymbol{x} - \boldsymbol{z}\|_1,$$

the ℓ_1 distance from $\boldsymbol{x}$ to the nearest s-sparse vector, which is simply the sum of the magnitudes of all but the s largest entries.

Theorem 8.8 (Stable and robust recovery). If $\delta_{2s}(\boldsymbol{A}) < \sqrt{2} - 1$, then the quadratically constrained basis pursuit solution $\hat{\boldsymbol{x}}$ obeys

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2 \;\le\; C_0 \, \frac{\sigma_s(\boldsymbol{x})_1}{\sqrt{s}} \;+\; C_1 \, \eta,$$

with constants C_0, C_1 depending only on δ_{2s}.

The bound is the theorem the field actually uses. Read it as two penalties added together. The **noise term** $C_1\eta$ says the recovery error grows at worst *linearly* in the measurement noise: the method does not amplify noise, it degrades gracefully (Fig. 8.1). The **approximation term** $C_0\sigma_s(\boldsymbol{x})_1/\sqrt{s}$ says a signal that is only nearly sparse is recovered nearly as well as if it were exactly sparse, with an error set by the tail it omits. Two special cases recover what we already know.

- **Exactly sparse, noiseless.** If $\boldsymbol{x}$ is s-sparse then $\sigma_s(\boldsymbol{x})_1 = 0$, and if $\eta = 0$ then both terms vanish: $\hat{\boldsymbol{x}} = \boldsymbol{x}$, the exact recovery of Theorem 8.4.
- **Sparse, noisy.** If $\boldsymbol{x}$ is s-sparse but $\eta > 0$, the error is at most $C_1\eta$, proportional to the noise: stable recovery.

Example 8.9 (Recovering a compressible signal). Suppose the sorted coefficient magnitudes of $\boldsymbol{x} \in \mathbb{R}^N$ decay like $|x|_{(k)} = k^{-2}$, a typical compressible profile. The best s-term approximation error is the omitted tail,

$$\sigma_s(\boldsymbol{x})_1 = \sum_{k>s} k^{-2} \;\approx\; \frac{1}{s},$$

so the approximation penalty is $C_0\sigma_s(\boldsymbol{x})_1/\sqrt{s} \approx C_0/s^{3/2}$, which falls off fast: at $s = 10$ it is about $C_0/32$, and at $s = 25$ about $C_0/125$. Keeping a handful of measurements per significant coefficient already drives the structural error to a few percent, and the remaining error is the noise floor $C_1\eta$. This is exactly why the images of Chapter 24 recover well from a modest fraction of measurements despite never being exactly sparse. ◊

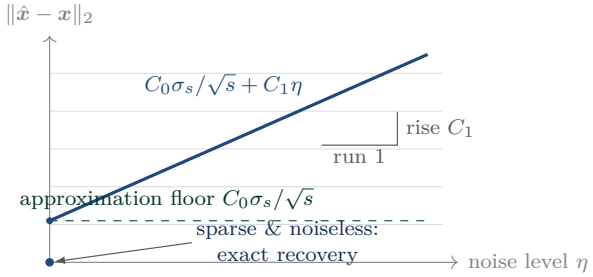


Figure 8.1. Stable and robust recovery. The error grows at most linearly in the noise η (the slope triangle shows rise C_1 per unit of noise) above a floor set by how compressible the signal is (the best s-term error). An exactly sparse, noiseless signal sits at the origin (highlighted), recovered exactly.

This is why the restricted isometry property, not the exact null space property, is the condition the literature leans on. The null space property is the sharp characterization of *exact* recovery, but it says nothing graceful about noise or near-sparsity. A single bound on δ_{2s} delivers all three at once: exact recovery of sparse signals, stable recovery under noise, and robust recovery of compressible signals. One checkable number, three guarantees.

8.4 Mutual Coherence

The restricted isometry constant is itself hard to compute, since Lemma 8.2 ranges over all $\binom{N}{s}$ subsets. A single pairwise statistic gives a computable upper bound on it.

Definition 8.10 (Mutual coherence). For $\boldsymbol{A}$ with unit-norm columns $\boldsymbol{a}_1, \dots, \boldsymbol{a}_N$, the **mutual coherence** is the largest correlation between distinct columns,

$$\mu(\boldsymbol{A}) = \max_{i \neq j} |\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle|.$$

Because the columns are unit vectors, $|\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle| = |\cos \theta_{ij}|$, so μ is the cosine of the smallest angle between any two columns. Small coherence means all columns are far from parallel. The value lies in $[0, 1]$: it is 0 only when the columns are orthonormal, possible only if $N \leq m$, and approaches 1 when two columns nearly align. Coherence is cheap to compute: form the Gram matrix $\boldsymbol{G} = \boldsymbol{A}^* \boldsymbol{A}$, whose off-diagonal entries are the inner products, and take the largest in magnitude. This is one matrix multiplication, $O(mN^2)$, in stark contrast to the combinatorial cost of δ_s.

An analogy: each column of $\boldsymbol{A}$ is a *question* the sensor asks the signal, and the inner product $\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle$ measures how alike two questions are. High coherence means two columns nearly coincide, so asking one almost re-asks the other, wasting a measurement. Low coherence means every column asks a genuinely different question, so a handful of measurements pin the signal down. Designing a good sensing matrix is designing a questionnaire with no redundant questions.

Example 8.11 (Coherence of a small matrix). For the matrix of Example 8.3, the inner products are $\langle \boldsymbol{a}_1, \boldsymbol{a}_2 \rangle = 0$ and $\langle \boldsymbol{a}_1, \boldsymbol{a}_3 \rangle = \langle \boldsymbol{a}_2, \boldsymbol{a}_3 \rangle = 1/\sqrt{2}$, so $\mu = 1/\sqrt{2}$. This matches $\delta_2 = 1/\sqrt{2}$ from Example 8.3, which is no accident, as the next result shows. ◊

8.5 Coherence Controls the Restricted Isometry Constant

The link between the two notions begins at order two, where it is an exact equality, and extends to all orders as a bound.

Proposition 8.12 (Order-two equality). $\delta_2(\boldsymbol{A}) = \mu(\boldsymbol{A})$.

Proof. For a two-element set $S = \{j, k\}$, the Gram matrix has unit diagonal and off-diagonal $\langle \boldsymbol{a}_j, \boldsymbol{a}_k \rangle$, so $\boldsymbol{A}_S^* \boldsymbol{A}_S - \boldsymbol{I}_2 = \left(\begin{smallmatrix} 0 & \overline{\alpha} \\ \alpha & 0 \end{smallmatrix}\right)$ with $\alpha = \langle \boldsymbol{a}_k, \boldsymbol{a}_j \rangle$. This Hermitian matrix has eigenvalues $\pm|\alpha|$, hence spectral norm $|\alpha|$. Maximizing over pairs and using Lemma 8.2 gives $\delta_2 = \max_{j \neq k} |\langle \boldsymbol{a}_j, \boldsymbol{a}_k \rangle| = \mu$. ∎

Theorem 8.13 (Coherence bounds the restricted isometry constant). For every $s \geq 2$, $\delta_s(\boldsymbol{A}) \leq (s-1)\,\mu(\boldsymbol{A})$.

Proof. Fix $|S| \leq s$ and set $\boldsymbol{B} = \boldsymbol{A}_S^* \boldsymbol{A}_S - \boldsymbol{I}_{|S|}$, a Hermitian matrix with zero diagonal and off-diagonal entries $\langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle$. For Hermitian $\boldsymbol{B}$ the spectral norm is bounded by the induced $1 \to 1$ norm (the maximum absolute column sum), since $\|\boldsymbol{B}\|_{2\to 2} \leq \sqrt{\|\boldsymbol{B}\|_{1\to 1} \|\boldsymbol{B}\|_{\infty \to \infty}} = \|\boldsymbol{B}\|_{1\to 1}$ using $\|\boldsymbol{B}\|_{1\to 1} = \|\boldsymbol{B}\|_{\infty \to \infty}$ for Hermitian $\boldsymbol{B}$. Each column sum of $\boldsymbol{B}$ has at most $|S| - 1 \leq s - 1$ off-diagonal terms, each at most μ, so $\|\boldsymbol{B}\|_{1\to 1} \leq (s-1)\mu$. Taking the maximum over S and applying Lemma 8.2 gives $\delta_s \leq (s-1)\mu$. ∎

This is the bound that makes coherence useful: it turns the one cheap number μ into a certificate for the geometric constant δ_s, and through Theorem 8.4 into a recovery guarantee.

Coherence guarantee for recovery

If $\mu(\boldsymbol{A}) < \dfrac{1}{2s-1}$, then $\operatorname{spark}(\boldsymbol{A}) \geq 2s+1$ and the exact recovery condition both hold, so (P_1) and orthogonal matching pursuit each recover every s-sparse signal. This is the bound previewed in Chapters 4 and 6, now proved through the spark inequality $\operatorname{spark}(\boldsymbol{A}) \geq 1 + 1/\mu$ and the restricted isometry chain above.

8.6 The Quadratic Bottleneck

Coherence is checkable, but the guarantee it gives is weak, and the reason is a hard lower bound on how small coherence can be. No wide matrix can have arbitrarily small coherence.

Proposition 8.14 (Welch bound). For any $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ with unit-norm columns and $N > m$,

$$\mu(\boldsymbol{A}) \;\geq\; \sqrt{\frac{N-m}{m(N-1)}}.$$

For $N \gg m$ this is approximately $1/\sqrt{m}$.

The Welch bound says coherence cannot beat about $1/\sqrt{m}$. Some matrices nearly achieve it: the concatenation $[\,\boldsymbol{I}\;\boldsymbol{F}\,]$ of the identity and the discrete Fourier matrix, the natural sensor for a signal sparse in either time or frequency, has $\mu = 1/\sqrt{m}$ exactly. Feed this into the coherence guarantee. To recover s-sparse signals we need $\mu < 1/(2s-1)$, that is $1/\sqrt{m} < 1/(2s-1)$, which rearranges to

$$m \;\gtrsim\; (2s-1)^2 \;\approx\; 4s^2.$$

This is the **quadratic bottleneck**: any coherence-based guarantee needs the number of measurements to grow like s^2. That is far worse than the $m \approx s\log(N/s)$ promised in Chapter 1. The bottleneck is not a weakness of the proof; it is forced by the Welch bound, an intrinsic limit on coherence. The only escape is to abandon the worst-case pairwise statistic and reason directly about δ_{2s} for a whole class of matrices at once. Random matrices have $\delta_{2s} < \frac{1}{3}$ with only $m \approx s\log(N/s)$ rows, beating the bottleneck by a square. Proving that is the entire business of Chapter 9.

Example 8.15 (The bottleneck in numbers). Take $N = 10^6$ and $s = 50$. The coherence guarantee demands $m \gtrsim (2s-1)^2 = 99^2 = 9801$, about 10^4 measurements. The restricted-isometry guarantee for a random matrix needs only $m \gtrsim s\log(N/s) = 50\ln(20000) \approx 50 \cdot 9.9 \approx 495$, about 500. The coherence bound asks for twenty times as many measurements, and the ratio is $(2s-1)^2/(s\log(N/s)) \approx 4s/\log(N/s)$, which *grows* with the sparsity: at $s = 100$ it is past fortyfold (Fig. 8.2). Worst-case coherence systematically overcounts. ◊

8.7 The Chain of Implications

The chapter assembles a chain from a computable quantity to the recovery we want. Figure 8.3 draws it. Reading left to right, mutual coherence bounds the restricted isometry constant, a small restricted isometry constant gives the null space property, and the null space property is exactly ℓ_1 recovery. Each arrow is a theorem of this chapter or the last.

The two routes through the chain trade against each other. Coherence is the only quantity here you can actually compute for a given matrix, but the Welch bound caps how much it can certify, at $m \gtrsim s^2$. The restricted isometry constant is not computable for a given matrix, but it is provably small for random matrices at the optimal rate

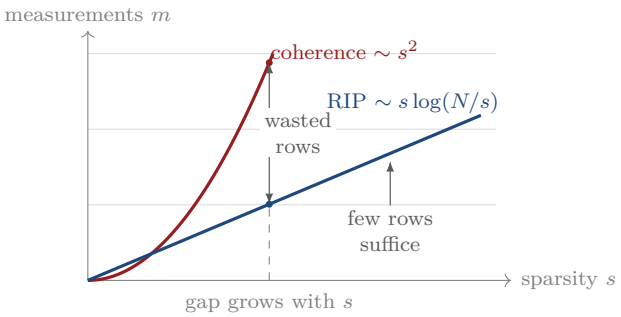


Figure 8.2. The quadratic bottleneck. The coherence-based measurement count grows quadratically in the sparsity, while the restricted-isometry count for a random matrix grows almost linearly. The vertical gap widens with s, which is why random matrices beat any worst-case coherence guarantee by a square.

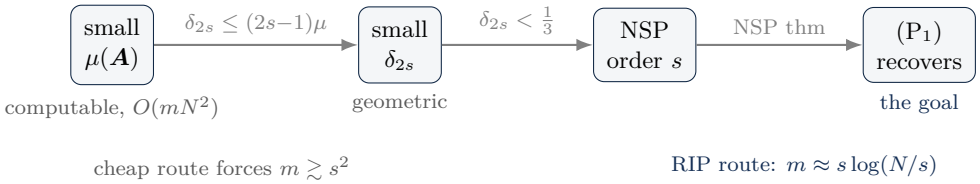


Figure 8.3. The certification chain. Coherence (cheap, $O(mN^2)$) bounds the restricted isometry constant (geometric), which gives the null space property (exact for ℓ_1), which is recovery. Coherence is checkable but forces $m \gtrsim s^2$; the restricted isometry route, fed by random matrices in Chapter 9, needs only $m \approx s\log(N/s)$.

$m \approx s\log(N/s)$. Compressed sensing in practice lives on the second route: design $\boldsymbol{A}$ at random, invoke the random-matrix theorem to know $\delta_{2s} < \frac{1}{3}$ with overwhelming probability, and recover by ℓ_1 or a greedy method. The next chapter proves that the second route exists.

Notes and References

The restricted isometry property and the recovery theorem are due to Candès and Tao [11, 12]; the sharp constant $\delta_{2s} < \sqrt{2}-1$ is Candès [9], with later work lowering it further. The clean $\delta_{2s} < \frac{1}{3}$ argument given here follows the development in Foucart and Rauhut [28], which also contains the three lemmas and the operator-norm form of δ_s. Mutual coherence and the bounds spark $\ge 1 + 1/\mu$ and $\delta_s \le (s-1)\mu$ are from Donoho and Elad [21] and Gribonval and Nielsen [31]; the coherence recovery guarantee $\mu < 1/(2s-1)$ also appears in Tropp [48]. The Welch bound is from Welch [52]; matrices attaining it (equiangular tight frames) are surveyed in the frame-theory literature. The spike-Fourier dictionary and its $m \gtrsim s^2$ bottleneck motivate the random-matrix theory of Chapter 9.

Exercises

Exercise 8.1. Compute δ_2 directly for $\boldsymbol{A} = \left(\begin{smallmatrix} 1 & 0 & \alpha \\ 0 & 1 & \beta \end{smallmatrix}\right)$ with the third column normalized ($\alpha^2 + \beta^2 = 1$), and confirm $\delta_2 = \mu = \max(|\alpha|, |\beta|)$ via Proposition 8.12.

Exercise 8.2. Verify Lemma 8.6 on $\boldsymbol{v} = (3, -4, 0, 0)^\top$ with $s = 2$: compute all three norms and check both inequalities, noting how much slack each has.

Exercise 8.3. Carry out the polarization step of Lemma 8.7 explicitly for $\boldsymbol{u} = \boldsymbol{e}_1$, $\boldsymbol{v} = \boldsymbol{e}_2$, and a 2×2 matrix $\boldsymbol{A}$ of your choosing; confirm $|\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{A}\boldsymbol{v} \rangle| \le \delta_2$.

Exercise 8.4. Using $\delta_s \le (s-1)\mu$, find the largest sparsity s for which the recovery theorem ($\delta_{2s} < \tfrac{1}{3}$) is certified by a matrix with $\mu = 0.05$. Compare with the coherence guarantee $\mu < 1/(2s-1)$.

Exercise 8.5. For the spike-Fourier matrix $[\,\boldsymbol{I}\ \boldsymbol{F}\,] \in \mathbb{C}^{m \times 2m}$ with $\mu = 1/\sqrt{m}$, find the number of measurements m the coherence guarantee requires to recover $s = 10$-sparse signals. Contrast with $m \approx s \log(N/s)$ for $N = 2m$.

CHAPTER 9

Random Matrices and the RIP

The best way to build a good sensing matrix is to not try, and roll the dice instead.

Chapter 8 proved that a small restricted isometry constant guarantees recovery, and that coherence can certify it, but only at the wasteful rate $m \gtrsim s^2$. This chapter delivers the optimal rate. The matrices that achieve it are not clever constructions but *random* ones: fill the entries with independent Gaussian or sign-valued draws, and with overwhelming probability the result has $\delta_{2s} < \frac{1}{3}$ using only $m \approx s \log(N/s)$ rows, a square-root improvement over coherence. The proof is the longest argument in the book, and it is worth every line, because it explains why compressed sensing works at all in practice. The strategy is three nested ideas: control one sparse vector (concentration), extend to all vectors on a fixed support (covering numbers), and union over all supports. We build each in turn.

9.1 Why Random Matrices

The certification chain of Chapter 8 has a computable end (coherence) and an optimal end (the restricted isometry constant), and they do not meet: coherence is stuck at $m \gtrsim s^2$ by the Welch bound, while a small δ_{2s} would need only $m \approx s \log(N/s)$ but cannot be computed for a given matrix. Random matrices break the deadlock. We cannot certify δ_{2s} for a *specific* matrix, but we can prove it is small for a *random* one, with probability so close to one that a single random draw works.

Definition 9.1 (Random ensembles). The **Gaussian ensemble** draws each entry of $\boldsymbol{A} \in \mathbb{R}^{m \times N}$ independently from $\mathcal{N}(0, 1/m)$, equivalently $\boldsymbol{A} = \frac{1}{\sqrt{m}}\boldsymbol{G}$ with $G_{ij} \sim \mathcal{N}(0,1)$ i.i.d. The **Bernoulli ensemble** sets each entry to $\pm 1/\sqrt{m}$ with equal probability. Both are special cases of **sub-Gaussian** ensembles, whose entries have tails no heavier than a Gaussian's. The $1/\sqrt{m}$ scaling normalizes so that $\mathbb{E}\|\boldsymbol{A}\boldsymbol{x}\|_2^2 = \|\boldsymbol{x}\|_2^2$.

The theorem we are after, stated now and proved by the end of the chapter, is the headline of the whole subject made rigorous.

Random matrices satisfy the RIP

Let $\boldsymbol{A}$ be drawn from the Gaussian or Bernoulli ensemble. For any target $\delta \in (0,1)$, if

$$m \;\geq\; C\,\delta^{-2}\, s\, \log(N/s)$$

for a universal constant C, then with probability at least $1 - 2e^{-cm}$ the matrix satisfies the restricted isometry property of order s with constant $\delta_s < \delta$.

Combined with Theorem 8.4 (take $\delta < \frac{1}{3}$ at order $2s$), this says a random matrix with $m \approx s \log(N/s)$ rows recovers every s-sparse signal by ℓ_1 minimization, with overwhelming probability. The rest of the chapter is the proof.

9.2 Concentration: The Point Estimate

The atom of the proof is a statement about a *single* fixed vector: a random matrix preserves its length, with a failure probability that decays exponentially in m. This is the engine; everything else extends it.

Lemma 9.2 (Norm preservation, the point estimate). Let $\boldsymbol{A} = \frac{1}{\sqrt{m}}\boldsymbol{G}$ be Gaussian. For any fixed $\boldsymbol{x}$ and any $\varepsilon \in (0,1)$,

$$\mathbb{P}\Big[\,\big|\,\|\boldsymbol{A}\boldsymbol{x}\|_2^2 - \|\boldsymbol{x}\|_2^2\,\big| > \varepsilon\|\boldsymbol{x}\|_2^2\Big] \;\le\; 2\,e^{-(\varepsilon^2-\varepsilon^3)m/4}.$$

The proof is a model Chernoff argument, and we give it in full because it is the technical heart of the chapter and a fair exam question on its own.

Proof. Normalize so $\|\boldsymbol{x}\|_2 = 1$. Each measurement $(\boldsymbol{G}\boldsymbol{x})_i = \sum_k G_{ik}x_k$ is a sum of independent Gaussians, hence Gaussian, with mean 0 and variance $\sum_k x_k^2 = 1$, so $(\boldsymbol{G}\boldsymbol{x})_i \sim \mathcal{N}(0,1)$, independent across the m rows. Writing $z_i = (\boldsymbol{G}\boldsymbol{x})_i$, the measured energy is

$$\|\boldsymbol{A}\boldsymbol{x}\|_2^2 = \frac{1}{m}\sum_{i=1}^{m} z_i^2 = \frac{1}{m}\,\chi^2, \qquad \chi^2 := \sum_{i=1}^{m} z_i^2,$$

a chi-squared variable with m degrees of freedom and mean m. The upper tail is $\mathbb{P}[\chi^2 > m(1+\varepsilon)]$.

Markov is not enough. Markov's inequality gives only $\mathbb{P}[\chi^2 > m(1+\varepsilon)] \le \mathbb{E}[\chi^2]/(m(1+\varepsilon)) = 1/(1+\varepsilon)$, a bound with no m in it, useless for concentration. The fix is to exponentiate before applying Markov.

The Chernoff step. For any $\lambda \in (0, \frac{1}{2})$, monotonicity of $t \mapsto e^{\lambda t}$ gives $\mathbb{P}[\chi^2 > m(1+\varepsilon)] = \mathbb{P}[e^{\lambda\chi^2} > e^{\lambda m(1+\varepsilon)}] \le e^{-\lambda m(1+\varepsilon)}\mathbb{E}[e^{\lambda\chi^2}]$. By independence the moment generating function factors, $\mathbb{E}[e^{\lambda\chi^2}] = \prod_i \mathbb{E}[e^{\lambda z_i^2}] = (\mathbb{E}[e^{\lambda z^2}])^m$, and a Gaussian integral evaluates the single factor: for $\lambda < \frac{1}{2}$,

$$\mathbb{E}[e^{\lambda z^2}] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-z^2(1-2\lambda)/2}\,dz = \frac{1}{\sqrt{1-2\lambda}}.$$

Hence $\mathbb{P}[\chi^2 > m(1+\varepsilon)] \le e^{-\lambda m(1+\varepsilon)}(1-2\lambda)^{-m/2} =: f(\lambda)$.

Optimize λ. Minimizing $\log f(\lambda) = -\lambda(1+\varepsilon)m - \frac{m}{2}\log(1-2\lambda)$ sets $-(1+\varepsilon)m + \frac{m}{1-2\lambda} = 0$, so $1-2\lambda = \frac{1}{1+\varepsilon}$ and $\lambda^\star = \frac{\varepsilon}{2(1+\varepsilon)} \in (0, \frac{1}{2})$ (the second derivative is positive, so this is the minimum). Substituting, the exponential factor becomes $e^{-\lambda^\star(1+\varepsilon)m} = e^{-\varepsilon m/2}$ and the polynomial factor becomes $(1-2\lambda^\star)^{-m/2} = (1+\varepsilon)^{m/2}$, giving

$$\mathbb{P}[\chi^2 > m(1+\varepsilon)] \;\le\; e^{-\varepsilon m/2}\,(1+\varepsilon)^{m/2}.$$

Taylor cleanup. Writing $(1+\varepsilon)^{m/2} = e^{(m/2)\log(1+\varepsilon)}$ and using $\log(1+\varepsilon) \le \varepsilon - \frac{\varepsilon^2-\varepsilon^3}{2}$ for $\varepsilon \in (0,1)$, the $\pm\varepsilon m/2$ terms cancel and leave

$$\mathbb{P}[\chi^2 > m(1+\varepsilon)] \;\le\; e^{-(\varepsilon^2-\varepsilon^3)m/4}.$$

The lower tail $\mathbb{P}[\chi^2 < m(1-\varepsilon)]$ obeys the same bound by an identical argument with $\lambda < 0$. Summing the two tails gives the factor of 2 and the claim. ∎

The bound shrinks like $e^{-c(\varepsilon)m}$ with $c(\varepsilon) \approx \varepsilon^2/4$ for small ε. The same statement holds for Bernoulli and general sub-Gaussian ensembles, with a different constant c, by the same exponential-moment method; the Gaussian case is the cleanest instance. We abbreviate the two-sided bound as

$$\mathbb{P}\Big[\,\big|\,\|\boldsymbol{A}\boldsymbol{x}\|_2^2 - \|\boldsymbol{x}\|_2^2\,\big| > t\,\|\boldsymbol{x}\|_2^2\Big] \le 2e^{-\tilde{c}\,t^2 m}. \tag{9.1}$$

9.3 Johnson–Lindenstrauss: Preserving a Finite Set

The point estimate already proves a famous theorem. If we apply it to the *differences* of a finite set of points and take a union bound, we learn that a random projection preserves all pairwise distances at once.

Theorem 9.3 (Johnson–Lindenstrauss). Let $\boldsymbol{p}_1, \dots, \boldsymbol{p}_n \in \mathbb{R}^N$ and $\varepsilon \in (0, \frac{1}{2})$. If $m \geq C\varepsilon^{-2} \log n$ for a universal constant C, then with high probability a Gaussian matrix $\boldsymbol{A} \in \mathbb{R}^{m \times N}$ satisfies, for all i, j,

$$(1-\varepsilon)\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2 \leq \|\boldsymbol{A}(\boldsymbol{p}_i - \boldsymbol{p}_j)\|_2^2 \leq (1+\varepsilon)\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2.$$

Proof. Apply Eq. (9.1) to each of the $\binom{n}{2} < n^2$ difference vectors $\boldsymbol{p}_i - \boldsymbol{p}_j$. The probability that any one fails is at most $2e^{-\tilde{c}\varepsilon^2 m}$, so by the union bound the probability that any pair fails is at most $n^2 \cdot 2e^{-\tilde{c}\varepsilon^2 m}$, which is below 1 once $m > \tilde{c}^{-1}\varepsilon^{-2} \cdot 2\log n$. ∎

The striking feature is that the target dimension $m \sim \varepsilon^{-2} \log n$ does not depend on the ambient dimension N at all: any n points, however high-dimensional, fit into $O(\log n)$ dimensions with distances nearly intact.

Example 9.4 (A million points from a billion dimensions). Take $n = 10^6$ points living in $N = 10^9$ dimensions, and ask to preserve all pairwise distances to within $\varepsilon = 0.1$. The target dimension is $m \sim \varepsilon^{-2} \log n = 100 \cdot \ln(10^6) \approx 100 \cdot 13.8 \approx 1380$, on the order of a few thousand once the constant is included. So a million points from a *billion*-dimensional space embed into a few thousand dimensions with every pairwise distance nearly intact, and the ambient 10^9 never enters the count: doubling it to 2×10^9 changes nothing. Only the number of points matters, and only through its logarithm. This dimension-free compression is why random projections are the workhorse of high-dimensional data analysis. ◊

The same union-bound idea drives the RIP proof, but with one difficulty. The set of s-sparse unit vectors is not finite; it is a continuum. Overcoming that is the role of covering numbers.

9.4 Covering Numbers

To run a union bound over a continuum, we first replace it by a finite set that approximates it, an *ε-net*.

Definition 9.5 (ε-net and covering number). A subset $\mathcal{U} \subseteq K$ of a metric space is a **ρ-net** of K if every point of K lies within distance ρ of some point of $\mathcal{U}$. The **covering number** $\mathcal{N}(K, \rho)$ is the size of the smallest such net.

The unit ball of an s-dimensional space has a net whose size is exponential in s but, crucially, finite.

Proposition 9.6 (Covering the unit ball). The unit ball $\mathcal{B}_s \subseteq \mathbb{R}^s$ has, for any $\rho \in (0, 1)$,

$$\mathcal{N}(\mathcal{B}_s, \rho) \;\leq\; \left(1 + \tfrac{2}{\rho}\right)^s.$$

Proof. Let $\mathcal{U} = \{\boldsymbol{u}_1, \dots, \boldsymbol{u}_K\}$ be a maximal ρ-separated subset of $\mathcal{B}_s$, meaning the points are pairwise more than ρ apart and no further point can be added. Maximality forces $\mathcal{U}$ to be a ρ-net: any uncovered point could be added. Now the balls of radius $\rho/2$ about the $\boldsymbol{u}_i$ are pairwise disjoint (the centers are more than ρ apart) and all lie inside the ball of radius $1 + \rho/2$ about the origin. Comparing volumes, and writing $V_s r^s$ for the volume of a radius-r ball,

$$K\, V_s(\rho/2)^s \;\leq\; V_s(1+\rho/2)^s \quad\Longrightarrow\quad K \;\leq\; \left(\frac{1+\rho/2}{\rho/2}\right)^s = \left(1 + \frac{2}{\rho}\right)^s. \qquad ∎$$

The bound is exponential in the dimension s, but its logarithm is only linear in s, and that linear factor is exactly what produces the $s\log(N/s)$ in the final count. For instance a $\rho = \frac{1}{2}$ net of $\mathcal{B}_5$ has at most $5^5 = 3125$ points, whose logarithm $5\ln 5 \approx 8$ is linear in the dimension; multiplying an e^{-cm} point estimate by this many net points still leaves e^{8-cm}, tiny once $m \gtrsim s$ (Fig. 9.1).

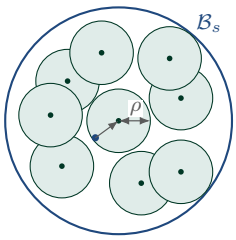


Figure 9.1. A ρ-net of the unit ball: finitely many points whose radius-ρ balls cover the whole continuum. Every point of $\mathcal{B}_s$ (e.g. the navy dot) is within ρ of a net center, so a union bound over the net controls the entire ball. The net has at most $(1+2/\rho)^s$ points.

9.5 The Three-Lemma Proof

We can now assemble the proof that a random matrix has a small restricted isometry constant. Fix a support S with $|S| = s$ and consider the deviation operator

$$\boldsymbol{B} = \boldsymbol{A}_S^* \boldsymbol{A}_S - \boldsymbol{I}_s,$$

a symmetric $s \times s$ matrix. By Lemma 8.2, controlling δ_s over all supports means controlling $\|\boldsymbol{B}\|_{2\to 2}$, and for symmetric $\boldsymbol{B}$ the spectral norm is the largest absolute Rayleigh quotient,

$$\|\boldsymbol{B}\|_{2\to 2} = \max_{\|\boldsymbol{x}\|_2 = 1} |\langle \boldsymbol{B}\boldsymbol{x}, \boldsymbol{x}\rangle|, \qquad \langle \boldsymbol{B}\boldsymbol{x}, \boldsymbol{x}\rangle = \|\boldsymbol{A}_S \boldsymbol{x}\|_2^2 - \|\boldsymbol{x}\|_2^2.$$

So the Rayleigh quotient is exactly the length distortion, and the point estimate controls it at any single $\boldsymbol{x}$.

Lemma 1 (point estimate). By Eq. (9.1), for a fixed unit vector $\boldsymbol{x}$ supported on S, $\mathbb{P}[\,|\langle \boldsymbol{B}\boldsymbol{x}, \boldsymbol{x}\rangle| > t\,] \le 2e^{-\tilde{c}t^2 m}$.

Lemma 2 (net to sphere). A net controls the whole sphere. Take a ρ-net $\mathcal{U}$ of the unit ball $\mathcal{B}_s$ with $\rho = \frac{1}{4}$, so $|\mathcal{U}| \le 9^s$ by Proposition 9.6. Suppose the point estimate holds at every net point: $|\langle \boldsymbol{B}\boldsymbol{u}, \boldsymbol{u}\rangle| \le t$ for all $\boldsymbol{u} \in \mathcal{U}$. For an arbitrary unit $\boldsymbol{x}$ with nearest net point $\boldsymbol{u}$ (so $\|\boldsymbol{x} - \boldsymbol{u}\|_2 \le \rho$), the polarization identity for the symmetric $\boldsymbol{B}$,

$$\langle \boldsymbol{B}\boldsymbol{x}, \boldsymbol{x}\rangle = \langle \boldsymbol{B}\boldsymbol{u}, \boldsymbol{u}\rangle + \langle \boldsymbol{B}(\boldsymbol{x}+\boldsymbol{u}), \boldsymbol{x}-\boldsymbol{u}\rangle\,,$$

bounds the distortion at $\boldsymbol{x}$ by the distortion at $\boldsymbol{u}$ plus a correction. Using $\|\boldsymbol{x}+\boldsymbol{u}\|_2 \le 2$ and $\|\boldsymbol{x}-\boldsymbol{u}\|_2 \le \rho$,

$$|\langle \boldsymbol{B}\boldsymbol{x}, \boldsymbol{x}\rangle| \le t + \|\boldsymbol{B}\|_{2\to 2} \cdot 2\rho.$$

Taking the maximum of the left side over $\boldsymbol{x}$ gives $\|\boldsymbol{B}\|_{2\to 2}$ on both sides, a self-referential inequality $\|\boldsymbol{B}\|_{2\to 2} \le t + 2\rho\|\boldsymbol{B}\|_{2\to 2}$. Solving with $\rho = \frac{1}{4}$,

$$\|\boldsymbol{B}\|_{2\to 2} \;\le\; \frac{t}{1-2\rho} = 2t.$$

So if the net is controlled to tolerance t, the entire support has $\delta_S \le 2t$. To force $\delta_S < \delta$ we take $t = \delta/2$.

Lemma 3 (union over supports). Combine the two with two union bounds. First, over the 9^s net points of one support, Lemma 9.2 fails somewhere with probability at most $9^s \cdot 2e^{-\tilde{c}t^2 m}$. Second, over all $\binom{N}{s} \le (eN/s)^s$ supports. The total failure probability that *any* support has $\delta_S \ge \delta$ is therefore at most

$$\binom{N}{s} \cdot 2 \cdot 9^s\, e^{-\tilde{c}(\delta/2)^2 m} \;\le\; 2\exp\!\Big(s\log\tfrac{eN}{s} + s\log 9 - \tfrac{\tilde{c}\delta^2}{4}m\Big).$$

Assemble. The exponent is negative, and the whole probability is exponentially small, as soon as the negative term dominates the two positive ones:

$$m \;\geq\; \frac{4}{\tilde{c}\,\delta^2}\Big(s\log\tfrac{eN}{s} + s\log 9 + \eta\Big) \;=\; C\,\delta^{-2}\,s\,\log(N/s) + \cdots,$$

which gives failure probability at most $2e^{-\eta}$. This is exactly the keybox theorem: $m \gtrsim \delta^{-2} s\log(N/s)$ rows force $\delta_s < \delta$ with overwhelming probability.

The grand chain

Reading the proof end to end: *one* sparse vector is preserved by concentration (Chernoff), *a net's worth* are preserved by a union bound over 9^s points, the *whole sphere* of a support is preserved by the polarization trick, and *all* $\binom{N}{s}$ supports are preserved by a final union bound. The three exponential factors, e^{-cm} down against 9^s and $(eN/s)^s$ up, balance exactly when $m \approx s\log(N/s)$.

9.6 Optimality and the Complete Picture

The measurement count $m \approx s\log(N/s)$ is not just good, it is essentially the best possible. An information-theoretic argument shows that *any* method that recovers all s-sparse signals in $\mathbb{R}^N$ needs at least $m \gtrsim s\log(N/s)$ measurements: there are roughly $\binom{N}{s} \approx (N/s)^s$ distinct supports to distinguish, and each linear measurement supplies a bounded amount of information, so about $s\log(N/s)$ of them are required just to identify the support. The random construction meets this floor up to the constant C. No deterministic construction is known to do as well at all sparsity levels, which is the deepest reason the field relies on randomness.

Figure 9.2 places the three regimes side by side. The trivial bound is $m = N$. Coherence, the only thing computable for a fixed matrix, certifies recovery but only at $m \gtrsim s^2$. Random matrices, certified not individually but as an ensemble, reach the information-theoretic optimum $m \approx s\log(N/s)$.

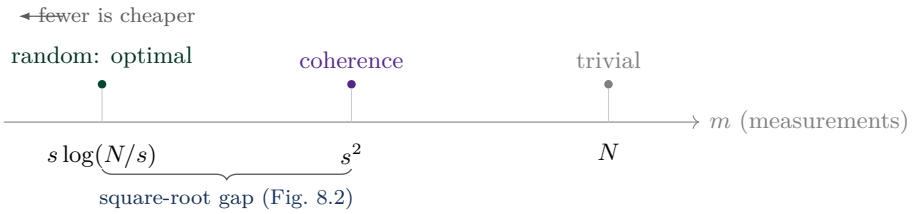


Figure 9.2. Measurement counts for recovering s-sparse signals in $\mathbb{R}^N$ (schematic, $s \ll N$). Random matrices reach the information-theoretic optimum $s\log(N/s)$; coherence certificates are stuck a square higher at s^2 (the braced gap); the trivial bound is N. Leftward is cheaper.

This closes the central question of the subject. A signal with s degrees of freedom in an N-dimensional ambient space can be recovered from about $s\log(N/s)$ random linear measurements, by a tractable convex program or a greedy algorithm, exactly and robustly. Chapter 10 returns to practice, where pure randomness is too expensive to store and apply, and builds *structured* random matrices that keep the guarantee while admitting a fast transform.

Notes and References

The concentration-based proof that random matrices satisfy the restricted isometry property is due to Baraniuk, Davenport, DeVore, and Wakin [3], which is the "simple proof" the three-lemma structure follows; the original random-matrix guarantees are Candès and Tao [12]. The Johnson–Lindenstrauss lemma is from [32]; its tight link to the restricted isometry property is in [3]. The chi-squared Chernoff computation and the covering-number bounds are standard high-dimensional probability, treated thoroughly in Vershynin [51]. The information-theoretic lower bound $m \gtrsim s \log(N/s)$ and the optimality of the random construction are discussed in Foucart and Rauhut [28]. The sub-Gaussian and sub-exponential machinery that extends the Gaussian point estimate to general ensembles, including Bernstein's inequality, is again in Vershynin [51].

Exercises

Exercise 9.1. Carry out the lower-tail half of Lemma 9.2: for $\lambda < 0$, bound $\mathbb{P}[\chi^2 < m(1-\varepsilon)]$ by the same Chernoff method and confirm the exponent $-(\varepsilon^2 - \varepsilon^3)m/4$.

Exercise 9.2. Use Theorem 9.3 to find the target dimension m needed to preserve all pairwise distances among $n = 10^6$ points to within $\varepsilon = 0.1$. Note that the answer does not depend on the ambient dimension N.

Exercise 9.3. Verify the covering bound of Proposition 9.6 for $s = 1$ (the interval $[-1, 1]$) and $\rho = \frac{1}{2}$ by exhibiting an explicit ρ-net and comparing its size to $(1 + 2/\rho)^1 = 5$.

Exercise 9.4. In Lemma 2 of Section 9.5, redo the self-referential bound with a general net radius ρ and find the resulting factor relating $\|\boldsymbol{B}\|_{2\to 2}$ to t. Why is $\rho = \frac{1}{4}$ a convenient choice, and what goes wrong as $\rho \to \frac{1}{2}$?

Exercise 9.5. Trace the constants through the assembly in Section 9.5 to express C in $m \geq C\delta^{-2} s \log(N/s)$ in terms of $\tilde{c}$. For $\delta = \frac{1}{3}$ (the recovery threshold) and a rough $\tilde{c} = \frac{1}{16}$, estimate C.

CHAPTER 10

Structured Random Matrices and Sensing in a Basis

Keep the randomness, but make it fast.

Chapter 9 proved that a fully random matrix achieves the optimal measurement count, but a dense Gaussian matrix is a poor engineering choice: storing mN real numbers and computing $\boldsymbol{A}\boldsymbol{x}$ in $O(mN)$ time is prohibitive when N runs to millions, and no physical sensor produces independent Gaussian inner products on demand. Real systems sense in a fixed basis, the Fourier basis of an MRI scanner or the binary patterns of a single-pixel camera, and acquire only a random subset of those measurements. This chapter develops the theory of such *structured* random matrices. The key requirement is *incoherence* between the basis you sense in and the basis the signal is sparse in, and the deepest fact behind it is a sparse uncertainty principle: a signal cannot be concentrated in two incoherent bases at once. We prove that principle in full, define the bounded orthonormal systems that formalize "good sensing bases," and state the recovery theorem that keeps nearly the optimal measurement count while admitting a fast transform.

10.1 Sensing in One Basis, Sparse in Another

A signal is rarely sparse in its raw coordinates; it is sparse in some transform domain. An image is sparse in wavelets, an audio clip in frequency. Write the signal as $\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{\beta}$, where $\boldsymbol{\Psi}$ is the sparsifying basis (its columns are the wavelet or Fourier atoms) and $\boldsymbol{\beta}$ is the sparse coefficient vector. A physical sensor, meanwhile, measures inner products against a fixed *sensing* basis $\boldsymbol{\Phi}$, and in compressed sensing it measures only a random subset of m of them. The composite sensing matrix is

$$\boldsymbol{A} = \boldsymbol{R}\,\boldsymbol{\Phi}^{*}\boldsymbol{\Psi},$$

where $\boldsymbol{R}$ selects m random rows. Recovery succeeds when the columns of this $\boldsymbol{A}$ are nearly orthonormal, and whether they are depends on how aligned the two bases are. The governing quantity is the coherence between them.

Definition 10.1 (Mutual coherence of two bases). For orthonormal bases $\boldsymbol{\Phi} = \{\boldsymbol{\phi}_i\}$ and $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_j\}$ of $\mathbb{C}^N$, the **mutual coherence** is

$$\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) = \max_{i,j} |\langle \boldsymbol{\phi}_i, \boldsymbol{\psi}_j \rangle|.$$

It lies in $[1/\sqrt{N},\, 1]$: it is at least $1/\sqrt{N}$ for any two orthonormal bases, and equals 1 when a sensing vector coincides with a sparsity atom.

When μ is small the two bases are *incoherent*: every sensing vector spreads its energy across all sparsity atoms, so a signal concentrated on a few atoms still shows up in every measurement. The extreme is the spike basis (raw coordinates) against the Fourier basis, where $\mu = 1/\sqrt{N}$, the smallest possible. The reason incoherence is exactly the right condition is a uncertainty principle, which we prove next.

10.2 The Sparse Uncertainty Principle

A single signal cannot be sparse in two incoherent bases simultaneously. If it has few nonzero coordinates in one, it must spread out in the other. This is the discrete analogue of the Heisenberg principle that a function and its Fourier transform cannot both be concentrated.

Theorem 10.2 (Sparse uncertainty principle). Let $\mathbf{\Phi}, \mathbf{\Psi}$ be orthonormal bases of $\mathbb{C}^N$ with coherence $\mu = \mu(\mathbf{\Phi}, \mathbf{\Psi})$, and let $\boldsymbol{v} \neq \mathbf{0}$ have coordinate vectors $\boldsymbol{\alpha}$ in $\mathbf{\Phi}$ and $\boldsymbol{\beta}$ in $\mathbf{\Psi}$. Then

$$\|\boldsymbol{\alpha}\|_0 \cdot \|\boldsymbol{\beta}\|_0 \;\geq\; \frac{1}{\mu^2}, \qquad \text{and} \qquad \|\boldsymbol{\alpha}\|_0 + \|\boldsymbol{\beta}\|_0 \;\geq\; \frac{2}{\mu}.$$

The proof is a sequence of four elementary inequalities, each squeezing the double sum that expresses $\|\boldsymbol{v}\|_2^2$ in both bases at once.

Proof. Let $S = \operatorname{supp}(\boldsymbol{\alpha})$ and $T = \operatorname{supp}(\boldsymbol{\beta})$, so $|S| = \|\boldsymbol{\alpha}\|_0$ and $|T| = \|\boldsymbol{\beta}\|_0$. Writing $\boldsymbol{v} = \sum_{i \in S} \alpha_i \boldsymbol{\phi}_i = \sum_{j \in T} \beta_j \boldsymbol{\psi}_j$ and expanding one copy in each basis,

$$\|\boldsymbol{v}\|_2^2 = \langle \boldsymbol{v}, \boldsymbol{v} \rangle = \sum_{i \in S} \sum_{j \in T} \alpha_i \, \overline{\beta_j} \, \langle \boldsymbol{\phi}_i, \boldsymbol{\psi}_j \rangle \, .$$

Take absolute values, apply the triangle inequality, and bound each inner product by μ:

$$\|\boldsymbol{v}\|_2^2 \leq \sum_{i \in S} \sum_{j \in T} |\alpha_i| \, |\beta_j| \, \mu = \mu \, \|\boldsymbol{\alpha}_S\|_1 \, \|\boldsymbol{\beta}_T\|_1 .$$

By Cauchy–Schwarz, a vector with $|S|$ nonzeros satisfies $\|\boldsymbol{\alpha}_S\|_1 \leq \sqrt{|S|}\, \|\boldsymbol{\alpha}_S\|_2$, and likewise for $\boldsymbol{\beta}_T$. By Parseval, the orthonormal expansions preserve length, so $\|\boldsymbol{\alpha}_S\|_2 = \|\boldsymbol{\beta}_T\|_2 = \|\boldsymbol{v}\|_2$. Substituting,

$$\|\boldsymbol{v}\|_2^2 \leq \mu \sqrt{|S| \, |T|} \, \|\boldsymbol{v}\|_2^2 .$$

Since $\boldsymbol{v} \neq \mathbf{0}$ we may cancel $\|\boldsymbol{v}\|_2^2$, leaving $1 \leq \mu \sqrt{|S||T|}$, that is $|S||T| \geq 1/\mu^2$, the product bound. The sum bound follows from the arithmetic-geometric mean inequality $\sqrt{|S||T|} \leq (|S| + |T|)/2$, which with $\sqrt{|S||T|} \geq 1/\mu$ gives $|S| + |T| \geq 2/\mu$. ∎

Example 10.3 (Spikes versus Fourier). Take $\mathbf{\Phi}$ the spike basis and $\mathbf{\Psi}$ the Fourier basis, where $\mu = 1/\sqrt{N}$. The uncertainty principle gives $\|\boldsymbol{\alpha}\|_0 \cdot \|\boldsymbol{\beta}\|_0 \geq N$ and $\|\boldsymbol{\alpha}\|_0 + \|\boldsymbol{\beta}\|_0 \geq 2\sqrt{N}$. A single spike, 1-sparse in time, is maximally spread in frequency: its Fourier representation uses all N coefficients, exactly meeting $1 \cdot N = N$. A signal cannot be sharply localized in both time and frequency, which is precisely what lets random time samples capture a frequency-sparse signal. ◊

Example 10.4 (The Dirac comb meets the bound exactly). Fix $N = 64$, so spike-Fourier coherence is $\mu = 1/\sqrt{64} = 1/8$ and the sum bound reads $\|\boldsymbol{\alpha}\|_0 + \|\boldsymbol{\beta}\|_0 \geq 2/\mu = 16$. Check three signals against it. A pure spike $\boldsymbol{e}_5$ has one time nonzero and, being flat in frequency, 64 Fourier nonzeros: $1 + 64 = 65 \geq 16$. A single real cosine has 2 Fourier nonzeros (at ± 1) but is dense in time (two of the 64 samples land on zero crossings): $62 + 2 = 64 \geq 16$. The extremal case is a *Dirac comb* of $\sqrt{N} = 8$ equally spaced spikes; its discrete Fourier transform is another Dirac comb, of 8 spikes at the reciprocal spacing, so $\|\boldsymbol{\alpha}\|_0 = \|\boldsymbol{\beta}\|_0 = 8$ and $8 + 8 = 16$, hitting the bound exactly. The comb is the most balanced signal in time and frequency: any signal sparser than 8 spikes in time must spend more than 8 coefficients in frequency. ◊

> The bound is tight, and the four steps reveal where: equality needs flat-magnitude coefficients (Cauchy–Schwarz), inner products all equal to μ (the coherence step), aligned phases (the triangle inequality), and equal sparsities $|S| = |T|$ (the arithmetic-geometric mean). The spike-Fourier pair with a flat-sparse signal hits all four, so no universal improvement is possible. The lesson for sensing is the converse: choose the sensing and sparsity bases to be *maximally incoherent*, and the uncertainty principle guarantees a sparse signal cannot hide from the measurements.

10.3 Bounded Orthonormal Systems

The right abstraction for "a good sensing basis" is a bounded orthonormal system, which packages orthonormality with the spreading property that incoherence requires.

Definition 10.5 (Bounded orthonormal system). A **bounded orthonormal system** on a domain D with probability measure ν is a family $\{\psi_1, \dots, \psi_N\}$ of functions $\psi_k : D \to \mathbb{C}$ that are orthonormal, $\int_D \psi_j \overline{\psi_k}\, d\nu = \delta_{jk}$, and uniformly bounded: there is a constant $K \geq 1$ with $\sup_{t \in D} |\psi_k(t)| \leq K$ for every k.

Proposition 10.6 (The bound is at least one). Every bounded orthonormal system has $K \geq 1$.

Proof. Orthonormality at $j = k$ gives $1 = \int_D |\psi_k|^2\, d\nu \leq \left(\sup_t |\psi_k(t)|^2\right) \int_D d\nu \leq K^2$, using that ν is a probability measure, so $K \geq 1$. ∎

The constant K measures concentration. When $K = 1$ every function has magnitude one everywhere on D: maximally spread, the ideal sensing basis. When $K = \sqrt{N}$ a function piles all its mass on one point: maximally concentrated, useless for random sampling.

Example 10.7 (The spectrum of K). The **Fourier** functions $\psi_k(t) = e^{2\pi i k t}$ satisfy $|\psi_k(t)| = 1$ everywhere, so $K = 1$; the same holds for the discrete Fourier transform and for the ± 1-valued **Walsh–Hadamard** transform, the basis the single-pixel camera actually uses. At the other extreme the **spike** basis has ψ_k equal to $\sqrt{N}$ at one point and 0 elsewhere, giving the worst possible $K = \sqrt{N}$. The bound K and the coherence are linked: sensing in a K-bounded system against the spike basis gives coherence $\mu = K/\sqrt{N}$, so $K = 1$ is exactly maximal incoherence. ◊

10.4 The Structured Random Matrix

A structured random matrix samples a random subset of rows from a bounded orthonormal system. For the discrete Fourier transform $\boldsymbol{F}$, draw m row indices $\ell_1, \dots, \ell_m$ uniformly at random and set

$$A_{ij} = \frac{1}{\sqrt{m}}\, F_{\ell_i,\, j} = \frac{1}{\sqrt{mN}}\, e^{2\pi i\, \ell_i j / N}.$$

The randomness is only in *which* rows are kept; the rows themselves are the deterministic, structured Fourier modes. This is what makes the matrix practical: applying it is a subsampled fast Fourier transform, $O(N \log N)$ time and $O(N)$ storage, against $O(mN)$ for a dense Gaussian.

Example 10.8 (A partial DFT on a spike, by hand). Take $N = 8$ with primitive root $\omega = e^{2\pi i/8} = e^{i\pi/4}$, and sample the three rows $\{1, 3, 6\}$, so $\boldsymbol{A}$ is the 3×8 matrix with $A_{kj} = \omega^{\ell_k j}$ (dropping the $1/\sqrt{8}$ scale), $\ell \in \{1, 3, 6\}$. Apply it to the 1-sparse spike $\boldsymbol{x} = \boldsymbol{e}_3$: only the $j = 3$ column survives, giving

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{e}_3 = \left(\omega^{1\cdot 3},\ \omega^{3\cdot 3},\ \omega^{6\cdot 3}\right)^{\top} = \left(\omega^{3},\ \omega^{9},\ \omega^{18}\right)^{\top} = \left(\omega^{3},\ \omega^{1},\ \omega^{2}\right)^{\top},$$

reducing exponents mod 8. Every entry has magnitude 1: the lone spike contributes *equally* to all three measurements, so any subset of rows sees it. Contrast the spike basis: sampling rows $\{1, 3, 6\}$ of the identity and applying to $\boldsymbol{e}_3$ gives $(0, 1, 0)^{\top}$, nonzero only where a sampled row happens to land on the spike. Almost every random subset of identity rows misses the spike entirely; the Fourier subset never does. ◊

Example 10.9 (Why the spike basis fails and Fourier succeeds). Sense a single spike $\boldsymbol{x} = \boldsymbol{e}_j$. Through the partial Fourier matrix every measurement is $A_{ij} = \frac{1}{\sqrt{mN}} e^{2\pi i \ell_i j / N}$, of equal magnitude: the spike's energy spreads across *all* m measurements, so any random subset of rows sees it. Through a subsampled spike basis (rows of the identity), the same spike appears in a measurement only if that row happens to be the spike's location; almost every random subset misses it entirely. This contrast, energy spread under an incoherent transform versus energy hidden under a coherent one, is the whole reason structured sensing works, and it is the $K = 1$ versus $K = \sqrt{N}$ distinction in action. ◊

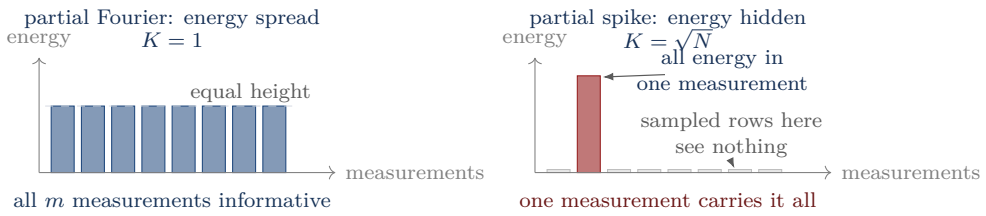


Figure 10.1. The same spike sensed two ways. Through a partial Fourier (or Walsh–Hadamard) operator its energy spreads evenly across every measurement (left, $K = 1$), so any random subset is informative. Through a partial spike basis the energy sits in a single measurement (right, $K = \sqrt{N}$), which a random subset usually misses. This is the $K = 1$ versus $K = \sqrt{N}$ contrast in action.

The recovery guarantee for these matrices is nearly as good as for Gaussians, with the bound K entering as the price of structure.

Recovery from a bounded orthonormal system

Let $\boldsymbol{A}$ be the m-row structured random matrix of a bounded orthonormal system with constant K. If

$$m \gtrsim K^2 \, s \, (\log s)^2 \, \log N,$$

then with high probability $\boldsymbol{A}$ satisfies the restricted isometry property of order s, and ℓ_1 minimization recovers every s-sparse signal.

For $K = 1$, the Fourier and Walsh–Hadamard case, the count is $m \gtrsim s \, (\log s)^2 \log N$, matching the Gaussian rate $s \log(N/s)$ up to logarithmic factors but with a fast transform and a real sensor behind it. The factor K^2 is the exact penalty for concentration: a more localized basis needs proportionally more measurements, and the spike basis with $K^2 = N$ needs $m \gtrsim N$, no compression at all, as it must.

In the Problem Sets

Problem Sets 5 and 6 (Chapters 24 and 25) build exactly these matrices. You construct the partial discrete Fourier and Walsh–Hadamard operators, apply them with the fast transform rather than a dense multiply, and reconstruct images from a random fraction of their coefficients, watching reconstruction quality track the $m \gtrsim s \, (\log s)^2 \log N$ prediction. The accompanying notebooks recover the standard test images from undersampled frequency data, the software model of an accelerated MRI scan.

10.5 Applications and the Practical Picture

Three of the applications from Chapter 1 are structured-random sensing in disguise. **Accelerated MRI** measures Fourier coefficients of the image at randomly chosen frequencies, exactly the partial-DFT construction, and recovers an image that is sparse in wavelets or in its gradient (the total-variation model of Chapter 7). The **single-pixel camera** measures inner products against random Walsh–Hadamard patterns, a real-valued bounded orthonormal system that a digital micromirror array can realize physically. **Hyperspectral imaging** uses the same idea across wavelength bands. In each, the sensing basis is fixed by the physics, the sparsity basis is fixed by the signal class, and the only design freedom is which measurements to take, chosen at random.

A further refinement makes even the dense Gaussian fast. The *fast Johnson–Lindenstrauss transform* factors a near-Gaussian sensing operator as a sparse random matrix times a fast unitary transform times a random sign-flip, $\boldsymbol{A} = \boldsymbol{PHD}$, which preserves the distance-preserving guarantee of Theorem 9.3 while costing $O(N \log N)$ to apply. The recurring theme is the chapter title: keep the randomness that makes the guarantee work, but wrap it around a structured transform that makes it cheap. With this, the theory of Chapter 1 through Chapter 9 becomes a deployable sensing system, and Part IV is complete.

Notes and References

The sparse uncertainty principle for two bases is due to Donoho and Elad [21], building on the time-frequency uncertainty principle; the product form $|S||T| \geq 1/\mu^2$ and the spike-Fourier extremal case are there. Bounded orthonormal systems and the recovery theorem $m \gtrsim K^2 s\,(\log s)^2 \log N$ are from the work of Rudelson and Vershynin [45] and Candès and Tao [12], with the definitive treatment in Foucart and Rauhut [28], whose Chapter 12 develops the restricted isometry property for structured random matrices in full. The single-pixel camera and the Walsh–Hadamard sensing model are from Duarte et al. [23]; compressed sensing MRI from Lustig, Donoho, and Pauly [36]. The fast Johnson–Lindenstrauss transform is Ailon and Chazelle [2].

Exercises

Exercise 10.1. Verify that $\mu(\mathbf{\Phi}, \mathbf{\Psi}) \geq 1/\sqrt{N}$ for any two orthonormal bases. *Hint: each $\boldsymbol{\phi}_i$ is a unit vector expanded in the basis $\mathbf{\Psi}$, so $\sum_j |\langle \boldsymbol{\phi}_i, \boldsymbol{\psi}_j \rangle|^2 = 1$ over N terms.*

Exercise 10.2. For the spike-Fourier pair in $\mathbb{C}^N$, exhibit a signal achieving equality $\|\boldsymbol{\alpha}\|_0 \cdot \|\boldsymbol{\beta}\|_0 = N$ in Theorem 10.2, and one achieving the sum bound $2\sqrt{N}$.

Exercise 10.3. Confirm Proposition 10.6 and compute K for the ± 1-valued Walsh–Hadamard system after the probability-measure normalization. Why is it $K = 1$ and not $K = \sqrt{N}$?

Exercise 10.4. For an 8×8 discrete Fourier transform, write the 3×8 partial matrix from sampling rows $\{1, 3, 6\}$ and compute its action on $\boldsymbol{e}_3$. Confirm every measurement has equal magnitude, and contrast with sampling the same rows of the identity.

Exercise 10.5. Using $m \gtrsim K^2 s \log^4 N$, compare the measurements needed to recover an s-sparse signal through a $K = 1$ Fourier system and through the $K = \sqrt{N}$ spike system. Explain why the latter forces $m \gtrsim N$ and amounts to no compression.

PART V

Low-Rank and Factorization Models

CHAPTER 11

Low-Rank Recovery and Matrix Completion

Sparsity for vectors, low rank for matrices: the same idea, one dimension up.

The first half of the book recovered sparse vectors. The second half repeats the entire story for matrices, and the translation is exact. A sparse vector has few nonzero entries; a *low-rank* matrix has few nonzero singular values. The count of nonzeros, $\|\cdot\|_0$, becomes the rank; the convex surrogate $\|\cdot\|_1$ becomes the nuclear norm; the null space property becomes a matrix null space property on singular values; and exact recovery from few measurements becomes the matrix completion that won the Netflix Prize. This chapter builds that dictionary, proves the matrix recovery theorem with the same two-direction argument as the vector case, and states the Candès–Recht theorem on completing a low-rank matrix from a random handful of its entries. The singular value decomposition of Chapter 2 is the engine throughout.

11.1 From Sparse Vectors to Low-Rank Matrices

Recall from Chapter 2 that any matrix $\boldsymbol{X} \in \mathbb{C}^{N_1 \times N_2}$ has a singular value decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, $n = \min(N_1, N_2)$, and that the rank is the number of nonzero singular values. Collect them into the *singular value vector* $\boldsymbol{\sigma}(\boldsymbol{X}) = (\sigma_1, \ldots, \sigma_n)$. Then the entire vector dictionary carries over by reading sparsity of $\boldsymbol{\sigma}$ as low rank of $\boldsymbol{X}$.

The vector–matrix dictionary

$$\operatorname{rank}(\boldsymbol{X}) = \|\boldsymbol{\sigma}(\boldsymbol{X})\|_0, \qquad \|\boldsymbol{X}\|_* = \|\boldsymbol{\sigma}(\boldsymbol{X})\|_1 = \sum_i \sigma_i(\boldsymbol{X}).$$

Rank is the ℓ_0 count of the singular values; the **nuclear norm** is their ℓ_1 sum. A low-rank matrix is exactly a matrix with a sparse singular value vector.

This is more than an analogy; it is the same object viewed through the SVD. A matrix of rank r in $\mathbb{C}^{N_1 \times N_2}$ has only about $r(N_1 + N_2 - r)$ degrees of freedom (the singular vectors and values), far fewer than the $N_1 N_2$ entries, just as an s-sparse vector has about $2s$ degrees of freedom rather than N. Low rank is the matrix form of having few degrees of freedom, and it pervades data: a ratings matrix is low rank because tastes follow a few latent factors, a video is low rank because its frames are nearly the same, a covariance matrix is low rank when the data lives near a low-dimensional subspace.

11.2 Rank Minimization and Its Convex Relaxation

Given a linear measurement operator $\mathcal{A} : \mathbb{C}^{N_1 \times N_2} \to \mathbb{C}^m$ and observations $\boldsymbol{y} = \mathcal{A}(\boldsymbol{X})$ of a low-rank matrix, the ideal recovery problem is to find the matrix of least rank consistent with the data,

$$(\mathrm{PM}_0): \qquad \min_{\boldsymbol{Z}} \ \operatorname{rank}(\boldsymbol{Z}) \quad \text{subject to} \quad \mathcal{A}(\boldsymbol{Z}) = \boldsymbol{y}.$$

This is the matrix analogue of (P_0) and is NP-hard for the same reason: rank, like $\|\cdot\|_0$, is a discontinuous count. Relaxing the count of singular values to their sum gives the tractable surrogate.

Definition 11.1 (Nuclear norm minimization).

$$(\mathrm{PM}_1): \qquad \min_{\boldsymbol{Z}} \ \|\boldsymbol{Z}\|_* \quad \text{subject to} \quad \mathcal{A}(\boldsymbol{Z}) = \boldsymbol{y}.$$

The nuclear norm is convex (it is a norm), and the constraint is linear, so (PM_1) is a convex program; in fact it is a semidefinite program, solvable by standard interior methods. It is the matrix basis pursuit. The Eckart–Young theorem of Chapter 2, that truncating the SVD gives the best low-rank approximation, is the matrix version of hard thresholding, and it plays the same role here that best s-term approximation played for vectors.

11.3 The Matrix Null Space Property

When does (PM_1) recover the true low-rank matrix? The answer is a null space property on singular values, exactly parallel to Theorem 4.10.

Theorem 11.2 (Matrix null space property). Let $\mathcal{A} : \mathbb{C}^{N_1 \times N_2} \to \mathbb{C}^m$ be linear and $r \le n = \min(N_1, N_2)$. The following are equivalent.

(A) **Recovery:** every $\boldsymbol{X}$ with $\operatorname{rank}(\boldsymbol{X}) \le r$ is the unique solution of (PM_1) with $\boldsymbol{y} = \mathcal{A}(\boldsymbol{X})$.

(B) **Matrix NSP:** every nonzero $\boldsymbol{M} \in \ker(\mathcal{A})$ has its singular values, sorted decreasing, obey

$$\sum_{i=1}^{r} \sigma_i(\boldsymbol{M}) \ < \ \sum_{i=r+1}^{n} \sigma_i(\boldsymbol{M}).$$

The condition says every null-space matrix is "top-heavy in the tail": its small singular values outweigh its large ones, so no null-space matrix can be confused with a low-rank one. In the vector case the support was an index set chosen from $\binom{N}{s}$ possibilities, and NSP had to hold for all of them; here the SVD picks the support for us, the top-r singular component, so there is one canonical split per matrix. The proof needs one inequality about how singular values change under a perturbation.

Lemma 11.3 (Singular-value perturbation: Weyl and Mirsky). Let $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{N_1 \times N_2}$ with singular values sorted decreasing, and let $n = \min(N_1, N_2)$. Then:

(i) **(Weyl, per index)** for every i, $\ |\sigma_i(\boldsymbol{A}) - \sigma_i(\boldsymbol{B})| \le \sigma_1(\boldsymbol{A} - \boldsymbol{B}) = \|\boldsymbol{A} - \boldsymbol{B}\|_2$;

(ii) **(Mirsky, summed)** $\ \displaystyle\sum_{i=1}^{n} |\sigma_i(\boldsymbol{A}) - \sigma_i(\boldsymbol{B})| \ \le \ \sum_{i=1}^{n} \sigma_i(\boldsymbol{A} - \boldsymbol{B}) \ = \ \|\boldsymbol{A} - \boldsymbol{B}\|_*$.

Remark 11.4. The naive per-index bound $\sigma_i(\boldsymbol{A} - \boldsymbol{B}) \ge |\sigma_i(\boldsymbol{A}) - \sigma_i(\boldsymbol{B})|$ is *false*: for $\boldsymbol{A} = \left(\begin{smallmatrix} -3 & -1 \\ -2 & 3 \end{smallmatrix}\right)$, $\boldsymbol{B} = \left(\begin{smallmatrix} 3 & -3 \\ 0 & 2 \end{smallmatrix}\right)$ one has $\sigma(\boldsymbol{A} - \boldsymbol{B}) = (6.70, 0.30)$ but $|\sigma(\boldsymbol{A}) - \sigma(\boldsymbol{B})| = (0.64, 1.52)$, and $0.30 \ge 1.52$ fails at $i = 2$. What survives is Weyl, which bounds *each* gap by the *top* singular value $\sigma_1(\boldsymbol{A} - \boldsymbol{B})$, and Mirsky, which bounds the gaps *summed*. The completion proof below uses Mirsky.

Proof. (i) Weyl. By Courant–Fischer, $\sigma_i(\boldsymbol{A}) = \min_{\dim S = N_2 - i + 1} \max_{\boldsymbol{x} \in S, \|\boldsymbol{x}\|_2 = 1} \|\boldsymbol{A}\boldsymbol{x}\|_2$. Let $S^\star$ attain the minimum for $\boldsymbol{B}$. On each unit $\boldsymbol{x} \in S^\star$ the triangle inequality gives $\|\boldsymbol{A}\boldsymbol{x}\|_2 \le \|\boldsymbol{B}\boldsymbol{x}\|_2 + \|(\boldsymbol{A} - \boldsymbol{B})\boldsymbol{x}\|_2 \le \|\boldsymbol{B}\boldsymbol{x}\|_2 + \|\boldsymbol{A} - \boldsymbol{B}\|_2$, since $\|\boldsymbol{A} - \boldsymbol{B}\|_2 = \sigma_1(\boldsymbol{A} - \boldsymbol{B})$ is the largest stretch over all unit vectors. Maximizing the left side over $\boldsymbol{x} \in S^\star$ and then

using that $S^\star$ is optimal for $\boldsymbol{B}$ yields $\sigma_i(\boldsymbol{A}) \le \sigma_i(\boldsymbol{B}) + \|\boldsymbol{A}-\boldsymbol{B}\|_2$. Swapping $\boldsymbol{A}$ and $\boldsymbol{B}$ gives the reverse, and the two together give the absolute value.

(ii) Mirsky. Write $a_i = \sigma_i(\boldsymbol{A})$, $b_i = \sigma_i(\boldsymbol{B})$, and $c_i = \sigma_i(\boldsymbol{A}-\boldsymbol{B})$, all sorted nonincreasing. The elementary part is a bound on *partial* sums. For each k the Ky Fan k-norm $\|\boldsymbol{X}\|_{(k)} := \sum_{i=1}^{k} \sigma_i(\boldsymbol{X})$ is a norm—it equals the maximum of $\Re\,\mathrm{tr}(\boldsymbol{W}^*\boldsymbol{X})$ over rank-k partial isometries $\boldsymbol{W}$, a maximum of linear functionals of $\boldsymbol{X}$—so it obeys the triangle inequality. Applying that to $\boldsymbol{A} = \boldsymbol{B} + (\boldsymbol{A}-\boldsymbol{B})$, and again with $\boldsymbol{A}$ and $\boldsymbol{B}$ swapped, gives for every k

$$\left|\sum_{i=1}^{k}(a_i - b_i)\right| \;\le\; \sum_{i=1}^{k} c_i.$$

The full statement $\sum_{i=1}^{n}|a_i - b_i| \le \sum_{i=1}^{n} c_i = \|\boldsymbol{A}-\boldsymbol{B}\|_*$ is Mirsky's theorem [38]. Passing from the partial sums to this summed bound is the substantive step, and it is exactly where the tempting termwise reading "the ℓ-th largest gap is $\le c_\ell$" fails: for the matrices above, the gaps sorted decreasingly are $(1.52, 0.64)$ while $(c_1, c_2) = (6.70, 0.30)$, so the second gap already exceeds c_2, yet the *sums* still obey $2.16 \le 7.00$. A clean route to the summed bound is the Hermitian dilation $\widetilde{\boldsymbol{A}} = \left(\begin{smallmatrix} 0 & \boldsymbol{A} \\ \boldsymbol{A}^* & 0 \end{smallmatrix}\right)$, whose eigenvalues are $\pm\sigma_1(\boldsymbol{A}), \ldots, \pm\sigma_n(\boldsymbol{A})$ (padded by zeros when $\boldsymbol{A}$ is rectangular): since $\widetilde{\boldsymbol{A}} - \widetilde{\boldsymbol{B}} = \widetilde{\boldsymbol{A}-\boldsymbol{B}}$, it turns the claim into Lidskii's eigenvalue majorization for the Hermitian pair $\widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{B}}$. See [5] for the majorization step. ∎

Proof of Theorem 11.2. (A)⇒(B). Let $\boldsymbol{M} \in \ker(\mathcal{A})$ be nonzero with SVD $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$. Split the singular values at r into a top part and a tail, and set $\boldsymbol{M}_1 = \boldsymbol{U}\boldsymbol{\Sigma}_{\text{top}}\boldsymbol{V}^*$ (rank $\le r$) and $\boldsymbol{M}_2 = -\boldsymbol{U}\boldsymbol{\Sigma}_{\text{tail}}\boldsymbol{V}^*$, so $\boldsymbol{M}_1 - \boldsymbol{M}_2 = \boldsymbol{M}$. Since $\mathcal{A}(\boldsymbol{M}) = \boldsymbol{0}$, we have $\mathcal{A}(\boldsymbol{M}_1) = \mathcal{A}(\boldsymbol{M}_2)$, so $\boldsymbol{M}_2$ is feasible for the recovery problem of the rank-$\le r$ matrix $\boldsymbol{M}_1$, and $\boldsymbol{M}_2 \ne \boldsymbol{M}_1$ (else $\boldsymbol{M} = \boldsymbol{0}$). Uniqueness in (A) forces $\|\boldsymbol{M}_1\|_* < \|\boldsymbol{M}_2\|_*$, which reads $\sum_{i\le r}\sigma_i(\boldsymbol{M}) < \sum_{i>r}\sigma_i(\boldsymbol{M})$, the matrix NSP.

(B)⇒(A). Let $\mathrm{rank}(\boldsymbol{X}) \le r$ and let $\boldsymbol{Z} \ne \boldsymbol{X}$ be feasible. Set $\boldsymbol{M} = \boldsymbol{X} - \boldsymbol{Z} \in \ker(\mathcal{A})$, nonzero, so $\boldsymbol{Z} = \boldsymbol{X} - \boldsymbol{M}$. Apply the Mirsky bound Lemma 11.3(ii) to the pair $\boldsymbol{X}$ and $\boldsymbol{M}$: their difference is $\boldsymbol{X} - \boldsymbol{M} = \boldsymbol{Z}$, hence

$$\|\boldsymbol{Z}\|_* = \|\boldsymbol{X}-\boldsymbol{M}\|_* \;\ge\; \sum_i |\sigma_i(\boldsymbol{X}) - \sigma_i(\boldsymbol{M})|.$$

Now split the sum at r. For $i \le r$, drop the absolute value: $|\sigma_i(\boldsymbol{X}) - \sigma_i(\boldsymbol{M})| \ge \sigma_i(\boldsymbol{X}) - \sigma_i(\boldsymbol{M})$. For $i > r$, the rank bound makes $\sigma_i(\boldsymbol{X}) = 0$, so the term is exactly $\sigma_i(\boldsymbol{M})$. Therefore

$$\|\boldsymbol{Z}\|_* \;\ge\; \underbrace{\sum_{i\le r}\sigma_i(\boldsymbol{X})}_{=\|\boldsymbol{X}\|_*} - \sum_{i\le r}\sigma_i(\boldsymbol{M}) + \sum_{i>r}\sigma_i(\boldsymbol{M}),$$

and the matrix NSP makes the last two terms strictly positive, so $\|\boldsymbol{Z}\|_* > \|\boldsymbol{X}\|_*$. Since $\boldsymbol{Z}$ was an arbitrary feasible competitor, $\boldsymbol{X}$ is the unique minimizer. ∎

> This is the vector recovery theorem of Chapter 4 with every word translated: $\ker(\boldsymbol{A})$ becomes $\ker(\mathcal{A})$, the support split becomes the SVD split at r, the ℓ_1 norm becomes the nuclear norm, and the scalar triangle inequality becomes Mirsky's summed singular-value bound. The same two-direction proof, the same role of the null space, the same convex-relaxation philosophy. Once you have seen the vector proof, the matrix proof writes itself.

11.4 Matrix Completion

The application that made this theory famous uses a special measurement operator: we observe a random subset of the matrix's *entries*.

Definition 11.5 (Sampling operator). For an observed set $\Omega \subseteq [N_1] \times [N_2]$, the **sampling operator** $\mathcal{P}_\Omega$ keeps the entries in Ω and zeros the rest. The **matrix completion** problem is to recover a low-rank $\boldsymbol{X}$ from $\mathcal{P}_\Omega(\boldsymbol{X})$, by

$$\min_{\boldsymbol{Z}} \|\boldsymbol{Z}\|_* \quad \text{subject to} \quad \mathcal{P}_\Omega(\boldsymbol{Z}) = \mathcal{P}_\Omega(\boldsymbol{X}).$$

Example 11.6 (Low rank determines the missing entries). Why should a few entries pin down the rest? Take the partly observed matrix

$$\boldsymbol{X} = \begin{pmatrix} 3 & 6 & ? \\ 1 & 2 & 7 \\ 2 & 4 & ? \\ 4 & ? & 28 \\ ? & ? & ? \end{pmatrix},$$

with question marks unknown. Under no assumption the missing cells are free: any values give a valid (generically full-rank) matrix, and there is no principled completion. Impose the *lowest-rank* prior, $\operatorname{rank}(\boldsymbol{X}) = 1$. Then every row is a scalar multiple of one fixed row vector $\boldsymbol{r}$. Reading the fully known second row gives $\boldsymbol{r} = (1, 2, 7)$. The first row $(3, 6, ?)$ must be $3\boldsymbol{r} = (3, 6, 21)$, so the missing entry is 21; the third row $(2, 4, ?) = 2\boldsymbol{r} = (2, 4, 14)$; the fourth row $(4, ?, 28) = 4\boldsymbol{r} = (4, 8, 28)$, since its last entry $28 = 4 \cdot 7$ fixes the scale. The fifth row, with no observations, is only determined up to its scalar, the residual ambiguity that more samples or an incoherence assumption removes. Low rank turned "infinitely many completions" into "essentially one," and the whole field is about making that "essentially" rigorous. ◊

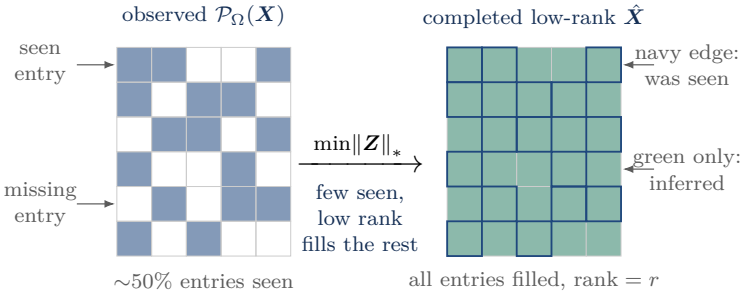


Figure 11.1. Matrix completion. The sampling operator $\mathcal{P}_\Omega$ reveals a random subset of entries (left, navy); nuclear-norm minimization fills the rest with the lowest-rank matrix consistent with what was seen (right). On the right the navy-bordered cells are the ones that were observed; every other cell was inferred from low rank. Incoherence ensures each observed entry carries information about the whole.

Completion is harder than the generic low-rank recovery of Section 11.3, because each measurement reveals only one entry rather than a full linear functional, and it needs an extra assumption. A matrix whose mass is concentrated in a single entry, $\boldsymbol{X} = \boldsymbol{e}_1\boldsymbol{e}_1^*$, is rank one but cannot be completed: unless that exact entry is sampled, the observations are all zero and give no information. The fix is to require the matrix be *incoherent*, its singular vectors spread out rather than aligned with the standard basis, so that every entry carries information about the whole. This is the matrix echo of the bounded-orthonormal-system condition of Chapter 10.

Theorem 11.7 (Candès–Recht completion). Let $\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2}$ have rank r with singular vectors drawn from the random orthogonal model (equivalently, satisfying a standard incoherence condition), and let Ω be a uniformly random set of m observed entries, with $N^\star = \max(N_1, N_2)$. There are constants $C, c > 0$ such that if

$$m \;\geq\; C\,(N^\star)^{5/4}\, r\, \log N^\star,$$

then with probability at least $1 - c/N^\star$, nuclear norm minimization recovers $\boldsymbol{X}$ exactly.

Reading the bound

The rank r plays the role of the sparsity s, and the factor $r(N_1 + N_2) \approx rN^\star$ is the degree-of-freedom floor, unavoidable. The extra $(N^\star)^{1/4}$ and the $\log N^\star$ are the price of the restrictive sampling: each observation is a single entry $\boldsymbol{e}_i\boldsymbol{e}_j^*$, not a rich Gaussian functional, so the geometry is less favorable than the compressed sensing of Chapter 9, where the count was only $s \log(N/s)$. Later work sharpened the exponent to a near-optimal $rN^\star \operatorname{polylog}(N^\star)$.

Example 11.8 (Reading the bound on a 100×100 matrix). Take $N_1 = N_2 = N^\star = 100$ and rank $r = 5$. Dropping the constant C, the bound asks for

$$m \gtrsim (N^\star)^{5/4} r \log N^\star = 100^{1.25} \cdot 5 \cdot \ln 100.$$

Now $100^{1.25} = 100 \cdot 100^{1/4} = 100 \cdot \sqrt[4]{100} \approx 100 \cdot 3.162 = 316.2$ and $\ln 100 \approx 4.605$, so $m \gtrsim 316.2 \cdot 5 \cdot 4.605 \approx 7280$, about 73% of the 10,000 entries (and once the constant $C \approx 32$ of the early proofs is included, the bound exceeds the matrix and says nothing). Yet in practice nuclear-norm minimization recovers a random rank-5 matrix of this size from only about 25% of its entries. The theorem is loose by a constant: the true phase transition is far sharper than the $(N^\star)^{5/4}$ exponent suggests, which is why empirical recovery beats the guarantee. ◊

Table 11.1 sets the vector and matrix theories side by side. Every row is the same statement told twice.

	Sparse vectors	**Low-rank matrices**
Structure	few nonzeros	few nonzero singular values
Complexity measure	$\|\boldsymbol{x}\|_0$	$\operatorname{rank}(\boldsymbol{X}) = \|\boldsymbol{\sigma}\|_0$
Convex surrogate	$\|\boldsymbol{x}\|_1$	$\|\boldsymbol{X}\|_* = \|\boldsymbol{\sigma}\|_1$
Hard problem	(P_0) (NP-hard)	(PM_0) (NP-hard)
Convex program	(P_1) (LP)	(PM_1) (SDP)
Recovery condition	null space property	matrix null space property
Random guarantee	$m \gtrsim s \log(N/s)$	$m \gtrsim rN^\star$ polylog

Table 11.1. The vector and matrix theories in parallel. The second half of the book is the first half with sparsity replaced by low rank and the ℓ_1 norm by the nuclear norm.

11.5 The Netflix Prize

In 2006 Netflix released a dataset of roughly 480,000 users rating 17,770 movies and offered one million dollars to anyone who could improve its recommendation accuracy by ten percent. The ratings form a matrix with more than eight billion entries, of which only about one percent were observed: most users rate few movies. The competition is exactly matrix completion. Fill in the missing entries, and you can recommend the movies a user has not seen but would rate highly.

The matrix is completable because it is approximately low rank. A user's taste is well described by a few latent factors, a leaning toward comedy or drama, old films or new, a given actor, and a movie by the same few factors, so a rating is close to the inner product of a short user-factor vector and a short movie-factor vector. That is exactly $\boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{V}^*$ with $\boldsymbol{U}, \boldsymbol{V}$ having only r columns, a rank-r model. In September 2009 the team "BellKor's Pragmatic Chaos" crossed the ten-percent threshold, at 10.06%, and won. Their solution was an ensemble built around low-rank matrix factorization, the practical cousin of the nuclear-norm minimization Theorem 11.7 certifies. The prize money was an advertisement; the lasting result was the demonstration that low-rank structure is real and exploitable in data at scale.

Remark 11.9 *(The manifold hypothesis).* Low rank is one instance of a broader principle, the *manifold hypothesis*: real high-dimensional data does not fill its ambient space but concentrates near a low-dimensional surface. Images of a face under varying illumination lie near a low-dimensional set; natural images occupy a thin sliver of pixel space; the ratings matrix lives near a low-rank model. Whenever data has few true degrees of freedom, the recovery machinery of this book applies, with sparsity, low rank, or a learned dictionary (Chapter 13) as the particular form the structure takes. This is why the same mathematics keeps reappearing across signal processing, vision, and machine learning.

Notes and References

Nuclear norm minimization for rank recovery is due to Fazel and to Recht, Fazel, and Parrilo [44], which proves a restricted-isometry analogue for generic linear measurements. The matrix null space property and its two-direction proof parallel the vector treatment in Foucart and Rauhut [28]. Exact matrix completion is Candès and Recht [10], with the sample complexity sharpened by Candès and Tao and by Recht; the singular-value thresholding solver is Cai, Candès, and Shen [8]. The Netflix Prize and the matrix-factorization methods that won it are described by Koren, Bell, and Volinsky [33]. The vector–matrix dictionary and the manifold hypothesis are developed throughout Wright and Ma [53].

Exercises

Exercise 11.1. For $\boldsymbol{X} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ and $\boldsymbol{Y} = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, compute the singular value vectors, the ranks, and the nuclear norms, and verify $\operatorname{rank} = \|\boldsymbol{\sigma}\|_0$ and $\|\cdot\|_* = \|\boldsymbol{\sigma}\|_1$ in each.

Exercise 11.2. Verify Lemma 11.3 numerically for two 2×2 matrices of your choice by computing all singular values of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{A} - \boldsymbol{B}$.

Exercise 11.3. Show that the rank-one matrix $\boldsymbol{X} = \boldsymbol{e}_1 \boldsymbol{e}_1^{*}$ cannot be completed from any set of observed entries that misses the $(1,1)$ position, and explain how this motivates the incoherence assumption in Theorem 11.7.

Exercise 11.4. A rank-r matrix in $\mathbb{R}^{N \times N}$ has about $r(2N - r)$ degrees of freedom. Compare this with N^2 for $N = 10^4$ and $r = 10$, and with the Candès–Recht sample complexity $m \gtrsim N^{5/4} r \log N$. What fraction of entries must be observed?

Exercise 11.5. Translate the vector NSP proof of Theorem 4.10 line by line into the matrix NSP proof of Theorem 11.2, identifying for each step (support split, triangle inequality, recombination) its matrix counterpart.

CHAPTER 12

Robust PCA

Classical PCA asks for the best subspace; robust PCA asks for it again, this time ignoring the vandals.

Principal component analysis finds the low-dimensional subspace that best explains a data matrix, and it is one of the most used tools in all of data analysis. It has a fatal weakness: a single grossly corrupted entry can swing the answer wildly, because the least-squares criterion it optimizes is dominated by outliers. Robust PCA repairs this by modeling the data as a sum of a low-rank matrix and a *sparse* matrix of arbitrary corruptions, and recovering both by a convex program. The remarkable result of this chapter, the Candès–Li–Ma–Wright theorem, is that the same nuclear-norm and ℓ_1 relaxations from Chapters 5 and 11, combined, separate the low-rank and sparse parts *exactly*, even when the corruption hits a constant fraction of all entries. We begin by deriving classical PCA from first principles, because the derivation exposes exactly where it breaks.

12.1 Principal Component Analysis from Scratch

Arrange N data points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathbb{R}^M$ (vectorized images, audio frames, spreadsheet rows) as the columns of a data matrix $\boldsymbol{X} \in \mathbb{R}^{M \times N}$. PCA seeks an S-dimensional subspace, $S \ll M$, from which every data point is nearly recoverable. Write the subspace through an orthonormal basis $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \mid \cdots \mid \boldsymbol{\phi}_S] \in \mathbb{R}^{M \times S}$, approximate each point by $\boldsymbol{x}_i \approx \boldsymbol{\Phi} \boldsymbol{w}_i$ with coordinates $\boldsymbol{w}_i \in \mathbb{R}^S$, and collect the coordinates into $\boldsymbol{W} \in \mathbb{R}^{S \times N}$. The total squared reconstruction error is a Frobenius norm, so PCA is the optimization

$$\min_{\boldsymbol{\Phi}, \boldsymbol{W}} \|\boldsymbol{\Phi} \boldsymbol{W} - \boldsymbol{X}\|_F^2 \quad \text{subject to} \quad \boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}_S. \tag{12.1}$$

> Keep $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ and $\boldsymbol{\Phi} \boldsymbol{\Phi}^\top$ apart. With $\boldsymbol{\Phi} \in \mathbb{R}^{M \times S}$ and $S < M$, the $S \times S$ Gram matrix $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}_S$ says the columns are orthonormal, a sensible constraint, whereas the $M \times M$ product $\boldsymbol{\Phi} \boldsymbol{\Phi}^\top$ is the orthogonal *projector* onto the subspace, of rank S, and is *not* the identity.

Solve for the coordinates. Fix $\boldsymbol{\Phi}$. The objective decouples across points, and for each one, expanding and using $\boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}_S$,

$$\|\boldsymbol{\Phi} \boldsymbol{w}_i - \boldsymbol{x}_i\|_2^2 = \|\boldsymbol{w}_i\|_2^2 - 2 \boldsymbol{w}_i^\top \boldsymbol{\Phi}^\top \boldsymbol{x}_i + \|\boldsymbol{x}_i\|_2^2.$$

Setting the gradient to zero gives $\boldsymbol{w}_i^\star = \boldsymbol{\Phi}^\top \boldsymbol{x}_i$: in an orthonormal basis the best coordinates are the inner products against the basis vectors, so $\boldsymbol{W}^\star = \boldsymbol{\Phi}^\top \boldsymbol{X}$.

Reduce to the subspace. Substituting $\boldsymbol{W}^\star$ leaves a problem in $\boldsymbol{\Phi}$ alone. With the projector $\boldsymbol{P} = \boldsymbol{\Phi} \boldsymbol{\Phi}^\top$ (so $\boldsymbol{P}^2 = \boldsymbol{P} = \boldsymbol{P}^\top$),

$$\|\boldsymbol{P} \boldsymbol{X} - \boldsymbol{X}\|_F^2 = \operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{X}) - \operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{P} \boldsymbol{X}),$$

and since $\operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{X}) = \|\boldsymbol{X}\|_F^2$ is constant, minimizing the error is the same as *maximizing* the captured variance $\operatorname{tr}(\boldsymbol{\Phi}^\top \boldsymbol{X} \boldsymbol{X}^\top \boldsymbol{\Phi}) = \sum_{j=1}^{S} \boldsymbol{\phi}_j^\top \boldsymbol{X} \boldsymbol{X}^\top \boldsymbol{\phi}_j$.

Read off the answer. By the Eckart–Young theorem (Appendix D), the best rank-S approximation of $\boldsymbol{X}$ is its truncated SVD, so the optimal basis is the top S left singular vectors of $\boldsymbol{X}$, equivalently the top eigenvectors of the covariance $\boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^\top$:

$$\boldsymbol{\Phi}^\star = \boldsymbol{U}_S, \qquad \boldsymbol{W}^\star = \boldsymbol{U}_S^\top \boldsymbol{X}.$$

Geometrically (Fig. 12.1) PCA fits the flat subspace that minimizes the summed squared perpendicular distances from the data: the best-fit line for $S = 1$, the best-fit plane for $S = 2$.

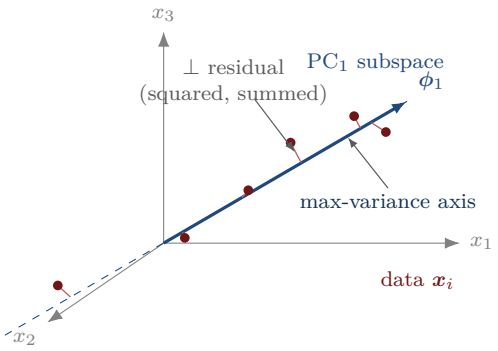


Figure 12.1. PCA in $\mathbb{R}^3$ with $S = 1$. The red points are data; the blue line is the best-fit principal direction $\boldsymbol{\phi}_1$, the axis of maximal variance; the red segments are the perpendicular residuals whose squared lengths PCA minimizes.

Example 12.1 (PCA on five points near a plane). Take

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 0.1 & 0.2 & 0.1 & 0.3 & 0.2 \end{bmatrix} \in \mathbb{R}^{3\times 5},$$

whose second row is twice the first and whose third row is small noise, so the data nearly lies on a line. The covariance is

$$\boldsymbol{X}\boldsymbol{X}^\top = \begin{bmatrix} 55 & 110 & 3.0 \\ 110 & 220 & 6.0 \\ 3.0 & 6.0 & 0.19 \end{bmatrix},$$

where $55 = 1^2+\cdots+5^2$, $110 = 1\cdot 2+\cdots+5\cdot 10$, $3.0 = 1\cdot 0.1+\cdots+5\cdot 0.2$, and $0.19 = 0.01+\cdots+0.04$. Its eigenvalues are $\lambda_1 \approx 275.16$, $\lambda_2 \approx 0.026$, $\lambda_3 \approx 0$, with top eigenvector $\boldsymbol{u}_1 \approx (0.447, 0.894, 0.024)^\top$, close to $(1, 2, 0)/\sqrt{5}$. The best rank-one PCA reconstruction has error $\|\boldsymbol{X} - \boldsymbol{X}_1\|_F^2 = \lambda_2 + \lambda_3 \approx 0.026$ against a total of $\lambda_1 + \lambda_2 + \lambda_3 = \operatorname{tr}(\boldsymbol{X}\boldsymbol{X}^\top) = 275.19$, capturing 99.99% of the variance along the first principal direction. ◊

PCA as low-rank-plus-dense-noise. There is an equivalent reading that sets up the upgrade to come. If we observe $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{W}$ with $\boldsymbol{X}$ low rank and $\boldsymbol{W}$ a *dense, small* noise ($\|\boldsymbol{W}\|_F \le \eta$), then the best rank-$r$ fit $\min_{\operatorname{rank}(\boldsymbol{Z})\le r}\|\boldsymbol{Y} - \boldsymbol{Z}\|_F^2$ is, by Eckart–Young, the truncated SVD of $\boldsymbol{Y}$, which is exactly PCA. The Frobenius norm is the right loss when the noise is dense and small, and perturbation theory then guarantees the top singular subspace of $\boldsymbol{Y}$ is a stable estimate of that of $\boldsymbol{X}$. The fatal word is *dense*.

12.2 Why PCA Breaks

PCA assumes the noise is dense and small. Real corruption is often the opposite: a few entries, wrong by a lot. Picture many photographs of the Mona Lisa under slightly varying light, a nearly rank-one data matrix with dense low-amplitude fluctuation, the PCA regime exactly. Now someone paints sunglasses on one copy. A few hundred pixels near the eyes are catastrophically altered: the noise is now *sparse* (few entries) and *gross* (huge values). Because PCA charges noise by its squared magnitude and barely by its count, it warps its principal subspace to chase the sunglasses, corrupting the recovered painting everywhere. The following computation makes the failure exact.

Example 12.2 (One gross entry destroys the top component). Let the clean signal be the rank-one matrix $\boldsymbol{X} = \boldsymbol{u}\boldsymbol{v}^\top$ with $\boldsymbol{u} = \boldsymbol{v} = \frac{1}{\sqrt{3}}(1,1,1)^\top$, so every entry of $\boldsymbol{X}$ equals $\frac{1}{3}$ and its singular values are $(1,0,0)$: exactly rank one, top singular vector $\frac{1}{\sqrt{3}}(1,1,1)^\top$. Corrupt a single entry,

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{E}, \qquad \boldsymbol{E} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 100 \end{bmatrix},$$

so $\|\boldsymbol{E}\|_0 = 1$ but $\|\boldsymbol{E}\|_F = 100$. The corrupted matrix is dominated by its $(3,3)$ entry $\frac{1}{3} + 100$. Its largest singular value is $\sigma_1 \approx 100.34$ and the top left singular vector is $\boldsymbol{u}_1 \approx (0.0033, 0.0033, 0.99998)^\top$, almost exactly $\boldsymbol{e}_3$. The clean direction $\frac{1}{\sqrt{3}}(1,1,1)^\top$ is gone, and a best rank-one fit of $\boldsymbol{Y}$, that is PCA with $S = 1$, recovers a matrix with almost nothing to do with $\boldsymbol{X}$. One bad pixel out of nine swung the answer completely. ◊

> Sparse gross corruption is not a contrived case; it is the default in many domains. A scratched DVD plays perfectly except for scattered loud glitches; a camera with dead pixels is crisp except where those pixels are stuck at 0 or 255; an occluded face is the clean low-rank face plus a sparse occluder; a lossy video stream drops a few frames; salt-and-pepper noise flips scattered pixels to pure black or white; a single misentered value drags a covariance off axis. In every case the corruption is sparse but unbounded, exactly what least squares cannot handle and what robust PCA is built for.

12.3 The Robust PCA Model and Principal Component Pursuit

Model the observation as $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{E}$ with $\boldsymbol{X}$ low rank (the true signal) and $\boldsymbol{E}$ sparse (the gross corruption, few nonzeros but arbitrary magnitudes). We see only $\boldsymbol{Y}$ and want both pieces. The honest formulation,

$$\min_{\boldsymbol{X},\boldsymbol{E}} \ \operatorname{rank}(\boldsymbol{X}) + \lambda\|\boldsymbol{E}\|_0 \quad \text{subject to} \quad \boldsymbol{X} + \boldsymbol{E} = \boldsymbol{Y},$$

is NP-hard, combining the two intractable measures of Chapters 3 and 11. Relax both at once.

Definition 12.3 (Principal Component Pursuit). Replacing rank by the nuclear norm and the corruption count by the entrywise ℓ_1 norm gives the convex program

$$\text{(PCP)} \qquad \min_{\boldsymbol{X},\boldsymbol{E}} \ \|\boldsymbol{X}\|_* + \lambda\|\boldsymbol{E}\|_1 \quad \text{subject to} \quad \boldsymbol{X} + \boldsymbol{E} = \boldsymbol{Y},$$

where $\|\boldsymbol{E}\|_1 = \sum_{i,j}|E_{ij}|$. Both terms are convex norms and the constraint is linear, so (PCP) is a semidefinite program.

Each substitution is the tightest convex one: the nuclear norm is the convex envelope of rank on the spectral-norm ball, and the ℓ_1 norm is the convex envelope of $\|\cdot\|_0$ on the ℓ_∞ ball, meaning each is the largest convex function lying below its combinatorial counterpart. The only question is whether the relaxation is exact.

12.4 The Candès–Li–Ma–Wright Theorem

It is, under strikingly weak conditions. As in matrix completion, the low-rank part must be *incoherent*, its singular vectors spread out rather than aligned with a few coordinates, so the low-rank and sparse models do not overlap. (A matrix that is both low rank and sparse, like $\boldsymbol{e}_1\boldsymbol{e}_1^*$, could belong to either part, and no method could separate it; this is exactly the corruption of Example 12.2.) Incoherence is measured by the parameter μ with $\max_i\|\boldsymbol{U}^\top\boldsymbol{e}_i\|_2^2 \le \mu r/N_1$ for the left singular vectors and $\max_j\|\boldsymbol{V}^\top\boldsymbol{e}_j\|_2^2 \le \mu r/N_2$ for the right singular vectors.

Theorem 12.4 (Candès–Li–Ma–Wright). Let $\boldsymbol{X} \in \mathbb{R}^{N_1\times N_2}$ have rank r with incoherent singular vectors (parameter μ), and let $\boldsymbol{E}$ have a uniformly random support with random signs. Write $n_* = \min(N_1, N_2)$, $n^* = \max(N_1, N_2)$. There are constants $\rho_r, \rho_s, c > 0$ such that if

$$r \ \le \ \rho_r \frac{n_*}{\mu\,(\log n^*)^2} \qquad \text{and} \qquad \|\boldsymbol{E}\|_0 \ \le \ \rho_s\, N_1 N_2,$$

then Principal Component Pursuit with $\lambda = 1/\sqrt{n^*}$ recovers $(\boldsymbol{X}, \boldsymbol{E})$ exactly, with probability at least $1 - c\,(n^*)^{-10}$.

What is surprising

Constant-fraction corruption. The sparse part may corrupt up to $\rho_s N_1 N_2$ entries, a positive fraction of the matrix, with arbitrarily large values, and recovery is still exact. Sparse vector recovery allowed only a sublinear number of nonzeros; here the low-rank constraint and the additive equation together are far stronger than either alone. **No tuning.** The weight $\lambda = 1/\sqrt{n^*}$ is fixed by the dimensions, not cross-validated; the theorem proves this exact value works. **Near-full rank.** For square matrices with benign coherence the rank may be as large as $n/\log^2 n$.

Example 12.5 (Reading the bound). Take $N_1 = N_2 = 1000$, so $n_* = n^* = 1000$, and $\mu = 1$. The rank cap with $\rho_r = 0.5$ is $r \le 0.5 \cdot 1000/(\ln 1000)^2 = 500/(6.9)^2 \approx 10.5$, so a rank up to about 10 is separable. The regularizer is $\lambda = 1/\sqrt{1000} \approx 0.0316$, fixed. With $\rho_s = 0.1$ the corruption may hit up to 10^5 of the 10^6 entries, a tenth of the matrix, and still be removed exactly. The failure probability $c \cdot 1000^{-10}$ is 10^{-30} small. ◊

12.5 Solving PCP by the Augmented Lagrangian

A general semidefinite-program solver cannot handle Principal Component Pursuit at the sizes that arise in imaging. The standard fast method is the inexact augmented Lagrangian, and its two work steps are exactly the proximal operators we already have: soft thresholding for the ℓ_1 part (Chapter 7) and its matrix analogue, singular-value thresholding, for the nuclear-norm part.

The augmented Lagrangian of (PCP) adds a multiplier $\boldsymbol{\Lambda}$ and a quadratic penalty to the constraint,

$$\mathcal{L}_\mu(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\Lambda}) = \|\boldsymbol{X}\|_* + \lambda\|\boldsymbol{E}\|_1 + \langle \boldsymbol{\Lambda},\, \boldsymbol{X} + \boldsymbol{E} - \boldsymbol{Y}\rangle + \frac{\mu}{2}\|\boldsymbol{X} + \boldsymbol{E} - \boldsymbol{Y}\|_F^2.$$

Minimizing over $\boldsymbol{X}$ with $\boldsymbol{E}$ fixed is the proximal problem for the nuclear norm, solved by *singular-value thresholding*: take the SVD of the argument and soft-threshold its singular values.

Definition 12.6 (Singular-value thresholding). For $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ and $\tau > 0$,

$$\mathcal{D}_\tau(\boldsymbol{M}) = \boldsymbol{U}\,\mathcal{S}_\tau(\boldsymbol{\Sigma})\,\boldsymbol{V}^* = \boldsymbol{U}\ \mathrm{diag}\big(\max(\sigma_k - \tau, 0)\big)\,\boldsymbol{V}^* = \arg\min_{\boldsymbol{X}}\ \tau\,\|\boldsymbol{X}\|_* + \tfrac{1}{2}\|\boldsymbol{X} - \boldsymbol{M}\|_F^2.$$

It is soft thresholding applied to the singular values, the exact matrix analogue of the ℓ_1 prox.

Example 12.7 (A singular-value threshold by hand). Let $\boldsymbol{M} = \left(\begin{smallmatrix} 3 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, already its own SVD with singular values $3, 1$. Then $\mathcal{D}_{1.5}(\boldsymbol{M})$ thresholds the singular values by 1.5: $3 \mapsto 1.5$ and $1 \mapsto 0$, giving $\mathcal{D}_{1.5}(\boldsymbol{M}) = \left(\begin{smallmatrix} 1.5 & 0 \\ 0 & 0 \end{smallmatrix}\right)$, rank one. Thresholding singular values both shrinks the matrix and *reduces its rank*, exactly as soft thresholding both shrinks entries and increases vector sparsity. ◊

Minimizing over $\boldsymbol{E}$ is the ordinary ℓ_1 prox, entrywise soft thresholding. Alternating these two closed-form updates and ascending on the multiplier gives the algorithm.

The structure is the recurring template of the book: each iteration is a gradient-like step on the penalty followed by a threshold, one for each variable. Figure 12.2 shows the data flow. Growing μ (the continuation trick) starts with large, aggressive thresholds and tightens the constraint over time, which converges far faster than a fixed penalty: small μ makes the thresholds $1/\mu$ and λ/μ large and the updates bold but the constraint loose, and increasing μ gradually enforces $\boldsymbol{X} + \boldsymbol{E} = \boldsymbol{Y}$.

Algorithm 8 Inexact ALM for Robust PCA

Require: $\boldsymbol{Y}$, $\lambda = 1/\sqrt{\max(N_1, N_2)}$, penalty $\mu > 0$, growth $\rho > 1$, tolerance ε

$\boldsymbol{X} \leftarrow \boldsymbol{0}$, $\boldsymbol{E} \leftarrow \boldsymbol{0}$, $\boldsymbol{\Lambda} \leftarrow \boldsymbol{0}$

repeat

  $\boldsymbol{X} \leftarrow \mathcal{D}_{1/\mu}(\boldsymbol{Y} - \boldsymbol{E} - \frac{1}{\mu}\boldsymbol{\Lambda})$ ▷ singular-value threshold (nuclear-norm prox)

  $\boldsymbol{E} \leftarrow \mathcal{S}_{\lambda/\mu}(\boldsymbol{Y} - \boldsymbol{X} - \frac{1}{\mu}\boldsymbol{\Lambda})$ ▷ soft threshold (ℓ_1 prox)

  $\boldsymbol{\Lambda} \leftarrow \boldsymbol{\Lambda} + \mu\,(\boldsymbol{X} + \boldsymbol{E} - \boldsymbol{Y})$ ▷ dual ascent

  $\mu \leftarrow \rho\,\mu$ ▷ continuation

until $\|\boldsymbol{Y} - \boldsymbol{X} - \boldsymbol{E}\|_F / \|\boldsymbol{Y}\|_F < \varepsilon$

return low-rank $\boldsymbol{X}$, sparse $\boldsymbol{E}$

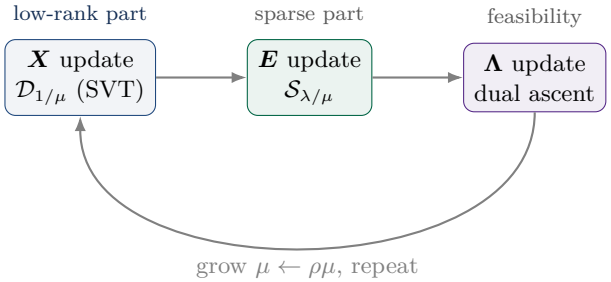


Figure 12.2. One inexact-ALM iteration: threshold the singular values to update the low-rank part, soft-threshold the entries to update the sparse part, ascend on the multiplier, grow the penalty, repeat.

12.6 Applications

The low-rank-plus-sparse model fits a surprising range of problems, each an instance of $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{E}$ with a different reading of the two parts.

Background subtraction in video. Stack the frames of a surveillance video as the columns of $\boldsymbol{Y}$, one vectorized frame per column. The static background is nearly identical across frames, so it is *low rank* (often nearly rank one); the moving objects occupy a few pixels per frame, so they are *sparse*. Principal Component Pursuit returns the background as $\boldsymbol{X}$ and the moving foreground as $\boldsymbol{E}$, separating them with no training and no motion model. This is the canonical demonstration of robust PCA, and it works strikingly well: a person walking through a lobby appears as a sparse silhouette in $\boldsymbol{E}$ while the lobby fills $\boldsymbol{X}$.

Face modeling under shadows. Images of one face under varying illumination lie near a low-rank subspace (the illumination cone), but cast shadows and specular highlights are sparse, large deviations. Robust PCA recovers the clean low-rank face and isolates the shadows, which classical PCA would smear across the whole image. The cleaned faces feed directly into the recognizer of Chapter 16.

Image inpainting and salt-and-pepper denoising. An image with scattered corrupted pixels (salt-and-pepper noise, dead sensors) is a low-rank or smooth image plus a sparse error, and PCP removes the corruption while preserving the rest, where a linear filter would blur it. The same model repairs occasional bad frames in a video by treating the frame stack as low rank.

Recommendation with corruption. A ratings matrix is low rank, but a few users rate maliciously or erratically; those entries are sparse outliers. Robust PCA, and its combination with the matrix completion of Chapter 11 when entries are also missing, recovers the true low-rank preference matrix despite both gaps and corruption.

These all rest on the manifold idea of Chapter 11: the clean data has few degrees of freedom (low rank), and what departs from that structure is sparse. When both halves of that description hold, Theorem 12.4 guarantees the convex program pulls them apart. The parallel to the vector theory is exact: Table 12.1 lines up robust PCA against the sparse recovery it generalizes.

	Sparse vectors (robust SRC)	**Low-rank matrices (robust PCA)**
Observation	$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$	$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{E}$
Structured part	sparse $\boldsymbol{x}$	low-rank $\boldsymbol{X}$
Corruption	sparse $\boldsymbol{e}$	sparse $\boldsymbol{E}$
Program	$\min \|\boldsymbol{x}\|_1 + \|\boldsymbol{e}\|_1$	$\min \|\boldsymbol{X}\|_* + \lambda \|\boldsymbol{E}\|_1$
Solver	soft thresholding	soft + singular-value thresholding

Table 12.1. Robust PCA is the matrix analogue of the robust sparse recovery of Chapter 16: vectors become matrices, the ℓ_1 norm becomes the nuclear norm, and soft thresholding gains a singular-value-thresholding twin.

In the Problem Sets

Problem Set 8 (Chapter 27) implements Algorithm 8 and runs it on a real task: separating the moving foreground of a video from its static background. You compute the two thresholding operators (entrywise soft thresholding and singular-value thresholding via an SVD), watch the residual $\|\boldsymbol{Y} - \boldsymbol{X} - \boldsymbol{E}\|_F$ fall, and verify the recovered background is low rank while the foreground is sparse, exactly as Theorem 12.4 predicts.

Notes and References

The PCA derivation and the Eckart–Young theorem behind it are classical; Appendix D has the proof, and the perturbation theory (Wedin, Davis–Kahan) that makes PCA stable under dense noise is standard. Robust principal component analysis and the Principal Component Pursuit program are due to Candès, Li, Ma, and Wright [15], whose paper title ends in a question mark because the authors found the exact-separation result surprising. The convex-envelope justification of the nuclear norm and ℓ_1 norm is standard convex analysis; see Boyd and Vandenberghe [7]. The singular-value thresholding operator and its use as the nuclear-norm prox are from Cai, Candès, and Shen [8]; the inexact augmented Lagrangian solver is the practical workhorse described in that literature. Background subtraction and the broader low-rank-plus-sparse modeling program are developed in Wright and Ma [53].

Exercises

Exercise 12.1. Carry out the PCA derivation's first step in detail: fix $\boldsymbol{\Phi}$ with $\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi} = \boldsymbol{I}_S$, expand $\|\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{x}\|_2^2$, and confirm the minimizer is $\boldsymbol{w}^{\star} = \boldsymbol{\Phi}^{\top}\boldsymbol{x}$.

Exercise 12.2. Reproduce Example 12.1 numerically: form $\boldsymbol{X}\boldsymbol{X}^{\top}$, find its eigenvalues, and verify the captured-variance fraction $\lambda_1/(\lambda_1 + \lambda_2 + \lambda_3)$.

Exercise 12.3. In Example 12.2, compute the top singular vector of $\boldsymbol{Y}$ for corruption magnitudes 10, 100, and 1000 in the $(3, 3)$ entry, and describe how fast the top component rotates toward $\boldsymbol{e}_3$ as the corruption grows.

Exercise 12.4. Compute $\mathcal{D}_{\tau}(\boldsymbol{M})$ for $\boldsymbol{M} = \left(\begin{smallmatrix} 4 & 0 \\ 0 & 3 \end{smallmatrix}\right)$ and $\tau = 1, 2, 3, 4$, and tabulate the rank of the result. At what τ does it first drop to rank one, and to rank zero?

Exercise 12.5. For $N_1 = N_2 = 5000$ and $\mu = 1$, use Theorem 12.4 (with $\rho_r = 0.5$, $\rho_s = 0.1$) to estimate the largest separable rank, the fraction of entries that may be corrupted, and the value of λ.

Exercise 12.6. Explain why a matrix that is simultaneously low rank and sparse, such as $\boldsymbol{e}_1\boldsymbol{e}_1^{*}$, cannot be uniquely separated by Principal Component Pursuit, and relate this to the incoherence hypothesis and to Example 12.2.

CHAPTER 13

Dictionary Learning

Stop choosing the dictionary by hand. Let the data choose it.

Every recovery method so far assumed the sparsifying basis was given: an image is sparse in wavelets, a signal in Fourier. But a fixed basis is a guess, and a basis *learned* from the data sparsifies it far better. Dictionary learning is the problem of finding, from a collection of example signals, a dictionary in which they all admit sparse representations. It turns the factorization $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ into a problem where *both* the dictionary $\boldsymbol{A}$ and the sparse codes $\boldsymbol{X}$ are unknown, and it is solved by alternating between the two: sparse-code the signals with the current dictionary, then update the dictionary to fit the codes. This chapter develops the two classic algorithms, the Method of Optimal Directions and K-SVD, derives the K-SVD update in full, traces a complete iteration of each by hand, and shows how the whole framework sits between classical clustering and modern deep learning.

13.1 Learning the Factorization

Collect N example signals as the columns of a data matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times N}$. We seek a dictionary $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ of m unit-norm atoms (typically overcomplete, $m > n$) and a coefficient matrix $\boldsymbol{X} \in \mathbb{R}^{m \times N}$ whose columns are each s-sparse, such that

$$\boldsymbol{Y} \approx \boldsymbol{A}\boldsymbol{X}, \qquad \|\boldsymbol{x}_i\|_0 \le s \text{ for every column } i.$$

Each piece has a name worth fixing. The dictionary $\boldsymbol{A}$ is the matrix of atoms; an **atom** $\boldsymbol{a}_k$ is one of its columns, normalized to $\|\boldsymbol{a}_k\|_2 = 1$; the **sparse code** $\boldsymbol{x}_i$ is the column of $\boldsymbol{X}$ that reconstructs signal $\boldsymbol{y}_i$ as $\boldsymbol{y}_i \approx \boldsymbol{A}\boldsymbol{x}_i = \sum_k X_{ki}\boldsymbol{a}_k$, a short recipe naming a few atoms and their weights. Both $\boldsymbol{A}$ and $\boldsymbol{X}$ are unknown, so the objective

$$\min_{\boldsymbol{A},\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|_F^2 \quad \text{subject to} \quad \|\boldsymbol{x}_i\|_0 \le s \ \forall i,$$

is not convex in the pair: it is bilinear, with a combinatorial constraint. Fixing either factor, though, leaves a problem we can solve. The universal strategy is then *alternating minimization*, holding one factor fixed and optimizing the other, then swapping.

The alternating loop

Sparse coding. Fix the dictionary $\boldsymbol{A}$ and solve, for each signal, $\min_{\boldsymbol{x}_i} \|\boldsymbol{y}_i - \boldsymbol{A}\boldsymbol{x}_i\|_2^2$ subject to $\|\boldsymbol{x}_i\|_0 \le s$. This is exactly the sparse recovery of Part II, solved by orthogonal matching pursuit (Chapter 6) or ℓ_1 minimization. **Dictionary update.** Fix the codes $\boldsymbol{X}$ and update the atoms to better fit them. How this step is done is what distinguishes the algorithms below.

Each half is a problem we can already solve or are about to; the loop alternates them until the factorization stabilizes. The objective decreases at every half-step, so the loop converges, though to a local minimum rather than a global one, as we discuss in Section 13.5.

Why normalize the atoms? Without the constraint $\|\boldsymbol{a}_k\|_2 = 1$ the factorization has a free scaling: replacing $\boldsymbol{a}_k$ by $c\boldsymbol{a}_k$ and the matching code row by $\boldsymbol{x}^k/c$ leaves the product $\boldsymbol{A}\boldsymbol{X}$ untouched, so the objective cannot distinguish them. The algorithm could drive an atom's norm to infinity and its codes to zero without changing the reconstruction, which makes the sparsity count meaningless. Fixing every atom to unit norm removes this gauge freedom and makes the coefficient magnitudes comparable across atoms.

13.2 K-Means as the One-Sparse Case

The simplest dictionary learning problem is clustering. Take the sparsity to its extreme, $s = 1$, and require the single nonzero coefficient to equal one, so the code is forced to a canonical basis vector $\boldsymbol{x}_i = \boldsymbol{e}_{k(i)}$. Then each signal is represented by a single atom, $\boldsymbol{y}_i \approx \boldsymbol{a}_{k(i)}$, and the factorization assigns every signal to its nearest atom. The dictionary update that best fits these assignments replaces each atom with the average of the signals assigned to it. This is exactly Lloyd's algorithm for k-means: alternate between assigning points to the nearest center (sparse coding with $s = 1$) and recomputing each center as the mean of its cluster (the dictionary update).

Theorem 13.1 (k-means is one-sparse dictionary learning). The k-means objective

$$\min_{\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_K} \sum_{i=1}^{N} \min_{j \in \{1,\dots,K\}} \|\boldsymbol{y}_i - \boldsymbol{\mu}_j\|_2^2$$

is identical to the dictionary learning problem $\min_{\boldsymbol{A},\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|_F^2$ with $m = K$ atoms and the extra constraint $\boldsymbol{x}_i \in \{\boldsymbol{e}_1, \dots, \boldsymbol{e}_K\}$, that is, each code is a canonical basis vector. The atoms $\boldsymbol{a}_j$ are the centroids $\boldsymbol{\mu}_j$.

Example 13.2 (k-means by hand on eight points). Take four points near $(1.5, 1.5)$ and four near $(8.5, 8.5)$,

$$\boldsymbol{y}_{1\dots4} = \tbinom{1}{1}, \tbinom{2}{1}, \tbinom{1}{2}, \tbinom{2}{2}; \qquad \boldsymbol{y}_{5\dots8} = \tbinom{8}{8}, \tbinom{9}{8}, \tbinom{8}{9}, \tbinom{9}{9},$$

and initialize $K = 2$ centroids *deliberately off*, at $\boldsymbol{\mu}_1 = (3, 3)$ and $\boldsymbol{\mu}_2 = (7, 6)$.

Assignment. For each point compare the squared distances. For $\boldsymbol{y}_1 = (1,1)$, $\|\boldsymbol{y}_1 - \boldsymbol{\mu}_1\|^2 = 4 + 4 = 8$ against $\|\boldsymbol{y}_1 - \boldsymbol{\mu}_2\|^2 = 36 + 25 = 61$, so it joins cluster 1. Running through all eight, the four low points land in cluster 1 ($8, 5, 5, 2$ versus $61, 50, 52, 41$) and the four high points in cluster 2 ($50, 61, 61, 72$ versus $5, 8, 10, 13$).

Update. Recompute each centroid as its cluster mean:

$$\boldsymbol{\mu}_1 = \tfrac{1}{4}\big[(1,1) + (2,1) + (1,2) + (2,2)\big] = (1.5, 1.5), \qquad \boldsymbol{\mu}_2 = \tfrac{1}{4}\big[(8,8) + (9,8) + (8,9) + (9,9)\big] = (8.5, 8.5).$$

Second assignment. Every low point is now within $\sqrt{0.5}$ of $\boldsymbol{\mu}_1$ and far from $\boldsymbol{\mu}_2$, and symmetrically for the high points, so the assignments do not change and the algorithm halts. One assignment/update cycle moved the centroids from their bad start to the exact cluster centers (Fig. 13.1). ◊

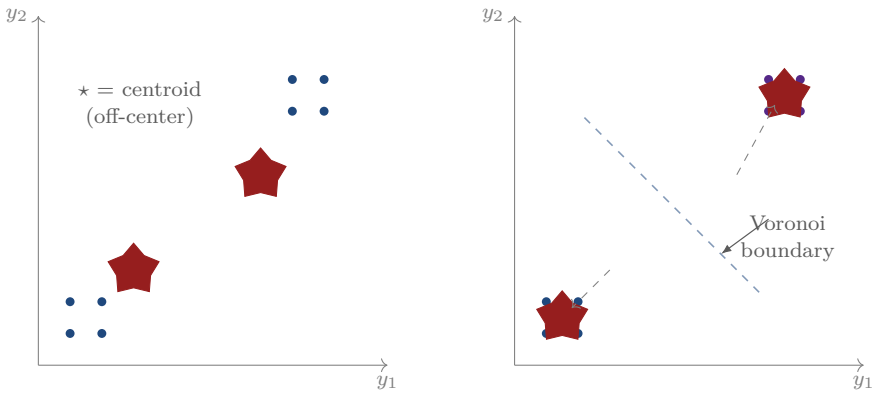


Figure 13.1. One cycle of k-means on the eight points of Example 13.2. The stars are centroids: starting badly placed at $(3,3)$ and $(7,6)$ (left), after a single assignment/update cycle they slide along the dashed paths to the exact cluster centers $(1.5, 1.5)$ and $(8.5, 8.5)$ (right). The dashed diagonal is the Voronoi boundary equidistant from the two centroids: every point is assigned to the centroid on its side. A second assignment changes nothing and the loop stops.

> k-means is dictionary learning with $s = 1$ and unit coefficients. The atoms are cluster centers, and the sparse code is a one-hot assignment. Its limitation is exactly that one-hot code: representing every face by one of K prototype faces is hopeless, because the face we want is a particular blend of eyes, nose, mouth, and lighting that no single centroid captures. Relaxing $s = 1$ to $s > 1$ lets a signal be a *combination* of a few atoms rather than a copy of one, turning hard clustering into sparse representation. Everything in this chapter is the general-s version of the familiar clustering loop.

Algorithm 9 Method of Optimal Directions (MOD)

Require: data $\boldsymbol{Y}$, dictionary size m, sparsity s
1: initialize $\boldsymbol{A}$ with unit-norm columns (e.g. random or DCT atoms)
2: **repeat**
3: **sparse coding:** for each i, $\boldsymbol{x}_i \leftarrow \mathrm{OMP}(\boldsymbol{A}, \boldsymbol{y}_i, s)$
4: **dictionary update:** $\boldsymbol{A} \leftarrow \boldsymbol{Y}\boldsymbol{X}^{\dagger} = \boldsymbol{Y}\boldsymbol{X}^{\top}(\boldsymbol{X}\boldsymbol{X}^{\top})^{-1}$
5: normalize each column of $\boldsymbol{A}$ to unit norm
6: **until** the residual $\|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|_F$ stops decreasing
7: **return** dictionary $\boldsymbol{A}$, codes $\boldsymbol{X}$

13.3 The Method of Optimal Directions

The Method of Optimal Directions (MOD) takes the simplest possible dictionary update: with the sparse codes fixed, choose the dictionary that minimizes the reconstruction error over *all* of $\boldsymbol{A}$ at once. That objective, $\min_{\boldsymbol{A}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|_F^2$, is an ordinary least-squares problem in $\boldsymbol{A}$, solved by the pseudoinverse of Chapter 2.

Example 13.3 (MOD on a synthetic dictionary). Build a ground-truth dictionary of $m = 3$ atoms in $\mathbb{R}^4$ and six 2-sparse codes:

$$\boldsymbol{A}^{\star} = \begin{bmatrix} 1 & 0 & 1/\sqrt{2} \\ 0 & 1 & 1/\sqrt{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{X}^{\star} = \begin{bmatrix} 1 & 1 & 0 & 2 & 1 & 0 \\ 1 & 0 & 1 & 1 & 2 & 0 \\ 0 & 1 & 1 & 0 & 0 & 2 \end{bmatrix},$$

each atom of unit norm. Multiplying out $\boldsymbol{Y}^{\star} = \boldsymbol{A}^{\star}\boldsymbol{X}^{\star}$ gives the data (only the first two rows are nonzero, since $\boldsymbol{A}^{\star}$ has two zero rows):

$$\boldsymbol{Y}^{\star} \approx \begin{bmatrix} 1.000 & 1.707 & 0.707 & 2.000 & 1.000 & 1.414 \\ 1.000 & 0.707 & 1.707 & 1.000 & 2.000 & 1.414 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

where, for instance, column 2 is $\boldsymbol{a}_1^{\star} + \boldsymbol{a}_3^{\star} = (1 + \tfrac{1}{\sqrt{2}}, \tfrac{1}{\sqrt{2}}, 0, 0) \approx (1.707, 0.707, 0, 0)$. Initialize $\boldsymbol{A}^{(0)}$ as four random unit-norm columns. The first sparse-coding pass runs OMP against this wrong dictionary and produces codes $\boldsymbol{X}^{(1)}$ that match the truth poorly: OMP cannot recover the right support from the wrong atoms. The dictionary update then solves the least squares $\boldsymbol{A}^{(1)} = \boldsymbol{Y}^{\star}(\boldsymbol{X}^{(1)})^{\dagger}$ and renormalizes. On the second pass the codes are far closer, because $\boldsymbol{A}^{(1)}$ has moved toward $\boldsymbol{A}^{\star}$, and the loop tightens. On a toy of this size MOD converges in roughly five to fifteen iterations, recovering $\boldsymbol{A}^{\star}$ *up to a permutation of its columns and sign flips.* The recovery is scored by the atom-recovery rate, the fraction of true atoms $\boldsymbol{a}_k^{\star}$ matched by some learned atom with $|\langle \boldsymbol{a}_k^{\star}, \boldsymbol{a}_j \rangle| > 0.99$. ◊

MOD is simple and effective, but it updates the dictionary while holding the codes fixed, so the two factors are never improved together, and the single matrix inversion couples all atoms at once. K-SVD improves on both points by updating each atom together with its coefficients, one atom at a time.

13.4 K-SVD

K-SVD, due to Aharon, Elad, and Bruckstein [1], updates the dictionary one atom at a time, and crucially updates each atom *and its coefficients together*. The derivation rests on reading the product $\boldsymbol{A}\boldsymbol{X}$ as a sum of rank-one pieces.

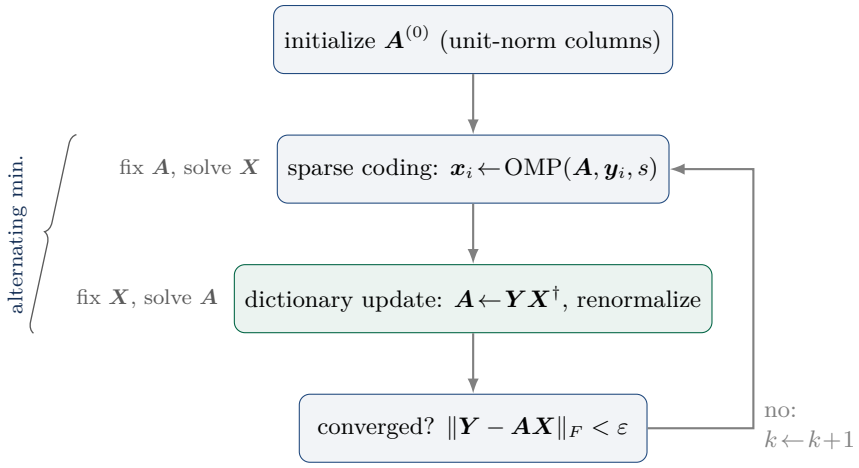


Figure 13.2. MOD alternates two steps until the Frobenius reconstruction error stops falling: fix the dictionary and sparse-code every signal, then fix those codes and replace the whole dictionary by the least-squares fit, renormalizing the atoms. The braced pair is one round of alternating minimization.

13.4.1 Isolating one atom

By the outer-product form of matrix multiplication,

$$\boldsymbol{A}\boldsymbol{X} = \sum_{j=1}^{m} \boldsymbol{a}_j \, (\boldsymbol{x}^j)^{\top},$$

where $\boldsymbol{a}_j$ is the j-th atom (a column of $\boldsymbol{A}$) and $\boldsymbol{x}^j$ is the j-th *row* of $\boldsymbol{X}$, the coefficients of atom j across all signals. The product is a sum of m rank-one matrices, one per atom (Fig. 13.3). To update atom j_0, pull its rank-one term out of the residual and collect everything else into a fixed error matrix,

$$\boldsymbol{E}_{j_0} = \boldsymbol{Y} - \sum_{j \neq j_0} \boldsymbol{a}_j (\boldsymbol{x}^j)^{\top}, \qquad \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|_F^2 = \|\boldsymbol{E}_{j_0} - \boldsymbol{a}_{j_0}(\boldsymbol{x}^{j_0})^{\top}\|_F^2.$$

The right side depends only on $\boldsymbol{a}_{j_0}$ and $\boldsymbol{x}^{j_0}$: the matrix $\boldsymbol{E}_{j_0}$ is what atom j_0 must explain, and we want the best rank-one fit to it. By the Eckart–Young theorem (Chapter 2), the best rank-one approximation of a matrix is its top singular component, so the naive update sets $\boldsymbol{a}_{j_0} = \boldsymbol{u}_1$ and $\boldsymbol{x}^{j_0} = \sigma_1 \boldsymbol{v}_1$ from the SVD of $\boldsymbol{E}_{j_0}$.

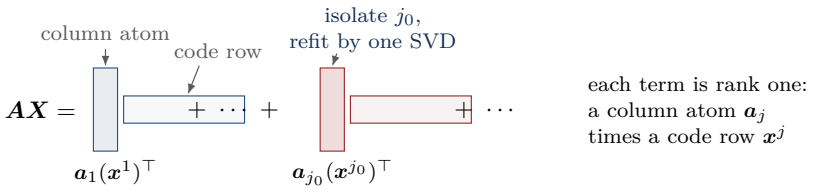


Figure 13.3. K-SVD's key picture: the product $\boldsymbol{A}\boldsymbol{X}$ is a sum of m rank-one outer products, each a column atom $\boldsymbol{a}_j$ times a code row $(\boldsymbol{x}^j)^{\top}$. Freezing all but the j_0-th term (red) leaves a single rank-one fit to the residual $\boldsymbol{E}_{j_0}$, solved by one SVD.

13.4.2 The restriction trick

The naive update has a fatal flaw: the top singular vector $\boldsymbol{v}_1$ is dense, so overwriting the whole row $\boldsymbol{x}^{j_0}$ with $\sigma_1 \boldsymbol{v}_1$ fills in nonzeros at signals that did not use atom j_0, breaking their sparsity budgets. The fix is to update the atom only on the signals that *currently use* it.

Definition 13.4 (Active set). The **active set** of atom j_0 is $\omega_{j_0} = \{i : X_{j_0,i} \neq 0\}$, the signals whose current code uses atom j_0.

Restrict the error matrix to the columns in ω_{j_0}, giving $\boldsymbol{E}_{j_0}^R$, and compute its top singular pair. Setting $\boldsymbol{a}_{j_0} \leftarrow \boldsymbol{u}_1$ and the restricted coefficients $\boldsymbol{x}_R^{j_0} \leftarrow \sigma_1 \boldsymbol{v}_1$ updates the atom and re-fits its coefficients *only where it was already used*; the zeros of $\boldsymbol{x}^{j_0}$ are untouched, so sparsity is preserved exactly. The restriction also makes the SVD far cheaper, since $|\omega_{j_0}|$ is usually a small fraction of N.

> The restriction is the most error-prone part of an implementation. The wrong order is to SVD the *full* residual $\boldsymbol{E}_{j_0}$, take the rank-one fit, and then mask the zeros: that fit already used all N columns and assumes a dense code. The correct order is *restrict first, then* take the SVD of $\boldsymbol{E}_{j_0}^R$, *then* write the result back only into the active positions ω_{j_0}. Restricting and masking are not interchangeable.

Algorithm 10 K-SVD (Aharon, Elad, Bruckstein)

Require: data $\boldsymbol{Y}$, dictionary size m, sparsity s
initialize $\boldsymbol{A}$ with unit-norm columns
repeat
  sparse coding: for each i, $\boldsymbol{x}_i \leftarrow \mathrm{OMP}(\boldsymbol{A}, \boldsymbol{y}_i, s)$
  for $j_0 = 1, \dots, m$ **do** ▷ dictionary update, one atom at a time
    $\omega_{j_0} \leftarrow \{i : X_{j_0,i} \neq 0\}$; **if** empty, reseed $\boldsymbol{a}_{j_0}$ at random
    $\boldsymbol{E}_{j_0}^R \leftarrow$ columns of $\left(\boldsymbol{Y} - \sum_{j \neq j_0} \boldsymbol{a}_j (\boldsymbol{x}^j)^\top\right)$ in ω_{j_0}
    $(\boldsymbol{u}_1, \sigma_1, \boldsymbol{v}_1) \leftarrow$ top singular triple of $\boldsymbol{E}_{j_0}^R$
    $\boldsymbol{a}_{j_0} \leftarrow \boldsymbol{u}_1$; $\boldsymbol{x}_R^{j_0} \leftarrow \sigma_1 \boldsymbol{v}_1$
  end for
until the residual stops decreasing
return dictionary $\boldsymbol{A}$, codes $\boldsymbol{X}$

13.4.3 One iteration, traced by hand

Example 13.5 (A single K-SVD atom update). Take $n = 5$, $m = 4$, $N = 6$, $s = 2$, with training matrix

$$\boldsymbol{Y} = \begin{pmatrix} 0.83 & -0.21 & 0.14 & 0.62 & -0.07 & 0.45 \\ -0.49 & 0.55 & 0.18 & -0.31 & 0.42 & -0.22 \\ 0.31 & 0.18 & -0.62 & 0.55 & -0.71 & 0.07 \\ 0.62 & -0.42 & 0.83 & -0.18 & 0.55 & 0.31 \\ -0.18 & 0.31 & -0.07 & 0.42 & 0.14 & -0.62 \end{pmatrix},$$

and $\boldsymbol{A}^{(0)}$ the first four columns of $\boldsymbol{Y}$, each normalized to unit norm. Then $\boldsymbol{a}_j = \boldsymbol{y}_j / \|\boldsymbol{y}_j\|$ for $j = 1, \dots, 4$, with $\|\boldsymbol{y}_1\| \approx 1.20$, $\|\boldsymbol{y}_2\| \approx 0.81$, $\|\boldsymbol{y}_3\| \approx 1.06$, $\|\boldsymbol{y}_4\| \approx 1.00$, so

$$\boldsymbol{A}^{(0)} \approx \begin{pmatrix} 0.69 & -0.26 & 0.13 & 0.62 \\ -0.41 & 0.68 & 0.17 & -0.31 \\ 0.26 & 0.22 & -0.58 & 0.55 \\ 0.52 & -0.52 & 0.78 & -0.18 \\ -0.15 & 0.38 & -0.07 & 0.42 \end{pmatrix}.$$

Sparse coding. For each column, OMP with budget $s = 2$ repeatedly picks the atom most correlated with the current residual and least-squares-fits on the chosen support. For $\boldsymbol{y}_1$ the inner products are $\boldsymbol{a}_1^\top \boldsymbol{y}_1 \approx 1.20$, $\boldsymbol{a}_2^\top \boldsymbol{y}_1 \approx -0.87$, $\boldsymbol{a}_3^\top \boldsymbol{y}_1 \approx 0.34$, $\boldsymbol{a}_4^\top \boldsymbol{y}_1 \approx 0.65$. The first equals $\|\boldsymbol{y}_1\|$ exactly, because $\boldsymbol{a}_1 = \boldsymbol{y}_1 / \|\boldsymbol{y}_1\|$; fitting atom 1 alone already drives the residual to zero, so OMP stops with a single term and $\boldsymbol{x}_1 = (1.20, 0, 0, 0)^\top$. The same happens for $\boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_4$: each is one of the initial atoms, hence exactly 1-sparse. Only $\boldsymbol{y}_5, \boldsymbol{y}_6$ genuinely mix two atoms. The code matrix is

$$\boldsymbol{X}^{(1)} \approx \begin{pmatrix} 1.20 & 0 & 0 & 0 & -0.43 & 0.90 \\ 0 & 0.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.06 & 0 & 1.02 & 0 \\ 0 & 0 & 0 & 1.00 & 0 & -0.42 \end{pmatrix}.$$

Initializing the dictionary from columns of $\boldsymbol{Y}$ makes those columns trivial to code; the first real work is the atom update.

Update atom 1. Its active set is $\omega_1 = \{i : X_{1,i} \neq 0\} = \{1, 5, 6\}$, the signals whose first coordinate fired. Form the residual without atom 1, $\boldsymbol{E}_1 = \boldsymbol{Y} - \sum_{j \neq 1} \boldsymbol{a}_j (\boldsymbol{x}^j)^\top$, and keep only its active columns:

$$\boldsymbol{E}_1^R \approx \begin{pmatrix} 0.83 & -0.20 & 0.71 \\ -0.49 & 0.25 & -0.35 \\ 0.31 & -0.12 & 0.30 \\ 0.62 & -0.25 & 0.23 \\ -0.18 & 0.21 & -0.44 \end{pmatrix} \in \mathbb{R}^{5 \times 3}.$$

(Column 1 of $\boldsymbol{E}_1^R$ is just $\boldsymbol{y}_1$, since $\boldsymbol{y}_1$ used only atom 1, so removing the other atoms leaves it untouched.) Its top singular triple is $\sigma_1 \approx 1.58$, $\boldsymbol{u}_1 \approx (0.70, -0.41, 0.28, 0.43, -0.29)^\top$, $\boldsymbol{v}_1 \approx (0.75, -0.28, 0.60)^\top$ (both unit norm: $0.70^2 + 0.41^2 + 0.28^2 + 0.43^2 + 0.29^2 \approx 1.00$ and $0.75^2 + 0.28^2 + 0.60^2 \approx 1.00$). The rank-one fit $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top$ only approximates $\boldsymbol{E}_1^R$, which still carries a smaller second singular value $\sigma_2 \approx 0.32$. The update overwrites the atom and its active coefficients,

$$\boldsymbol{a}_1 \leftarrow \boldsymbol{u}_1, \qquad \boldsymbol{x}_R^1 \leftarrow \sigma_1 \boldsymbol{v}_1 \approx (1.18, -0.44, 0.95)^\top,$$

so the first code row becomes $\boldsymbol{x}^1 \approx (1.18, 0, 0, 0, -0.44, 0.95)^\top$, with the inactive positions $\{2, 3, 4\}$ left at zero. Sweeping atoms $2, 3, 4$ the same way, each using the freshly updated earlier atoms, completes one dictionary update.

Objective. Before the update $\|\boldsymbol{Y} - \boldsymbol{A}^{(0)} \boldsymbol{X}^{(1)}\|_F^2 \approx 0.20$; after the atom-1 step alone it falls to ≈ 0.13, and after the full four-atom sweep $\|\boldsymbol{Y} - \boldsymbol{A}^{(1)} \boldsymbol{X}^{(1)}\|_F^2 \approx 0.11$. The error fell, and the loop proceeds; over twenty iterations it decays monotonically to a noise floor near 0.035. ◊

Two features of the trace are worth naming. The active sets differ sharply across atoms, so the restricted SVDs have different sizes and the algorithm naturally spends more work on heavily used atoms: here ω_1 has three columns while atoms used by a single signal yield a one-column restricted SVD. And the new atom is the top singular vector of a brand-new residual rather than a small nudge of the old one, so K-SVD can genuinely re-orient an atom instead of only perturbing it.

The atom sweep is Gauss–Seidel: each update uses the most recent values of the other atoms, so every per-atom step is the exact minimizer in its coordinates and the objective decreases monotonically. This joint update of atom and coefficients is why K-SVD converges faster and to better dictionaries than MOD, which freezes the codes during the dictionary step.

Remark 13.6 *(The name, and the cost).* The algorithm performs m singular value decompositions per dictionary update, one per atom, generalizing the k mean computations of k-means; hence "K-SVD." When $s = 1$ and coefficients are forced to one, the restricted rank-one fit reduces to averaging the active signals, and K-SVD becomes k-means exactly, closing the circle with Section 13.2. The cost is dominated by the restricted SVDs: a thin SVD of an $n \times |\omega_{j_0}|$ matrix is $O(n^2 |\omega_{j_0}|)$, and since $\sum_{j_0} |\omega_{j_0}| = sN$ (each signal contributes s atom uses), the per-iteration total is $O(n^2 sN)$, cheaper than MOD's $O(nm^2 N + m^3)$ when $s \ll m$.

13.5 Identifiability, Convergence, and Modern Connections

Both algorithms decrease the Frobenius objective at every step, so they converge, but only to a local minimum: the problem is nonconvex, and a poor initialization can trap either method. Monotonicity has a one-line proof. The sparse-coding step replaces each code with the minimizer over the sparse set, so it cannot raise the objective; the dictionary step replaces each atom-and-row pair with the exact rank-one Eckart–Young minimizer of its sub-problem, for which the old value is a feasible point, so it cannot raise it either. The objective is bounded below by zero, so the value sequence converges, though the iterates themselves may drift among equally good optima.

Whether the *true* dictionary is even recoverable, the question of *identifiability*, is separate from convergence and is settled only up to natural symmetries. Permuting the atoms (and the matching code rows), flipping the sign of an atom and its row, or rescaling an atom by c and its row by $1/c$ all leave $\boldsymbol{AX}$ unchanged, so the best one can hope to recover is the dictionary up to permutation, sign, and (absent the unit-norm constraint) scale. Under enough diverse training signals, low coherence among the true atoms, and a sparsity level s small relative to the atom count, the true dictionary is the unique sparse-generating one up to those symmetries, and the alternating loop finds it from a reasonable start. Because local minima abound, the standard safeguard is a multi-start: run from several random initializations and keep the dictionary with the lowest final objective. In practice a good start (often the overcomplete discrete cosine dictionary) and many examples make the learned dictionary reliable.

Remark 13.7 *(Sparse coding as a neural network).* Dictionary learning is the direct ancestor of representation learning. Unrolling a few iterations of the iterative soft-thresholding algorithm of Chapter 7 into a fixed-depth, trainable network gives LISTA, a learned sparse coder that runs in milliseconds. A dictionary $\boldsymbol{A}$ with a sparse code is structurally an autoencoder: $\boldsymbol{A}$ is the decoder, the sparse coding step is the encoder, and K-SVD is an alternating way to train both. The modern sparse autoencoder, used to interpret large language models, is this same object with a learned nonlinear encoder in place of the OMP step. The line from k-means through K-SVD to deep representation learning is unbroken.

13.6 Application: Image Denoising

The signature application of K-SVD is image denoising. Break a noisy image into small overlapping patches (say 8×8), treat each patch as a training signal, and learn a dictionary in which clean patches are sparse. Because noise is not sparse in any fixed direction, sparse-coding each noisy patch over the learned dictionary and reconstructing keeps the signal and discards the noise. Two stopping rules separate the two phases: during *training* OMP stops at $\|\boldsymbol{x}\|_0 = s$ atoms, while during *denoising* it stops when the residual reaches the noise level, $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \le C\sigma\sqrt{n}$, so each patch is explained only to the precision the noise allows. Averaging the overlapping reconstructed patches, weighting each pixel by how many patches cover it, yields a denoised image.

The payoff is concrete. On the standard cameraman test image corrupted with Gaussian noise of standard deviation σ, a learned dictionary beats a fixed overcomplete discrete cosine dictionary across the noise range, precisely because the learned atoms are tuned to the textures of the image at hand (Table 13.1). The gain peaks at moderate noise, where there is both real signal to protect and real noise to remove; at very low noise both dictionaries do well, and at very high noise both lose information.

σ	noisy PSNR	DCT PSNR	K-SVD PSNR	K-SVD gain
10	28.14	33.61	34.53	+0.92 dB
15	24.62	30.74	32.02	+1.28 dB
25	20.18	27.04	29.08	+2.04 dB
50	14.16	22.41	24.05	+1.64 dB

Table 13.1. Denoising the cameraman image with 8×8 patches and a learned dictionary of 256 atoms, against a fixed discrete cosine dictionary, in peak signal-to-noise ratio (higher is better). The learned dictionary wins throughout, by the widest margin at moderate noise.

The same idea reaches well beyond photographs. Learning a dictionary of an artist's brushstrokes *authenticates art*: a genuine patch codes sparsely over the learned dictionary while a forged patch codes poorly, and the reconstruction error flags it. Learning a dictionary of normal machine-vibration spectra performs *anomaly detection*: a healthy reading is sparse in the dictionary and a fault is not, so the residual spikes when a bearing starts to fail. Learning a dictionary of clean audio frames *denoises and inpaints sound*, filling short dropouts the way patch averaging fills corrupted pixels. In every case the recipe is identical: learn atoms on clean examples, then use sparse-coding residual as the signal of interest.

In the Problem Sets

Problem Set 8 (Chapter 27) implements K-SVD from scratch and applies it to image denoising. You alternate OMP sparse coding with the per-atom singular-value dictionary update, watch the learned atoms organize into oriented edges and textures, and compare the denoising quality of the learned dictionary against a fixed overcomplete discrete cosine dictionary. The accompanying notebook reports the reconstruction quality as the dictionary trains.

13.7 Beyond K-SVD: Online Learning, Locality, and Subspace Clustering

K-SVD is the canonical dictionary learner, but the course followed the idea into three directions that matter in practice: learning at scale, coding for speed, and using the sparse code itself to discover structure.

Online dictionary learning. K-SVD processes the entire training set every iteration, which is hopeless when there are millions of patches. **Online dictionary learning** (Mairal and collaborators) instead streams the data in small batches: draw a few signals, sparse-code them against the current dictionary, and take one cheap stochastic update of the atoms that accumulates sufficient statistics rather than re-solving from scratch. It converges to the same kind of dictionary as batch K-SVD but scales to data far too large to hold in memory, and it adapts online as the data distribution drifts.

Locality-constrained linear coding. Sparse coding by OMP or ℓ_1 is accurate but slow, a small optimization per signal. When the dictionary is used as a *feature extractor* for classification, speed matters more than exactness, and **locality-constrained linear coding** (LLC) replaces the sparsity constraint with a *locality* one: code each signal using only its few nearest atoms, with weights that decay with distance. The code is found in closed form (a small local least squares), so it is orders of magnitude faster than OMP, and because nearby atoms vary smoothly the codes are stable, which is what a downstream classifier wants.

Sparse subspace clustering. The most striking use turns dictionary learning inside out. Suppose the data lies in a *union of subspaces*, several low-dimensional planes, and we want to discover which point belongs to which without knowing the subspaces. The key is **self-expressiveness**: a point in a subspace is a linear combination of *other* points in the *same* subspace. So use the data itself as the dictionary and sparse-code each point against all the others,

$$\min_{\boldsymbol{c}_i} \|\boldsymbol{c}_i\|_1 \quad \text{s.t.} \quad \boldsymbol{y}_i = \boldsymbol{Y}\boldsymbol{c}_i, \quad c_{ii} = 0,$$

the constraint $c_{ii} = 0$ forbidding the trivial self-representation. The sparse weights $\boldsymbol{c}_i$ are nonzero almost only at points sharing $\boldsymbol{y}_i$'s subspace (Fig. 13.4), so the matrix $|\boldsymbol{C}| + |\boldsymbol{C}|^\top$ is an affinity graph whose connected pieces are the subspaces; a spectral clustering then reads off the groups. This **sparse subspace clustering** (Elhamifar and Vidal) segments video into rigidly moving objects, clusters faces by illumination subspace, and separates motions, all from the sparse code alone, with no labels.

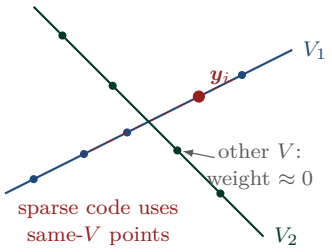


Figure 13.4. Sparse subspace clustering. Each point (red $\boldsymbol{y}_i$) is sparsely expressed by *other* points on its own subspace (dashed links to blue points on V_1), while points on V_2 get essentially zero weight. The resulting affinity graph splits into the subspaces, which spectral clustering recovers.

Notes and References

The Method of Optimal Directions is due to Engan, Aase, and Husøy [27]; K-SVD is Aharon, Elad, and Bruckstein [1]. The connection to k-means and the general sparse-modeling framework are developed in Elad's book [25]. Image denoising with learned dictionaries is Elad and Aharon [26]. The unrolling of iterative thresholding into a trainable network (LISTA) is Gregor and LeCun [30]; the broader link from sparse coding to representation learning is surveyed in Wright and Ma [53]. The Eckart–Young theorem underlying the per-atom update is proved in Appendix D.

Exercises

Exercise 13.1. Show that with $s = 1$ and the nonzero coefficient fixed to 1, the K-SVD per-atom update reduces to replacing $\boldsymbol{a}_{j_0}$ with the mean of the signals in its active set, recovering the k-means center update.

Exercise 13.2. Repeat Example 13.2 from a different initialization, $\boldsymbol{\mu}_1 = (5, 5)$, $\boldsymbol{\mu}_2 = (6, 5)$. Do the assignments still split the two clusters correctly after one cycle? Construct an initialization that converges to a worse (higher-objective) local optimum.

Exercise 13.3. Verify the outer-product identity $\boldsymbol{A}\boldsymbol{X} = \sum_j \boldsymbol{a}_j (\boldsymbol{x}^j)^\top$ on a 2×2 times 2×3 example, and confirm each summand is rank one.

Exercise 13.4. In Example 13.3, compute $\boldsymbol{Y}^\star = \boldsymbol{A}^\star \boldsymbol{X}^\star$ entry by entry and confirm the printed numerical matrix. Which columns of $\boldsymbol{Y}^\star$ are equal, and why does that make atom 3 harder to distinguish from a combination of atoms 1 and 2?

Exercise 13.5. Explain precisely why overwriting the full row $\boldsymbol{x}^{j_0}$ with $\sigma_1 \boldsymbol{v}_1$ (the naive update) can violate the column sparsity constraints, and how restricting to the active set ω_{j_0} prevents it. Relate this to the implementation pitfall in the subtlety box of Section 13.4.

Exercise 13.6. In Example 13.5, verify that $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$ are unit vectors to the printed precision, and check that $\sigma_1 \boldsymbol{v}_1$ reproduces the stated updated code row $\boldsymbol{x}_R^1$.

Exercise 13.7. Both MOD and K-SVD decrease the Frobenius objective monotonically, yet may converge to different dictionaries. Using the per-iteration costs $O(nm^2N + m^3)$ for MOD and $O(n^2 sN)$ for K-SVD, determine for which regime of s, m, n K-SVD's per-iteration step is cheaper, and explain why its joint atom-and-coefficient update tends to reach a lower objective than MOD's frozen-code update.

CHAPTER 14

Nonnegative Matrix Factorization

When you can only add, never subtract, you are forced to build from parts.

The factorizations so far allowed coefficients of any sign. But much real data is *nonnegative*: pixel intensities, word counts, spectra, ratings. Nonnegative matrix factorization (NMF) factors such data, $\boldsymbol{Y} \approx \boldsymbol{A}\boldsymbol{X}$, while requiring both factors to be nonnegative. This one constraint has a striking consequence. Because the model can only *add* atoms, never cancel them, the learned atoms become interpretable *parts*: factor a set of faces and the atoms become noses, eyes, and mouths; factor a set of documents and the atoms become topics. This chapter states the problem, derives the elegant Lee–Seung multiplicative update that solves it, proves it converges by a majorization argument, traces it by hand on a small matrix, replays the original faces experiment, and confronts the one place NMF is fragile: its solutions are not unique unless the data is separable.

14.1 Nonnegativity and Parts-Based Representation

Definition 14.1 (Nonnegative matrix factorization). Given a nonnegative data matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times N}_{\geq 0}$ and a target rank m, NMF seeks nonnegative factors $\boldsymbol{A} \in \mathbb{R}^{n \times m}_{\geq 0}$ and $\boldsymbol{X} \in \mathbb{R}^{m \times N}_{\geq 0}$ minimizing

$$\min_{\boldsymbol{A} \geq 0,\ \boldsymbol{X} \geq 0} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|_F^2.$$

Each data column is then $\boldsymbol{y}_i \approx \sum_k X_{ki} \boldsymbol{a}_k$, a nonnegative combination of the nonnegative atoms.

The constraint is what makes NMF special. In a signed factorization like the SVD, atoms can cancel: a face might be built as one broad pattern minus another, and the atoms themselves look like ghostly whole faces (the eigenfaces of Chapter 16). With nonnegativity, cancellation is forbidden, so every atom must contribute only what is actually present. The result, demonstrated in Lee and Seung's 1999 paper, is that factoring a database of faces yields atoms that are localized facial *parts*, and each face is reconstructed by switching a few parts on. Subtraction-free representation forces a parts-based one.

> This is the conceptual payoff of NMF, and it distinguishes it from every other method in the book. PCA and the SVD give the most compact representation but in terms of abstract, often uninterpretable, signed components. NMF gives a slightly less compact representation in terms of additive parts that a human can read. When interpretability matters more than compression, nonnegativity is the constraint to add.

14.2 The Lee–Seung Multiplicative Updates

The objective is, like dictionary learning, bilinear and nonconvex, so again we alternate: fix $\boldsymbol{A}$ and update $\boldsymbol{X}$, then swap. The difficulty is the nonnegativity constraint. A plain gradient step can push entries negative, and clamping them to zero destroys the descent guarantee. Lee and Seung found an update that respects nonnegativity automatically, with no projection at all.

Write the loss as a trace and expand,

$$\mathcal{L} = \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|_F^2 = \operatorname{tr}(\boldsymbol{Y}^\top \boldsymbol{Y}) - 2\operatorname{tr}(\boldsymbol{Y}^\top \boldsymbol{A}\boldsymbol{X}) + \operatorname{tr}(\boldsymbol{X}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X}).$$

The gradients in the two factors are

$$\nabla_{\boldsymbol{X}} \mathcal{L} = 2\big(\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X} - \boldsymbol{A}^\top \boldsymbol{Y}\big), \qquad \nabla_{\boldsymbol{A}} \mathcal{L} = 2\big(\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^\top - \boldsymbol{Y}\boldsymbol{X}^\top\big).$$

A gradient step on entry X_{ij} would be $X_{ij} \leftarrow X_{ij} - \eta\,[(\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X})_{ij} - (\boldsymbol{A}^\top \boldsymbol{Y})_{ij}]$. The trick is to choose a *different step size for every entry*,

$$\eta_{ij} = \frac{X_{ij}}{(\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X})_{ij}},$$

which is positive and exactly cancels the awkward term. Substituting,

$$X_{ij} \leftarrow X_{ij} - \frac{X_{ij}}{(\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X})_{ij}}\big[(\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X})_{ij} - (\boldsymbol{A}^\top \boldsymbol{Y})_{ij}\big] = X_{ij} - X_{ij} + \frac{X_{ij}\,(\boldsymbol{A}^\top \boldsymbol{Y})_{ij}}{(\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X})_{ij}},$$

and the first two terms cancel, leaving a pure multiplication.

Lee–Seung multiplicative updates

From any nonnegative start, iterate (with elementwise product and division)

$$\boldsymbol{X} \;\leftarrow\; \boldsymbol{X} \circ \frac{\boldsymbol{A}^\top \boldsymbol{Y}}{\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X}}, \qquad \boldsymbol{A} \;\leftarrow\; \boldsymbol{A} \circ \frac{\boldsymbol{Y}\boldsymbol{X}^\top}{\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^\top}.$$

Each factor is multiplied by a ratio of nonnegative quantities, so the factors stay nonnegative at every step with no projection. Lee and Seung proved these updates decrease the Frobenius objective monotonically.

Why nonnegativity is preserved for free: with $\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y} \geq 0$, every entry of $\boldsymbol{A}^\top \boldsymbol{Y}$ and $\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X}$ is a sum of products of nonnegative numbers, hence nonnegative. A nonnegative value times a nonnegative ratio is nonnegative. The constraint that would have required an awkward projection is enforced by the very form of the update. In practice a tiny ε is added to each denominator for numerical safety. The update is two lines of code with no inner solver, which is why it has endured for twenty-five years.

Algorithm 11 Lee–Seung NMF via multiplicative updates

Require: nonnegative data $\boldsymbol{Y} \in \mathbb{R}^{n \times N}_{\geq 0}$, rank m, max iterations T, tolerance ε

initialize $\boldsymbol{A}, \boldsymbol{X}$ with random nonnegative entries (e.g. unif$[0, 1)$)
for $t = 0, 1, \ldots, T-1$ **do**
  $\boldsymbol{X} \leftarrow \boldsymbol{X} \circ (\boldsymbol{A}^\top \boldsymbol{Y}) / (\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X} + \varepsilon)$ ▷ elementwise
  $\boldsymbol{A} \leftarrow \boldsymbol{A} \circ (\boldsymbol{Y}\boldsymbol{X}^\top) / (\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^\top + \varepsilon)$ ▷ elementwise
  if the relative loss change is below ε, **stop**
end for
return nonnegative factors $\boldsymbol{A}$, $\boldsymbol{X}$

Example 14.2 (One multiplicative update by hand). Take the exactly rank-one nonnegative matrix

$$\boldsymbol{Y} = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 2 & 3 \end{bmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \end{pmatrix},$$

and seek a rank-$m = 1$ factorization from the start $\boldsymbol{A} = (1, 1)^\top$, $\boldsymbol{X} = (1, 1, 1)$.

Update $\boldsymbol{X}$. Here $\boldsymbol{A}^\top \boldsymbol{Y} = (2{+}1,\ 4{+}2,\ 6{+}3) = (3, 6, 9)$ and $\boldsymbol{A}^\top \boldsymbol{A} = 2$, so $\boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{X} = (2, 2, 2)$ and

$$\boldsymbol{X} \leftarrow (1, 1, 1) \circ \tfrac{(3,6,9)}{(2,2,2)} = (1.5,\ 3,\ 4.5).$$

Update $\boldsymbol{A}$. Now $\boldsymbol{Y}\boldsymbol{X}^\top = (2{\cdot}1.5{+}4{\cdot}3{+}6{\cdot}4.5,\ 1{\cdot}1.5{+}2{\cdot}3{+}3{\cdot}4.5) = (42, 21)$ and $\boldsymbol{X}\boldsymbol{X}^\top = 1.5^2{+}3^2{+}4.5^2 = 31.5$, so $\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^\top = (31.5, 31.5)$ and

$$\boldsymbol{A} \leftarrow (1, 1) \circ \tfrac{(42,21)}{(31.5,31.5)} = (1.333,\ 0.667)^\top.$$

Result. The product is now $\boldsymbol{A}\boldsymbol{X} = \left[\begin{smallmatrix} 1.333 \\ 0.667 \end{smallmatrix}\right]\left[\begin{smallmatrix} 1.5 & 3 & 4.5 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 2 & 4 & 6 \\ 1 & 2 & 3 \end{smallmatrix}\right] = \boldsymbol{Y}$, exact after a single sweep, because $\boldsymbol{Y}$ was genuinely rank one. Notice the recovered factors are not the "obvious" $(2, 1)^\top$ and $(1, 2, 3)$ but scaled versions of them, $\boldsymbol{A} = (2, 1)^\top \cdot \frac{2}{3}$ and $\boldsymbol{X} = (1, 2, 3) \cdot \frac{3}{2}$. The product is identical because the scale $\frac{2}{3} \cdot \frac{3}{2} = 1$ cancels. That cancellation is the scaling ambiguity of Section 14.6, here made concrete. ◊

14.3 Convergence by Majorization

The multiplicative updates are not an arbitrary recipe; they are the exact minimizers of a carefully chosen upper bound on the loss. This is the *majorization–minimization* (MM) framework, and it both proves monotone descent and explains where the updates come from.

Theorem 14.3 (Lee–Seung monotone convergence). Each multiplicative update satisfies $\|\boldsymbol{Y} - \boldsymbol{A}^{(t+1)}\boldsymbol{X}^{(t+1)}\|_F^2 \le \|\boldsymbol{Y} - \boldsymbol{A}^{(t)}\boldsymbol{X}^{(t)}\|_F^2$, with equality only at a fixed point. The loss sequence is nonincreasing and bounded below by zero, hence convergent.

Proof sketch. An **auxiliary function** for $\mathcal{L}$ at the current iterate $\boldsymbol{X}^{(t)}$ is a function $G(\boldsymbol{X}, \boldsymbol{X}^{(t)})$ with three properties: it dominates the loss, $G(\boldsymbol{X}, \boldsymbol{X}^{(t)}) \ge \mathcal{L}(\boldsymbol{A}^{(t)}, \boldsymbol{X})$ for all $\boldsymbol{X} \ge 0$; it touches it at the current point, $G(\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(t)}) = \mathcal{L}(\boldsymbol{A}^{(t)}, \boldsymbol{X}^{(t)})$; and it is minimized in closed form. Given such a G, the iterate $\boldsymbol{X}^{(t+1)} = \arg\min_{\boldsymbol{X} \ge 0} G(\boldsymbol{X}, \boldsymbol{X}^{(t)})$ never increases the loss, because

$$\mathcal{L}(\boldsymbol{X}^{(t+1)}) \le G(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)}) \le G(\boldsymbol{X}^{(t)}, \boldsymbol{X}^{(t)}) = \mathcal{L}(\boldsymbol{X}^{(t)}),$$

the first step by domination, the second because $\boldsymbol{X}^{(t+1)}$ minimizes G, the third by touching. For NMF the working auxiliary function is the quadratic

$$G(\boldsymbol{X}, \boldsymbol{X}^{(t)}) = \mathcal{L}(\boldsymbol{X}^{(t)}) + \sum_{ij} (\nabla_{\boldsymbol{X}} \mathcal{L})_{ij} (X_{ij} - X_{ij}^{(t)}) + \tfrac{1}{2} \sum_{ij} \frac{(\boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{X}^{(t)})_{ij}}{X_{ij}^{(t)}} (X_{ij} - X_{ij}^{(t)})^2,$$

a separable upper bound whose curvature coefficient $(\boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{X}^{(t)})_{ij} / X_{ij}^{(t)}$ is chosen exactly large enough to dominate the true quadratic. Setting $\partial G / \partial X_{ij} = 0$ and solving gives, after the cancellation, the multiplicative update. The same construction with the roles of $\boldsymbol{A}$ and $\boldsymbol{X}$ swapped handles the $\boldsymbol{A}$ step. The EM algorithm is the same MM idea with a different majorant. ■

> Monotone descent is not convergence to the global minimum. NMF is nonconvex, and the updates settle at a stationary point that may be a local minimum, a saddle, or a boundary point of $\mathbb{R}_{\ge 0}^{m \times N}$. The standard safeguard is the same as for k-means and K-SVD: several random restarts, keep the lowest-loss result. A deterministic warm start, NNDSVD (a sign-rectified truncated SVD of $\boldsymbol{Y}$), often converges faster than a random one.

14.4 The Parts-Based Representation in Action

The experiment that put NMF on the map is Lee and Seung's 1999 decomposition of a face database. The CBCL set has 2,429 frontal faces at 19×19 pixels; vectorize each to length 361 and stack them into $\boldsymbol{Y} \in \mathbb{R}_{\geq 0}^{361 \times 2429}$, every entry a pixel intensity in $[0, 255]$, naturally nonnegative. Choosing $m = 49$ atoms (a 7×7 display grid) and running Algorithm 11 for a few hundred iterations from a random nonnegative start, the loss falls monotonically to a plateau.

The payoff is in the atoms. Reshaped to 19×19, each learned atom is a *piece* of a face: a left eye, a right eye, a nose, a mouth, a chin contour, a forehead. Together the forty-nine atoms tile the face into recognizable parts, and any individual face is a nonnegative sum of a few of them (Fig. 14.1). Run on the same data, the signed methods behave differently: PCA's eigenfaces are ghostly whole-face templates with mixed-sign pixels, and K-SVD's atoms are blurry whole faces combined with signed coefficients. The nonnegativity constraint, alone, with no prior on locality or support, is what produces the parts, because the alternative (holistic templates that must be subtracted from each other) is simply unreachable inside the nonnegative orthant.

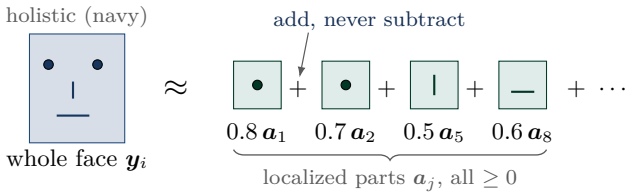


Figure 14.1. NMF on a face: the whole image is reconstructed as a nonnegative weighted sum of localized parts (eye, eye, nose, mouth, ...). Each atom $\boldsymbol{a}_j$ is nonzero only on a small region and each coefficient $X_{ji} \geq 0$. The model has no eraser: once a part is added, no negative coefficient can remove it, which is exactly what forces the atoms to be local.

The parts emergence is not a quirk of faces. It is a generic consequence of imposing nonnegativity on a bilinear factorization of nonnegative data: with only additive composition the model must find ingredients that combine cleanly, and clean additive composition favors localized, sparse atoms. The same effect turns documents into topics and audio into source spectra. The 1999 observation seeded a research thread, from parts-based models of the visual cortex to the early layers of convolutional networks, that is still active.

14.5 Where NMF Sits Among the Factorizations

NMF, dictionary learning, and PCA are three constrained versions of the same factorization $\boldsymbol{Y} \approx \boldsymbol{A}\boldsymbol{X}$, differing in what they ask of the factors. PCA (the SVD) asks for orthonormal atoms and gives the most compact, signed, holistic representation. Dictionary learning (Chapter 13) asks for *sparse codes* and an overcomplete dictionary, putting the structure in $\boldsymbol{X}$. NMF asks for *nonnegativity* of both factors and puts the structure in the additivity, which produces parts. Table 14.1 lines them up.

Method	Constraint	Representation
PCA / SVD	orthonormal atoms	compact, signed, holistic
Dictionary learning	sparse codes, overcomplete	sparse, signed
NMF	both factors ≥ 0	additive, parts-based, interpretable

Table 14.1. Three constrained factorizations of $\boldsymbol{Y} \approx \boldsymbol{A}\boldsymbol{X}$. The constraint chosen determines what the atoms mean.

The contrast with K-SVD is the sharpest, because both learn a dictionary of the same size, yet they place the sparsity in different factors. K-SVD demands a sparse *code*: every signal uses few atoms, but those atoms are dense, signed, whole-object templates. NMF demands nonnegative factors and gets sparse *atoms* for free: the parts are localized

and mostly zero, while a signal may use many of them. K-SVD answers "which few templates," NMF answers "which parts are present." One puts structure in $\boldsymbol{X}$, the other in $\boldsymbol{A}$.

14.6 Ill-Posedness and Separability

NMF has one genuine weakness we must be honest about: in general its solution is not unique. The optimization can have infinitely many distinct nonnegative factor pairs giving the same product.

The algebraic picture. If $(\boldsymbol{A}, \boldsymbol{X})$ solves $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ and $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is invertible, then $\boldsymbol{A}' = \boldsymbol{A}\boldsymbol{Q}$ and $\boldsymbol{X}' = \boldsymbol{Q}^{-1}\boldsymbol{X}$ give $\boldsymbol{A}'\boldsymbol{X}' = \boldsymbol{A}\boldsymbol{X} = \boldsymbol{Y}$ as well. The question is which $\boldsymbol{Q}$ keep both factors nonnegative. A permutation matrix does, and just relabels the atoms; a positive diagonal $\boldsymbol{Q} = \mathrm{diag}(c_1, \dots, c_m)$ does, and just rescales atom j by c_j while rescaling its code row by $1/c_j$ (the scaling already seen in Example 14.2). These two are the unavoidable trivial ambiguities. The trouble is that *non-trivial* $\boldsymbol{Q}$, neither permutation nor diagonal, can also keep both factors nonnegative, giving genuinely different parts.

The geometric picture. Read each data column as a point in $\mathbb{R}^n_{\geq 0}$. The factorization says every $\boldsymbol{y}_i$ is a nonnegative combination of the atoms, so it lies in their *conic hull* $\mathrm{cone}(\boldsymbol{a}_1, \dots, \boldsymbol{a}_m) = \{\sum_j \alpha_j \boldsymbol{a}_j : \alpha_j \geq 0\}$. NMF asks for m rays whose cone contains all the data. The ambiguity is now visible: if one cone wraps the data, a different, tilted cone usually wraps it too (Fig. 14.2, left and center). Both are valid NMFs with different atoms.

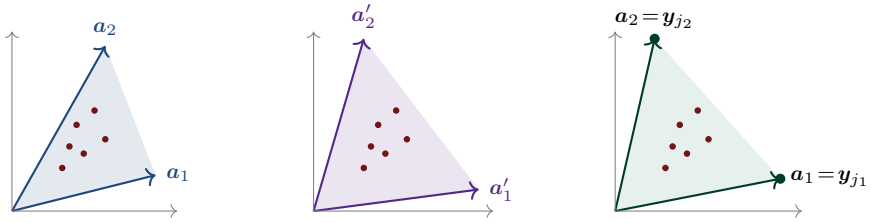


Figure 14.2. Why NMF is ill-posed, and how separability fixes it. Left and center: two different cones both contain the same data, so two different nonnegative factorizations are equally valid. Right: under separability the atoms are themselves data points, the extreme rays of the data cone, which pins the cone down and makes the factorization unique up to permutation and scaling.

Separability restores uniqueness. The cleanest repair, due to Donoho and Stodden, is the *separability* assumption: each atom appears essentially alone in at least one data column, so the columns of $\boldsymbol{A}$ are themselves (scaled) columns of $\boldsymbol{Y}$. Geometrically the atoms are the extreme rays of the data cone, and every other point is a convex combination of those pure ones (Fig. 14.2, right). When separability holds, the cone, and hence the atoms, are uniquely determined, and finding the factorization reduces to identifying the m pure columns, a sparse-row selection problem solvable by convex methods. When it fails, NMF is still a useful summary, just not a canonical one.

14.7 NMF Meets K-Means: The Ding–He–Simon Theorem

A final connection ties this chapter back to where dictionary learning began. Drop nonnegativity on $\boldsymbol{A}$ but keep it on $\boldsymbol{X}$, and add an orthogonality constraint on the rows of $\boldsymbol{X}$:

$$\min_{\boldsymbol{A},\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{X}\|_F^2 \quad \text{subject to} \quad \boldsymbol{X} \geq 0,\ \boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{I}_m.$$

Nonnegative rows that are also orthonormal must have disjoint supports, so each column of $\boldsymbol{X}$ has at most one nonzero: the one-hot code of k-means.

Theorem 14.4 (Ding–He–Simon). The orthogonal semi-NMF above is equivalent to k-means clustering of the columns of $\boldsymbol{Y}$ into m clusters.

Proof sketch. Encode a k-means assignment by an indicator matrix $\boldsymbol{H} \in \{0,1\}^{m\times N}$ with one nonzero per column and centroids in the columns of $\boldsymbol{M}$. Then the assigned centroid of point i is the i-th column of $\boldsymbol{MH}$, so k-means is exactly $\min_{\boldsymbol{M},\boldsymbol{H}} \|\boldsymbol{Y} - \boldsymbol{MH}\|_F^2$ over one-hot $\boldsymbol{H}$. Relaxing $\boldsymbol{H}$ to a nonnegative $\boldsymbol{X}$ with orthonormal rows (after scaling each cluster by $|\cdot|^{-1/2}$) is the orthogonal semi-NMF, and the relaxed optimum coincides with k-means up to a column permutation. Maximizing the equivalent $\operatorname{tr}(\boldsymbol{XY}^\top\boldsymbol{YX}^\top)$ over orthonormal $\boldsymbol{X}$ is achieved by the top m eigenvectors of the Gram matrix $\boldsymbol{Y}^\top\boldsymbol{Y}$, which also ties the problem to spectral clustering. ∎

The theorem is more useful for intuition than computation. It says k-means, NMF, and spectral clustering are corners of one factorization framework: same Frobenius loss, different constraint (one-hot codes, nonnegative codes with orthogonality, general nonnegative codes). Walking the constraint set continuously interpolates between them. The whole of Part V is one factorization seen through different constraints, each adapted to a different kind of data.

14.8 Application: Topic Modeling

The canonical text application is topic modeling. Build a nonnegative *word-by-document* matrix $\boldsymbol{Y}$ whose entry Y_{ij} counts how often word i appears in document j. NMF factors it as $\boldsymbol{Y} \approx \boldsymbol{AX}$, where each column of $\boldsymbol{A}$ is a *topic*, a nonnegative distribution over words (a topic about finance puts weight on "market," "stock," "rate"), and each column of $\boldsymbol{X}$ expresses a document as a nonnegative mixture of topics. Because nothing can be subtracted, a topic can only *add* words and a document can only *add* topics, which is exactly how text is generated, so the parts NMF finds are genuine themes. The same factorization underlies the parts-based image decomposition of Section 14.4 and the spectral unmixing of hyperspectral images into constituent materials, where each atom is a pure material spectrum and separability is the assumption that every material appears pure in at least one pixel.

In the Problem Sets

Problem Set 9 (Chapter 28) implements the Lee–Seung updates from scratch and applies them to topic discovery on a document collection. You build the word-document count matrix, run the two multiplicative updates to convergence, and read off the learned topics as the top words of each atom, verifying that nonnegativity yields interpretable themes where a signed factorization would not. The accompanying notebook reports the reconstruction error as the factorization converges.

14.9 Entropy as a Sparsity Measure

The ℓ_0 count and the ℓ_1 norm are not the only ways to ask for sparsity. The course met a third, which fits nonnegative data especially naturally: **entropy**. A nonnegative vector $\boldsymbol{x} \geq \boldsymbol{0}$ normalized to sum one, $p_i = x_i / \sum_j x_j$, is a probability distribution, and its **Shannon entropy**

$$H(\boldsymbol{p}) = -\sum_i p_i \log p_i$$

measures how spread out it is. It is 0 when all the mass sits on one entry (maximally concentrated, maximally sparse) and $\log N$ when the mass is uniform (maximally spread). So *low entropy means concentrated means sparse*, and minimizing entropy is an alternative sparsity prior, smooth where ℓ_0 is discrete.

Example 14.5 (Entropy tracks concentration). Three distributions on $N = 3$ entries. The spike $\boldsymbol{p} = (1, 0, 0)$ has $H = 0$: all mass in one place. The concentrated $\boldsymbol{p} = (0.8, 0.1, 0.1)$ has $H = -0.8\ln 0.8 - 2(0.1\ln 0.1) \approx 0.18 + 0.46 = 0.64$. The uniform $\boldsymbol{p} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ has $H = \ln 3 \approx 1.10$, the maximum. Entropy rises monotonically from spike to spread, so driving it down drives the vector toward a few dominant entries. ◊

Adding an entropy penalty to nonnegative matrix factorization, $\min \|\boldsymbol{Y} - \boldsymbol{AX}\|_F^2 + \beta \sum_i H(\boldsymbol{X}_{:,i})$, makes each document's topic mixture concentrate on a few topics, sharpening the parts-based representation beyond what nonnegativity

alone gives, in the same spirit as an ℓ_1 penalty but matched to the probability-simplex structure of the coefficients. The same idea drives **robust entropy minimization**, which replaces the combinatorial sparse-or-low-rank objective by a smooth entropy of the spectrum or of the rows, and is robust to the heavy-tailed corruption that breaks a least-squares fit. Entropy joins ℓ_1 and the nuclear norm as a third surrogate for the same goal: prefer the explanation that concentrates its mass.

Notes and References

Nonnegative matrix factorization and the parts-based representation are from Lee and Seung [34]; the multiplicative update rules and their monotone-convergence proof via auxiliary functions are in the follow-up [35]. The equivalence of orthogonal NMF with k-means clustering is Ding, He, and Simon [19]. The separability condition that makes NMF identifiable, and the cone geometry behind it, are due to Donoho and Stodden, and are discussed in the broader treatment of Wright and Ma [53]; the connection to sparse modeling and dictionary learning is in Elad [25].

Exercises

Exercise 14.1. Verify that the Lee–Seung step size $\eta_{ij} = X_{ij}/(\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X})_{ij}$ turns the gradient step into the multiplicative update by carrying out the cancellation explicitly, and confirm the analogous computation for the $\boldsymbol{A}$ update.

Exercise 14.2. Continue Example 14.2 for a second full update and confirm the factors do not change (the factorization is already exact). Then rerun from the start $\boldsymbol{A} = (1, 2)^\top$, $\boldsymbol{X} = (1, 1, 1)$ and verify it still converges to a scaling of $(2, 1)^\top$ and $(1, 2, 3)$.

Exercise 14.3. Show that if $\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y}$ are entrywise nonnegative, then one multiplicative update keeps them nonnegative. Where could a division by zero occur, and how is it handled?

Exercise 14.4. State the three defining properties of an auxiliary function G, and use them to prove the two-line inequality $\mathcal{L}(\boldsymbol{X}^{(t+1)}) \le \mathcal{L}(\boldsymbol{X}^{(t)})$ in Theorem 14.3.

Exercise 14.5. Contrast the atoms produced by PCA and by NMF on a dataset of faces: explain, in terms of cancellation, why PCA atoms look like whole faces while NMF atoms look like parts.

Exercise 14.6. Exhibit a small nonnegative matrix with two genuinely different nonnegative factorizations of the same rank (find a non-trivial $\boldsymbol{Q}$ keeping both factors nonnegative), demonstrating non-uniqueness, and state what the separability condition would add to pin down a canonical one.

Exercise 14.7. Explain why nonnegative orthonormal rows must have disjoint supports, and use this to argue that the orthogonal semi-NMF of Theorem 14.4 reduces to one-hot (k-means) codes.

PART VI

Extensions and Applications

CHAPTER 15

Joint Sparsity and Multiple Measurement Vectors

When many signals share the same support, recovering them together beats recovering each alone.

So far each recovery problem concerned a single sparse vector. Many applications, though, present a whole family of sparse signals that share the same support: the pixels of a hyperspectral image at different wavelengths, the readings of a multi-electrode array over time, several snapshots of one scene. Recovering them one at a time wastes the most valuable fact about them, that their nonzeros line up. This chapter develops the multiple-measurement-vector model, the mixed norms that express shared support, the three guarantees that make joint recovery provably easier than single-vector recovery, and the simultaneous greedy algorithm that exploits the structure. The whole theory is the vector theory of Part II lifted from vectors to matrices, with the ℓ_1 norm replaced by a norm that couples the columns.

15.1 One Support, Many Signals

Stack K measurement vectors as the columns of $\boldsymbol{Y} = [\boldsymbol{y}_1 \mid \cdots \mid \boldsymbol{y}_K]$ and the unknown signals as $\boldsymbol{X} = [\boldsymbol{x}_1 \mid \cdots \mid \boldsymbol{x}_K]$. If all are sensed through the same matrix $\boldsymbol{A}$, the K equations $\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{x}_k$ stack into one,

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}, \qquad \boldsymbol{Y} \in \mathbb{R}^{m\times K},\ \boldsymbol{A} \in \mathbb{R}^{m\times N},\ \boldsymbol{X} \in \mathbb{R}^{N\times K}.$$

This is the **multiple-measurement-vector** (MMV) model. Nothing forces us to use the matrix structure; the columns are decoupled, and we could run K separate single-vector recoveries. The point is the structural assumption that makes joint recovery pay.

Definition 15.1 (Joint sparsity). The columns of $\boldsymbol{X}$ are **jointly sparse** if they all have the same support: the active atoms are common to every column, only the coefficient values differ. Then $\boldsymbol{X}$ is *row-sparse*, a few fully populated rows in a sea of zero rows, and the quantity to minimize is the number of nonzero rows,

$$\|\boldsymbol{X}\|_{\text{row},0} = \#\{\text{nonzero rows of } \boldsymbol{X}\}.$$

Figure 15.1 draws the structure. Each column is a sparse code for one measurement; joint sparsity aligns those codes into a few active rows. Why exploit it? Because the K columns are K independent pieces of evidence about the *same* support. A wrong support must be consistent with all K columns at once, a far more stringent test than matching one, so identifying the support gets *easier* as K grows. The next section makes this precise: the uniqueness threshold improves by the rank of $\boldsymbol{Y}$, and fewer measurements per signal are needed than recovering each independently.

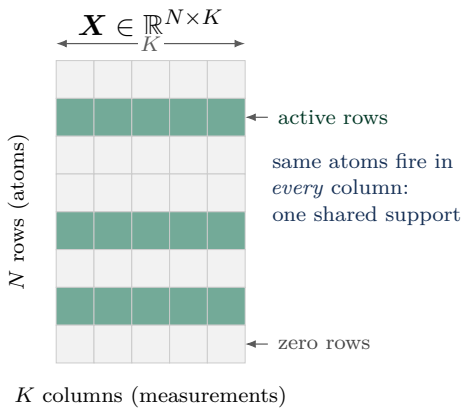


Figure 15.1. Row-sparse structure in the MMV model. Joint sparsity forces all K columns to share a support, so the nonzeros align into a few fully active rows (green) while the rest of the rows vanish. The same atoms fire in every column, which is the structure recovery exploits: it minimizes the number of active rows.

15.2 Mixed Norms

To count and then relax the active rows we need norms that act on rows as units.

Definition 15.2 (Mixed $\ell_{p,q}$ norm). For $\boldsymbol{X} \in \mathbb{R}^{N\times K}$ with rows $\boldsymbol{X}_{k,:}$,

$$\|\boldsymbol{X}\|_{p,q} = \Big(\sum_{k=1}^{N} \|\boldsymbol{X}_{k,:}\|_q^p\Big)^{1/p}:$$

first take the inner ℓ_q norm *along* each row, producing N row-norms, then take the outer ℓ_p norm *down* that list.

Two choices matter. With outer ℓ_0, the row-norm of a row is nonzero exactly when the row is, so $\|\boldsymbol{X}\|_{0,q} = \|\boldsymbol{X}\|_{\text{row},0}$ counts active rows regardless of q. With outer ℓ_1 and inner ℓ_2,

$$\|\boldsymbol{X}\|_{1,2} = \sum_{k=1}^{N} \|\boldsymbol{X}_{k,:}\|_2,$$

the convex surrogate, which is the *group lasso*. The two stages do two jobs: the inner ℓ_2 treats each row as an indivisible group (all K entries of a surviving row are kept or killed together, which enforces the *joint* structure), while the outer ℓ_1 sum over rows drives most row-energies to zero (which enforces *sparsity*). Take the ℓ_2 by row for jointness, the ℓ_1 across rows for sparsity.

Example 15.3 (A mixed-norm computation). For $\boldsymbol{X} = \left(\begin{smallmatrix} 3 & 4 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$, the row ℓ_2 norms are $5, 0, 1, 0$. The row count is $\|\boldsymbol{X}\|_{\text{row},0} = 2$, and the surrogate is $\|\boldsymbol{X}\|_{1,2} = 5 + 0 + 1 + 0 = 6$. Exactly as $\|\cdot\|_{\text{row},0}$ counts active rows the way $\|\cdot\|_0$ counts active entries, $\|\cdot\|_{1,2}$ is the convex stand-in the way $\|\cdot\|_1$ was. ◊

The group lasso has its own proximal operator, the matrix analogue of soft thresholding, and it shows exactly how the mixed norm enforces joint structure. The prox of $\lambda\|\cdot\|_2$ acting on a single row $\boldsymbol{r}$ is the **block soft threshold**

$$\text{bst}_\lambda(\boldsymbol{r}) = \max\Big(1 - \frac{\lambda}{\|\boldsymbol{r}\|_2},\, 0\Big)\,\boldsymbol{r},$$

which shrinks the row's magnitude while keeping its direction, and zeroes the *whole* row at once when its energy falls below λ.

Example 15.4 (Keep or kill a whole row). Apply the block soft threshold with $\lambda = 2$ to two rows. A strong row $\boldsymbol{r} = (3, 4)$ has $\|\boldsymbol{r}\|_2 = 5$, so it survives, scaled: $\text{bst}_2(\boldsymbol{r}) = (1 - \frac{2}{5})(3, 4) = 0.6\,(3, 4) = (1.8, 2.4)$, norm 3. A weak row

$\boldsymbol{r}' = (0.6, 0.8)$ has $\|\boldsymbol{r}'\|_2 = 1 < 2$, so $1 - \frac{2}{1} < 0$ and the whole row is set to zero. Both entries of a row are kept or killed together, never one without the other, which is precisely the joint-support structure the group lasso enforces. Compare ordinary soft thresholding, which would treat each entry independently and could keep one coordinate of a row while dropping another, breaking the shared support. ◊

The recovery problems then mirror the vector case exactly. The combinatorial problem minimizes the row count,

$$(\mathrm{P}_{R0}) \qquad \min_{\boldsymbol{X}} \|\boldsymbol{X}\|_{\mathrm{row},0} \ \text{ s.t. } \ \boldsymbol{A}\boldsymbol{X} = \boldsymbol{Y},$$

NP-hard like its ancestor (P_0), and the convex relaxation replaces it by the group lasso

$$(\mathrm{P}_{1,2}) \qquad \min_{\boldsymbol{X}} \|\boldsymbol{X}\|_{1,2} \ \text{ s.t. } \ \boldsymbol{A}\boldsymbol{X} = \boldsymbol{Y}.$$

The rest of the chapter answers three questions about these two problems, each the matrix lift of something from Part II: when is the row-sparse solution unique, when does the convex relaxation find it, and when does the greedy algorithm find it.

15.3 Three Guarantees

15.3.1 Uniqueness improves with the rank of the data

The first question is the matrix version of the spark bound of Chapter 4: when is the row-sparse solution of (P_{R0}) the only one? The answer is strictly better than the single-vector bound, by exactly the rank of the measurements.

Theorem 15.5 (Joint-sparse uniqueness). For $\boldsymbol{Y} \in \mathbb{R}^{m \times K}$ there is *at most one* jointly S-row-sparse $\boldsymbol{X}$ with $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ if and only if

$$S < \frac{\operatorname{spark}(\boldsymbol{A}) - 1 + \operatorname{rank}(\boldsymbol{Y})}{2}.$$

A coherence-based sufficient form, using $\operatorname{spark}(\boldsymbol{A}) \geq 1 + 1/\mu(\boldsymbol{A})$, is $S < \frac{1}{2}\big(\mu(\boldsymbol{A})^{-1} + \operatorname{rank}(\boldsymbol{Y})\big)$.

Set $K = 1$. Then $\boldsymbol{Y}$ is a single nonzero column, $\operatorname{rank}(\boldsymbol{Y}) = 1$, and the bound collapses to $S < \operatorname{spark}(\boldsymbol{A})/2$, the single-vector spark bound of Theorem 4.1 exactly. The matrix theorem contains the vector theorem as its $K = 1$ case, as a good generalization should. As K grows and the active rows produce a high-rank measurement, $\operatorname{rank}(\boldsymbol{Y})$ can be as large as $\min(m, K, S)$, and the admissible row-sparsity rises with it.

Why matrices are easier: the collision argument

A failure of uniqueness is a *collision*: two distinct jointly sparse matrices $\boldsymbol{X} \neq \boldsymbol{X}'$ with $\boldsymbol{A}\boldsymbol{X} = \boldsymbol{A}\boldsymbol{X}'$. Their difference $\boldsymbol{D} = \boldsymbol{X} - \boldsymbol{X}'$ is row-sparse (row support inside the union of the two, of size at most $2S$) and lies in the null space, $\boldsymbol{A}\boldsymbol{D} = \boldsymbol{0}$. So *every column* of $\boldsymbol{D}$ must be a sparse null vector of $\boldsymbol{A}$ on the same row set. Forcing one column into the null space on a given support is one constraint; forcing all K columns in, while $\boldsymbol{D}$ carries rank up to $\operatorname{rank}(\boldsymbol{Y})$, is many constraints at once. The more independent directions $\boldsymbol{Y}$ explores, the harder the collision must work, and that extra difficulty is exactly the $+\operatorname{rank}(\boldsymbol{Y})$ term. More columns sharing one support makes collisions harder, hence uniqueness easier.

The bound buys a trade in either direction. For a *fixed* sensing matrix you may recover matrices with more active rows than you could recover active entries in a vector. For a *fixed* target sparsity you may use a worse sensing matrix: $\operatorname{spark}(\boldsymbol{A}) > 2S$ becomes $\operatorname{spark}(\boldsymbol{A}) > 2S + 1 - \operatorname{rank}(\boldsymbol{Y})$, a weaker demand. The same effect shows up in measurement counts: where a single s-sparse vector needs $m \gtrsim Cs\log(N/s)$ random rows, joint recovery of an S-row-sparse matrix needs $m \gtrsim C_R\, S \log(N/S)$ with a constant C_R that shrinks as K grows. The many columns amortize the cost of pinning down the shared support.

15.3.2 The convex relaxation, through coherence

Theorem 15.5 says when the row-sparse matrix is the unique solution of the combinatorial problem; it says nothing about whether the convex relaxation finds it. The second guarantee closes that gap with a coherence condition, the matrix lift of the ℓ_1 recovery theorem of Chapter 8.

Theorem 15.6 ($\ell_{1,2}$ exact recovery via coherence). If $\boldsymbol{X} \in \mathbb{R}^{N\times K}$ is jointly S-row-sparse with $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ and

$$S < \frac{1}{2}\Big(1 + \frac{1}{\mu(\boldsymbol{A})}\Big),$$

then $\boldsymbol{X}$ is the unique solution of the convex program $(\mathrm{P}_{1,2})$, and it coincides with the solution of (P_{R0}).

This is *identical* to the single-vector ℓ_1 recovery threshold, with the vector sparsity s replaced by the row-sparsity S. The coherence guarantee does not improve in the matrix case, which seems to clash with Theorem 15.5.

> There is no contradiction; the two theorems live at different sharpness. Theorem 15.5 is the *exact* uniqueness condition for the combinatorial problem, and it genuinely improves with $\operatorname{rank}(\boldsymbol{Y})$. Theorem 15.6 is a *sufficient* condition for the convex relaxation, written through the coherence $\mu(\boldsymbol{A})$, the largest inner product between two columns. Coherence is a worst-case summary of $\boldsymbol{A}$ alone, and a worst-case summary cannot see the rank of $\boldsymbol{Y}$, so the coherence guarantee is just the single-vector one lifted verbatim. The improvement Theorem 15.5 promises is real but invisible to a coherence-only analysis; sharper average-case analyses for random $\boldsymbol{A}$ do recover the benefit, and that is where the smaller measurement count C_R comes from.

15.4 Simultaneous Orthogonal Matching Pursuit

The greedy method lifts OMP from vectors to matrices. Every step has a matrix counterpart, and the only new ingredient is how the K columns vote on which atom to add.

Algorithm 12 Simultaneous Orthogonal Matching Pursuit (SOMP)

Require: $\boldsymbol{A}$, measurements $\boldsymbol{Y}$, row-sparsity s, inner index q (usually 2)

$\hat{\boldsymbol{X}} \leftarrow \boldsymbol{0}$, $\boldsymbol{R} \leftarrow \boldsymbol{Y}$, $S \leftarrow \emptyset$

repeat

  $n^\star \leftarrow \arg\max_n \|\boldsymbol{a}_n^\top \boldsymbol{R}\|_q$ ▷ atoms vote: row of $\boldsymbol{A}^\top\boldsymbol{R}$ of largest ℓ_q norm

  $S \leftarrow S \cup \{n^\star\}$

  $\hat{\boldsymbol{X}}_S \leftarrow \boldsymbol{A}_S^\dagger \boldsymbol{Y}$ ▷ joint least-squares refit of all K columns

  $\boldsymbol{R} \leftarrow \boldsymbol{Y} - \boldsymbol{A}_S \hat{\boldsymbol{X}}_S$

until $\|\boldsymbol{R}\|_F$ small or $|S| = s$

return row-sparse $\hat{\boldsymbol{X}}$

The selection rule is the heart of the matter. In single-vector OMP the score of atom n was the scalar $|\boldsymbol{a}_n^\top \boldsymbol{r}|$; here the residual is a matrix, so $\boldsymbol{a}_n^\top \boldsymbol{R}$ is a length-K row of correlations, one per measurement, and the score is its ℓ_q norm. The ℓ_q fuses the K pieces of evidence into a single number, and the atom that is *collectively* most correlated across all measurements wins. This is the one place the joint structure enters, and it is exactly where the columns vote together. The refit then projects *all* K columns through the same projector onto the chosen atoms, so the estimate is row-sparse by construction, sharing one support across every column.

Example 15.7 (One SOMP iteration by hand). Take $m = 3$, $N = 5$, $K = 2$ with the unit-norm dictionary and the jointly 2-row-sparse truth (support $\{1, 3\}$)

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}, \qquad \boldsymbol{X} = \begin{bmatrix} 2 & -1 \\ 0 & 0 \\ 3 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad \boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} = \begin{bmatrix} 2 & -1 \\ 0 & 0 \\ 3 & 4 \end{bmatrix},$$

since $\boldsymbol{Y}_{:,1} = 2\boldsymbol{a}_1 + 3\boldsymbol{a}_3 = (2, 0, 3)^\top$ and $\boldsymbol{Y}_{:,2} = -\boldsymbol{a}_1 + 4\boldsymbol{a}_3 = (-1, 0, 4)^\top$.

Select. With $\boldsymbol{R}_0 = \boldsymbol{Y}$ the correlations and their row ℓ_2 scores are

$$\boldsymbol{A}^\top \boldsymbol{R}_0 = \begin{bmatrix} 2 & -1 \\ 0 & 0 \\ 3 & 4 \\ 2/\sqrt{2} & -1/\sqrt{2} \\ 3/\sqrt{2} & 4/\sqrt{2} \end{bmatrix}, \qquad \text{scores} = \left(\sqrt{5},\ 0,\ 5,\ \sqrt{2.5},\ \sqrt{12.5}\right) \approx (2.24, 0, 5, 1.58, 3.54).$$

The winner is row 3 with score 5, a true support atom, so $S_1 = \{3\}$.

Refit. With $\boldsymbol{A}_{S_1} = \boldsymbol{a}_3 = (0, 0, 1)^\top$ the pseudoinverse is $\boldsymbol{a}_3^\top$, and $\hat{\boldsymbol{X}}_{S_1} = \boldsymbol{a}_3^\top \boldsymbol{Y} = (3,\ 4)$, the exact row-3 coefficients. The residual

$$\boldsymbol{R}_1 = \boldsymbol{Y} - \boldsymbol{a}_3\,(3,\ 4) = \begin{bmatrix} 2 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

still has $\|\boldsymbol{R}_1\|_F = \sqrt{5} > 0$, so SOMP continues.

Second iteration. Now row 1 scores $\sqrt{5}$ and row 3 scores 0 (the residual is orthogonal to the chosen atom), so atom 1 wins, $S_2 = \{1, 3\}$, and the joint least-squares solve returns $(2, -1)$ on row 1 and $(3, 4)$ on row 3, driving the residual to zero. SOMP has recovered the exact shared support $\{1, 3\}$ and the exact coefficients in two iterations. ◊

The trace shows the two defining moves: the selection step fused the two columns of correlations into one score per atom and picked the collectively strongest, and the refit solved both measurement columns at once through a single pseudoinverse, producing a row-sparse estimate by construction.

15.4.1 When SOMP succeeds

The guarantee for SOMP is, strikingly, the *same* exact-recovery condition that governs single-vector OMP, the Tropp ERC.

Theorem 15.8 (SOMP exact recovery condition). Let $\boldsymbol{X}$ be jointly S-row-sparse with $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ and true row support S. If

$$\mathrm{ERC}(\boldsymbol{A}, S) \equiv \max_{j \notin S} \|\boldsymbol{A}_S^{\dagger} \boldsymbol{a}_j\|_1 < 1,$$

then SOMP recovers $\boldsymbol{X}$ exactly, regardless of the inner index q used in the selection rule.

The vector $\boldsymbol{A}_S^{\dagger}\boldsymbol{a}_j$ measures how well an off-support atom $\boldsymbol{a}_j$ can be synthesized from the true atoms. If its ℓ_1 norm stays below 1 for every outside atom, no intruder can masquerade as a better match to the residual than a genuine support atom, and the greedy rule stays honest at every step. The condition is a statement about the geometry of $\boldsymbol{A}$ on the support, not about how the columns vote, which is why it is *independent of* q: fusing the K correlations by their ℓ_1, ℓ_2, or ℓ_∞ norm gives the same exact-recovery guarantee. The choice of q affects behavior on noisy or near-degenerate data, but the clean guarantee does not depend on it. This is the identical inequality, on the identical quantity, that governs OMP in Chapter 8; the only change is that the residual is a matrix and the score a fused ℓ_q norm.

15.5 Applications

Joint sparsity arises wherever many signals share a structure. In **hyperspectral imaging**, a scene is captured at hundreds of wavelengths; each pixel's spectrum is sparse in a dictionary of material signatures, and a given material's signature is the same set of atoms across all wavelength bands, so the spectral images are jointly sparse, with K equal to the number of bands. In **multi-electrode neural recording**, many electrodes record overlapping populations of neurons; the spikes of a given neuron appear on the same few electrodes, a shared support across the recording channels, so SOMP across channels localizes the active neurons. In **distributed sensing** and **direction-of-arrival estimation**, several sensors or snapshots observe one event through correlated sparse signals, and the shared sparsity is the few sources present. In each case SOMP or the group lasso recovers all the signals together from fewer measurements per signal than independent recovery would need, because the shared support is identified once, using all the evidence at once, exactly as Theorem 15.5 predicts.

15.6 Hierarchical and Collaborative Group Sparsity

The mixed norm has a richer life than the row-sparse MMV model. Two refinements, both used in the course, structure sparsity at more than one level.

The group lasso. Sometimes the coefficients of a *single* signal come in predefined **groups**, $\{1, \dots, N\} = G_1 \cup \cdots \cup G_K$, and we want a few whole groups active rather than a few scattered entries (the wavelet coefficients of one image band, the genes of one pathway, the atoms of one sub-dictionary). The **group lasso** penalty is the sum of group ℓ_2 norms,

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \sum_{g=1}^{K} \|\boldsymbol{x}_{G_g}\|_2,$$

the single-vector analogue of the row-$\ell_{1,2}$ norm of Definition 15.2: the inner ℓ_2 keeps each group whole (all of it on or all off), the outer sum drives most groups to zero. Its prox is the block soft threshold of Example 15.4 applied group by group.

Hierarchical sparsity (HiLasso). A group can be active yet still use only a few of its own atoms. **Hierarchical Lasso** captures both levels at once by adding an ordinary ℓ_1 term to the group lasso,

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \lambda_2 \sum_g \|\boldsymbol{x}_{G_g}\|_2 + \lambda_1 \|\boldsymbol{x}\|_1,$$

so the group term selects a few active groups and the ℓ_1 term thins each active group to a few atoms (Fig. 15.2). The result is sparse *across* groups and *within* them, the right model when the dictionary is organized into classes.

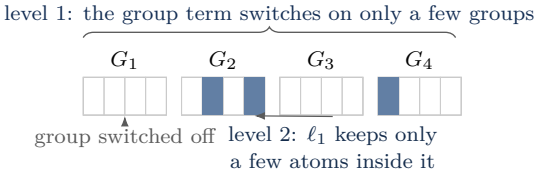


Figure 15.2. Hierarchical sparsity at two levels. Level 1 (the group term) lights only a few of the candidate groups: here two of four (group 2 and group 4). Level 2 (the ℓ_1 term) thins each active group to a few atoms. The result is sparse *across* groups and *within* them.

Collaborative hierarchical sparsity (C-HiLasso). Now combine hierarchy with the joint structure of this chapter. Given many signals as columns of $\boldsymbol{X}$ that share the same active *groups* but use different atoms within them, **collaborative HiLasso** couples the group selection across all signals (a Frobenius norm on each group's block, $\sum_g \|\boldsymbol{X}_{G_g,:}\|_F$, forcing the columns to agree on *which* groups) while letting an entrywise ℓ_1 pick different atoms per column. It is the model behind collaborative source separation: a set of mixed recordings all draw from the same few source dictionaries (the shared groups), but each instant activates different atoms within them. The three levels, collaborative group selection, then per-signal atom selection, are three mixed norms stacked, all solved by the same proximal machinery.

Notes and References

The multiple-measurement-vector model, mixed norms, and simultaneous orthogonal matching pursuit are developed by Tropp, Gilbert, and Strauss [50]; the improvement of the uniqueness threshold by the rank of the measurement matrix (Theorem 15.5) and the group-lasso relaxation are treated there and in Foucart and Rauhut [28]. The exact recovery condition (Theorem 15.8) is the matrix lift of Tropp's single-vector ERC. The mixed $\ell_{1,2}$ norm is the group lasso of the statistics literature. Hyperspectral and neural-recording applications of joint sparsity are surveyed in the sparse-modeling treatments of Elad [25] and Wright and Ma [53].

Exercises

Exercise 15.1. Compute $\|\boldsymbol{X}\|_{\mathrm{row},0}$, $\|\boldsymbol{X}\|_{1,2}$, and the Frobenius norm for $\boldsymbol{X} = \left(\begin{smallmatrix} 1 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 3 & 4 \end{smallmatrix}\right)$, and explain why the Frobenius norm fails to promote row sparsity.

Exercise 15.2. Show that $\|\boldsymbol{X}\|_{0,q}$ does not depend on the inner index $q \geq 1$, but that the convex surrogate $\|\boldsymbol{X}\|_{p,q}$ does. Why does the inner norm only start to matter after relaxation?

Exercise 15.3. Specialize Theorem 15.5 to $K = 1$ and confirm it reproduces the single-vector spark bound $S < \operatorname{spark}(\boldsymbol{A})/2$. Then give a numerical $\boldsymbol{A}$ and a rank-2 $\boldsymbol{Y}$ for which the matrix bound permits a strictly larger S than the vector bound.

Exercise 15.4. For $K = 1$ column, show that SOMP reduces exactly to single-vector OMP, the $\ell_{1,2}$ norm reduces to the ℓ_1 norm, and Theorem 15.8 reduces to the single-vector ERC.

Exercise 15.5. Continue Example 15.7 through its second iteration in full: form $\boldsymbol{A}^\top \boldsymbol{R}_1$, compute the row ℓ_2 scores, confirm atom 1 wins, and verify the final least-squares refit on $\{\boldsymbol{a}_1, \boldsymbol{a}_3\}$ drives the residual to zero in both columns.

Exercise 15.6. Explain, in terms of the collision argument, why the shared support becomes easier to identify as K grows, and relate this directly to the $+\operatorname{rank}(\boldsymbol{Y})$ term in Theorem 15.5.

Exercise 15.7. Reconcile Theorem 15.5 and Theorem 15.6: one improves with $\operatorname{rank}(\boldsymbol{Y})$ and the other does not. Explain why a coherence-only analysis is blind to the rank of the data, and what kind of analysis recovers the benefit.

CHAPTER 16

Sparse Representation-Based Classification

A face is best explained by other faces of the same person, and by no one else's.

The book closes with the application that turns the whole theory into a working recognition system. Sparse representation-based classification (SRC) treats a test image as a sparse combination of labeled training images and reads its identity off the sparse code: an image of a given person is reconstructed almost entirely from training images of that same person, so the coefficients concentrate on the correct class. SRC recovers the recovery machinery of Part II in service of a classifier, inherits its robustness to corruption directly from the ℓ_1 formulation, and ties the course's mathematics to the concrete task of recognizing faces under varying light and occlusion.

16.1 From Recovery to Recognition

Collect all labeled training images as the columns of a dictionary $\boldsymbol{A} = [\boldsymbol{A}_1 \mid \boldsymbol{A}_2 \mid \cdots \mid \boldsymbol{A}_C]$, grouped into C class blocks, one per subject, with unit-norm columns. A test image $\boldsymbol{y}$ of subject $i^\star$ is, to good approximation, a linear combination of the training images of subject $i^\star$ alone, so the coefficient vector $\boldsymbol{x}$ in $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is *sparse* and *supported on the block of class* $i^\star$. The classification problem is thus a sparse recovery problem: solve for $\boldsymbol{x}$, see which block its mass lands in, and that is the identity.

Algorithm 13 Sparse Representation-Based Classification (Wright et al.)

Require: dictionary $\boldsymbol{A}$ (unit-norm columns), class labels, test image $\boldsymbol{y}$

1: **sparse code:** $\hat{\boldsymbol{x}} \leftarrow \arg\min_{\boldsymbol{z}} \|\boldsymbol{z}\|_1$ s.t. $\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{y}\|_2 \le \varepsilon$
2: **for** each class $i = 1, \dots, C$ **do**
3: form $\delta_i(\hat{\boldsymbol{x}})$, the coefficients of $\hat{\boldsymbol{x}}$ on class i with the rest zeroed
4: $r_i \leftarrow \|\boldsymbol{y} - \boldsymbol{A}\,\delta_i(\hat{\boldsymbol{x}})\|_2$ ▷ how well class i alone explains $\boldsymbol{y}$
5: **end for**
6: **return** $\hat{i} = \arg\min_i r_i$ ▷ the class that best reconstructs $\boldsymbol{y}$

The algorithm sparse-codes the test image over the entire training set at once, then asks, class by class, how well that class's coefficients *alone* reconstruct $\boldsymbol{y}$. The class with the smallest residual wins. Crucially, the ℓ_1 step does not know the labels; it simply finds the sparsest explanation, and the labels are read off afterward.

Example 16.1 (SRC by hand: Alice, Bob, Charlie). Three subjects, two training images each, ambient dimension 4, so everything fits on a page. We use OMP in place of ℓ_1 minimization because OMP is exactly tractable by hand. The unit-norm dictionary, in class blocks, is

$$\boldsymbol{A} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 \end{bmatrix} = \big[\, \underbrace{\boldsymbol{a}_{A,1}\ \boldsymbol{a}_{A,2}}_{\text{Alice}} \mid \underbrace{\boldsymbol{a}_{B,1}\ \boldsymbol{a}_{B,2}}_{\text{Bob}} \mid \underbrace{\boldsymbol{a}_{C,1}\ \boldsymbol{a}_{C,2}}_{\text{Charlie}} \,\big],$$

each column of norm 1. A new photo of Alice under fresh illumination is $\boldsymbol{y} = \frac{1}{\sqrt{2}}(1.0,\, 0.3,\, 0.1,\, 0.0)^\top$.

OMP step 1. The six correlations $\boldsymbol{A}^\top \boldsymbol{y}$ are

$$\langle \boldsymbol{a}_{A,1},\, \boldsymbol{y}\rangle = 0.65, \quad \langle \boldsymbol{a}_{A,2},\, \boldsymbol{y}\rangle = 0.35, \quad \langle \boldsymbol{a}_{B,1},\, \boldsymbol{y}\rangle = 0.05, \quad \langle \boldsymbol{a}_{B,2},\, \boldsymbol{y}\rangle = 0.05, \quad \langle \boldsymbol{a}_{C,1},\, \boldsymbol{y}\rangle = 0.55, \quad \langle \boldsymbol{a}_{C,2},\, \boldsymbol{y}\rangle = -0.45,$$

for example $\langle \boldsymbol{a}_{A,1},\, \boldsymbol{y}\rangle = \frac{1}{2}(1{\cdot}1.0 + 1{\cdot}0.3) = 0.65$. The winner is $\boldsymbol{a}_{A,1}$ at 0.65, so the support is $\{A, 1\}$ and the fit coefficient is 0.65. The residual $\boldsymbol{r}_1 = \boldsymbol{y} - 0.65\,\boldsymbol{a}_{A,1} = \frac{1}{\sqrt{2}}(0.35, -0.35, 0.10, 0)^\top$.

OMP step 2. Re-correlating, $\langle \boldsymbol{a}_{A,2},\, \boldsymbol{r}_1\rangle = 0.35$ now leads (the chosen atom scores 0 by orthogonality), so $\boldsymbol{a}_{A,2}$ joins the support. Since $\boldsymbol{a}_{A,1} \perp \boldsymbol{a}_{A,2}$, the refit is just the two projections, $\hat{\boldsymbol{x}} = (0.65,\, 0.35,\, 0, 0, 0, 0)^\top$, both nonzeros in Alice's block.

Class residuals. Reconstructing with each class block alone,

$$r_A = \|\boldsymbol{y} - 0.65\boldsymbol{a}_{A,1} - 0.35\boldsymbol{a}_{A,2}\|_2 = \tfrac{0.1}{\sqrt{2}} \approx 0.071, \qquad r_B = r_C = \|\boldsymbol{y}\|_2 \approx 0.742,$$

because $0.65\boldsymbol{a}_{A,1} + 0.35\boldsymbol{a}_{A,2} = \frac{1}{\sqrt{2}}(1.0, 0.3, 0, 0)^\top$ leaves only the tiny third-coordinate residual, while Bob's and Charlie's blocks contribute nothing. The decision $\hat{i} = \arg\min_i r_i = A$ classifies $\boldsymbol{y}$ as Alice, with a clean margin $r_B / r_A \approx 10.5$. ◊

16.2 Why It Works

The geometry is a union of subspaces (Fig. 16.1). Each class i spans a subspace $V_i = \text{range}(\boldsymbol{A}_i)$, the span of its training images, and the images of subject i under varying illumination and expression lie close to V_i. A test image of subject $i^\star$ lies near $V_{i^\star}$ and far from every other V_j, so projecting it onto its own class leaves a small residual while projecting onto a wrong class leaves a large one.

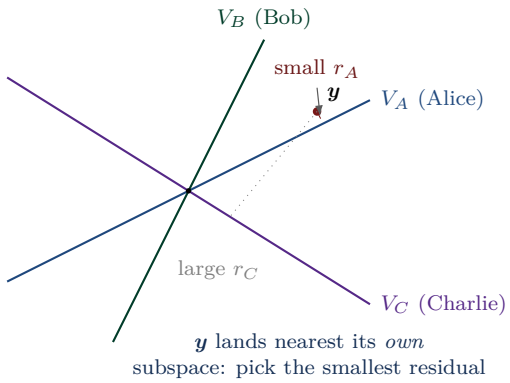


Figure 16.1. The union-of-subspaces picture behind SRC. Each class is a subspace through the origin; a test image $\boldsymbol{y}$ of Alice sits close to V_A (small residual r_A) and far from V_B and V_C (large residuals). SRC finds the nearest subspace, but with the ℓ_1 code selecting which few training images within that subspace to use.

Proposition 16.2 (Residual concentration). Suppose $\boldsymbol{y} = \boldsymbol{A}_{i^\star}\boldsymbol{x}_{i^\star} + \boldsymbol{n}$ with $\|\boldsymbol{n}\|_2 \le \nu$, the class subspaces are separated so that $\min_{\boldsymbol{w}\in V_i}\|\boldsymbol{y}-\boldsymbol{w}\|_2 \ge \nu+\gamma$ for every $i \ne i^\star$, and the ℓ_1 solution $\hat{\boldsymbol{x}}$ fits $\boldsymbol{y}$ to within ν with support on class $i^\star$. Then $r_{i^\star} \le \nu$ while $r_i \ge \nu+\gamma$ for $i \ne i^\star$, so SRC classifies correctly.

Proof. Since $\hat{\boldsymbol{x}}$ is supported on class $i^\star$, $\boldsymbol{A}\delta_{i^\star}(\hat{\boldsymbol{x}}) = \boldsymbol{A}\hat{\boldsymbol{x}}$, so $r_{i^\star} = \|\boldsymbol{A}\hat{\boldsymbol{x}}-\boldsymbol{y}\|_2 \le \nu$. For $i \ne i^\star$, $\boldsymbol{A}\delta_i(\hat{\boldsymbol{x}}) \in V_i$, so $r_i \ge \min_{\boldsymbol{w}\in V_i}\|\boldsymbol{y}-\boldsymbol{w}\|_2 \ge \nu+\gamma$. ∎

One could instead just project onto each class subspace and pick the nearest, the classical nearest-subspace classifier. The ℓ_1 step buys two things over that: within each class it selects only the few most relevant training images rather than using all of them, and, as the next section shows, it makes the classifier robust to gross corruption of the test image. The classifier's confidence can be quantified by how concentrated the coefficients are on a single class.

Definition 16.3 (Sparsity concentration index). For a code $\hat{\boldsymbol{x}}$ over C classes, write $p_i = \|\delta_i(\hat{\boldsymbol{x}})\|_1/\|\hat{\boldsymbol{x}}\|_1$ for the fraction of ℓ_1 mass in class i (so $\sum_i p_i = 1$). The **sparsity concentration index** is

$$\text{SCI}(\hat{\boldsymbol{x}}) = \frac{C\,\max_i p_i - 1}{C-1} \in [0,1].$$

It is 1 when all the coefficient mass sits in one class (a confident, valid query) and 0 when the mass spreads evenly across classes (an ambiguous one). Rejecting queries with SCI below a threshold τ lets SRC decline to classify an outlier rather than guess; a threshold in the low hundredths to a tenth is typical (Chapter 26 uses $\tau \approx 0.06$ for a 5% false-rejection rate).

Example 16.4 (SCI of a confident and an ambiguous query). For the Alice code $\hat{\boldsymbol{x}} = (0.65, 0.35, 0, 0, 0, 0)^\top$ of Example 16.1, all the mass is in Alice's block, so $p_{\max} = 1$ and $\text{SCI} = \frac{3\cdot 1-1}{3-1} = 1$: maximally confident. Now take a spread-out code $\hat{\boldsymbol{x}} = (0.30, 0.10, 0.20, 0.05, 0.25, 0.10)^\top$ with $\|\hat{\boldsymbol{x}}\|_1 = 1$; the class masses are $p_A = 0.40$, $p_B = 0.25$, $p_C = 0.35$, so $p_{\max} = 0.40$ and $\text{SCI} = \frac{3\cdot 0.40-1}{2} = 0.10$, at the edge of the typical reject threshold. The index cleanly separates the two regimes. ◊

16.3 Robust SRC: Occlusion and Corruption

The feature that made SRC famous is its handling of occlusion. A face wearing sunglasses or a scarf, or a portrait with a block of pixels destroyed, is the clean image plus a *sparse* error: only the occluded pixels are wrong, but they are wrong by a lot. This is exactly the sparse gross corruption of Chapter 12, now on a single image, and the fix is the same idea, applied through the dictionary.

Model the corrupted image as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$ with $\boldsymbol{e}$ sparse, and absorb the error into an augmented dictionary by appending the identity,

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{e} \end{pmatrix},$$

then solve a single ℓ_1 problem for the stacked unknown $(\boldsymbol{x}; \boldsymbol{e})$. The identity block gives each pixel its own atom, so the solver can charge the occluded pixels to $\boldsymbol{e}$, where they cost little because they are few, and keep $\boldsymbol{x}$ concentrated on the true class. The classification then proceeds on $\boldsymbol{x}$ as before, with the corruption explained away in $\boldsymbol{e}$. Because the occlusion is sparse in the pixel basis and the face is sparse in the training-image basis, and the two bases are incoherent, the ℓ_1 program separates them, exactly the principle of Chapter 10. SRC can recover identity from faces that are half covered.

Example 16.5 (Corruption charged to the error term). Return to Alice from Example 16.1, with clean image $\boldsymbol{y} = \frac{1}{\sqrt{2}}(1.0, 0.3, 0.1, 0)^\top$, and corrupt the third pixel by adding 5: $\tilde{\boldsymbol{y}} = \frac{1}{\sqrt{2}}(1.0, 0.3, 5.1, 0)^\top$. Plain SRC now sees a huge third coordinate that matches no Alice training image, and the residual test is thrown off. The augmented program $\min \|[\boldsymbol{x}; \boldsymbol{e}]\|_1$ subject to $[\boldsymbol{A}\ \boldsymbol{I}][\boldsymbol{x}; \boldsymbol{e}] = \tilde{\boldsymbol{y}}$ has a cheap explanation available: set $\boldsymbol{e} = \frac{5.1}{\sqrt{2}}\boldsymbol{e}_3$, one nonzero costing $5.1/\sqrt{2} \approx 3.61$ in ℓ_1, which absorbs the whole third coordinate and leaves the clean Alice code $\boldsymbol{x} = (0.65, 0.35, 0, 0, 0, 0)^\top$ on the training block. Any explanation that instead forced the face block to fit the corrupted pixel would spend far more ℓ_1 mass across many atoms, so the minimizer charges the damage to $\boldsymbol{e}$. Classification then runs on $\boldsymbol{x}$ alone and still returns Alice. This is why robust SRC held 54% accuracy at 30% random-pixel corruption in Chapter 26, where plain SRC had collapsed to 28%. ◊

Method	Idea	Yale B	Character
Eigenfaces (PCA)	project on top principal directions, nearest neighbor	∼75%	fast, no class info
Fisherfaces (LDA)	class-aware projection, maximize between- vs. within-class scatter	∼85%	discriminative, needs labels
SRC	ℓ_1 over the training dictionary, class-wise residual	∼95%	interpretable, robust, slow
CNN (FaceNet)	deep embeddings, triplet/contrastive loss	∼99%	needs huge data, opaque

Table 16.1. SRC against the main face recognizers on the Extended Yale B benchmark. Accuracies are representative figures from the literature. SRC trades raw accuracy for interpretability and robustness, and needs no training.

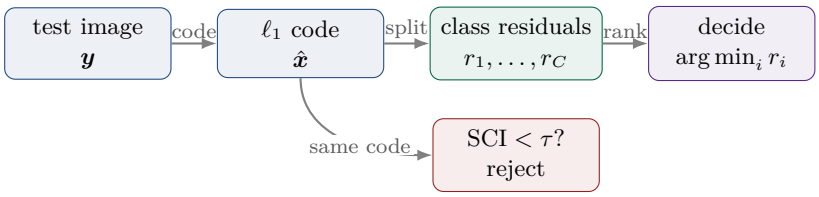


Figure 16.2. The SRC pipeline: sparse-code the test image over the training dictionary, measure how well each class block alone reconstructs it, and pick the smallest residual. The concentration index of the same code gives a reject option for out-of-database queries.

16.4 In Context

SRC sits among a family of face recognizers and is distinguished by its use of sparsity. The classical *eigenfaces* and *Fisherfaces* methods project every image onto a single fixed low-dimensional subspace (the principal or discriminant directions) and classify in that space; they are fast but fragile to occlusion, since a corrupted pixel contaminates the projection. SRC instead represents the test image over all training images and lets the ℓ_1 program choose, per query, which few to use, which is what gives it robustness. Modern convolutional networks outperform SRC on raw accuracy given enough training data, but SRC needs no training beyond storing the labeled images, comes with the interpretable residual and concentration index, and degrades gracefully under corruption that the geometry explains. Table 16.1 places the methods side by side.

The historical arc explains the trade. Eigenfaces (Turk–Pentland, 1991) first showed linear algebra could recognize faces at all; Fisherfaces (Belhumeur–Hespanha–Kriegman, 1997) added class-aware projection; the Yale B benchmark (2001) standardized the illumination-invariance test; SRC (Wright et al., 2009) brought ℓ_1 minimization and held the state of the art for about five years; FaceNet (Schroff et al., 2015) and the deep models after it pushed past 99% given millions of images. SRC remains the right tool in the low-data, high-stakes regime: a handful of images per class, a need to know *which* training images explain a query, formal robustness to sparse corruption, and a principled reject option through the concentration index.

> SRC is the whole book in one application. It uses the union-of-subspaces geometry of Chapter 3, the ℓ_1 recovery of Chapter 5 solved by the greedy or proximal methods of Chapters 6 and 7, the incoherence principle of Chapter 10 to separate face from occlusion, and the sparse-plus-structured decomposition of Chapter 12 to handle corruption. A working face recognizer falls out of the same mathematics that began with $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$.

In the Problem Sets

Problem Set 7 (Chapter 26) builds SRC end to end on the Extended Yale B face database. You assemble the class-block dictionary from training faces, sparse-code test images by ℓ_1 minimization, classify by the smallest class-wise residual, add the identity block to recognize faces under synthetic occlusion, and use the sparsity concentration index to reject non-face outliers. The accompanying notebook reports recognition accuracy as the occlusion level and the measurement dimension vary.

16.5 The Classifiers SRC Competes With

SRC is one entry in a long line of classifiers, and it is worth seeing the others, both because they are the baselines the course compared SRC against and because SRC turns out to generalize one of them. A **classifier** takes a labeled training set $\{(\boldsymbol{a}_i, \ell_i)\}$ and assigns a label to a new point $\boldsymbol{y}$. The classical methods differ in what they store and what geometry they trust.

Nearest neighbor and K-NN. The simplest classifier stores every training point and labels $\boldsymbol{y}$ by its single nearest neighbor, $\hat{\ell} = \ell_{i^\star}$, $i^\star = \arg\min_i \|\boldsymbol{y} - \boldsymbol{a}_i\|_2$. The **$K$-nearest-neighbor** rule is steadier: find the K closest training points and take a majority vote of their labels. The one parameter K is set by *cross-validation*, holding out part of the training data and choosing the K with the best held-out accuracy. K-NN needs no training beyond storing the data, but it keeps the whole training set, is slow at test time, and trusts raw Euclidean distance, which a few corrupted pixels can wreck.

Nearest subspace. Instead of comparing to individual points, the **nearest-subspace** classifier compares to each class's *span*. Each class i defines a subspace $V_i = \mathcal{R}(\boldsymbol{A}_i)$, and the rule projects $\boldsymbol{y}$ onto each and picks the closest,

$$\hat{\ell} = \arg\min_i \ \|\boldsymbol{y} - \boldsymbol{P}_{V_i}\boldsymbol{y}\|_2, \qquad \boldsymbol{P}_{V_i} = \boldsymbol{A}_i(\boldsymbol{A}_i^*\boldsymbol{A}_i)^{-1}\boldsymbol{A}_i^*,$$

the residual of the orthogonal projection of Appendix B. This is exactly the geometry behind SRC (Section 16.2): a face of subject i lies near V_i. The difference is that nearest subspace uses *all* of class i's training images at once, while SRC's ℓ_1 step selects only the few most relevant ones per query and, with the identity block, shrugs off gross corruption. SRC is the sparse, robust refinement of nearest subspace.

Support vector machines. The **support vector machine** (SVM) takes a different view: find the separating boundary, not the class prototypes. For two classes, it fits the hyperplane $\boldsymbol{w}^\top\boldsymbol{x} + b = 0$ that separates them with the *widest margin*,

$$\min_{\boldsymbol{w},b} \ \tfrac{1}{2}\|\boldsymbol{w}\|_2^2 \quad \text{subject to} \quad \ell_i(\boldsymbol{w}^\top\boldsymbol{a}_i + b) \geq 1 \ \ \forall i,$$

where $\ell_i \in \{\pm 1\}$. The margin width is $2/\|\boldsymbol{w}\|_2$, so minimizing $\|\boldsymbol{w}\|_2$ maximizes it (Fig. 16.3). Only the training points lying exactly on the margin, the **support vectors**, determine the boundary; the rest could be deleted without changing it. For data that no hyperplane separates, the **kernel trick** maps the points into a higher-dimensional feature space through $\boldsymbol{x} \mapsto \boldsymbol{\phi}(\boldsymbol{x})$ and fits a linear boundary there, which is curved back in the original space. The map is never computed: the SVM uses only inner products, and a **kernel** $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \rangle$ supplies them directly, the Gaussian (radial basis) kernel $k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\|\boldsymbol{x}-\boldsymbol{x}'\|_2^2/2\sigma^2}$ being the common choice.

Where SRC fits

The classifiers split by what they trust: K-NN trusts *distance to points*, the SVM trusts a *separating boundary*, and nearest subspace and SRC trust *distance to class subspaces*. SRC sharpens the subspace view with a per-query sparse selection and an explicit sparse-error model, which is what buys its robustness to occlusion. The trade is cost: SRC solves an ℓ_1 program per test image, where K-NN and a trained SVM classify in a single pass. SRC wins when training data is scarce and corruption is real; the others win on raw speed at scale.

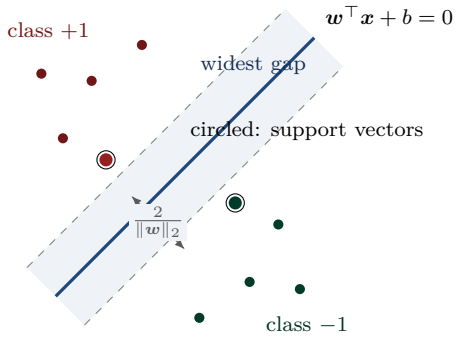


Figure 16.3. A linear SVM. Among all separating hyperplanes it picks the one with the widest margin: the shaded corridor of width $2/\|\boldsymbol{w}\|_2$ that holds no training point. Only the support vectors (circled), the points on the margin, determine the boundary. The kernel trick fits the same maximum-margin rule in a feature space, giving curved boundaries in the original space.

Notes and References

Sparse representation-based classification, robust SRC via the augmented dictionary, and the sparsity concentration index are all from Wright, Yang, Ganesh, Sastry, and Ma [54], the paper that established the method. The Extended Yale B face database is from Georghiades, Belhumeur, and Kriegman [29]; eigenfaces are due to Turk and Pentland and Fisherfaces to Belhumeur, Hespanha, and Kriegman, with the deep-embedding successor FaceNet due to Schroff, Kalenichenko, and Philbin. The union-of-subspaces view and the connection to the rest of compressed sensing are developed in Wright and Ma [53] and in Elad [25]. The robustness of the ℓ_1 formulation to sparse corruption is the same phenomenon analyzed in the robust PCA of Chapter 12.

Exercises

Exercise 16.1. Rework Example 16.1 with the test image $\boldsymbol{y} = \frac{1}{\sqrt{2}}(0.5, 0, 0.5, 0)^\top$. Compute the six correlations, run OMP, find the class residuals, and explain why this query is ambiguous. Which atom does OMP select first, and why is the margin small?

Exercise 16.2. Compute $\mathrm{SCI}(\hat{\boldsymbol{x}})$ for a code whose mass is entirely in one class, and for one spread evenly across C classes, confirming the values 1 and 0. Then verify the two numbers in Example 16.4.

Exercise 16.3. Explain why appending the identity, $[\boldsymbol{A}\ \boldsymbol{I}]$, lets SRC handle occlusion, and why it is essential that the occlusion be sparse in the pixel basis while the face is sparse in the training-image basis. Relate this to incoherence.

Exercise 16.4. Contrast SRC with eigenfaces: why does projecting onto a single fixed subspace make eigenfaces fragile to a few corrupted pixels, while SRC's per-query sparse selection is robust? Use Table 16.1 to frame the accuracy/robustness trade.

Exercise 16.5. Show that when the class subspaces are well separated and the test image lies exactly in one of them, the residual decision rule and the maximum-coefficient-mass rule $\hat{\imath} = \arg\max_i \|\delta_i(\hat{\boldsymbol{x}})\|_1$ agree.

CHAPTER 17

Compressed Sensing in Practice

A clean theory, followed quickly by a striking machine, is the mark of an idea that was ready to happen.

Chapter 1 opened with the promise that a sparse signal can be acquired far below the Nyquist rate. This chapter cashes that promise in hardware. We work through the two instruments that made compressed sensing tangible, the single-pixel camera and the compressed-sensing MRI scanner, in enough engineering detail to see exactly how the abstract $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ becomes an optical path or a pulse sequence, and we compute, for real image sizes, how many measurements each actually needs. A short survey of further applications closes the chapter. Throughout, the mathematics is the mathematics of the earlier parts; what is new is the map from physics to the sensing matrix.

17.1 The Single-Pixel Camera

A conventional digital camera puts one photodiode behind every pixel: a megapixel image needs a million detectors. The single-pixel camera, built at Rice University by Duarte and collaborators [23], replaces the entire array with *one* photodiode and a programmable mask, and reconstructs the image by the ℓ_1 minimization of Chapter 5.

17.1.1 The optical measurement

Light from the scene is focused onto a **digital micromirror device** (DMD), a chip carrying an array of tiny mirrors, one per image pixel, each of which tilts independently to send its share of the light either toward the photodiode or away from it. A DMD pattern is therefore a binary spatial mask $\boldsymbol{\phi}_k \in \{0,1\}^N$ (or, with a differencing trick, a ± 1 mask). With pattern $\boldsymbol{\phi}_k$ loaded, the photodiode integrates all the light that reaches it, producing a single scalar that is exactly the inner product of the mask with the vectorized scene $\boldsymbol{x} \in \mathbb{R}^N$,

$$y_k = \langle \boldsymbol{\phi}_k, \boldsymbol{x} \rangle = \boldsymbol{\phi}_k^\top \boldsymbol{x}.$$

Load m different patterns in sequence, record m scalars, and the camera has measured $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}$ with $\boldsymbol{\Phi}$ the $m \times N$ matrix of masks (Fig. 17.1). One detector, m exposures, a full image.

17.1.2 Why it works

The image is not sparse in pixels, but natural images are sparse in a wavelet or DCT basis $\boldsymbol{\Psi}$, so $\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{s}$ with $\boldsymbol{s}$ sparse. The DMD masks are realizations of a bounded orthonormal system (Walsh–Hadamard, $K = 1$), so the effective sensing matrix $\boldsymbol{\Phi}\boldsymbol{\Psi}$ satisfies the restricted isometry property with high probability (Chapter 10), and ℓ_1 minimization recovers $\boldsymbol{s}$, and hence the image, from $m \sim s \log(N/s)$ masks. The Walsh–Hadamard choice is not incidental: its ± 1 entries are exactly what a tilting mirror can realize, and its fast transform makes the recovery cheap.

Example 17.1 (How many masks for a 64×64 image?). The original Rice experiments imaged $N = 64\times64 = 4096$ pixels. For a scene whose DCT representation is about $s = 100$-sparse, the structured-matrix bound $m \gtrsim s \log(N/s)$ asks for on the order of a few hundred masks for an exactly sparse image. The published camera used $m = 1600$ masks,

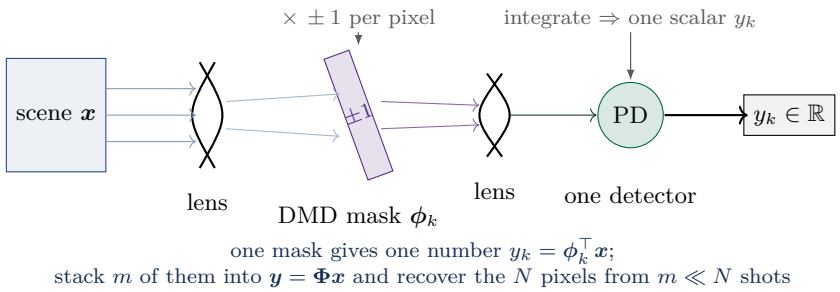


Figure 17.1. The Rice single-pixel camera. The scene is imaged onto a digital micromirror device whose mirrors implement a ± 1 mask $\boldsymbol{\phi}_k$; a lens collects the masked light onto one photodiode, which integrates it to a single number $y_k = \boldsymbol{\phi}_k^\top \boldsymbol{x}$. Repeating with m masks yields $\boldsymbol{y} = \boldsymbol{\Phi x}$, and the N-pixel image is reconstructed by ℓ_1 minimization from $m \ll N$ measurements.

about 39% of the pixel count, to get a clearly recognizable reconstruction. The gap from the few-hundred prediction is the usual one: a real photograph is only *approximately* sparse, with a heavy coefficient tail, so more measurements are needed than the idealized count. At roughly 1 ms per mask (set by the photodiode integration time), 1600 masks take about 1.6 seconds, fine for a still scene and the practical floor on the camera's speed. ◊

17.1.3 Why one detector is worth the trouble

The point is not to replace a phone camera, where silicon megapixel sensors are cheap. It is to image at wavelengths where detector arrays are expensive or unavailable: the infrared, the terahertz, certain hyperspectral bands. A single detector for such a band can cost orders of magnitude more than a visible-light pixel, so replacing a million of them with one, at the price of m sequential masked exposures and a recovery solve, is a real engineering win. The single-pixel camera turned the abstract promise of compressed sensing into a device within two years of the founding papers, much as Schwarzschild wrote down a black hole within months of general relativity. A clean theory followed quickly by a striking machine is the mark of an idea whose time had come.

17.2 Compressed Sensing MRI

Magnetic resonance imaging is the application that motivated the field, and the one where shorter scans matter most. An MRI scanner does not measure pixels; it measures *Fourier coefficients* of the image. The physics is the point.

17.2.1 The scanner measures k-space

A magnetic field gradient makes the spins at position $\boldsymbol{r}$ precess at a position-dependent frequency, so the signal the receiver coil records at a given gradient setting is the value of the image's two-dimensional Fourier transform at one spatial frequency $\boldsymbol{k}$,

$$y(\boldsymbol{k}) = \int \rho(\boldsymbol{r})\, e^{-2\pi i\, \boldsymbol{k}\cdot\boldsymbol{r}}\, d\boldsymbol{r} = \hat{\rho}(\boldsymbol{k}),$$

where ρ is the proton density being imaged. Sweeping the gradients traces out points of this *k-space*, one frequency per readout. The total scan time is proportional to the number of k-space points acquired, because each point needs its own gradient-encoding step, and that time is what keeps a child sedated in the bore or a patient holding their breath. The sensing matrix is therefore a *partial discrete Fourier transform*, exactly the bounded orthonormal system with $K = 1$ of Chapter 10: $\boldsymbol{A} = \boldsymbol{RF}$, keep a subset Ω of the Fourier rows.

17.2.2 What makes it compressible

Two facts combine. First, medical images are sparse in a wavelet basis, and piecewise-smooth images like an axial brain slice are sparse in their gradient, so total-variation recovery (Chapter 7) applies. Second, the Fourier (sensing) and wavelet (sparsity) bases are incoherent, so a *random* subset of k-space carries near-independent information about the wavelet coefficients. The recipe of Lustig, Donoho, and Pauly [36] is then: sample k-space on a variable-density random pattern (dense near the center, where image energy concentrates, sparse in the periphery), and reconstruct by

$$\min_{\boldsymbol{x}} \ \tfrac{1}{2}\|\boldsymbol{R}\boldsymbol{F}\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \lambda_1\|\boldsymbol{\Psi}^*\boldsymbol{x}\|_1 + \lambda_2\,\mathrm{TV}(\boldsymbol{x}),$$

a wavelet-plus-total-variation objective solved by the proximal methods of Chapter 7.

Example 17.2 (Scan-time speedup on a 512×512 slice). Take a clinical brain slice, $N = 512\times512 = 262{,}144$ pixels, with about $s = 2000$ significant wavelet coefficients. The structured-matrix guarantee for a partial Fourier operator ($K = 1$) needs

$$m \ \gtrsim\ C\,s\,(\log s)^2\log N = C\cdot 2000\cdot(\log 2000)^2\cdot\log(2.6\times10^5) \approx C\cdot 1.4\times10^6.$$

With the empirical constant $C \approx 0.05$ for random Fourier sampling, this gives $m \approx 72{,}000$, about 27% of the 262,144 Nyquist samples, a 3.6× scan-time speedup. That figure matches what accelerated clinical protocols actually achieve, and the saving is largest exactly where it is most needed, in pediatric and cardiac imaging where the patient cannot hold still for a full scan. ◊

The same matrix, two machines

The single-pixel camera and the MRI scanner are the *same* compressed-sensing problem wearing different hardware. Both sense a wavelet-sparse image through a $K = 1$ bounded orthonormal system, Walsh–Hadamard masks in one, partial Fourier in the other, and both recover by an ℓ_1 or total-variation program. The theory of Chapter 10 was written once and deployed twice.

17.3 A Survey of Further Applications

The same template reaches well beyond imaging. Table 17.1 lists the recurring ones, each an instance of sensing a structured signal through an incoherent operator.

Application	Signal and sparsity	Sensing operator
Single-pixel camera	image, wavelet-sparse	Walsh–Hadamard DMD masks
Compressed-sensing MRI	medical image, wavelet/gradient-sparse	variable-density partial Fourier
Hyperspectral imaging	spatial-spectral cube, joint DCT-sparse	coded aperture (random mask + prism)
Sub-Nyquist sampling	multiband signal, sparse in frequency	random demodulator / analog-to-information
Radar and DOA	few targets, sparse in delay-Doppler	random or structured waveforms
Genomic group testing	few infected samples, sparse	pooled (combined) assays

Table 17.1. Compressed sensing across domains. In every row a structured signal is sensed through an operator incoherent with its sparsity basis, and recovered by a convex program.

Two are worth a sentence each. **Hyperspectral imaging** stacks a scene at hundreds of wavelengths into a data cube that is jointly sparse in a spatial-spectral transform; a coded aperture (a random mask followed by a dispersive prism)

senses it, and such cubes have been recovered from a small fraction of their voxels, in line with the few-percent the theory predicts. **Sub-Nyquist sampling** attacks the analog-to-digital converter directly: a signal that is sparse in frequency is mixed with a fast random sign sequence and lowpass filtered, the analog-to-information converter, so a wideband signal is digitized at a rate set by its information content rather than its bandwidth.

17.4 Three Tiers of Measurement Bounds

The worked counts above used the structured-matrix bound, but it is worth seeing all three tiers side by side, because the engineering choice of sensing matrix is a trade between the number of measurements and the cost of taking and processing them.

Sensing matrix	Measurements m	Cost to apply
Gaussian (dense)	$\sim s\log(N/s)$, the fewest	$O(mN)$, and $O(mN)$ storage
Structured random (BOS)	$\sim s(\log s)^2 \log N$, slightly more	$O(N \log N)$, no storage
Coherence (worst case)	$\sim s^2$, the most	checkable, but pessimistic

Table 17.2. The three measurement-count tiers. Gaussian matrices need the fewest measurements but are expensive to apply; structured random matrices need a few more but run with a fast transform; the worst-case coherence bound is checkable but quadratically loose.

The middle row is what real instruments use, and Example 17.2 shows why. A dense Gaussian operator on a 512×512 image would need fewer measurements but cost $O(mN) \approx 2 \times 10^9$ operations to apply and an enormous matrix to store, while the partial Fourier operator runs in $O(N \log N) \approx 5 \times 10^6$ with nothing to store. The Gaussian saves on the order of ten times the measurements but pays a few hundred times the compute, so structured sensing wins on total cost by a wide margin. This is the resolution of the apparent paradox of Chapter 10: the optimal-measurement matrix is not the optimal *system*, and the quadratic bottleneck of Chapter 8 is exactly why neither extreme is used in practice. Structured randomness, fast and near-optimal, is the working compromise.

Notes and References

The single-pixel camera is Duarte, Davenport, Takhar, Laska, Sun, Kelly, and Baraniuk [23]. Compressed-sensing MRI is Lustig, Donoho, and Pauly [36], building on the founding recovery theory of Candès, Romberg, and Tao [14] and Donoho [20]. The structured-random-matrix bounds behind the worked measurement counts are developed in Foucart and Rauhut [28]; the hyperspectral, sub-Nyquist, and group-testing applications are surveyed there and in the broader literature. The total-variation reconstruction model is the proximal machinery of Chapter 7.

Exercises

Exercise 17.1. A single-pixel camera images $N = 128 \times 128$ pixels of a scene that is $s = 300$-sparse in the DCT. Using $m \gtrsim s \log(N/s)$ with constant 1, estimate the number of masks, and the acquisition time at 1 ms per mask. Compare with acquiring all N pixels.

Exercise 17.2. Repeat the MRI count of Example 17.2 for a 256×256 slice with $s = 1000$ wavelet coefficients and $C = 0.05$. What fraction of Nyquist, and what scan-time speedup, do you predict?

Exercise 17.3. Explain, in terms of incoherence, why an MRI scanner (which measures Fourier coefficients) is naturally suited to recovering a wavelet-sparse image, and why sampling k-space on a regular grid would be a poor choice.

Exercise 17.4. Using Table 17.2, a 1000×1000 image, and $s = 5000$, estimate the measurement count and the per-measurement apply-cost for a dense Gaussian operator versus a partial-Fourier operator. Which has the lower total cost, and by roughly how much?

Exercise 17.5. The single-pixel camera's ± 1 masks come from a Walsh–Hadamard system. Explain why a ± 1-valued mask is what a tilting-mirror array can physically realize, and why the Walsh–Hadamard bound $K = 1$ makes it near-optimal as a sensing basis.

CHAPTER 18

Sparse Models and Deep Learning

Take a principled iterative algorithm, unroll it, and learn its parameters.

Compressed sensing did not stop in 2006. The convex-recovery viewpoint the book has built, find the low-dimensional structure, write a convex surrogate, solve it by an iterative algorithm, has propagated into much of modern machine learning, often without the new field noticing where it came from. This chapter names the bridges. Each is a direct descendant of a method already in the book: the iterative thresholding of Chapter 7, the dictionary learning of Chapter 13, and the manifold hypothesis of Chapter 11. The throughline is unbroken from $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ to the networks now used to interpret large language models.

18.1 Algorithm Unrolling: From ISTA to LISTA

The most direct bridge is *algorithm unrolling*. Recall the iterative soft-thresholding algorithm of Chapter 7: each iteration applies a linear map and then a soft threshold,

$$\boldsymbol{x}^{k+1} = \mathcal{S}_{\lambda/L}\Big(\boldsymbol{x}^k - \tfrac{1}{L}\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y})\Big) = \mathcal{S}_\theta\big(\boldsymbol{W}_e\boldsymbol{y} + \boldsymbol{S}\boldsymbol{x}^k\big),$$

where the second form collects the constants into an encoding matrix $\boldsymbol{W}_e = \frac{1}{L}\boldsymbol{A}^\top$ and a mutual-inhibition matrix $\boldsymbol{S} = \boldsymbol{I} - \frac{1}{L}\boldsymbol{A}^\top\boldsymbol{A}$. Each iteration is therefore exactly one layer of a feedforward neural network: a linear transform followed by a fixed pointwise nonlinearity, the soft threshold, which is nothing but a shrinking version of the ReLU activation.

Gregor and LeCun [30] made the consequence precise. Unroll a fixed number K of ISTA iterations into a K-layer network, then *untie* the matrices $\boldsymbol{W}_e$ and $\boldsymbol{S}$ from their formulas and treat them as *learnable weights*, trained on examples of $(\boldsymbol{y}, \boldsymbol{x})$ pairs to minimize reconstruction error. The result, **LISTA** (learned ISTA), recovers sparse signals in ten or twenty layers to an accuracy that plain ISTA needs hundreds of iterations to reach (Fig. 18.1). The network learns a problem-specific preconditioner that ISTA, bound to the generic formulas, cannot.

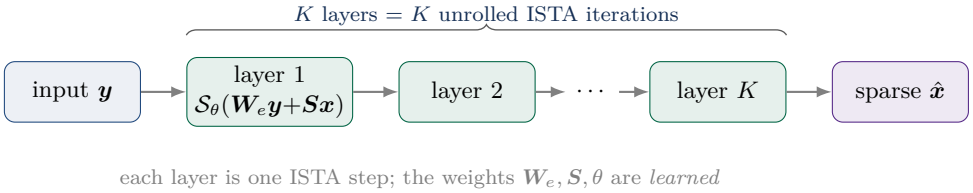


Figure 18.1. Unrolling ISTA into LISTA. A fixed number K of iterations becomes a K-layer network (the braced span), one layer per iteration, with the soft threshold as the activation. Untying the linear maps from their ISTA formulas and training them on data yields a sparse coder that runs in a handful of layers.

The recipe, take a principled iterative algorithm, unroll it to fixed depth, and learn its parameters, became a template for modern computational imaging, with FISTA, ADMM, and the proximal methods of Chapter 7 all unrolled into trainable networks for MRI, microscopy, and inverse problems generally. Unrolling keeps the interpretable structure of the optimization while gaining the speed and adaptivity of learning.

18.2 Learned Sensing and Learned Regularization

The book took the measurement matrix $\boldsymbol{A}$ and the sparsifying basis $\boldsymbol{\Psi}$ as given or random. Modern work learns them, in two ways that both keep the proximal skeleton of Chapter 7.

Learned regularization. The proximal-gradient iteration alternates a gradient step on the data term with the prox of the regularizer. The prox of the ℓ_1 norm is the soft threshold, but it is, abstractly, just a *denoiser*: it takes a noisy estimate and returns a cleaner one consistent with the prior. The *plug-and-play* idea replaces that soft threshold with any off-the-shelf denoiser, including a deep network trained to remove noise, inside the same iteration. The *deep image prior* goes further, using an untrained network's own structure as the regularizer. In both, the smooth-plus-structured splitting of ISTA is the backbone, and only the prox changes.

Learned sensing. Where the book chose $\boldsymbol{A}$ to be incoherent at random, one can instead train $\boldsymbol{A}$ end to end for a specific task, learning which measurements to take. The union of learned sensing and learned regularization is a fully data-driven version of the recovery pipeline, but its skeleton, measure, then alternate a data step with a prior step, is exactly the one assembled in Parts III and IV.

18.3 Sparse Coding as the Ancestor of Representation Learning

The dictionary learning of Chapter 13 is, in hindsight, the first representation-learning algorithm. Olshausen and Field [42] showed that learning a sparse code for natural image patches produces atoms that look like the oriented-edge receptive fields of the primary visual cortex, the same atoms K-SVD recovers. A dictionary with a sparse code is structurally an **autoencoder**: the dictionary $\boldsymbol{A}$ is the decoder, the sparse-coding step is the encoder, and K-SVD is an alternating way to train both (Fig. 18.2).

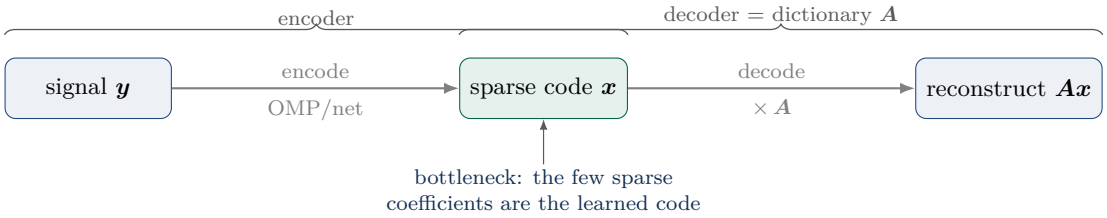


Figure 18.2. Dictionary learning is an autoencoder. The encoder maps a signal to a sparse code (by OMP in K-SVD, by a learned network in a modern sparse autoencoder); the sparse code is the bottleneck; the decoder is multiplication by the dictionary $\boldsymbol{A}$. K-SVD trains both factors by alternation.

The modern **sparse autoencoder** is this same object with a learned nonlinear encoder in place of the OMP step, and it has become a leading tool for *interpretability*: trained on the internal activations of a large language model, a sparse autoencoder discovers a dictionary of human-readable features, the parts-based decomposition of Chapter 14 applied to a network's hidden states. The line from k-means through K-SVD and nonnegative factorization to the interpretation of large language models is unbroken, and every link is in this book.

18.4 The Manifold View in Generative Models

The deepest connection is conceptual. The recurring premise of the book is that high-dimensional data has few true degrees of freedom: it lies near a low-dimensional set, a sparse model, a low-rank matrix, a union of subspaces. The **manifold hypothesis** of Chapter 11 states this generally: real data concentrates near a low-dimensional manifold in its ambient space. Modern generative models replace the hand-specified structure (sparse, low-rank) with a *learned* manifold, the range of a trained generative map, and recovery becomes inverting that map: find the low-dimensional latent code whose generated output matches the measurements.

The vocabulary changes, sparse coefficients become latent codes, the dictionary becomes a deep decoder, but the principle is the one this book is about. A signal with s degrees of freedom needs measurements proportional to s, not to its ambient dimension, whether those degrees of freedom are the support of a sparse vector or the coordinates on a learned manifold. Compressed sensing was an early, fully analyzable instance of a phenomenon now everywhere in machine learning, and its theorems remain the cleanest place to understand why measuring little and recovering by structure works at all.

Notes and References

Algorithm unrolling and LISTA are due to Gregor and LeCun [30]; the unrolling template now spans a large literature in computational imaging. The sparse-coding model of natural images is Olshausen and Field [42], the empirical root of the dictionary learning in Chapter 13. Plug-and-play priors and the deep image prior are the standard references for learned regularization inside proximal iterations; the sparse-autoencoder approach to interpretability builds on the same parts-based factorization as Chapter 14. The manifold hypothesis and its role across recovery and generative modeling are developed in Wright and Ma [53].

Exercises

Exercise 18.1. Write one ISTA iteration in the form $\boldsymbol{x}^{k+1} = \mathcal{S}_\theta(\boldsymbol{W}_e\boldsymbol{y} + \boldsymbol{S}\boldsymbol{x}^k)$, giving $\boldsymbol{W}_e$, $\boldsymbol{S}$, and θ explicitly in terms of $\boldsymbol{A}$, L, and λ. Explain which quantities LISTA makes learnable and why that can accelerate convergence.

Exercise 18.2. Explain why the soft-thresholding nonlinearity is a shifted, two-sided relative of the ReLU activation, and sketch both. What role does the threshold θ play that a plain ReLU lacks?

Exercise 18.3. Describe the encoder and decoder of the K-SVD autoencoder of Fig. 18.2, and state what a modern sparse autoencoder changes about the encoder. Relate the learned atoms to the parts-based factors of Chapter 14.

Exercise 18.4. In the plug-and-play framework, the prox of ℓ_1 is replaced by a general denoiser. Explain why any denoiser can stand in for a prox in the proximal-gradient iteration, and what is gained and lost relative to the provable ℓ_1 recovery of Chapter 5.

Exercise 18.5. State the manifold hypothesis and explain how recovering a sparse signal, completing a low-rank matrix, and inverting a generative model are three instances of the same principle, identifying the "degrees of freedom" in each.

CHAPTER 19

One Template, Many Problems

Low-dimensional structure makes high-dimensional recovery possible from few measurements.

The book has covered sparse vectors, low-rank matrices, dictionaries, nonnegative factors, joint sparsity, and a working face recognizer, a long list of algorithms with intimidating names. They are not a list. Almost everything here is a variation on a single template, solved by variations on a small set of ideas, and justified by one principle. This closing chapter lays that out. It introduces nothing new; it shows that the whole book was one idea all along.

19.1 The One Template

Nearly every problem in the book has the same shape: *recover an object with low-dimensional structure from few linear measurements*, written as

$$\min \ (\text{a structure-promoting penalty}) \quad \text{subject to} \quad (\text{linear measurements match}).$$

The structure changes from chapter to chapter; the shape never does. Table 19.1 fills in the template for each model in the book.

Model	Structure	Combinatorial penalty	Convex surrogate
Sparse vector	few nonzero entries	$\|\boldsymbol{x}\|_0$	$\|\boldsymbol{x}\|_1$
Low-rank matrix	few nonzero singular values	$\operatorname{rank}(\boldsymbol{X})$	$\|\boldsymbol{X}\|_*$
Robust PCA	low rank + sparse	$\operatorname{rank} + \|\cdot\|_0$	$\|\cdot\|_* + \lambda\|\cdot\|_1$
Joint sparsity	shared support across columns	$\|\boldsymbol{X}\|_{\text{row},0}$	$\|\boldsymbol{X}\|_{1,2}$
Dictionary / NMF	sparse (or nonnegative) code over learned atoms	$\|\boldsymbol{x}_i\|_0$	alternating fit

Table 19.1. One template, filled in five ways. Each model replaces a combinatorial measure of structure by a convex surrogate, and recovers from few linear measurements.

In every row the combinatorial penalty (ℓ_0, rank, row-ℓ_0) is replaced by a convex surrogate (ℓ_1, nuclear norm, $\ell_{1,2}$), and the surrogate is provably exact under the right condition on the measurement operator. That single move, trading hard combinatorial structure for a tractable convex norm that happens to recover it, is the most important idea in the book.

19.2 The Recurring Recipe

Within the template, every chapter followed the same four steps (Fig. 19.1). Naming them makes the unity explicit.

1. **Model the structure.** Identify the few degrees of freedom: sparsity, rank, shared support, a learned dictionary.
2. **Relax the penalty.** Replace the intractable combinatorial count by its convex envelope, the tightest convex function below it.
3. **Recover by a tractable program.** Solve the convex problem with a linear program, a greedy pursuit, or a proximal iteration.
4. **Certify the recovery.** Prove the relaxation is exact under a condition on the operator, the null space property, the restricted isometry property, incoherence, which random matrices satisfy with overwhelming probability.

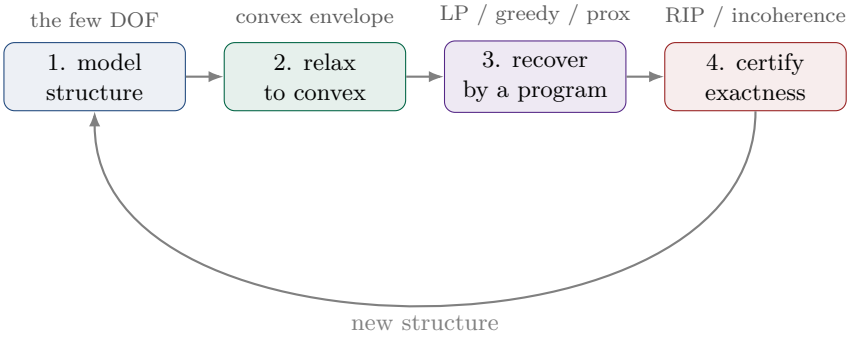


Figure 19.1. The recurring recipe, one template every chapter fills in. It models a low-dimensional structure (the few degrees of freedom), relaxes its combinatorial penalty to a convex surrogate, recovers by a tractable program (a linear program, greedy pursuit, or proximal iteration), and certifies the relaxation under a condition the sensing operator meets (RIP, incoherence, the null space property). Each new model re-enters at step one.

19.3 The Grand Chain

The pieces connect into one chain from the basic question to its many elaborations. The vector theory of Parts II–IV is the trunk: sparsity, the ℓ_0 problem, its ℓ_1 relaxation, the uniqueness conditions (spark, coherence, null space property), the restricted isometry property, and the random matrices that satisfy it. Part V lifts the entire trunk from vectors to matrices, reading sparsity of the singular values as low rank, the ℓ_1 norm as the nuclear norm, and the null space property as its matrix analogue; from there branch matrix completion, robust PCA, dictionary learning, and nonnegative factorization. Part VI carries the ideas into joint sparsity and into classification, where recovery becomes recognition. The applications of Chapter 17 are the trunk and branches deployed in hardware. Each arrow in this chain is a theorem proved in the book, and reading it from left to right is reading the table of contents as a single argument.

19.4 The Arc of the Course

Table 19.2 maps the parts of the book to their central results, the ten-week arc in one view.

Topic	Central result
Sparsity, ℓ_0 vs. ℓ_1	ℓ_1 relaxation recovers the sparsest vector under coherence and spark conditions.
Uniqueness and stability	Spark bound $s < \operatorname{spark}(\boldsymbol{A})/2$; coherence bound $s < \frac{1}{2}(1 + 1/\mu)$; the null space property.
Restricted isometry property	RIP implies stable, robust ℓ_1 recovery, the workhorse sufficient condition.
Random sensing matrices	Gaussian, sub-Gaussian, and structured random matrices satisfy RIP with $m \gtrsim s \log(N/s)$ rows.
Recovery algorithms	Greedy (OMP, CoSaMP), iterative hard thresholding, and convex solvers (ISTA, FISTA).
Matrix completion	Nuclear-norm minimization recovers a low-rank matrix from $O(n^{5/4} r \log n)$ entries (Candès–Recht).
Robust PCA	$\min \|\boldsymbol{X}\|_* + \lambda \|\boldsymbol{E}\|_1$ separates low rank from sparse corruption (Candès–Li–Ma–Wright).
Dictionary learning	MOD and K-SVD learn the dictionary and the sparse code by alternation; k-means is the $s = 1$ case.
Nonnegative factorization	Lee–Seung multiplicative updates; nonnegativity yields parts-based atoms; the bridge to topic modeling.
Joint sparsity and SRC	MMV uniqueness improves with $\operatorname{rank}(\boldsymbol{Y})$; SOMP; sparse representation turns recovery into recognition.

Table 19.2. The arc of the book, with the central result of each segment. Every row is one instance of the template of Section 19.1.

19.5 The Unifying Principle

One sentence to remember

Low-dimensional structure makes high-dimensional recovery possible from few measurements. A signal in $\mathbb{R}^N$ with only s degrees of freedom, an s-sparse vector, a rank-r matrix, a row-sparse code, does not need N measurements to pin down; it needs a number proportional to its true complexity, s or r, up to logarithmic factors. The combinatorial way to exploit that structure is intractable, but the convex relaxation, ℓ_1 for sparsity, the nuclear norm for rank, $\ell_{1,2}$ for joint sparsity, recovers the same answer in polynomial time whenever the measurement operator is incoherent or satisfies the restricted isometry property, conditions that random matrices meet with overwhelming probability.

Every theorem in the book is a precise version of that sentence, and every algorithm is a way to carry it out. The subject began with a question that sounded impossible, how to reconstruct a signal from far fewer measurements than its dimension, and answered it by noticing that real signals are never as complicated as their dimension suggests. They live on thin, low-dimensional sets, and a few well-chosen measurements plus a convex program are enough to find them. That is compressed sensing, and it is why the same mathematics keeps reappearing across imaging, vision, recommendation, and machine learning. The structure was always there; the achievement was learning to measure little and recover it.

PART VII

The Problem Sets

About the Problem Sets

The course was half theory and half code, and the nine problem sets are where the theorems of the preceding chapters became working programs. Each problem set pairs a mathematical task with a computational one, building from brute-force sparse recovery on toy instances up to a face-recognition system and a topic model on real data. The chapters that follow restate each problem and work each solution, and every problem set points back to the chapter whose theory it implements and forward to the accompanying notebook that runs the experiment.

The arc of the nine sets mirrors the arc of the book.

Set	Chapter	What you build
PS1	Chapter 3	brute-force ℓ_0 recovery and oracle least squares; watch the cost explode as $\binom{N}{s}$ predicts
PS2	Chapter 5	ℓ_1 minimization as a linear program; recovery in time and frequency; incoherence
PS3	Chapter 6	OMP, subspace pursuit, and CoSaMP head to head
PS4	Chapters 2 and 6	proofs: Cauchy–Schwarz, the reverse triangle and polarization identities, induced norms, OMP optimality, why ℓ_1 is sparse
PS5	Chapter 10	structured random (partial-Fourier) sensing and image reconstruction
PS6	Chapter 7	a recovery-algorithm shootout: ISTA, FISTA, IHT, and basis pursuit on images
PS7	Chapter 16	face recognition by sparse representation on the Extended Yale B database, with occlusion
PS8	Chapters 11 to 13	matrix completion, K-SVD dictionary learning, and robust PCA
PS9	Chapter 14	nonnegative matrix factorization and topic discovery

The mathematical problems are solved here in full; the computational problems are described with the algorithm, the expected behavior, and the figure each produces, so the two sides stay connected. Where a problem set asks for a proof that also appears in a chapter, the solution here is the short exam-ready version and the chapter has the long one.

CHAPTER 20

Problem Set 1: Brute-Force ℓ_0 Recovery

This first problem set makes the intractability of (P_0) concrete and demonstrates oracle recovery. It implements the brute-force search of Chapter 3 on two real sensing scenarios, watches its cost grow exactly as $\binom{N}{s}$ predicts, and ties the success of recovery to the conditioning of the sensing matrix.

Problem 1: Recover a sparse signal by exhaustive search

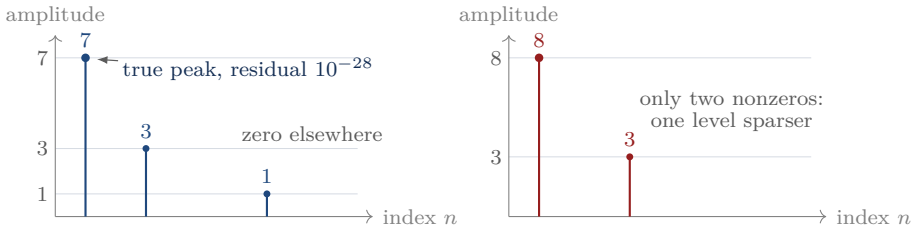
A signal $\boldsymbol{x} \in \mathbb{R}^{100}$ has at most three nonzero entries ($N = 100$, $s \leq 3$), with unknown locations and magnitudes. Two sensing matrices were applied, producing measurements $\boldsymbol{y}_1 = \boldsymbol{A}_1\boldsymbol{x}$ and $\boldsymbol{y}_2 = \boldsymbol{A}_2\boldsymbol{x}$. Recover $\boldsymbol{x}$ in each case by solving the ℓ_0 problem $\min\|\boldsymbol{x}\|_0$ subject to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$.

Solution. The ℓ_0 problem has no shortcut, so we search by *increasing sparsity*, smallest first (Occam's razor). For each candidate size $k = 1, 2, 3$ and each support S of size k, extract the column submatrix $\boldsymbol{A}_S$, run oracle recovery $\hat{\boldsymbol{x}}_S = \boldsymbol{A}_S^{\dagger}\boldsymbol{y}$, and test the residual $\|\boldsymbol{y} - \boldsymbol{A}_S\hat{\boldsymbol{x}}_S\|_2$. The first support whose residual is numerically zero is the sparsest explanation, and by the uniqueness theory of Chapter 4 it is the true signal. The four steps (enumerate supports, slice columns, pseudoinverse-fit, test residual) are the oracle recipe of Section 3.4.

The search for the first scenario climbs through the sparsity levels and locks on at $k = 3$:

k	best support	coefficients	residual^2
1	$\{1\}$	7.03	1.56×10^{2}
2	$\{1, 3\}$	7.46, 2.95	1.58×10^{1}
3	$\{1, 3, 7\}$	7.0, 3.0, 1.0	3.4×10^{-28}

The residual collapses to machine zero only at $k = 3$, recovering $\boldsymbol{x}_1$ supported on $\{1, 3, 7\}$ with values $(7, 3, 1)$. The second scenario locks on one level earlier, at $k = 2$: support $\{1, 5\}$ with values $(8, 3)$ and residual 7.7×10^{-14}. The two recovered signals are genuinely different, two and three nonzeros, so searching only at the maximum $s = 3$ would have mislabeled the second signal; the increasing-sparsity order is essential to the ℓ_0 objective. Figure 20.1 shows both.



Case 1 ($\boldsymbol{A}_1$): 3-sparse $\{1, 3, 7\}$ Case 2 ($\boldsymbol{A}_2$): 2-sparse $\{1, 5\}$

Figure 20.1. The two recovered signals. Brute-force ℓ_0 returns a 3-sparse signal in the first scenario and a 2-sparse one in the second, confirming the two sensing matrices encode different signals.

Problem 2: Why brute force does not scale

For $N = 100$, how large can s be before the search is infeasible?

Solution. The number of supports of size s is $\binom{100}{s}$. At the recovered sparsities the counts are still small, $\binom{100}{2} = 4950$ and $\binom{100}{3} = 161{,}700$, a fraction of a second each. They explode immediately after: $\binom{100}{5} = 7.53 \times 10^7$ (seconds to minutes), $\binom{100}{10} = 1.73 \times 10^{13}$ (out of reach), and worse beyond. The back-of-the-envelope estimate of Section 3.5 holds exactly: brute force is fine for tiny s and hopeless past about $s = 8$ even at this modest N, which is the entire motivation for the convex and greedy relaxations of the chapters that follow.

Problem 3: The sensing matrix is the real constraint

Why does recovery succeed for one matrix and not necessarily for the other? Examine the two matrices.

Solution. Recoverability is governed by the conditioning of small column submatrices, which the mutual coherence $\mu(\boldsymbol{A})$ and the condition number $\kappa(\boldsymbol{A})$ summarize:

	coherence μ	condition κ	uniqueness bound $\frac{1}{2}(1 + 1/\mu)$
$\boldsymbol{A}_1$	0.70	2.5	1.2
$\boldsymbol{A}_2$	1.00	3.7×10^{16}	1.0

$\boldsymbol{A}_1$ is well conditioned: low coherence means every small column submatrix is near-orthonormal, the oracle pseudoinverse is stable, and recovery is exact. $\boldsymbol{A}_2$ has coherence 1, meaning two of its columns are essentially parallel, so a support containing both gives a nearly singular $\boldsymbol{A}_S$ whose pseudoinverse amplifies error, and the residual test becomes delicate. The coherence bound $\frac{1}{2}(1 + 1/\mu)$ formally guarantees uniqueness only up to $s = 1$ for both matrices, yet brute force still succeeded at $s = 2$ and $s = 3$: the bound is sufficient, not necessary, and is conservative in practice. This is the first encounter with the theme of the whole book, that the spark and coherence of the sensing matrix, not the recovery algorithm, set the limit (Chapters 4 and 8).

In the Problem Sets

The accompanying notebook generates the two sensing scenarios, runs the brute-force solver across sparsity levels with `itertools.combinations`, and produces the residual-versus-sparsity curve, the recovered-signal stem plots, and the coherence and condition-number comparison of the two matrices.

CHAPTER 21

Problem Set 2: Sensing Bases and Incoherence

This set replaces the brute-force search with ℓ_1 minimization and asks the central practical question: which sensing matrix should you use? Sweeping six designs against both time-sparse and frequency-sparse signals turns the incoherence principle of Chapter 10 into a measured phase transition, and shows that the right sensing basis depends entirely on where the signal is sparse.

Problem 1: Six sensing designs on a time-sparse signal

For $N = 256$ and an $s = 5$ signal sparse in time, compare six ways to build the $m \times N$ sensing matrix, sweeping $m \in \{10, 20, \ldots, 100\}$ over 100 random trials each and recording the fraction with exact ℓ_1 recovery ($\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2 \le 10^{-6}$). The six designs are (a) random rows of the identity (random time samples), (b) uniformly spaced rows of the identity, (c) random rows of the DCT (random frequency samples), (d) the first m rows of the DCT (low frequencies), (e) uniformly spaced rows of the DCT, and (f) a Gaussian matrix with orthonormalized rows.

Solution. Solve $\min\|\boldsymbol{z}\|_1$ subject to $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}$ as the linear program of Chapter 5, via the split $\boldsymbol{z} = \boldsymbol{u} - \boldsymbol{v}$ with $\boldsymbol{u}, \boldsymbol{v} \ge 0$. The recovery rates split the six designs cleanly:

design	m=20	m=30	m=40	m=60	m=100
(c) random frequency	0.17	0.87	1.00	1.00	1.00
(f) random domain	0.10	0.79	1.00	1.00	1.00
(d) low frequency	0.55	0.79	0.82	0.85	0.99
(a) random time	0.00	0.00	0.00	0.00	0.02
(e) equispaced frequency	0.00	0.00	0.01	0.00	0.04
(b) uniform time	0.00	0.00	0.00	0.01	0.00

The two incoherent designs, random frequency (c) and random Gaussian (f), reach perfect recovery by $m = 40$, about 16% of N, matching the $m \sim s\log(N/s) \approx 5\ln 51 \approx 20$ prediction up to the usual constant. Sampling in time (a, b) fails almost completely on a time-sparse signal, because a spike basis is maximally coherent with the signal's own support: most random time samples simply miss the few nonzeros. Low-frequency sampling (d) also recovers well, reaching about 0.8 by $m = 30$: a time-sparse signal spreads its energy across the whole spectrum, so even the low band stays incoherent with the spike basis. Equispaced frequency (e) is the exception among the frequency designs, recovering only erratically because its periodic sampling pattern aliases. Figure 21.1 plots the phase transitions.

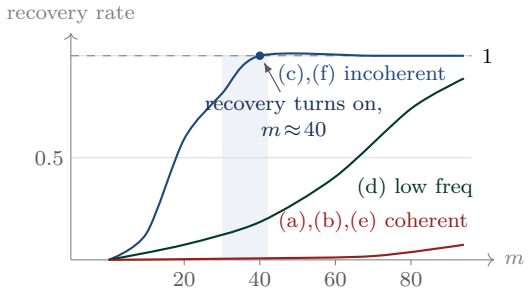


Figure 21.1. Phase transitions for a time-sparse signal. Incoherent designs (random frequency, random Gaussian) jump to perfect recovery near $m = 40$ (highlighted band, marked •); coherent time-domain designs stay near zero. The sharp transition is the experimental signature of the $s \log(N/s)$ theory.

Problem 2: The same six designs on a frequency-sparse signal

Now let the signal be sparse in the DCT domain, $\boldsymbol{x} = \boldsymbol{\Psi\alpha}$ with $\boldsymbol{\alpha}$ s-sparse, and repeat. How must the ℓ_1 program change?

Solution. Recover the coefficients, not the samples: since $\boldsymbol{y} = \boldsymbol{Ax} = \boldsymbol{A\Psi\alpha}$, solve $\min\|\boldsymbol{\alpha}\|_1$ subject to $(\boldsymbol{A\Psi})\boldsymbol{\alpha} = \boldsymbol{y}$, then reconstruct $\hat{\boldsymbol{x}} = \boldsymbol{\Psi}\hat{\boldsymbol{\alpha}}$. The recovery table is the *mirror image* of Problem 1: now the time-domain designs win and the frequency-domain ones fail.

design	m=20	m=30	m=40	m=60	m=100
(a) random time	0.15	0.87	1.00	1.00	1.00
(f) random domain	0.15	0.83	1.00	1.00	1.00
(b) uniform time	0.18	0.46	0.78	0.80	0.94
(c) random frequency	0.00	0.00	0.00	0.00	0.00
(d),(e) low/equi. freq	0.00	0.00	0.00	0.00	0.00

Random time samples (a) and the universal Gaussian design (f) recover perfectly by $m = 40$; uniform time samples (b) work but more slowly, reaching 94% only at $m = 100$. The frequency-domain designs (c)–(e), now maximally coherent with the DCT-sparse signal, never get off the floor. This is the exact mirror of Problem 1, with the time and frequency roles swapped.

The incoherence principle, measured

A sensing basis works when it is *incoherent* with the sparsity basis. Time samples recover frequency-sparse signals and frequency samples recover time-sparse signals, because each spreads a spike in one domain across all measurements in the other. The Gaussian design (f) is *universal*: incoherent with every fixed basis, it wins in both tables. This is the experimental face of the uncertainty principle of Theorem 10.2.

Problem 3: Recovering the challenge data without a sparsity prior

Apply ℓ_1 to the two scenarios of Chapter 20, this time without being told the signal is 3-sparse.

Solution. The single ℓ_1 program needs no sparsity input: it returns the sparsest feasible signal automatically. For the well-conditioned $\boldsymbol{A}_1$ it recovers exactly the ℓ_0 answer of Chapter 20, support $\{1, 3, 7\}$ with values $(7, 3, 1)$ and residual $\approx 6 \times 10^{-15}$ (order 10^{-14}). For the pathological $\boldsymbol{A}_2$ ($\mu \approx 1$, $\kappa \approx 10^{16}$) the ℓ_1 solution can differ from the ℓ_0 one, because the coherence is far above the threshold $\frac{1}{2}(1 + 1/\mu)$ that would guarantee $\ell_1 = \ell_0$. The lesson repeats: ℓ_1

costs one linear program of $2N = 512$ variables instead of the $\binom{256}{5} \approx 8.8 \times 10^9$ subsets a brute-force ℓ_0 would need, and it succeeds whenever the matrix is incoherent enough.

In the Problem Sets

The accompanying notebook builds all six sensing designs, runs the ℓ_1 linear program over the measurement sweep and 100 trials per point, and produces the two phase-transition plots and the challenge-data recovery, confirming the mirror-image structure between the time-sparse and frequency-sparse experiments.

CHAPTER 22

Problem Set 3: Greedy Algorithms

This set implements the greedy recovery family of Chapter 6, races the methods against ℓ_1 minimization across the six sensing designs of Chapter 21, and measures the speed-versus-robustness trade that makes greedy methods the practical default.

Problem 1: Implement OMP, SP, and CoSaMP

Code orthogonal matching pursuit, subspace pursuit, and CoSaMP against a common interface (dictionary, measurements, sparsity), and verify each recovers an $s = 5$ signal in $\mathbb{R}^{256}$.

Solution. All three share the structure of Chapter 6: a correlation scan $\boldsymbol{A}^*\boldsymbol{r}$, a support update, a least-squares refit by pseudoinverse, and a residual update. **OMP** adds the single best-correlated atom each iteration and never removes one, so the residual is orthogonal to the chosen atoms and no atom is reselected. **Subspace pursuit** adds the top s correlations to the current support (size $\leq 2s$), refits, and hard-thresholds back to s, which lets it discard a bad early pick. **CoSaMP** is identical but adds the top $2s$ (support size $\leq 3s$), a wider safety net. The pruning step is the only structural difference, and it is two lines of code.

Problem 2: A recovery and runtime shootout

Across all six sensing designs and the measurement sweep, measure each method's exact-recovery probability and average runtime, against the ℓ_1 linear program as baseline.

Solution. The sensing matrix dominates, but the solver matters on the hard cases. At $m = 50$ the fully incoherent designs are solved by everyone and the badly coherent designs are solved by no one; the interesting regime is the partially structured frequency designs, where ℓ_1 pulls far ahead of the greedy methods:

design (m=50)	OMP	SP	CoSaMP	ℓ_1
(c) random frequency	1.00	1.00	1.00	1.00
(f) random domain	1.00	1.00	1.00	1.00
(d) low frequency	0.57	0.08	0.03	0.89
(e) equispaced frequency	0.52	0.05	0.01	0.88
(a) random time, (b) uniform time	0.00	0.00	0.00	0.00

On the structured frequency designs (d) and (e) the methods are far from indistinguishable: ℓ_1 recovers about 88–89% of signals while SP and CoSaMP collapse to single-digit percentages and OMP lands in between at roughly half. The greedy correlation scan gets trapped by aliased columns and the hard-thresholding prune in SP/CoSaMP then discards correct atoms with no way to recover them, whereas ℓ_1 sees the whole feasible polytope. OMP, which only ever adds atoms, is more robust here than its backtracking cousins.

The runtimes, in contrast, differ by more than an order of magnitude:

average ms (m=50)	OMP	SP	CoSaMP	ℓ_1
random frequency	0.15	0.13	0.16	7.5
random domain	0.18	0.16	0.31	7.4

On these incoherent designs all three greedy methods finish in a fraction of a millisecond, making the ℓ_1 linear program roughly 40–50× slower, which the complexity counts explain: OMP costs $O(s\,mN)$, the batch methods a few iterations of $O(mN)$ correlation plus an $O(m\,s^2)$ refit, while the ℓ_1 program solves a $2N = 512$-variable linear program at $O(N^3)$. Figure 22.1 shows the gap.

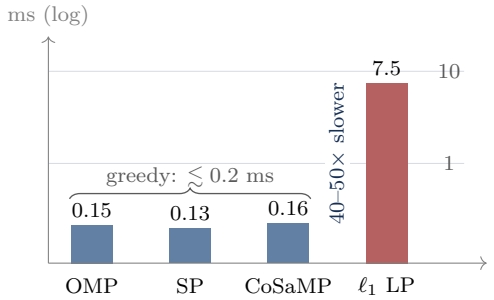


Figure 22.1. Average recovery time per signal at $m = 50$ on the random-frequency design (log-scaled bars, milliseconds), with 1 and 10 ms reference levels. The three greedy methods finish in under a fifth of a millisecond; the ℓ_1 linear program takes about 7.5, some 40–50× slower (dimension arrow). On these incoherent designs all four recover the same signals, so the greedy methods are the practical choice.

Problem 3: The challenge data in the greedy lens

Run all four methods on the two scenarios of Chapter 20.

Solution. On the well-conditioned $\boldsymbol{A}_1$ all four agree perfectly with the brute-force ℓ_0 answer: support $\{1, 3, 7\}$, values $(7, 3, 1)$, residuals at machine precision (10^{-14} for OMP, 10^{-15} for ℓ_1). This is the golden case where the exact-recovery condition of Theorem 6.7 holds. On the pathological $\boldsymbol{A}_2$ the methods *diverge*: OMP latches onto a wrong atom (index 2 with value 11.0 instead of the true $\{1, 5\}$), and ℓ_1 returns a different spread-out support, each consistent with the measurements but none matching the truth. The exact-recovery condition fails because $\mu \approx 1 \gg 1/(2s-1) = 0.2$. When the sensing matrix is the problem, no algorithm can rescue the recovery; the limit is the matrix, not the solver.

In the Problem Sets

The accompanying notebook implements the three greedy algorithms behind a shared interface, runs the Monte-Carlo recovery and timing experiments across all six designs, and reports per-algorithm recovery probability and average runtime alongside the challenge-data comparison, where the methods agree on the good matrix and diverge on the bad one.

CHAPTER 23

Problem Set 4: Proofs and Inequalities

This set is the course's written proof set. The inequalities it asks for are the ones the recovery theorems are built from, and the proofs are exam-ready.

Problem 1: Cauchy–Schwarz

Prove $|\langle \boldsymbol{u}, \boldsymbol{v}\rangle| \le \|\boldsymbol{u}\|_2\|\boldsymbol{v}\|_2$.

Solution. For real t the quadratic $q(t) = \|\boldsymbol{u} - t\boldsymbol{v}\|_2^2 = \|\boldsymbol{u}\|_2^2 - 2t\langle \boldsymbol{u}, \boldsymbol{v}\rangle + t^2\|\boldsymbol{v}\|_2^2 \ge 0$ has nonpositive discriminant, $4\langle \boldsymbol{u}, \boldsymbol{v}\rangle^2 - 4\|\boldsymbol{u}\|_2^2\|\boldsymbol{v}\|_2^2 \le 0$, which is the claim. Equality holds iff $\boldsymbol{u}$ and $\boldsymbol{v}$ are linearly dependent.

Problem 2: Reverse triangle inequality

Prove $\big|\|\boldsymbol{u}\|_2 - \|\boldsymbol{v}\|_2\big| \le \|\boldsymbol{u} - \boldsymbol{v}\|_2$.

Solution. By the triangle inequality $\|\boldsymbol{u}\|_2 = \|(\boldsymbol{u}-\boldsymbol{v}) + \boldsymbol{v}\|_2 \le \|\boldsymbol{u}-\boldsymbol{v}\|_2 + \|\boldsymbol{v}\|_2$, so $\|\boldsymbol{u}\|_2 - \|\boldsymbol{v}\|_2 \le \|\boldsymbol{u}-\boldsymbol{v}\|_2$; swapping the roles gives the other side, and together they give the absolute value.

Problem 3: The polarization identity

Prove $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \frac{1}{4}(\|\boldsymbol{u}+\boldsymbol{v}\|_2^2 - \|\boldsymbol{u}-\boldsymbol{v}\|_2^2)$ *(real case).*

Solution. Expand both squares: $\|\boldsymbol{u}+\boldsymbol{v}\|_2^2 = \|\boldsymbol{u}\|_2^2 + 2\langle \boldsymbol{u}, \boldsymbol{v}\rangle + \|\boldsymbol{v}\|_2^2$ and $\|\boldsymbol{u}-\boldsymbol{v}\|_2^2 = \|\boldsymbol{u}\|_2^2 - 2\langle \boldsymbol{u}, \boldsymbol{v}\rangle + \|\boldsymbol{v}\|_2^2$. Subtracting gives $4\langle \boldsymbol{u}, \boldsymbol{v}\rangle$, hence the identity. This is the identity used in Lemma 8.7 to bound disjoint-support cross terms.

Problem 4: Induced matrix norms

Show $\|\boldsymbol{A}\|_{1\to 1}$ *equals the maximum absolute column sum and* $\|\boldsymbol{A}\|_{\infty\to\infty}$ *the maximum absolute row sum.*

Solution. For the $1\to 1$ norm, the supremum of $\|\boldsymbol{A}\boldsymbol{x}\|_1/\|\boldsymbol{x}\|_1$ is attained at an extreme point of the ℓ_1 ball, a signed standard basis vector $\pm\boldsymbol{e}_j$, where $\|\boldsymbol{A}\boldsymbol{e}_j\|_1 = \sum_i |A_{ij}|$ is the j-th column sum; the maximum over j is the claim. The $\infty\to\infty$ norm is the same statement for $\boldsymbol{A}^\top$, giving the maximum row sum.

Problem 5: OMP optimality

Show that after the OMP least-squares refit, the residual is orthogonal to every selected atom.

Solution. The refit sets $\hat{\boldsymbol{x}}_S = \boldsymbol{A}_S^\dagger \boldsymbol{y}$, so $\boldsymbol{A}_S^*(\boldsymbol{y} - \boldsymbol{A}_S\hat{\boldsymbol{x}}_S) = \boldsymbol{A}_S^*\boldsymbol{y} - \boldsymbol{A}_S^*\boldsymbol{A}_S(\boldsymbol{A}_S^*\boldsymbol{A}_S)^{-1}\boldsymbol{A}_S^*\boldsymbol{y} = \boldsymbol{0}$. Hence $\langle \boldsymbol{a}_j, \boldsymbol{r}\rangle = 0$ for every $j \in S$, so no selected atom is ever reselected (Proposition 6.4).

Problem 6: ℓ_1 minimization encourages sparsity

Show that a unique (P_1) *solution has at most* m *nonzeros.*

Solution. If $\hat{\boldsymbol{x}}$ is the unique (P_1) solution, the columns of $\boldsymbol{A}$ on its support are linearly independent (else a null-space perturbation within the support would give an equally good or better solution, contradicting uniqueness, Theorem 5.5). An independent set of columns has at most $\operatorname{rank}(\boldsymbol{A}) \leq m$ members, so $\|\hat{\boldsymbol{x}}\|_0 \leq m$.

In the Problem Sets

The computational companion verifies each inequality numerically on random vectors and matrices, confirming the proofs hold with the expected equality cases.

CHAPTER 24

Problem Set 5: Compressed Sensing of Images

This set leaves toy vectors for real images and confirms that compressed sensing works on signals whose sparsity is only approximate and whose level is unknown. Each image is sensed patch by patch, recovered by ℓ_1 regression in the DCT domain, and scored by peak signal-to-noise ratio, the quantitative form of the recovery miracle of Chapter 1.

Problem 1: A patch-based compressed-sensing pipeline

Reconstruct three 128×128 *images (a phantom, a brain MRI, the cameraman) from* m *measurements per* 8×8 *patch, for two sensing designs (random Gaussian and random pixel subsampling), and measure quality as* m *varies over* $\{8, 12, 16, 20, 28, 36, 48, 56\}$.

Solution. Each 8×8 patch is a vector of $N = 64$ pixels, sparse in the two-dimensional DCT: writing $\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{s}$ with $\boldsymbol{\Psi}$ the orthonormal DCT synthesis matrix, the patch coefficients $\boldsymbol{s}$ are nearly sparse. For each patch, simulate $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x} = (\boldsymbol{\Phi}\boldsymbol{\Psi})\boldsymbol{s}$, recover $\hat{\boldsymbol{s}}$ by the ℓ_1-penalized least squares (Lasso) of Chapter 7, and debias by an ordinary least-squares refit on the detected support (the post-Lasso estimator, which removes the shrinkage bias of the penalty). Synthesize $\hat{\boldsymbol{x}} = \boldsymbol{\Psi}\hat{\boldsymbol{s}}$ and clip to $[0, 255]$. Quality is

$$\mathrm{PSNR} = 10 \log_{10} \frac{255^2}{\mathrm{MSE}}, \qquad \mathrm{MSE} = \tfrac{1}{P} \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2,$$

in decibels, with P the number of pixels in the reconstructed image and higher being better. The DCT round-trip is exact to machine precision ($\|\boldsymbol{x} - \boldsymbol{\Psi}\boldsymbol{\Psi}^\top \boldsymbol{x}\|_2 \approx 2 \times 10^{-15}$), so all error is the recovery's.

Problem 2: Quality versus number of measurements

Tabulate and plot PSNR against m for each image and design.

Solution. Quality climbs monotonically with m and depends strongly on the image's DCT decay:

PSNR (dB)	m=8	m=16	m=28	m=36	m=48	m=56
brain, Gaussian	19.4	22.9	28.1	30.8	37.2	41.5
brain, subsample	20.4	22.4	28.5	31.3	37.5	42.8
phantom, Gaussian	15.8	16.6	23.2	25.8	32.3	36.5
phantom, subsample	16.0	18.0	22.4	24.7	31.8	38.7
cameraman, Gaussian	17.0	19.8	23.9	25.7	33.0	36.8
cameraman, subsample	17.4	20.3	25.1	27.3	32.3	37.7

The brain MRI recovers best at every m (smooth tissue concentrates its DCT energy in a few coefficients). The cameraman and the piecewise-constant phantom trail and trade places, with the phantom generally the lowest of the three, especially under random subsampling. Figure 24.1 plots the brain curve.

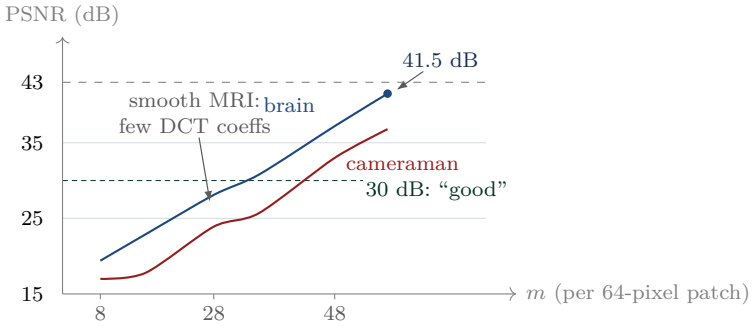


Figure 24.1. PSNR versus measurements per patch (Gaussian sensing), with 25/30/35 dB reference levels. The smooth brain MRI crosses the 30 dB "good" line earlier and peaks higher (41.5 dB at m=56, marked •) than the textured cameraman, because its DCT spectrum decays faster. Both rise monotonically, the imaging form of the phase transition.

Problem 3: Which sensing design, and why?

Compare the two sensing designs and state a preference.

Solution. The two are close, with a revealing crossover. At moderate m the Gaussian design has a slight edge, consistent with its tighter restricted isometry constants. At high m (above about half of N) random subsampling matches or beats it, by up to 2 dB on the phantom at $m = 56$ (38.7 vs 36.5). The reason is that the images are only *approximately* sparse: subsampling reports exact pixel values, while a Gaussian measurement mixes the un-modeled DCT tail into every reading. When measurements are plentiful the clean pixel values win. For a general-purpose, low-measurement system the Gaussian (or a structured random matrix, Chapter 25) is preferable for its universal guarantee; when measurements are cheap and abundant, subsampling is simpler and competitive.

In the Problem Sets

The accompanying notebook builds the DCT synthesis matrix and both sensing designs, runs the patch-wise Lasso-with-debiasing reconstruction across the measurement sweep for all three images, and produces the PSNR-versus-measurement curves and the progressive reconstructions that sharpen as m grows.

CHAPTER 25

Problem Set 6: Structured Random Matrices and Coherence Bounds

This set adds the structured random matrix of Chapter 10 to the imaging pipeline of Chapter 24 and pairs the experiment with the proof that pins down the range of mutual coherence. Together they show why a fast, structured sensing operator recovers nearly as well as a dense Gaussian, and why incoherence has a hard floor of $1/\sqrt{N}$.

Problem 1: Gaussian versus subsampling versus SRM

Add a structured random matrix to the three-image reconstruction of Chapter 24 and compare. The SRM is block-diagonal, each block a scaled product $\boldsymbol{RFD}$: $\boldsymbol{D}$ a random sign (or permutation) diagonal, $\boldsymbol{F}$ a Hadamard or DCT block, and $\boldsymbol{R}$ a random row subset of the identity.

Solution. The product $\boldsymbol{RFD}$ is exactly the recipe of Section 10.4: the sign flip $\boldsymbol{D}$ randomizes the signal, the fast transform $\boldsymbol{F}$ spreads its energy, and the row selector $\boldsymbol{R}$ subsamples, all applied in $O(N \log N)$ time with no dense matrix stored. Its columns come out essentially unit-norm (mean 0.999, standard deviation 0.04), matching the Gaussian. The three designs track each other across the measurement sweep:

PSNR (dB), brain	m=8	m=20	m=36	m=56
Gaussian	19.4	25.0	30.8	41.5
subsampling	20.4	24.3	31.3	42.8
SRM	20.1	22.9	32.1	42.7

At $m = 36$ the SRM actually leads on the brain image (32.1 vs Gaussian's 30.8); at smaller m it trails the best design by as much as about 2 dB (for instance brain at $m = 20$, 22.9 vs 25.0), closing the gap as m grows. The point is the cost: the Gaussian needs $O(mN)$ work and $O(mN)$ storage per patch, while the SRM needs $O(N \log N)$ and almost no storage, the same recovery quality for a fraction of the compute. This is the practical payoff of Chapter 10: keep the randomness that makes the guarantee work, wrap it around a fast transform that makes it cheap.

Problem 2: The mutual coherence between two bases is between $1/\sqrt{N}$ and 1

For orthonormal bases $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N]$ of $\mathbb{R}^N$, with $\mu(\boldsymbol{U}, \boldsymbol{V}) = \max_{i,j} |\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle|$, prove $1/\sqrt{N} \le \mu(\boldsymbol{U}, \boldsymbol{V}) \le 1$ and that both bounds are sharp.

Solution.

Upper bound. Each $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ is a unit vector, so by Cauchy–Schwarz $|\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle| \le \|\boldsymbol{u}_i\|_2 \|\boldsymbol{v}_j\|_2 = 1$. Taking the maximum over i, j gives $\mu \le 1$. It is attained: with $\boldsymbol{U} = \boldsymbol{V} = \boldsymbol{I}$, the diagonal inner products are $\langle \boldsymbol{e}_i, \boldsymbol{e}_i \rangle = 1$, so $\mu(\boldsymbol{I}, \boldsymbol{I}) = 1$.

Lower bound. Fix any i. Since $\{\boldsymbol{v}_j\}$ is an orthonormal basis, Parseval's identity gives $\sum_{j=1}^{N} |\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle|^2 = \|\boldsymbol{u}_i\|_2^2 = 1$. A maximum is at least an average, so

$$\max_j |\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle|^2 \;\geq\; \frac{1}{N} \sum_{j=1}^{N} |\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle|^2 = \frac{1}{N},$$

hence $\max_j |\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle| \geq 1/\sqrt{N}$, and a fortiori $\mu \geq 1/\sqrt{N}$. It is attained by the spike-Fourier pair: with $\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{V} = \boldsymbol{F}$ the DFT, every entry has $|\langle \boldsymbol{e}_i, \boldsymbol{f}_j \rangle| = |F_{ij}| = 1/\sqrt{N}$, so $\mu(\boldsymbol{I}, \boldsymbol{F}) = 1/\sqrt{N}$ exactly. The real-valued Hadamard basis achieves the same floor.

The numerics confirm the spike-Fourier floor across dimensions:

N	4	8	16	32	64
$\mu(\boldsymbol{I}, \boldsymbol{F})$	0.500	0.354	0.250	0.177	0.125
$1/\sqrt{N}$	0.500	0.354	0.250	0.177	0.125

Why the floor matters

The lower bound $\mu \geq 1/\sqrt{N}$ says no sensing basis can be more than $\sqrt{N}$-incoherent with the sparsity basis. Since the measurement count scales as $\mu^2 N s \log N$, the floor $\mu^2 \geq 1/N$ caps the savings: the very best pairings (spike-Fourier, spike-Hadamard) hit $\mu^2 N = 1$ and give the $s \log N$ count, and nothing does better. The structured random matrix of Problem 1 is engineered to sit near this floor while still admitting a fast transform.

In the Problem Sets

The accompanying notebook builds the block-diagonal SRM with the fast Hadamard/DCT transform, adds it to the three-image reconstruction sweep alongside the Gaussian and subsampling designs, and verifies the coherence bounds numerically across dimensions, confirming the $1/\sqrt{N}$ floor is met exactly by the spike-Fourier and spike-Hadamard pairs.

CHAPTER 26

Problem Set 7: Face Recognition via SRC

This set builds the sparse-representation classifier of Chapter 16 end to end on the Extended Yale B face database, turning the recovery machinery into a working recognizer, hardening it against occlusion, and giving it a principled way to refuse a stranger.

Problem 1: Build the dictionary and classify

From the 2,414 *images of the cropped Extended Yale B face set (*38 *usable subjects, from the* 39 *the assignment lists) at* 96×84 *pixels, build a sparse-representation dictionary on a random half and classify the other half.*

Solution. Vectorize each 96×84 image to a length-8064 vector and reduce dimension by a single random Gaussian projection $\boldsymbol{R} \in \mathbb{R}^{150\times 8064}$, the random-features trick that makes the ℓ_1 solves fast while preserving the union-of-subspaces geometry. Stack the projected, unit-normalized training faces class by class into $\boldsymbol{A} \in \mathbb{R}^{150\times 1205}$. For each test face $\boldsymbol{y}$, solve $\min\|\boldsymbol{x}\|_1$ subject to $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \le \varepsilon$, form the class-wise residual $r_i = \|\boldsymbol{y} - \boldsymbol{A}\delta_i(\hat{\boldsymbol{x}})\|_2$ for each subject, and predict the smallest (Algorithm 13). On the 1,209 test faces this reaches 95.0% **accuracy** (1,149 correct), with the coefficient mass landing on the correct subject for nearly every query (median sparsity concentration index 0.47). Column normalization is essential: without it, brighter training images dominate the correlations and bias the prediction.

Problem 2: Robust SRC under corruption

Corrupt a fraction p *of each test image, by random pixels or a contiguous patch, and compare plain SRC against the identity-augmented robust SRC.*

Solution. Model the corrupted image as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e}$ with $\boldsymbol{e}$ sparse, and solve the augmented problem $\min\|[\boldsymbol{x};\boldsymbol{e}]\|_1$ subject to $[\boldsymbol{A}\ \boldsymbol{I}][\boldsymbol{x};\boldsymbol{e}] = \boldsymbol{y}$. The identity block gives each pixel its own atom, so the solver charges the corrupted pixels to $\boldsymbol{e}$ and keeps $\boldsymbol{x}$ on the true class. The robust formulation roughly doubles accuracy in the damaging mid-corruption range:

accuracy	p=0	p=0.1	p=0.2	p=0.3	p=0.5
random pixels, plain	94%	63%	34%	28%	11%
random pixels, robust	87%	80%	56%	54%	15%
patch, plain	94%	72%	49%	22%	14%
patch, robust	87%	70%	53%	35%	13%

At 30% random-pixel corruption the robust classifier holds 54% where the plain one has collapsed to 28% (Fig. 26.1). Contiguous-patch corruption is harder than scattered pixels at the same fraction, because a solid block violates the

sparse-error model more severely. Both methods fail once corruption passes about half the image, where there is simply too little clean signal left.

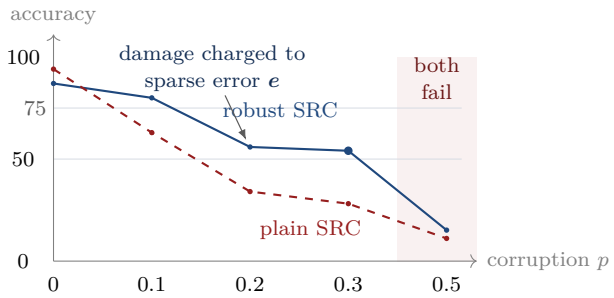


Figure 26.1. Accuracy under random-pixel corruption. The identity-augmented robust SRC (solid) charges corrupted pixels to a sparse error term and holds far above plain SRC (dashed): at p=0.3 it keeps 54% where plain collapses to 28% (marked •, roughly 2×). Both fail past $p \approx 0.5$ (shaded).

Problem 3: Rejecting strangers with the concentration index

Add faces of people not in the database (held-out subjects and synthetic selfies) and reject them using the sparsity concentration index.

Solution. Compute SCI($\hat{\boldsymbol{x}}$) from Definition 16.3: a genuine face concentrates its coefficient mass on one class and scores high, while a stranger spreads its mass and scores low. The two populations separate cleanly, in-database faces at a mean index of 0.49 and strangers at 0.07, so a single threshold discriminates them. Setting $\tau = 0.063$ to allow a 5% false-rejection rate on genuine faces, the index rejects strangers with area under the ROC curve 0.94: an excellent detector built from the same sparse code already computed for classification, at no extra cost. The classifier can now say "not in the database" instead of forcing a wrong guess.

In the Problem Sets

The accompanying notebook loads and preprocesses the Extended Yale B faces, builds the projected class-block dictionary, runs ℓ_1 classification with and without the identity-augmented robust formulation across corruption levels, and computes the concentration-index distributions and ROC curve for stranger rejection.

CHAPTER 27

Problem Set 8: Matrix Completion and Dictionary Learning

This set spans the matrix recovery half of the book in two topics: completing a low-rank matrix from a few entries (Chapter 11), and learning a dictionary by K-SVD to denoise an image (Chapter 13). Both turn the theory into a measured phase transition or a decibel gain.

Topic A.1: Nuclear-norm completion and the phase transition

Generate a 50×50 *rank-5 matrix* $\boldsymbol{M} = \boldsymbol{A}\boldsymbol{B}^\top$*, observe a random fraction* p *of its entries, and recover it by* $\min \|\boldsymbol{Z}\|_*$ *subject to* $\mathcal{P}_\Omega(\boldsymbol{Z}) = \mathcal{P}_\Omega(\boldsymbol{M})$*. Plot the relative error against* p*.*

Solution. Solve the semidefinite program of Theorem 11.7 with a convex modeling layer. The recovery shows a sharp phase transition:

fraction p	0.1	0.2	0.3	0.4	0.5
observed $\lvert\Omega\rvert$	249	524	748	1000	1245
relative error	0.89	0.63	0.38	0.16	2.9×10^{-6}

The error decreases steadily through the undersampled regime (0.89 at $p = 0.1$ down to 0.16 at $p = 0.4$) and then collapses to the solver's numerical floor ($\sim 10^{-6}$) at $p = 0.5$. The cliff sits between $p = 0.4$ and $p = 0.5$ (Fig. 27.1). The *degrees-of-freedom floor* is $r(N_1 + N_2 - r)/(N_1 N_2) = 5(50+50-5)/2500 = 19\%$, the information-theoretic minimum: a rank-5 50×50 matrix has only 475 free parameters. The *Candès–Recht* estimate $\sim rN \log N/(N_1 N_2) \approx 39\%$ predicts the order of magnitude. The empirical transition at $p \approx 0.45$–0.5 sits at or just above the Candès–Recht estimate and well above the degrees-of-freedom floor: exact recovery needs meaningfully more samples than the 475-parameter count alone would suggest, and the convex bound tracks the truth rather than over-promising.

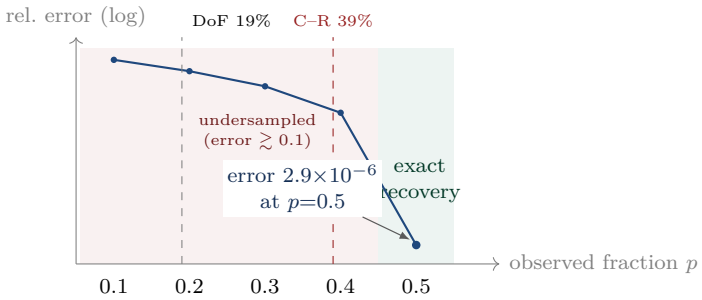


Figure 27.1. Phase transition for nuclear-norm completion of a 50×50 rank-5 matrix. The relative error (log scale) drifts down through the undersampled regime and then falls off a cliff between $p = 0.4$ and $p = 0.5$, reaching the solver's numerical floor (marked •, error $2.9{\times}10^{-6}$) at $p = 0.5$. The empirical transition sits at or just above the Candès–Recht estimate (39%) and well above the degrees-of-freedom floor (19%).

Topic A.2: Recommendations by alternating least squares

*On the MovieLens 100K ratings (*943 *users,* 1,682 *movies,* 100,000 *ratings,* 94% *missing), hold out* 20% *and complete the factored model* $\boldsymbol{R} \approx \boldsymbol{U}\boldsymbol{V}^\top$ *by alternating least squares, sweeping rank* r *and regularization* λ*.*

Solution. Alternate the closed-form ridge updates of Section 11.5, fixing $\boldsymbol{V}$ to solve for each user row and vice versa. The held-out root-mean-square error is

test RMSE	λ=0.01	λ=0.1	λ=1.0
$r = 5$	1.099	1.011	**0.959**
$r = 10$	1.468	1.204	1.045
$r = 20$	1.746	1.424	1.174
$r = 50$	1.937	1.693	1.333

The best configuration is low rank with strong regularization, $r = 5$ and $\lambda = 1.0$, at RMSE 0.959, beating the global-mean baseline of 1.124 by 15%. The pattern is instructive: raising the rank without regularization *worsens* the error, because the extra factors overfit the sparse observations. Low rank plus regularization is the right prior, exactly the manifold hypothesis of Chapter 11. The recovered factors give sensible recommendations: a user who rated several art-house films highly is recommended more of the same.

Topic B: Dictionary learning and denoising

Add Gaussian noise to the cameraman image, learn a dictionary of $K = 256$ *atoms on its* 8×8 *patches by K-SVD, denoise by sparse coding, and compare against a fixed DCT dictionary across noise levels.*

Solution. Run the alternating loop of Chapter 13: OMP sparse coding then the restricted rank-one atom update. On synthetic data the objective $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2$ falls monotonically over 30 iterations to ≈ 19.5 against a signal energy of ≈ 1457, explaining 98.7% of the signal energy. On the image, the learned atoms organize into oriented edges and textures (unlike the DCT's pure sinusoids), and denoising beats the fixed dictionary at every noise level up to severe:

PSNR (dB)	σ=10	σ=15	σ=25	σ=50
noisy input	28.2	24.8	20.5	15.1
fixed DCT	29.6	28.6	26.1	**21.4**
learned K-SVD	**31.1**	**29.6**	**26.6**	21.3

The learned dictionary wins by 1.5 dB at low noise, where it can tune itself to the image's textures; at $\sigma = 50$ the two tie, because the dictionary is now learned from patches too noisy to teach it anything the DCT does not already know. The gain is largest exactly where there is clean structure to exploit.

In the Problem Sets

The accompanying notebooks implement nuclear-norm completion and alternating least squares for Topic A, and K-SVD with patch-based denoising for Topic B, producing the completion phase-transition curve, the rank-versus-regularization RMSE grid, and the learned-versus-DCT denoising comparison with the atom visualizations.

CHAPTER 28

Problem Set 9: Nonnegative Matrix Factorization

The final set implements nonnegative matrix factorization (Chapter 14) by the Lee–Seung multiplicative updates and applies it to discovering topics in a real document collection, then contrasts the parts-based NMF topics against the signed components of a truncated SVD.

Problem 1: The multiplicative updates

Implement the Lee–Seung updates and verify they decrease the objective.

Solution. From a nonnegative initialization, iterate the two elementwise updates of Section 14.2,

$$\boldsymbol{X} \leftarrow \boldsymbol{X} \circ \frac{\boldsymbol{A}^\top \boldsymbol{Y}}{\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{X}}, \qquad \boldsymbol{A} \leftarrow \boldsymbol{A} \circ \frac{\boldsymbol{Y}\boldsymbol{X}^\top}{\boldsymbol{A}\boldsymbol{X}\boldsymbol{X}^\top},$$

adding a small ε to each denominator for safety. Each is two lines of array arithmetic with no projection, and both factors stay nonnegative automatically. On a synthetic 20×30 random nonnegative matrix (full rank, entries drawn uniformly on $[0, 1]$) factored at rank 5, the Frobenius error $\|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_F^2$ falls monotonically to ≈ 23.5 over 200 iterations, a steep drop in the first dozen iterations then a long flat tail at the best rank-5 approximation of a full-rank target, exactly the monotone convergence the Lee–Seung theorem (Theorem 14.3) guarantees.

Problem 2: Topic discovery and the case for nonnegativity

*Build a TF-IDF word-by-document matrix from five newsgroup categories (*4,854 *documents,* 1,500 *vocabulary terms), factor it at rank* 5*, and read off the topics. Then repeat with a truncated SVD and compare interpretability.*

Solution. Factor the term-document matrix $\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}$ with five topics. Each column of $\boldsymbol{W}$ is a topic, a nonnegative weight over words, read off by its top entries. The five NMF topics map cleanly, one-to-one, onto the five true newsgroups, recovered with no access to the labels:

topic (top words)	true category
car, cars, engine, dealer, oil, ford	autos
thanks, graphics, image, file, format, program	computer graphics
space, nasa, shuttle, launch, orbit, moon, mission	space
gun, guns, people, fbi, government, right	politics / guns
game, team, baseball, games, hit, players, runs	baseball

The truncated SVD at the same rank produces *signed* components instead of topics. Its first component is a generic "common words" direction mixing every category, and the later ones are contrasts: one category's words with positive

weight set against another's with negative weight. A negative word-weight has no meaning as a topic (a document cannot contain "minus baseball"), so the SVD components are not human-readable themes.

Why nonnegativity buys interpretability

NMF can only *add* atoms, so to reconstruct a baseball document it must switch on a topic whose top words are baseball words. The nonnegativity forces every factor to concentrate on one coherent theme. The SVD, optimizing variance with no sign constraint, spends its components on signed contrasts that cover the data efficiently but mean nothing individually. NMF trades a little reconstruction accuracy for topics a person can read, the parts-based representation of Chapter 14.

Problem 3: The effect of rank and sparsity

Sweep the rank and an optional ℓ_1 penalty on the coefficients, and watch the reconstruction error and the sparsity of the document encodings.

Solution. The reconstruction error decreases smoothly and gently as the rank climbs (from ≈ 4485 at $r = 2$ to ≈ 4174 at $r = 20$, sampled at $r \in \{2, 4, 6, 8, 10, 15, 20\}$), a steady 7% reduction with no sharp elbow: on this TF-IDF matrix extra topics keep buying a little accuracy by splitting existing themes, so the curve gives no single "correct rank" on its own. The coefficient matrix $\boldsymbol{H}$ grows sparser with rank (from $\sim 19\%$ near-zero entries at $r = 2$ to $\sim 55\%$ at $r = 20$): with more topics available, each document uses only a few. Adding an explicit ℓ_1 penalty α sparsifies $\boldsymbol{H}$ further, from 39% near-zero at $\alpha = 0$ to 81% at $\alpha = 0.5$, while leaving the top topic words unchanged and the reconstruction error essentially flat. The penalty trims weak topic memberships rather than rewriting the topics, confirming that nonnegativity already induces sparsity and the ℓ_1 term is a fine-tuning knob.

In the Problem Sets

The accompanying notebook implements the multiplicative updates from scratch, builds the TF-IDF term-document matrix for five newsgroups, factors it at rank 5, reports the converging reconstruction error and the learned topics as the top words of each atom, and contrasts them against the signed components of a truncated SVD and across the rank and ℓ_1 sweeps.

PART VIII

Practice Examinations

CHAPTER 29

Practice Examinations and Problem Bank

This part collects fully worked problems of the kind a written examination asks, sorted by the three categories of Chapter 1: *compute* something concrete, *prove* an equivalence or inequality, and *explain* a design choice. Every problem here is solved in full. A self-contained practice exam closes the chapter.

29.1 Compute

Exercise 29.1 (Spark, coherence, maximum sparsity). The matrix $\boldsymbol{A} \in \mathbb{R}^{m \times N}$ has unit-norm columns and mutual coherence $\mu = 0.1$. What sparsity levels s does the coherence guarantee certify for exact ℓ_1 recovery, and what lower bound on the spark does it give?

Solution. The Donoho–Elad bound gives $\operatorname{spark}(\boldsymbol{A}) \geq 1 + 1/\mu = 1 + 10 = 11$. Unique ℓ_0 recovery needs spark $\geq 2s + 1$, so $2s + 1 \leq 11$ gives $s \leq 5$. The ℓ_1/OMP coherence guarantee $\mu < 1/(2s - 1)$ reads $0.1 < 1/(2s - 1)$, i.e. $2s - 1 < 10$, i.e. $s \leq 5$. So coherence certifies recovery of every 5-sparse signal. ■

Exercise 29.2 (Soft thresholding). Compute $\mathcal{S}_2(\boldsymbol{v})$ for $\boldsymbol{v} = (5, -1, 3, -4, 1.5)^\top$.

Solution. Threshold each entry, $\operatorname{sgn}(v_i)\max(|v_i| - 2, 0)$: $5 \mapsto 3$, $-1 \mapsto 0$, $3 \mapsto 1$, $-4 \mapsto -2$, $1.5 \mapsto 0$. So $\mathcal{S}_2(\boldsymbol{v}) = (3, 0, 1, -2, 0)^\top$: the two small entries die, the survivors shrink by 2. ■

Exercise 29.3 (An SVD by hand). Find the singular values of $\boldsymbol{A} = \left(\begin{smallmatrix} 3 & 0 \\ 4 & 5 \end{smallmatrix}\right)$ and its spectral and nuclear norms.

Solution. $\boldsymbol{A}^\top \boldsymbol{A} = \left(\begin{smallmatrix} 25 & 20 \\ 20 & 25 \end{smallmatrix}\right)$ has eigenvalues $25 \pm 20 = 45, 5$, so $\sigma_1 = \sqrt{45} = 3\sqrt{5}$, $\sigma_2 = \sqrt{5}$. Then $\|\boldsymbol{A}\|_{2\to 2} = 3\sqrt{5} \approx 6.71$ and $\|\boldsymbol{A}\|_* = 4\sqrt{5} \approx 8.94$. Check: $\sigma_1^2 + \sigma_2^2 = 50 = \|\boldsymbol{A}\|_F^2$ and $\sigma_1\sigma_2 = 15 = |\det \boldsymbol{A}|$. ■

Exercise 29.4 (Measurement count). Roughly how many random measurements recover a 20-sparse signal in $\mathbb{R}^{10^6}$? Take the constant in $m \gtrsim Cs\log(N/s)$ to be $C \approx 2$.

Solution. $m \gtrsim 2 \cdot 20 \cdot \ln(10^6/20) = 40\ln(50000)$. Since $\ln(50000) \approx 10.8$, $m \gtrsim 430$. About four hundred measurements suffice, against a million for full sampling: the scaling $m \sim s\log(N/s)$ is the point. ■

Exercise 29.5 (One OMP step). With unit-norm atoms and residual $\boldsymbol{r}$, the correlations $|\langle \boldsymbol{a}_j, \boldsymbol{r}\rangle|$ are $(0.2, 0.9, 0.5, 0.7)$. Which atom does OMP select, and what is the new coefficient before the least-squares refit?

Solution. OMP picks the largest correlation, atom 2, and (for a fresh one-element support) sets its coefficient to $\langle \boldsymbol{a}_2, \boldsymbol{r}\rangle$. After adding atom 2 the support is refit jointly by the pseudoinverse, and the new residual is orthogonal to $\boldsymbol{a}_2$, so atom 2 cannot be reselected. ■

Exercise 29.6 (One ISTA step). With $\boldsymbol{A} = \boldsymbol{I}_2$ (so the Lipschitz constant is $L = 1$), $\boldsymbol{y} = (3, 0.5)^\top$, penalty $\lambda = 1$, and start $\boldsymbol{x}_0 = \boldsymbol{0}$, compute one iteration of ISTA.

Solution. ISTA is a gradient step then a soft threshold, $\boldsymbol{x}_1 = \mathcal{S}_{\lambda/L}\big(\boldsymbol{x}_0 - \frac{1}{L}\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{y})\big)$. The gradient step from $\mathbf{0}$ gives $\boldsymbol{A}^\top\boldsymbol{y} = (3, 0.5)^\top$, and $\mathcal{S}_1((3, 0.5)^\top) = (2, 0)^\top$: the large coordinate survives (shrunk by 1), the small one dies. One step already finds the correct one-sparse support. ■

Exercise 29.7 (A singular-value threshold). Compute $\mathcal{D}_2(\boldsymbol{M})$ for $\boldsymbol{M} = \mathrm{diag}(5, 2, 1)$ and give the resulting rank.

Solution. Singular-value thresholding soft-thresholds the singular values: $5 \mapsto 3$, $2 \mapsto 0$, $1 \mapsto 0$, so $\mathcal{D}_2(\boldsymbol{M}) = \mathrm{diag}(3, 0, 0)$, of rank 1. Thresholding the spectrum both shrinks the matrix and reduces its rank, the matrix analogue of Exercise 29.2. ■

Exercise 29.8 (Degrees of freedom of a low-rank matrix). How many free parameters does a rank-3 matrix in $\mathbb{R}^{100\times 100}$ have, and what fraction of its entries is that?

Solution. A rank-r matrix in $\mathbb{R}^{N\times N}$ has $r(2N - r)$ degrees of freedom (the singular vectors and values). Here $3(200 - 3) = 591$, against $100^2 = 10^4$ entries, about 6%. This is the floor below which no completion can succeed, and it is why matrix completion from a few percent of the entries is even conceivable (Chapter 11). ■

29.2 Prove

Exercise 29.9 (ℓ_0 is not a norm). Show $\|\cdot\|_0$ violates absolute homogeneity, so it is not a norm.

Solution. Take $\boldsymbol{x} = (1, 0, 2)$, so $\|\boldsymbol{x}\|_0 = 2$, and $\alpha = 5$. Then $5\boldsymbol{x} = (5, 0, 10)$ still has two nonzeros, $\|5\boldsymbol{x}\|_0 = 2$, but homogeneity demands $|5|\|\boldsymbol{x}\|_0 = 10 \neq 2$. Scaling never changes how many entries are nonzero, so ℓ_0 ignores magnitude and fails the axiom. ■

Exercise 29.10 (Where $\sqrt{2}-1$ comes from). Explain why the restricted isometry recovery threshold is $\delta_{2s} < \sqrt{2}-1$.

Solution. The argument that $\mathrm{RIP}_{2s} \Rightarrow \mathrm{NSP}_s$ produces a contraction $\|\boldsymbol{v}_S\|_1 \leq \frac{\sqrt{2}\,\delta_{2s}}{1-\delta_{2s}}\|\boldsymbol{v}_{S^c}\|_1$ for every null vector. The null space property requires this constant to be below 1: $\sqrt{2}\,\delta = 1 - \delta$ gives $\delta(\sqrt{2}+1) = 1$, i.e. $\delta = 1/(\sqrt{2}+1) = \sqrt{2}-1 \approx 0.414$. Below that value the contraction is strict, no null vector concentrates on s coordinates, and ℓ_1 recovery holds. (The cleaner $\delta_{2s} < \frac{1}{3}$ of Chapter 8 follows the same scheme with looser bookkeeping.) ■

Exercise 29.11 (Nuclear-norm completion is convex). Show that minimizing $\|\boldsymbol{Z}\|_*$ subject to $\mathcal{P}_\Omega(\boldsymbol{Z}) = \mathcal{P}_\Omega(\boldsymbol{X})$ is a convex problem.

Solution. The nuclear norm is a norm, hence convex. The constraint set $\{\boldsymbol{Z} : \mathcal{P}_\Omega(\boldsymbol{Z}) = \mathcal{P}_\Omega(\boldsymbol{X})\}$ is the solution set of a linear equation, an affine (hence convex) set. Minimizing a convex function over a convex set is a convex problem, so every local minimum is global. ■

Exercise 29.12 (Soft thresholding is the ℓ_1 proximal map). Prove that $\mathrm{prox}_{\lambda|\cdot|}(v) = \arg\min_x \left[\lambda|x| + \frac{1}{2}(x-v)^2\right]$ equals $\mathcal{S}_\lambda(v) = \mathrm{sgn}(v)\max(|v| - \lambda, 0)$.

Solution. The objective is convex, so the minimizer is where 0 lies in the subdifferential $\lambda\,\partial|x| + (x - v)$. If $x > 0$, then $\partial|x| = \{1\}$ and $\lambda + x - v = 0$ gives $x = v - \lambda$, valid when $v > \lambda$. If $x < 0$, then $x = v + \lambda$, valid when $v < -\lambda$. If $|v| \leq \lambda$, then $0 \in \lambda[-1, 1] - v$ is solved by $x = 0$. Collecting the three cases is exactly $\mathrm{sgn}(v)\max(|v| - \lambda, 0)$. The same computation entrywise gives the vector soft threshold, and on singular values it gives the operator of Exercise 29.7. ■

Exercise 29.13 (OMP never reselects an atom). Show that after OMP's least-squares refit on a support S, the residual is orthogonal to every selected atom, so no atom is chosen twice.

Solution. The refit sets $\hat{\boldsymbol{x}}_S = \boldsymbol{A}_S^\dagger\boldsymbol{y}$, so the residual is $\boldsymbol{r} = \boldsymbol{y} - \boldsymbol{A}_S\boldsymbol{A}_S^\dagger\boldsymbol{y} = (\boldsymbol{I} - \boldsymbol{P}_S)\boldsymbol{y}$, the projection onto the orthogonal complement of $\mathcal{R}(\boldsymbol{A}_S)$. Hence $\boldsymbol{A}_S^*\boldsymbol{r} = \boldsymbol{A}_S^*(\boldsymbol{I} - \boldsymbol{P}_S)\boldsymbol{y} = \mathbf{0}$, so $\langle \boldsymbol{a}_j, \boldsymbol{r}\rangle = 0$ for every $j \in S$. A selected atom has zero correlation with the new residual, so the next correlation scan cannot pick it again. ■

29.3 Explain

Exercise 29.14 (Structured versus Gaussian). Why would one use a partial-Fourier sensing matrix instead of a dense Gaussian one?

Solution. A Gaussian matrix is optimal in measurement count but dense: $O(mN)$ storage and time to apply. A partial-Fourier matrix has a fast transform ($O(N \log N)$) and tiny storage, corresponds to a real sensor (an MRI scanner), and, being a bounded orthonormal system with $K = 1$, keeps a near-optimal $m \gtrsim s \log^4 N$ guarantee. Structure buys practicality at almost no statistical cost. ■

Exercise 29.15 (Why nonnegativity gives parts). Explain why nonnegative matrix factorization yields parts-based representations while PCA does not.

Solution. PCA allows signed coefficients, so atoms can cancel, and the best compact basis comes out as holistic, ghostly whole-object components. NMF forbids subtraction: every atom can only add what is present, so the factorization is forced to build each datum from additive parts, and the atoms become localized, interpretable features (facial parts, topics). ■

Exercise 29.16 (FISTA versus ISTA). ISTA and FISTA cost the same per iteration, yet FISTA is preferred. Why?

Solution. Both perform a gradient step and a soft threshold each iteration, so the per-step cost is identical. ISTA converges at rate $O(1/k)$ in objective value, while FISTA adds a momentum extrapolation (a weighted combination of the last two iterates) that accelerates it to the optimal first-order rate $O(1/k^2)$. Reaching a target accuracy thus needs about the square root of the iterations, a large saving on large problems, for two extra lines of code (Chapter 7). ■

Exercise 29.17 (Why robust PCA needs incoherence). Explain why robust PCA cannot separate every low-rank-plus-sparse matrix, and what condition rules out the bad cases.

Solution. A matrix that is *both* low rank and sparse, such as $\boldsymbol{e}_1\boldsymbol{e}_1^*$ (rank one, one nonzero), could be assigned to either component, so the decomposition is not unique and no method can recover the intended one. The *incoherence* condition forbids this overlap: it requires the low-rank part's singular vectors to be spread out rather than aligned with a few coordinates, so a low-rank matrix cannot also be sparse. Under incoherence (and a uniformly random corruption support), the Candès–Li–Ma–Wright theorem guarantees exact separation (Chapter 12). ■

Exercise 29.18 (Reading the concentration index). An SRC query returns coefficient mass 0.5 on one class and 0.5 spread over the other four (five classes total). Compute the sparsity concentration index and say whether to accept the classification.

Solution. With $C = 5$ and the largest class fraction $p_{\max} = 0.5$, the index is $\mathrm{SCI} = \frac{5 \cdot 0.5 - 1}{5 - 1} = \frac{1.5}{4} = 0.375$. This is well above a typical reject threshold of 0.1 but below the near-1 value of a confident query, so the prediction is usable but not strongly concentrated; a conservative system might flag it for review. The index turns the shape of the sparse code into a single confidence number (Chapter 16). ■

29.4 A Practice Examination

Closed book conditions: one hour, calculator allowed, no electronics. Solutions follow.

1. (*Compute*) For unit-norm columns with $\mu = 0.05$, give the largest sparsity the coherence guarantee certifies and the implied spark bound.
2. (*Compute*) Evaluate $\mathcal{S}_1((4, -0.5, 2, -3)^\top)$ and $\mathcal{H}_2((4, -0.5, 2, -3)^\top)$.
3. (*Prove*) State the null space property of order s and prove that it implies every s-sparse signal is the unique (P_1) solution.
4. (*Prove*) Show that for an s-sparse vector, $\|\boldsymbol{x}\|_1 \le \sqrt{s}\,\|\boldsymbol{x}\|_2$, and identify when equality holds.
5. (*Explain*) State the recovery chain from a random matrix to exact ℓ_1 recovery, naming the condition on each arrow.

Solutions

1. spark $\ge 1 + 1/0.05 = 21$, so $2s + 1 \le 21$ gives $s \le 10$; equivalently $\mu < 1/(2s-1)$ gives $s \le 10$.

2. $\mathcal{S}_1 = (3, 0, 1, -2)^\top$ (survivors shrink by 1, the -0.5 dies); $\mathcal{H}_2 = (4, 0, 0, -3)^\top$ (keep the two largest magnitudes, 4 and 3, untouched).

3. NSP of order s: for every nonzero $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{A})$ and every $|S| \le s$, $\|\boldsymbol{v}_S\|_1 < \|\boldsymbol{v}_{S^c}\|_1$. Proof: let $\boldsymbol{x}$ be s-sparse on $S^\star$ and $\boldsymbol{z} \neq \boldsymbol{x}$ feasible; then $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{z}$ is a nonzero null vector, and NSP at $S^\star$ gives $\|\boldsymbol{x} - \boldsymbol{z}_{S^\star}\|_1 < \|\boldsymbol{z}_{(S^\star)^c}\|_1$. By the triangle inequality $\|\boldsymbol{x}\|_1 \le \|\boldsymbol{x} - \boldsymbol{z}_{S^\star}\|_1 + \|\boldsymbol{z}_{S^\star}\|_1 < \|\boldsymbol{z}_{(S^\star)^c}\|_1 + \|\boldsymbol{z}_{S^\star}\|_1 = \|\boldsymbol{z}\|_1$, so $\boldsymbol{x}$ is the unique minimizer (full version in Chapter 4).

4. Let $\boldsymbol{\sigma}$ have unit modulus on $\operatorname{supp}(\boldsymbol{x})$ and zero off it; then $\|\boldsymbol{x}\|_1 = \langle \boldsymbol{\sigma}, \boldsymbol{x} \rangle \le \|\boldsymbol{\sigma}\|_2 \|\boldsymbol{x}\|_2 = \sqrt{s}\,\|\boldsymbol{x}\|_2$ by Cauchy–Schwarz, with equality when all nonzero entries have equal magnitude.

5. random $\boldsymbol{A} \xrightarrow{m \gtrsim s \log(N/s),\ \text{whp}} \mathrm{RIP}_{2s}(\delta < \tfrac{1}{3}) \Rightarrow \mathrm{NSP}_s \Leftrightarrow (\mathrm{P}_1)$ recovers every s-sparse $\boldsymbol{x}$. The first arrow is the random-matrix theorem, the second is Theorem 8.4, the last is the equivalence Theorem 4.10. ∎

29.5 A Second Practice Examination

This exam covers the matrix half of the course. Same conditions; solutions follow.

1. (*Compute*) Find the singular values of $\boldsymbol{A} = \left(\begin{smallmatrix} 2 & 0 \\ 0 & 3 \\ 0 & 4 \end{smallmatrix}\right)$ and its nuclear norm.
2. (*Compute*) Apply singular-value thresholding $\mathcal{D}_3$ to a matrix with singular values $(8, 4, 2)$, and give the rank of the result.
3. (*Compute*) A robust-PCA program uses $\lambda = 1/\sqrt{n^*}$ on a 400×900 matrix. What is λ?
4. (*Prove*) Show the nuclear norm equals $\|\boldsymbol{\sigma}\|_1$ and the rank equals $\|\boldsymbol{\sigma}\|_0$, where $\boldsymbol{\sigma}$ is the singular-value vector.
5. (*Explain*) In one sentence each, say what structure NMF, K-SVD, and PCA each impose on $\boldsymbol{Y} \approx \boldsymbol{A}\boldsymbol{X}$.

Solutions

1. $\boldsymbol{A}^\top \boldsymbol{A} = \left(\begin{smallmatrix} 4 & 0 \\ 0 & 25 \end{smallmatrix}\right)$, so the singular values are $\sqrt{4} = 2$ and $\sqrt{25} = 5$ (the second column has norm $\sqrt{9+16} = 5$). The nuclear norm is $2 + 5 = 7$, and $\|\boldsymbol{A}\|_F = \sqrt{4 + 25} = \sqrt{29}$ checks against $\sqrt{2^2 + 5^2}$.

2. Soft-threshold the singular values by 3: $8 \mapsto 5$, $4 \mapsto 1$, $2 \mapsto 0$, giving spectrum $(5, 1, 0)$, rank 2.

3. $n^* = \max(400, 900) = 900$, so $\lambda = 1/\sqrt{900} = 1/30 \approx 0.033$.

4. By definition the nuclear norm is $\|\boldsymbol{X}\|_* = \sum_i \sigma_i = \|\boldsymbol{\sigma}\|_1$, and the rank is the number of nonzero singular values, $\operatorname{rank}(\boldsymbol{X}) = |\{i : \sigma_i \neq 0\}| = \|\boldsymbol{\sigma}\|_0$. These are the ℓ_1 and ℓ_0 of the spectrum, which is the vector–matrix dictionary of Chapter 11.

5. NMF requires both factors nonnegative, forcing additive parts; K-SVD requires the code columns sparse, learning an overcomplete dictionary; PCA requires the atoms orthonormal, giving the most compact signed basis. Same factorization, three different constraints. ■

CHAPTER 30

A Full Practice Examination, Worked in Detail

This chapter works a complete practice examination end to end, in the same fully-worked, diagram-rich style as the rest of the book. It comes in three parts: a set of True/False statements with rigorous justifications, a set of long questions spanning the vector and matrix theory, and a pair of proofs. Every solution restates the question, names each theorem it invokes, shows every algebraic step, and boxes the answer. Read it as a self-test: cover the solution, attempt the problem, then check.

30.1 Part I: True / False

*For each statement, state **True** or **False** with a rigorous one- to three-sentence justification, or a clean counterexample. The verdict alone is worth little; the justification is the point.*

30.1.1 Problem 1: ℓ_0 uniqueness $\not\Rightarrow$ ℓ_1 uniqueness

Statement

If $\boldsymbol{x}^\star$ is the unique minimizer of $\min\|\boldsymbol{z}\|_0$ s.t. $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}$, then $\boldsymbol{x}^\star$ is necessarily also the unique minimizer of $\min\|\boldsymbol{z}\|_1$ s.t. $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}$.

Verdict. FALSE.

Reasoning. The two uniqueness conditions are governed by different invariants of $\boldsymbol{A}$:

- Unique (P_0) for every k-sparse signal $\iff$ $\operatorname{spark}(\boldsymbol{A}) > 2k$ (Donoho–Elad).
- Unique (P_1) for every k-sparse signal $\iff$ NSP of order k (Ch. 4 (Spark/NSP) in our textbook).

NSP $\Rightarrow$ spark $> 2k$, but the converse is *false*, so a matrix can be ℓ_0-uniquely decoding yet ℓ_1-non-uniquely decoding.

Concrete counterexample. Take

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \in \mathbb{R}^{2\times 3}, \qquad \boldsymbol{x}^\star = \boldsymbol{e}_2 = (0,1,0)^\top, \qquad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^\star = (1,1)^\top.$$

The kernel is the line $\{c\,(1,-2,1)^\top : c \in \mathbb{R}\}$, so the smallest support of a nonzero kernel vector is 3 and $\operatorname{spark}(\boldsymbol{A}) = 3 > 2(1) = 2k$. Hence $\boldsymbol{x}^\star$ is the unique (P_0) minimizer.

For (P_1), parameterize $\boldsymbol{z} = \boldsymbol{x}^\star + c\,(1,-2,1)^\top = (c,\, 1-2c,\, c)^\top$:

$$\|\boldsymbol{z}\|_1 = |c| + |1-2c| + |c| = 2|c| + |1-2c|.$$

For $c \in [0, \tfrac{1}{2}]$ the right side equals $2c + (1-2c) = 1 = \|\boldsymbol{x}^\star\|_1$. Thus a whole interval of distinct vectors achieves the same ℓ_1 value, so the (P_1) minimizer is *not* unique. The NSP test $\|\boldsymbol{v}_S\|_1 < \|\boldsymbol{v}_{S^c}\|_1$ at $S = \{2\}$ with $\boldsymbol{v} = (1,-2,1)^\top$ reads $2 < 2$, which fails (equality), exactly as predicted.

Final answer: FALSE. spark $> 2k \not\Rightarrow \mathrm{NSP}_k$, so ℓ_0 uniqueness is strictly weaker than ℓ_1 uniqueness.

30.1.2 Problem 2: $\delta_{2k} = 0.6$ and basis pursuit

Statement

If $\boldsymbol{A}$ satisfies the RIP of order $2k$ with $\delta_{2k} = 0.6$, then basis pursuit recovers every k-sparse vector exactly.

Verdict. FALSE.

Reasoning. The standard sufficient threshold for basis pursuit recovery in our textbook (the RIP recovery theorem, Candès) is

$$\delta_{2k} < \sqrt{2} - 1 \approx 0.4142.$$

With $\delta_{2k} = 0.6$ this sufficient condition is *violated*, so the theorem does not apply. The value 0.6 falls in the gap $(\sqrt{2}-1,\ 1/\sqrt{2}) \approx (0.414,\ 0.707)$: above the recovery threshold $\sqrt{2}-1$, but below $1/\sqrt{2}$, the level at which Davies and Gribonval (2009) exhibited matrices for which (P_1) provably fails on some k-sparse signal. In this gap the recovery theorem gives no guarantee, and no general counterexample is known either, so $\delta_{2k} = 0.6$ on its own does not certify recovery.

Sanity check. The condition $\delta_{2k} < 1$ alone is enough to make distinct k-sparse vectors yield distinct measurements (injectivity on Σ_k), but is *not* enough for the ℓ_1-relaxation to find the truth; the latter needs the geometric margin $\delta_{2k} < \sqrt{2} - 1$ to convert RIP into NSP via the contraction $\frac{\sqrt{2}\,\delta_{2k}}{1-\delta_{2k}} < 1$.

Final answer: FALSE. Below the $\sqrt{2} - 1$ threshold, recovery is guaranteed; above it (here 0.6), counterexamples exist.

30.1.3 Problem 3: K-SVD does not update all atoms at once

Statement

In K-SVD, the dictionary-update stage updates all atoms at once by taking a single SVD of the full residual matrix $\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}$.

Verdict. FALSE.

Reasoning. K-SVD updates atoms *one at a time*. The defining feature of K-SVD is the per-atom rank-one update of the *partial* residual:

1. For each atom index k, form the error matrix without atom k,
$$\boldsymbol{E}_k = \boldsymbol{Y} - \sum_{j \neq k} \boldsymbol{d}_j \, (\boldsymbol{x}^j)^\top.$$
2. Restrict to the active set $\omega_k = \{i : X_{k,i} \neq 0\}$, giving $\boldsymbol{E}_k^R = \boldsymbol{E}_k(:, \omega_k)$.
3. Take the top SVD triple of $\boldsymbol{E}_k^R = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, and set $\boldsymbol{d}_k \leftarrow \boldsymbol{u}_1$ and the restricted coefficient row $\boldsymbol{x}_R^k \leftarrow \sigma_1 \boldsymbol{v}_1$.

This is K *separate* rank-one SVDs (one per atom), *not* a single SVD of $\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}$. The whole-dictionary one-shot least-squares update $\boldsymbol{D} \leftarrow \boldsymbol{Y}\boldsymbol{X}^\dagger$ describes the *MOD* algorithm, not K-SVD.

Why the distinction matters. K-SVD's restriction to ω_k preserves the sparsity pattern fixed by OMP, while a single SVD of the full $\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}$ would yield a dense update that breaks the support constraints (the K-SVD chapter).

Final answer: FALSE. K-SVD performs K separate rank-one SVDs of restricted residuals, one atom at a time.

30.1.4 Problem 4: The Tropp coherence bound is sharp

Statement

OMP run for k steps recovers every k-sparse signal whenever $\mu < \frac{1}{2k-1}$, and this cannot in general be relaxed to $\mu < \frac{1}{k}$.

Verdict. TRUE.

Reasoning, sufficiency. The Tropp (2004) coherence guarantee asserts that if $\mu(\boldsymbol{A}) < \frac{1}{2k-1}$ then OMP (and basis pursuit) recovers every k-sparse signal exactly. The mechanism: at every iteration the on-support correlation strictly dominates every off-support correlation, so OMP selects a true atom each step. (See Tropp's ERC theorem in our textbook.)

Reasoning, tightness. The bound *cannot* be uniformly relaxed to $\mu < \frac{1}{k}$ because the gap

$$\frac{1}{k} - \frac{1}{2k-1} = \frac{k-1}{k(2k-1)} > 0 \quad \text{for } k \geq 2$$

is occupied by sparse-signal instances on which OMP demonstrably fails.

Counterexample sketch ($k = 2$, $\mu = \frac{1}{3}$). The Donoho–Elad pair from the textbook: unit columns $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$ with $\langle \boldsymbol{a}_1, \boldsymbol{a}_2 \rangle = \langle \boldsymbol{a}_1, \boldsymbol{a}_3 \rangle = \langle \boldsymbol{a}_2, \boldsymbol{a}_3 \rangle = \frac{1}{3}$, i.e. an equiangular triple. Take $\boldsymbol{x} = (1, 1, 0)^\top$, so $\boldsymbol{y} = \boldsymbol{a}_1 + \boldsymbol{a}_2$.

$$\langle \boldsymbol{a}_1, \boldsymbol{y} \rangle = 1 + \tfrac{1}{3} = \tfrac{4}{3}, \quad \langle \boldsymbol{a}_2, \boldsymbol{y} \rangle = \tfrac{1}{3} + 1 = \tfrac{4}{3}, \quad \langle \boldsymbol{a}_3, \boldsymbol{y} \rangle = \tfrac{1}{3} + \tfrac{1}{3} = \tfrac{2}{3}.$$

Note $\mu = \frac{1}{3}$ here satisfies $\frac{1}{3} < \frac{1}{k} = \frac{1}{2}$, so the relaxed bound holds. Both on-support correlations *tie* at $\frac{4}{3}$ (no strict margin), so a tie-breaking rule can leave OMP stuck on a wrong support after one step in the presence of any infinitesimal perturbation that lifts $\boldsymbol{a}_3$'s score. With the rigorous Tropp bound $\mu < \frac{1}{2k-1} = \frac{1}{3}$ (strict), this exact tie cannot occur.

Final answer: TRUE. $\mu < \frac{1}{2k-1}$ is sharp; $\mu < \frac{1}{k}$ is not enough in general.

30.1.5 Problem 5: NSP is necessary *and* sufficient

Statement

The Null Space Property of order k is sufficient but not necessary for basis pursuit to recover all k-sparse vectors exactly.

Verdict. FALSE.

Reasoning. The NSP *characterizes* (P_1) recovery: it is both sufficient *and* necessary. The textbook theorem (the NSP equivalence theorem) reads:

$$\text{(A) } (\mathrm{P}_1) \text{ recovers every } k\text{-sparse } \boldsymbol{x} \quad \Longleftrightarrow \quad \text{(B) } \boldsymbol{A} \text{ has NSP of order } k.$$

Proof sketch of (A) $\Rightarrow$ (B) (the part that makes NSP *necessary*). Suppose recovery holds. Take any nonzero $\boldsymbol{v} \in \ker \boldsymbol{A}$ and any S with $|S| \leq k$. Then $\boldsymbol{A}\boldsymbol{v}_S = -\boldsymbol{A}\boldsymbol{v}_{S^c}$, so $\boldsymbol{v}_S$ and $-\boldsymbol{v}_{S^c}$ are two feasible vectors for the measurement $\boldsymbol{A}\boldsymbol{v}_S$. The first is k-sparse on S; by recovery it is the unique minimizer of (P_1), so its ℓ_1 norm is *strictly* smaller than the competitor's:

$$\|\boldsymbol{v}_S\|_1 < \|-\boldsymbol{v}_{S^c}\|_1 = \|\boldsymbol{v}_{S^c}\|_1.$$

This is exactly NSP of order k. Hence the statement that NSP is "not necessary" is wrong.

Final answer: FALSE. NSP_k is sufficient *and* necessary for uniform (P_1) recovery of k-sparse signals.

30.1.6 Problem 6: $\delta_{2k} < 1$ gives injectivity but not (P_1) recovery

Statement

*If $\boldsymbol{A}$ satisfies RIP of order $2k$ with $\delta_{2k} < 1$, then distinct k-sparse vectors always yield distinct measurements; but $\delta_{2k} < 1$ alone does **not** guarantee ℓ_1-recovery of every k-sparse vector.*

Verdict. TRUE.

First claim (distinct measurements). Let $\boldsymbol{x}, \boldsymbol{x}'$ be distinct k-sparse vectors. Their difference $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{x}' \neq \boldsymbol{0}$ is at most $2k$-sparse. RIP gives

$$\|\boldsymbol{A}\boldsymbol{v}\|_2^2 \;\geq\; (1 - \delta_{2k})\|\boldsymbol{v}\|_2^2 \;>\; 0,$$

so $\boldsymbol{A}\boldsymbol{x} \neq \boldsymbol{A}\boldsymbol{x}'$. ✓

Second claim (not enough for (P_1)). For BP recovery via the textbook chain RIP $\Rightarrow$ NSP, the proof requires a *contraction* of the form $\|\boldsymbol{v}_S\|_1 < \rho\,\|\boldsymbol{v}_{S^c}\|_1$ with $\rho < 1$, where $\rho = \frac{\sqrt{2}\,\delta_{2k}}{1-\delta_{2k}}$. Setting $\rho < 1$ forces $\delta_{2k} < \sqrt{2} - 1 \approx 0.414$. A constant δ_{2k} above this threshold but still below 1 (e.g. 0.7, near the $1/\sqrt{2}$ reach of their construction) keeps injectivity but breaks the contraction, and explicit counterexamples (Davies–Gribonval, 2009) show (P_1) can fail.

Final answer: TRUE. $\delta_{2k} < 1$ ensures injectivity of $\boldsymbol{A}$ on Σ_{2k}; the BP threshold needs $\delta_{2k} < \sqrt{2} - 1$.

30.1.7 Problem 7: LASSO uniqueness can fail

Statement

The LASSO objective $\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$ always has a unique minimizer $\hat{\boldsymbol{x}}$ for every $\lambda > 0$.

Verdict. FALSE.

Reasoning. The objective is convex but *not* strictly convex whenever $\ker \boldsymbol{A} \neq \{\boldsymbol{0}\}$ (e.g. underdetermined systems): the quadratic term has Hessian $\boldsymbol{A}^\top\boldsymbol{A} \succeq \boldsymbol{0}$, which is rank-deficient in the CS regime, and ℓ_1 is convex but not strictly convex. Strict convexity can fail, and then multiple optima exist.

Concrete counterexample. Take $\boldsymbol{A} = (1, 1) \in \mathbb{R}^{1\times 2}$, $\boldsymbol{y} = 1$, $\lambda = 0.1$. Write $\boldsymbol{x} = (x_1, x_2)^\top$. The objective is

$$F(\boldsymbol{x}) = \tfrac{1}{2}(x_1 + x_2 - 1)^2 + 0.1(|x_1| + |x_2|).$$

Set $s = x_1 + x_2$. Among nonnegative pairs with fixed $s \geq 0$, $|x_1| + |x_2| = s$ regardless of the split. So $F = \frac{1}{2}(s-1)^2 + 0.1s$ depends only on s; minimize over s: $s^\star = 1 - 0.1 = 0.9$. Any nonneg pair with $x_1 + x_2 = 0.9$ is a minimizer: $(0.9, 0)$, $(0.45, 0.45)$, $(0.3, 0.6)$, etc., a one-parameter family.

When uniqueness is guaranteed. Sufficient conditions include (i) $\boldsymbol{A}$ has full column rank (strict convexity of the quadratic), or (ii) the active support's columns are linearly independent *and* a strict complementary slackness condition holds at the optimum.

Final answer: FALSE. When $\ker \boldsymbol{A} \neq \{\boldsymbol{0}\}$, the LASSO can have a continuum of optima.

30.1.8 Problem 8: Nuclear norm is ℓ_1 on singular values

Statement

Nuclear-norm minimization is to low-rank matrix recovery what ℓ_1 minimization is to sparse vector recovery, because the nuclear norm is the ℓ_1 norm of the vector of singular values.

Verdict. TRUE.

Reasoning. Let $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n)$ collect the singular values of $\boldsymbol{X}$. The dictionary is

$$\operatorname{rank}(\boldsymbol{X}) = \|\boldsymbol{\sigma}\|_0, \qquad \|\boldsymbol{X}\|_* = \sum_i \sigma_i = \|\boldsymbol{\sigma}\|_1, \qquad \|\boldsymbol{X}\|_F = \|\boldsymbol{\sigma}\|_2, \qquad \|\boldsymbol{X}\|_{2\to 2} = \|\boldsymbol{\sigma}\|_\infty,$$

so the whole matrix theory mirrors the vector theory applied to $\boldsymbol{\sigma}$. Just as ℓ_1 is the tightest convex relaxation of ℓ_0, the nuclear norm is the tightest convex relaxation of rank (it is the convex envelope of rank on the spectral-norm ball $\{\boldsymbol{X} : \|\boldsymbol{X}\|_{2\to 2} \le 1\}$).

Parallel theorems.

- Vector NSP $\Leftrightarrow$ (P_1) recovery $\quad\longleftrightarrow\quad$ Matrix NSP $\Leftrightarrow$ (PM_1) recovery (the matrix NSP theorem).
- ℓ_1 prox = soft thresholding $\quad\longleftrightarrow\quad$ nuclear-norm prox = singular-value thresholding $\mathcal{D}_\tau$ (the SVT definition).
- Vector sample complexity $m \gtrsim s \log(N/s)$ $\quad\longleftrightarrow\quad$ matrix completion $m \gtrsim nr\,\mathrm{polylog}(n)$ (Candès–Recht).

Final answer: TRUE. Matrix story = vector story applied to $\boldsymbol{\sigma}(\boldsymbol{X})$.

30.2 Part II: Long Questions

30.2.1 Long Question 1: Orthogonal Matching Pursuit

Setup

The dictionary has unit-norm columns

$$\boldsymbol{a}_1 = \begin{pmatrix}1\\0\\0\end{pmatrix},\ \boldsymbol{a}_2 = \begin{pmatrix}0\\1\\0\end{pmatrix},\ \boldsymbol{a}_3 = \begin{pmatrix}3/5\\4/5\\0\end{pmatrix},\ \boldsymbol{a}_4 = \begin{pmatrix}0\\0\\1\end{pmatrix},\quad \boldsymbol{y} = \begin{pmatrix}2\\0\\1\end{pmatrix}.$$

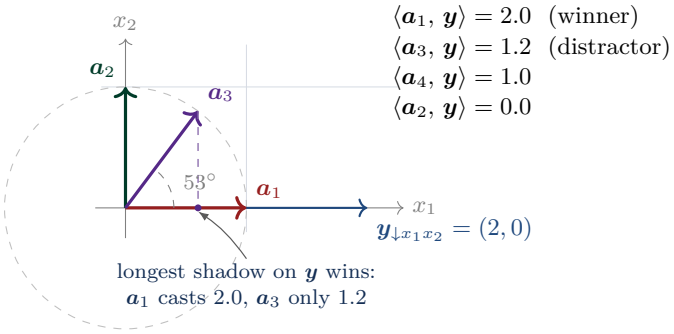
At each step OMP selects $j^\star = \arg\max_j |\langle \boldsymbol{a}_j, \boldsymbol{r}\rangle|$, augments the support, and subtracts the projection of $\boldsymbol{y}$ onto the chosen columns. Start $\boldsymbol{r} = \boldsymbol{y}$.

(a) Correlations and first selection. Compute

$$\begin{aligned}
\langle \boldsymbol{a}_1, \boldsymbol{y}\rangle &= 1\cdot 2 + 0\cdot 0 + 0\cdot 1 = \mathbf{2.000},\\
\langle \boldsymbol{a}_2, \boldsymbol{y}\rangle &= 0\cdot 2 + 1\cdot 0 + 0\cdot 1 = \mathbf{0.000},\\
\langle \boldsymbol{a}_3, \boldsymbol{y}\rangle &= \tfrac{3}{5}\cdot 2 + \tfrac{4}{5}\cdot 0 + 0\cdot 1 = \tfrac{6}{5} = \mathbf{1.200},\\
\langle \boldsymbol{a}_4, \boldsymbol{y}\rangle &= 0\cdot 2 + 0\cdot 0 + 1\cdot 1 = \mathbf{1.000}.
\end{aligned}$$

Sorted by magnitude: $|2| > |1.2| > |1| > |0|$, so the winner is $\boxed{\boldsymbol{a}_1}$ and the support becomes $S_1 = \{1\}$.

Why $\boldsymbol{a}_3$ is a *distractor*, not the winner. The atom $\boldsymbol{a}_3 = \frac{1}{5}(3,4,0)^\top$ shares its first coordinate with $\boldsymbol{y}$ and is only $\approx 53^\circ$ away from $\boldsymbol{a}_1$ in the x_1x_2-plane; its correlation 1.2 is significant. But OMP's selection rule is arg max *of magnitude*, and $2 > 1.2$ strictly. The natural worry "could $\boldsymbol{a}_3$ inject a spurious atom into the recovery?" is answered *no*: $\boldsymbol{a}_1$ beats it on the very first scan, and after the update the residual no longer carries the x_1-component that fed $\boldsymbol{a}_3$'s score.



The figure plots $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$ and the x_1x_2-shadow of $\boldsymbol{y}$. $\boldsymbol{a}_1$ lies exactly along that shadow; $\boldsymbol{a}_3$ leans toward it but at an angle, giving a smaller projection.

(b) Update the residual. With $S_1 = \{1\}$ and $\|\boldsymbol{a}_1\|_2 = 1$, the least-squares coefficient is $\hat{x}_1 = \langle \boldsymbol{a}_1, \boldsymbol{y}\rangle = 2$, and

$$\boldsymbol{r}_1 \;=\; \boldsymbol{y} - \hat{x}_1\,\boldsymbol{a}_1 \;=\; \begin{pmatrix}2\\0\\1\end{pmatrix} - 2\begin{pmatrix}1\\0\\0\end{pmatrix} \;=\; \boxed{\begin{pmatrix}0\\0\\1\end{pmatrix}}.$$

Orthogonality check (free): $\langle \boldsymbol{a}_1, \boldsymbol{r}_1\rangle = 0 \cdot 1 = 0$ ✓.

(c) Step 2: second atom, support, recovered $\boldsymbol{x}$. Compute correlations with $\boldsymbol{r}_1 = (0,0,1)^\top$:

$$\langle \boldsymbol{a}_1, \boldsymbol{r}_1 \rangle = 0, \quad \langle \boldsymbol{a}_2, \boldsymbol{r}_1 \rangle = 0, \quad \langle \boldsymbol{a}_3, \boldsymbol{r}_1 \rangle = \tfrac{3}{5}\cdot 0 + \tfrac{4}{5}\cdot 0 + 0\cdot 1 = 0, \quad \langle \boldsymbol{a}_4, \boldsymbol{r}_1 \rangle = 1.$$

The unique nonzero correlation is at index 4, so $S_2 = \{1, 4\}$.

Since $\boldsymbol{a}_1 = \boldsymbol{e}_1$ and $\boldsymbol{a}_4 = \boldsymbol{e}_3$ are orthogonal unit vectors, the LS refit is the diagonal

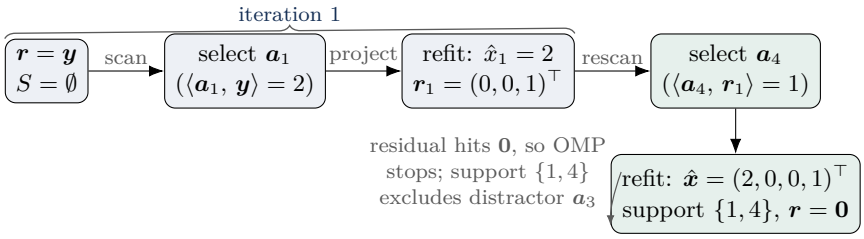
$$\hat{\boldsymbol{x}}_{S_2} = \big(\langle \boldsymbol{a}_1, \boldsymbol{y} \rangle, \langle \boldsymbol{a}_4, \boldsymbol{y} \rangle \big)^\top = (2, 1)^\top.$$

Reassembling on the four-dimensional ambient, $\boxed{\hat{\boldsymbol{x}} = (2, 0, 0, 1)^\top}$ with final support $\{1, 4\}$.

Verification. The reconstruction is

$$\boldsymbol{A}\hat{\boldsymbol{x}} = 2\boldsymbol{a}_1 + 1\cdot\boldsymbol{a}_4 = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \boldsymbol{y} \checkmark,$$

the final residual is $\mathbf{0}$, and the recovered support $\{1, 4\}$ does *not* contain the distractor $\boldsymbol{a}_3$, as expected.



30.2.2 Long Question 2: The ℓ_1 "Diamond"

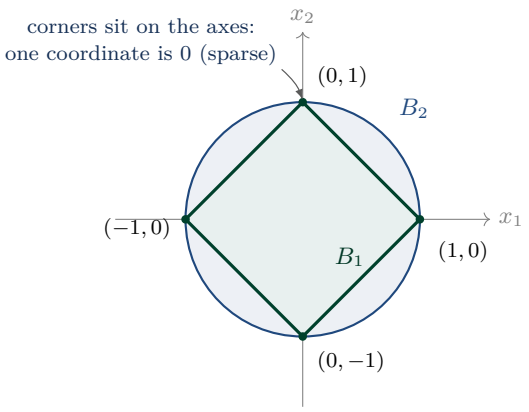
Setup

(a) In $\mathbb{R}^2$, draw the unit balls $B_1 = \{\boldsymbol{x} : \|\boldsymbol{x}\|_1 \le 1\}$ and $B_2 = \{\boldsymbol{x} : \|\boldsymbol{x}\|_2 \le 1\}$ on the same axes, labeling the four vertices of B_1.
(b) Draw an affine solution set (a line) $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}\}$ and depict $\min\|\boldsymbol{x}\|_1$ s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ as inflating B_1 until it first meets the line. Explain why first contact typically occurs at a vertex, why a vertex has a zero coordinate, and contrast with B_2.
(c) State precisely why we relax $\min\|\boldsymbol{x}\|_0$ to $\min\|\boldsymbol{x}\|_1$ rather than solving the ℓ_0 problem directly.

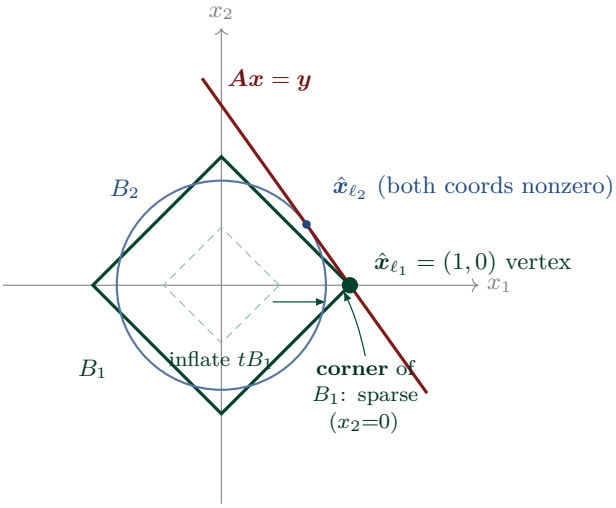
(a) The two unit balls and the vertices. B_1 is the convex hull of $\pm\boldsymbol{e}_1, \pm\boldsymbol{e}_2$, a diamond/cross-polytope with the four vertices

$$(1, 0), \quad (0, 1), \quad (-1, 0), \quad (0, -1).$$

B_2 is the closed unit disk.



(b) Inflating B_1 vs. B_2 to meet $\boldsymbol{Ax} = \boldsymbol{y}$. Pick a generic affine line. As we scale tB_1 with $t \uparrow$, the boundary expands outward; the optimum $\hat{\boldsymbol{x}} = \arg\min\{\|\boldsymbol{x}\|_1 : \boldsymbol{Ax} = \boldsymbol{y}\}$ is the first point of contact between the line and tB_1.



Why vertex contact. The four vertices of B_1 "poke out" in the axis directions: from the origin, the diamond reaches distance 1 at $\pm\boldsymbol{e}_1, \pm\boldsymbol{e}_2$ but only $\frac{1}{\sqrt{2}} \approx 0.707$ along its edges (since the midpoint of an edge is $\frac{1}{2}(\pm e_1 \pm e_2)$ with ℓ_1-norm 1 but ℓ_2-norm $\frac{1}{\sqrt{2}}$). So a generic line that is not exactly parallel to one of the four edges hits a *vertex* first as tB_1 inflates.

Why a vertex is sparse. Each vertex $\pm\boldsymbol{e}_i$ has exactly one nonzero coordinate. So a vertex minimizer has $\|\boldsymbol{x}\|_0 \leq 1$, automatically sparse, with no sparsity constraint imposed.

Contrast with B_2. The disk has no vertices: it is smooth and rotationally symmetric. A generic line touches it at a tangent point with no special coordinate structure; both x_1 and x_2 are nonzero, so the ℓ_2-minimum is *dense*.

Higher dimensions. In $\mathbb{R}^N$, B_1 is the *cross-polytope* with $2N$ vertices at $\pm\boldsymbol{e}_i$ and many lower-dimensional faces aligned with small index sets. A generic $(N-m)$-dimensional affine set first meets B_1 at a low-dimensional face, which lives in a few coordinates, giving a few-nonzero minimizer.

(c) Why ℓ_1 relaxation. Two clean reasons:

1. **Computational.** $\min\|\boldsymbol{x}\|_0$ s.t. $\boldsymbol{Ax}=\boldsymbol{y}$ is NP-hard (the NP-hardness section); the natural method is to enumerate $\binom{N}{s}$ supports, infeasible already for $N=10^3$, $s=10$. Relaxing to $\min\|\boldsymbol{x}\|_1$ gives a *linear program* (Ch. 5) solvable in polynomial time by simplex or interior-point methods.
2. **Geometric tightness.** Among all convex norms on $\mathbb{R}^N$, ℓ_1 is the largest one whose unit ball has its vertices on the coordinate axes. Any ℓ_q with $q>1$ bulges out and loses the sparse-corner property; ℓ_q with $q<1$ remains sparsity-seeking but is non-convex, restoring NP-hardness. So ℓ_1 is the *tightest convex surrogate* of ℓ_0.

Under conditions like NSP, RIP, or low coherence, the two minima coincide and we recover the truth at polynomial cost (the NSP and RIP recovery theorems).

30.2.3 Long Question 3: Soft-thresholding and ISTA

Setup

Consider $\operatorname{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{z})=\arg\min_{\boldsymbol{x}}\ \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2+\lambda\|\boldsymbol{x}\|_1,\ \ \lambda>0$.
(a) Show the problem decouples across coordinates.
(b) Derive $x_i=\operatorname{sgn}(z_i)\max(|z_i|-\lambda,0)$.
(c) ISTA minimizes $F(\boldsymbol{x})=\frac{1}{2}\|\boldsymbol{Ax}-\boldsymbol{y}\|_2^2+\lambda\|\boldsymbol{x}\|_1$ *by proximal gradient. Write one iteration with step* $1/L$ *(give the gradient of the smooth part) and state the* L *guaranteeing convergence and why.*

(a) Coordinate decoupling. Both the quadratic penalty and the ℓ_1 penalty are sums over coordinates:

$$\tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2+\lambda\|\boldsymbol{x}\|_1=\tfrac{1}{2}\sum_{i=1}^{N}(x_i-z_i)^2+\lambda\sum_{i=1}^{N}|x_i|=\sum_{i=1}^{N}\Big[\tfrac{1}{2}(x_i-z_i)^2+\lambda|x_i|\Big].$$

There is no cross-term coupling x_i to x_j for $i\neq j$; minimizing the sum is the same as minimizing each term independently:

$$x_i^\star=\arg\min_{x_i\in\mathbb{R}}\ \phi_i(x_i),\qquad \phi_i(x)=\tfrac{1}{2}(x-z_i)^2+\lambda|x|.$$

So the N-dimensional prox reduces to N *scalar* problems, ϕ_i depending only on the corresponding pair (x_i,z_i). ✓

(b) Derivation of the soft-threshold formula. Fix one coordinate and drop the subscript. Solve

$$x^\star\ =\ \arg\min_{x\in\mathbb{R}}\ \phi(x)\ =\ \tfrac{1}{2}(x-z)^2+\lambda|x|.$$

ϕ is strictly convex (curvature $1>0$ plus a convex penalty), so $x^\star$ is unique. The subdifferential at x is

$$\partial\phi(x)=(x-z)+\lambda\,\partial|x|=\begin{cases}(x-z)+\lambda, & x>0,\\ (x-z)+\lambda\,[-1,+1], & x=0,\\ (x-z)-\lambda, & x<0.\end{cases}$$

Optimality is $0\in\partial\phi(x^\star)$. Three cases.

Case 1: $z>\lambda$. Guess $x^\star>0$, so $0=(x^\star-z)+\lambda\Rightarrow x^\star=z-\lambda$. Consistency: $z-\lambda>0$ iff $z>\lambda$, satisfied. So $x^\star=z-\lambda$.

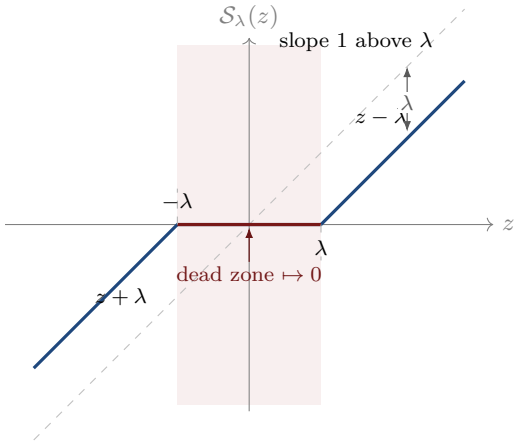
Case 2: $z<-\lambda$. Guess $x^\star<0$, so $0=(x^\star-z)-\lambda\Rightarrow x^\star=z+\lambda$. Consistency: $z+\lambda<0$ iff $z<-\lambda$, satisfied. So $x^\star=z+\lambda$.

Case 3: $|z|\le\lambda$. Test $x^\star=0$: the subdifferential at 0 is $\{(0-z)+\lambda u:u\in[-1,+1]\}=[-z-\lambda,-z+\lambda]$. This interval contains 0 iff $-z-\lambda\le 0\le -z+\lambda$, i.e. $-\lambda\le z\le\lambda$. So $x^\star=0$ when $|z|\le\lambda$.

Combining the three cases:

$$\boxed{x^\star=\operatorname{sgn}(z)\ \max\big(|z|-\lambda,\,0\big)=\begin{cases}z-\lambda, & z>\lambda,\\ 0, & |z|\le\lambda,\\ z+\lambda, & z<-\lambda.\end{cases}}$$

This is exactly the soft-threshold $\mathcal{S}_\lambda(z)$. Applied coordinatewise to $\boldsymbol{z}$, $\operatorname{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{z}) = \mathcal{S}_\lambda(\boldsymbol{z})$. ✓



The dashed diagonal is the identity (no threshold); soft-thresholding shifts the diagonal down by λ on the right, up by λ on the left, and flattens to zero in $[-\lambda, \lambda]$. Continuous (no jump), *killing* small magnitudes and *shrinking* survivors by λ.

(c) One ISTA iteration with step $1/L$. Write $F = g + h$ with smooth part $g(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ and nonsmooth part $h(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$. The gradient of g is

$$\nabla g(\boldsymbol{x}) = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}),$$

since $\nabla_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 = \nabla_{\boldsymbol{x}} \frac{1}{2}(\boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{y}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{y}) = \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top \boldsymbol{y}$.

Proximal-gradient step:

$$\boldsymbol{x}^{k+1} \;=\; \operatorname{prox}_{(\lambda/L)\,\|\cdot\|_1}\Big(\boldsymbol{x}^k - \tfrac{1}{L}\,\nabla g(\boldsymbol{x}^k)\Big) \;=\; \mathcal{S}_{\lambda/L}\Big(\boldsymbol{x}^k - \tfrac{1}{L}\,\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y})\Big).$$

That is, one gradient descent step on g at stepsize $1/L$, followed by coordinatewise soft-thresholding at level λ/L.

Choice of L. The descent lemma asks for a Lipschitz constant of ∇g:

$$\|\nabla g(\boldsymbol{u}) - \nabla g(\boldsymbol{v})\|_2 = \|\boldsymbol{A}^\top \boldsymbol{A}(\boldsymbol{u} - \boldsymbol{v})\|_2 \le \|\boldsymbol{A}^\top \boldsymbol{A}\|_{2\to 2}\,\|\boldsymbol{u} - \boldsymbol{v}\|_2 = \sigma_{\max}(\boldsymbol{A})^2\,\|\boldsymbol{u} - \boldsymbol{v}\|_2.$$

So $L = \sigma_{\max}(\boldsymbol{A})^2 = \|\boldsymbol{A}\|_{2\to 2}^2 = \lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A})$ is a valid Lipschitz constant.

Why L guarantees convergence. The descent lemma then yields

$$g(\boldsymbol{x}) \;\le\; g(\boldsymbol{x}^k) + \big\langle \nabla g(\boldsymbol{x}^k),\, \boldsymbol{x} - \boldsymbol{x}^k \big\rangle + \tfrac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}^k\|_2^2,$$

i.e. a quadratic upper bound that touches g at $\boldsymbol{x}^k$. Minimizing the majorant $+\lambda\|\boldsymbol{x}\|_1$ in $\boldsymbol{x}$ gives *exactly* the ISTA step above. Since the majorant decreases F at each step ($F(\boldsymbol{x}^{k+1}) \le F(\boldsymbol{x}^k)$), ISTA is a monotone descent method, and the rate is $F(\boldsymbol{x}^k) - F^\star = O(1/k)$ (Ch. 7 (Proximal/ISTA)).

Summary of one iteration:

$$\boldsymbol{x}^{k+1} = \mathcal{S}_{\lambda/L}\Big(\boldsymbol{x}^k - \tfrac{1}{L}\,\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}^k - \boldsymbol{y})\Big), \qquad L = \sigma_{\max}(\boldsymbol{A})^2.$$

Remark (FISTA). Replacing $\boldsymbol{x}^k$ inside the gradient by the Nesterov-extrapolated point $\boldsymbol{w}^k = \boldsymbol{x}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{x}^k - \boldsymbol{x}^{k-1})$ upgrades the rate to $O(1/k^2)$ at the same per-iteration cost, the default first-order LASSO solver.

30.2.4 Long Question 4: K-SVD

Problem statement

Given training signals as columns of $\boldsymbol{Y} \in \mathbb{R}^{n\times N}$, K-SVD seeks a dictionary $\boldsymbol{D} \in \mathbb{R}^{n\times K}$ and sparse codes $\boldsymbol{X} \in \mathbb{R}^{K\times N}$ solving

$$\min_{\boldsymbol{D},\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \quad \text{s.t.} \quad \|\boldsymbol{x}_i\|_0 \le T_0 \text{ for each column } \boldsymbol{x}_i,$$

by alternating a sparse coding step (fix $\boldsymbol{D}$, solve for $\boldsymbol{X}$ via OMP) with a dictionary update step (one atom at a time). Let $\boldsymbol{d}_k$ be the k-th atom and $\boldsymbol{x}_T^k$ the k-th *row* of $\boldsymbol{X}$.

(a) Show $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 = \|\boldsymbol{E}_k - \boldsymbol{d}_k\boldsymbol{x}_T^k\|_F^2$, where $\boldsymbol{E}_k := \boldsymbol{Y} - \sum_{j\neq k} \boldsymbol{d}_j\boldsymbol{x}_T^j$, and interpret $\boldsymbol{E}_k$.

(b) A naive fix takes a rank-1 SVD of $\boldsymbol{E}_k$. Explain why this destroys sparsity, and define the active set ω_k, the restricted error $\boldsymbol{E}_k^R$, and the restricted row $\boldsymbol{x}_R^k$.

(c) Using Eckart–Young, state the update: compute $\boldsymbol{E}_k^R = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and set $\boldsymbol{d}_k =?$ and $\boldsymbol{x}_R^k =?$. Verify the new atom is unit-norm and the support of $\boldsymbol{X}$ is preserved.

Solution recipe

Recipe

1. Read $\boldsymbol{D}\boldsymbol{X} = \sum_j \boldsymbol{d}_j\boldsymbol{x}_T^j$ via the outer-product (sum-of-rank-one) view; isolate the k-th term.
2. State Eckart–Young: the best rank-one Frobenius approximation of a matrix is its leading singular component.
3. Restrict to the active columns ω_k to preserve the OMP sparsity pattern; the rank-one fit on the restricted residual gives the K-SVD per-atom update.

(a) Isolating one atom

By the outer-product (sum-of-rank-one) form of matrix multiplication, the product $\boldsymbol{D}\boldsymbol{X} \in \mathbb{R}^{n\times N}$ decomposes as a sum of K rank-one terms,

$$\boldsymbol{D}\boldsymbol{X} = \sum_{j=1}^{K} \boldsymbol{d}_j\, \boldsymbol{x}_T^j, \tag{30.1}$$

where $\boldsymbol{d}_j \in \mathbb{R}^n$ is the j-th *column* of $\boldsymbol{D}$ (an atom) and $\boldsymbol{x}_T^j \in \mathbb{R}^{1\times N}$ is the j-th *row* of $\boldsymbol{X}$ (its coefficients across all signals). Pull out the k-th term and define the rest as $\boldsymbol{E}_k$:

$$\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} = \boldsymbol{Y} - \sum_{j\neq k} \boldsymbol{d}_j\boldsymbol{x}_T^j - \boldsymbol{d}_k\boldsymbol{x}_T^k = \boldsymbol{E}_k - \boldsymbol{d}_k\boldsymbol{x}_T^k. \tag{30.2}$$

Taking squared Frobenius norms of both sides establishes the identity:

$$\boxed{\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 = \|\boldsymbol{E}_k - \boldsymbol{d}_k\boldsymbol{x}_T^k\|_F^2.} \tag{30.3}$$

Interpreting $\boldsymbol{E}_k$. The matrix $\boldsymbol{E}_k$ is the **residual after all atoms except the k-th have done their work**: it is the share of $\boldsymbol{Y}$ that atom k must explain. So the dictionary update for atom k becomes

$$\min_{\boldsymbol{d}_k,\boldsymbol{x}_T^k} \|\boldsymbol{E}_k - \boldsymbol{d}_k\boldsymbol{x}_T^k\|_F^2,$$

the best rank-one Frobenius fit to $\boldsymbol{E}_k$.

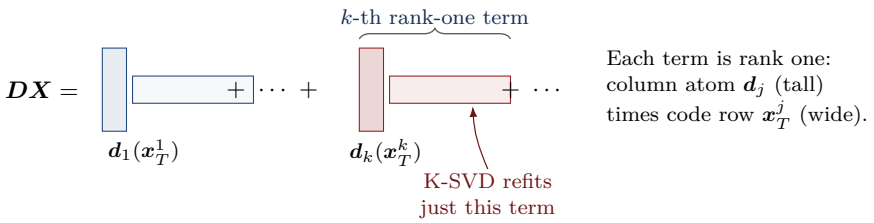


Figure 30.1. The product $\boldsymbol{DX}$ as a sum of K rank-one outer products. K-SVD freezes all but the k-th term and asks for the best rank-one fit to the leftover residual $\boldsymbol{E}_k$.

(b) Why the naive SVD destroys sparsity, and the fix

The Eckart–Young theorem gives the best rank-one approximation of $\boldsymbol{E}_k$ as the leading SVD term:

$$\boldsymbol{E}_k = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \qquad \boldsymbol{E}_k \approx \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top, \quad \boldsymbol{d}_k \leftarrow \boldsymbol{u}_1,\ \boldsymbol{x}_T^k \leftarrow \sigma_1 \boldsymbol{v}_1^\top. \tag{30.4}$$

The trouble. The top right singular vector $\boldsymbol{v}_1$ is **generically dense**: it has a nonzero entry at essentially every training signal. Overwriting the entire row $\boldsymbol{x}_T^k$ with $\sigma_1 \boldsymbol{v}_1^\top$ therefore lights up atom k at signals where OMP had not used it; the carefully maintained sparsity pattern from the sparse-coding step (each column has at most T_0 nonzeros) is gone.

> The order matters. *Restrict first, then SVD, then write back.* SVDing the full residual and masking the result is a different (and worse) operation: that SVD has already used all N columns as if the code were dense, so masking it back to ω_k throws away most of its fit.

The restriction. Define

$$\omega_k := \{i : x_T^k(i) \neq 0\}, \qquad \text{the } \textit{active set} \text{ of atom } k, \tag{30.5}$$

$$\boldsymbol{E}_k^R := \boldsymbol{E}_k\big|_{\omega_k} = \boldsymbol{E}_k(:, \omega_k), \qquad \text{the columns of } \boldsymbol{E}_k \text{ in } \omega_k, \tag{30.6}$$

$$\boldsymbol{x}_R^k := \boldsymbol{x}_T^k\big|_{\omega_k}, \qquad \text{the nonzero coefficients of atom } k. \tag{30.7}$$

ω_k is exactly the set of signals that currently use atom k. $\boldsymbol{E}_k^R$ is a slimmed matrix with $|\omega_k|$ columns (typically far smaller than N). $\boldsymbol{x}_R^k$ collects the live coefficients; the rest of $\boldsymbol{x}_T^k$ is zero and stays zero.

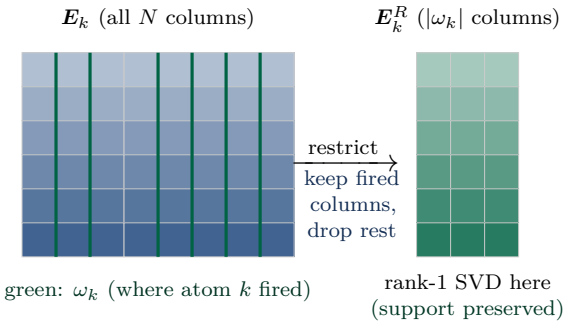


Figure 30.2. The restriction trick. Keeping only the columns of $\boldsymbol{E}_k$ in the active set ω_k yields a much smaller matrix $\boldsymbol{E}_k^R$. Its rank-one fit updates atom k and its live coefficients without ever touching the zero positions of $\boldsymbol{x}_T^k$, so the sparsity pattern survives intact.

(c) The K-SVD per-atom update

Compute the SVD of the restricted residual,

$$\boldsymbol{E}_k^R = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^\top, \tag{30.8}$$

and keep its leading triple $(\sigma_1, \boldsymbol{u}_1, \boldsymbol{v}_1)$. By Eckart–Young, the best rank-one Frobenius approximation of $\boldsymbol{E}_k^R$ is $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top$. Set

K-SVD per-atom update

$$\boxed{\boldsymbol{d}_k \leftarrow \boldsymbol{u}_1, \qquad \boldsymbol{x}_R^k \leftarrow \sigma_1\, \boldsymbol{v}_1^\top.} \tag{30.9}$$

The full row $\boldsymbol{x}_T^k$ has its entries at ω_k overwritten by $\sigma_1 \boldsymbol{v}_1^\top$ and all other entries kept at zero.

Unit-norm check. The left singular vector $\boldsymbol{u}_1$ of any matrix has $\|\boldsymbol{u}_1\|_2 = 1$ by definition (it is a column of an orthogonal matrix $\boldsymbol{U}$). So $\boldsymbol{d}_k = \boldsymbol{u}_1$ is unit-norm automatically; no separate renormalization is needed, in contrast to MOD which renormalizes at every iteration.

Support preservation. The positions of $\boldsymbol{x}_T^k$ *outside* ω_k are not touched, so they remain 0. The positions *inside* ω_k change in value but not in pattern: they were nonzero before, and after the update they are $\sigma_1 \boldsymbol{v}_1^\top$ component-wise, which is generically nonzero (zero only in a measure-zero set of cases). Hence

$$\operatorname{supp}(\boldsymbol{x}_T^k) \text{ before } = \omega_k = \operatorname{supp}(\boldsymbol{x}_T^k) \text{ after},$$

and the column-sparsity constraint $\|\boldsymbol{x}_i\|_0 \le T_0$ continues to hold for every i.

> The reason K-SVD updates atom *and* its coefficients together is exactly this rank-one fit: $\boldsymbol{d}_k$ and $\boldsymbol{x}_R^k$ are the left and (scaled) right singular vectors of the same matrix, so they are jointly optimal. MOD by contrast freezes the codes during the dictionary solve, which is why K-SVD typically reaches a lower objective in fewer iterations.

30.2.5 Long Question 5: Nonnegative Matrix Factorization

Problem statement

Given a nonnegative data matrix $\boldsymbol{V} \in \mathbb{R}^{n\times m}_{\geq 0}$ (columns are data points: images, documents), NMF seeks factors $\boldsymbol{W} \in \mathbb{R}^{n\times r}_{\geq 0}$ and $\boldsymbol{H} \in \mathbb{R}^{r\times m}_{\geq 0}$ with $\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}$.

(a) Write the optimization problem with its constraints, and explain why nonnegativity yields an additive, parts-based (often sparse) representation in contrast to SVD/PCA.

(b) The gradient of $f(\boldsymbol{H}) = \frac{1}{2}\|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_F^2$ in $\boldsymbol{H}$ is $\nabla_{\boldsymbol{H}} f = \boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{H} - \boldsymbol{W}^\top\boldsymbol{V}$. Starting from gradient descent $\boldsymbol{H} \leftarrow \boldsymbol{H} - \eta \odot \nabla_{\boldsymbol{H}} f$ with per-entry step size $\eta = \boldsymbol{H} \oslash (\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{H})$, derive the multiplicative update $\boldsymbol{H} \leftarrow \boldsymbol{H} \odot (\boldsymbol{W}^\top\boldsymbol{V}) \oslash (\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{H})$, and explain why it preserves nonnegativity.

(c) Name one concrete application, stating what the columns of $\boldsymbol{W}$ and $\boldsymbol{H}$ represent there, and state one way the NMF problem differs from the SVD regarding convexity or uniqueness.

Solution recipe

Recipe

1. Write the Frobenius minimization with $\boldsymbol{W} \geq \boldsymbol{0}, \boldsymbol{H} \geq \boldsymbol{0}$. Forbidding subtraction forces atoms to be additive "parts."
2. Substitute the data-dependent step into a gradient descent step on $\boldsymbol{H}$; watch the subtraction collapse algebraically into a pure multiplicative rule. Verify each factor of the result is a ratio of nonnegatives.
3. Topic modeling, faces, or hyperspectral unmixing all illustrate. SVD is convex (Eckart–Young is global); NMF is bilinear and nonconvex with non-unique solutions.

(a) The NMF problem and parts-based representation

The NMF optimization problem

$$\min_{\boldsymbol{W},\boldsymbol{H}} \ \tfrac{1}{2}\|\boldsymbol{V} - \boldsymbol{W}\boldsymbol{H}\|_F^2 \quad \text{subject to} \quad \boldsymbol{W} \geq \boldsymbol{0},\ \boldsymbol{H} \geq \boldsymbol{0}. \tag{30.10}$$

The data fidelity is the same Frobenius reconstruction error as in PCA/dictionary learning; the new ingredient is the entrywise nonnegativity of both factors.

Why nonnegativity yields parts. Each data column expands as

$$\boldsymbol{v}_i \approx \sum_{k=1}^{r} H_{ki}\,\boldsymbol{w}_k, \qquad H_{ki} \geq 0,\ \boldsymbol{w}_k \geq \boldsymbol{0}.$$

The model can only *add* atoms; it can never subtract one. To build a face it must turn on a topic of localized pixels (an eye region, a nose region); to build a document it must turn on a topic of co-occurring words. The result is that each $\boldsymbol{w}_k$ ends up *localized*: a small region of pixels, a small cluster of words. The representation is parts-based: data are nonnegative combinations of nonnegative parts. Since each datum typically uses only a few of the r parts, the coefficient matrix $\boldsymbol{H}$ is naturally sparse.

Contrast with SVD/PCA. Without sign constraints, the SVD finds the most compact factorization of $\boldsymbol{V}$, but the columns of $\boldsymbol{U}$ are signed: a face component might have positive pixel weights on the eyes and negative weights on the cheeks. Two such atoms can cancel each other when added: an eigenface is a holistic, often ghostly, whole-image

template that only makes sense in combination with others. PCA atoms are abstract directions, not interpretable parts. The constraint $\boldsymbol{W} \geq \mathbf{0}, \boldsymbol{H} \geq \mathbf{0}$ alone, with no prior on locality or sparsity, is enough to flip from holistic to parts-based.

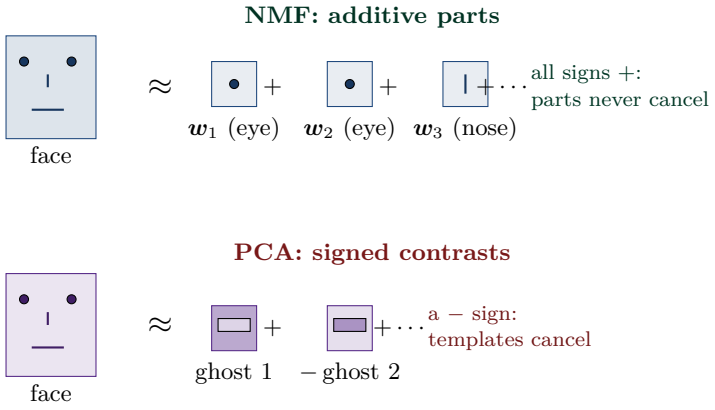


Figure 30.3. NMF (top) builds a face as a nonnegative sum of localized parts: an eye, an eye, a nose. PCA (bottom) builds the same face as a signed sum of holistic "ghosts": one whole-face template can be added, another subtracted. The same data, two interpretation regimes.

(b) Deriving the multiplicative update

Starting from gradient descent with a per-entry step,

$$H_{ij} \;\leftarrow\; H_{ij} - \eta_{ij} \cdot \big(\nabla_{\boldsymbol{H}} f\big)_{ij}, \qquad \nabla_{\boldsymbol{H}} f = \boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H} - \boldsymbol{W}^\top \boldsymbol{V}. \tag{30.11}$$

Choose the data-dependent, entrywise step

$$\eta_{ij} = \frac{H_{ij}}{(\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H})_{ij}}, \tag{30.12}$$

which is positive whenever $\boldsymbol{H}, \boldsymbol{W}$ are nontrivially nonnegative. Substituting Eq. (30.12) into Eq. (30.11):

$$\begin{aligned} H_{ij} &\leftarrow H_{ij} - \frac{H_{ij}}{(\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H})_{ij}} \left[(\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H})_{ij} - (\boldsymbol{W}^\top \boldsymbol{V})_{ij}\right] \\ &= H_{ij} - H_{ij} + H_{ij} \cdot \frac{(\boldsymbol{W}^\top \boldsymbol{V})_{ij}}{(\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H})_{ij}} \\ &= H_{ij} \cdot \frac{(\boldsymbol{W}^\top \boldsymbol{V})_{ij}}{(\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H})_{ij}}. \end{aligned} \tag{30.13}$$

The first two terms cancel, and the awkward subtraction collapses into a pure multiplication. In matrix form:

Lee–Seung multiplicative update for $\boldsymbol{H}$

$$\boxed{\boldsymbol{H} \;\leftarrow\; \boldsymbol{H} \odot \frac{\boldsymbol{W}^\top \boldsymbol{V}}{\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H}},} \tag{30.14}$$

where $\odot$ is entrywise (Hadamard) product and the fraction bar is entrywise division. (A small ε is added to the denominator in practice for numerical safety.)

Why nonnegativity is preserved (no projection). The new H_{ij} in Eq. (30.14) is the product of three nonnegative factors:

- The current $H_{ij} \geq 0$ (nonnegative by assumption).

- The numerator $(\boldsymbol{W}^\top \boldsymbol{V})_{ij} = \sum_k W_{ki} V_{kj}$ is a sum of products of nonnegatives, hence ≥ 0.
- The denominator $(\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H})_{ij}$ is likewise a sum of products of nonnegatives, hence ≥ 0.

A nonnegative value times a nonnegative ratio is nonnegative. So $\boldsymbol{H} \geq \boldsymbol{0}$ is preserved at every step *without any projection or clipping*. The constraint that would have required an awkward, descent-breaking projection step is enforced for free by the very form of the update. The symmetric update $\boldsymbol{W} \leftarrow \boldsymbol{W} \odot (\boldsymbol{V}\boldsymbol{H}^\top) \oslash (\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^\top)$ preserves $\boldsymbol{W} \geq \boldsymbol{0}$ by the same argument.

> The Lee–Seung updates also *monotonically decrease* the Frobenius objective. The proof uses a majorization–minimization argument with a separable quadratic auxiliary function; the $1/(\boldsymbol{W}^\top \boldsymbol{W}\boldsymbol{H})_{ij}$ step is exactly the choice that makes the majorant tight (Lee–Seung, 2001).

(c) Application and how NMF differs from SVD

One concrete application: topic modeling. Build the word-by-document count matrix $\boldsymbol{V} \in \mathbb{R}^{n \times m}_{\geq 0}$, with V_{ij} = count (or TF-IDF weight) of word i in document j. Choose r = number of topics and factor $\boldsymbol{V} \approx \boldsymbol{W}\boldsymbol{H}$ by Eq. (30.14).

- **Columns of $\boldsymbol{W}$** are the *topics*: each column $\boldsymbol{w}_k \in \mathbb{R}^n_{\geq 0}$ is a nonnegative distribution over words, naturally readable as "the words that tend to co-occur in topic k." A finance topic puts mass on "market," "stock," "rate."
- **Columns of $\boldsymbol{H}$** are the *topic mixtures* per document: column j gives the nonnegative weights with which document j draws from each topic.

The same recipe gives parts-based image decompositions (Lee–Seung faces), hyperspectral unmixing (atoms are pure material spectra), and audio source separation.

NMF vs. SVD: convexity and uniqueness.

- **Convexity.** The SVD is the unique global solution to the best-rank-r Frobenius problem, computed in closed form by Eckart–Young. NMF minimizes the same Frobenius objective but *bilinearly* over $(\boldsymbol{W}, \boldsymbol{H})$ with extra inequality constraints; the joint problem is nonconvex, has multiple local minima, and is NP-hard in general.
- **Uniqueness.** The SVD is unique up to sign of singular vectors when singular values are distinct. NMF has trivial ambiguities (permutation and positive diagonal rescaling of the factors), and in general also nontrivial ones: different cones spanned by the atoms can wrap the data equally well. Identifiability requires extra structural assumptions (e.g. *separability*, where each atom appears alone in at least one data column).

NMF vs. SVD in one line each

SVD: signed, orthonormal, unique, optimal reconstruction; components are abstract directions.
NMF: nonnegative, not orthonormal, not unique, slightly suboptimal reconstruction; components are interpretable parts.

30.2.6 Long Question 6: Matrix Completion

Problem statement

We observe a subset $\Omega \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$ of entries of $\boldsymbol{M} \in \mathbb{R}^{n \times n}$, believed approximately low rank, and wish to fill in the rest.

(a) Write the rank-minimization problem and its convex relaxation. Which norm plays the role of ℓ_1, and why is it the right surrogate for rank? Relate to singular values.

(b) State the incoherence assumption and explain, with a one-line example, why a "spiky" matrix such as $\boldsymbol{e}_1 \boldsymbol{e}_1^\top$ cannot be completed from random samples.

(c) Concretely: the matrix $\left(\begin{smallmatrix} 2 & 4 \\ 3 & ? \end{smallmatrix}\right)$ has rank 1. Find the missing entry.

Solution recipe

Recipe

1. Write the combinatorial rank problem; relax rank (ℓ_0 of the singular value vector) to the nuclear norm (ℓ_1 of the singular value vector), the tightest convex surrogate.
2. Translate incoherence: singular vectors should be "spread," not aligned with the standard basis. Spike example: a rank-one matrix with mass at one entry vanishes under random sampling.
3. For rank-one 2×2: rows are proportional; read the ratio from one fully known column and propagate.

(a) Rank minimization and the nuclear-norm surrogate

The hard problem. Among all matrices consistent with the observations, find the one of smallest rank:

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times n}} \operatorname{rank}(\boldsymbol{X}) \quad \text{subject to} \quad X_{ij} = M_{ij} \ \forall (i,j) \in \Omega. \tag{30.15}$$

This is the matrix analogue of (P_0) (the sparse-vector problem). Rank is combinatorial and the problem is NP-hard.

The convex relaxation. Replace rank by the **nuclear norm** $\|\boldsymbol{X}\|_* = \sum_i \sigma_i(\boldsymbol{X})$:

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times n}} \|\boldsymbol{X}\|_* \quad \text{subject to} \quad X_{ij} = M_{ij} \ \forall (i,j) \in \Omega. \tag{30.16}$$

The nuclear norm is convex (it is a norm) and the equality constraints are linear, so Eq. (30.16) is a semidefinite program, solvable in polynomial time.

Why nuclear is the right surrogate. Let $\boldsymbol{\sigma}(\boldsymbol{X}) = (\sigma_1, \ldots, \sigma_n)$ be the singular-value vector. The two complexity measures of the matrix are exactly the two sparsity measures of $\boldsymbol{\sigma}$:

$$\operatorname{rank}(\boldsymbol{X}) = \|\boldsymbol{\sigma}(\boldsymbol{X})\|_0, \qquad \|\boldsymbol{X}\|_* = \|\boldsymbol{\sigma}(\boldsymbol{X})\|_1. \tag{30.17}$$

Rank counts the nonzero singular values (a low-rank matrix has a sparse singular spectrum); the nuclear norm sums them. So nuclear is to rank exactly what ℓ_1 is to ℓ_0: the tightest convex envelope on the spectral-norm ball, lifted to act on singular values rather than on the entries. Every theorem from the vector half translates word for word.

The vector–matrix dictionary at recovery

$$\|\boldsymbol{x}\|_0 \leftrightarrow \operatorname{rank}(\boldsymbol{X}), \qquad \|\boldsymbol{x}\|_1 \leftrightarrow \|\boldsymbol{X}\|_*, \qquad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y} \leftrightarrow \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}).$$

Minimizing nuclear norm finds few nonzero singular values exactly as ℓ_1 minimization finds few nonzero entries.

(b) Incoherence and the spiky matrix

The incoherence assumption. Let $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the thin SVD with $\boldsymbol{U} \in \mathbb{R}^{n \times r}, \boldsymbol{V} \in \mathbb{R}^{n \times r}$ and rank r. The matrix is μ_0-*incoherent* if there is $\mu_0 \geq 1$ with

$$\max_{1 \leq i \leq n} \|\boldsymbol{U}^\top \boldsymbol{e}_i\|_2^2 \leq \frac{\mu_0 r}{n}, \qquad \max_{1 \leq j \leq n} \|\boldsymbol{V}^\top \boldsymbol{e}_j\|_2^2 \leq \frac{\mu_0 r}{n}. \tag{30.18}$$

Reading this: each row of $\boldsymbol{U}$ and $\boldsymbol{V}$ has norm at most $\sqrt{\mu_0 r/n}$, so no singular vector concentrates on a few coordinates. Equivalently, the singular vectors are spread out, not aligned with the standard basis. The Candès–Recht theorem (Candès–Recht, 2009) needs $\mu_0 = O(1)$ for nuclear-norm completion from $m \gtrsim n^{5/4} r \log n$ random samples (later sharpened to $\mu_0 n r \operatorname{polylog}(n)$).

Why the spike kills completion. Take $\boldsymbol{M} = \boldsymbol{e}_1\boldsymbol{e}_1^\top \in \mathbb{R}^{n\times n}$:

- Rank 1, singular value $\sigma_1 = 1$, left/right singular vector $\boldsymbol{u}_1 = \boldsymbol{v}_1 = \boldsymbol{e}_1$.
- Coherence: $\|\boldsymbol{U}^\top \boldsymbol{e}_1\|_2^2 = 1$, far above the allowed $\mu_0 r/n = \mu_0/n$. So μ_0 would have to be at least n, blowing up the sample complexity to $m \gtrsim n^2$, i.e. no compression.
- Sampling: $\boldsymbol{M}$ has exactly one nonzero entry, at $(1,1)$. A uniformly random sample of $m \ll n^2$ entries misses $(1,1)$ with probability $1 - m/n^2 \to 1$. Conditional on missing it, every observed $M_{ij} = 0$ for $(i,j) \in \Omega$, and the completion program sees an all-zeros observation set.
- Recovery: the all-zeros matrix $\boldsymbol{X} = \boldsymbol{0}$ is feasible (it satisfies $X_{ij} = 0 = M_{ij}$ on Ω) and has nuclear norm zero, the smallest possible. So Eq. (30.16) returns $\hat{\boldsymbol{X}} = \boldsymbol{0}$, not $\boldsymbol{M}$. Recovery fails.

Why incoherence is needed

The spike $\boldsymbol{e}_1\boldsymbol{e}_1^\top$ is simultaneously rank one *and* sparse: its mass hides in a single entry. Random sampling that misses that entry sees only zeros and cannot tell the matrix from the zero matrix. Incoherence Eq. (30.18) rules out exactly these "mass-in-one-spot" matrices, ensuring every entry carries information about the whole.

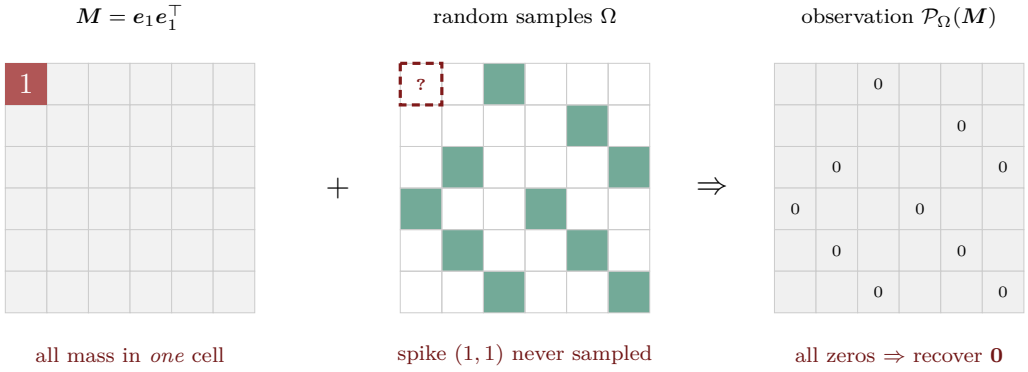


Figure 30.4. Why a spiky matrix cannot be completed. All mass sits at $(1,1)$. A random sampling pattern misses that entry with high probability, so every observation is zero, and nuclear-norm minimization returns the zero matrix.

(c) Concrete: complete the rank-one 2×2

We must complete

$$\boldsymbol{M} = \begin{pmatrix} 2 & 4 \\ 3 & ? \end{pmatrix}$$

to rank 1. A rank-one matrix is exactly one whose rows are proportional (equivalently, whose 2×2 minors vanish).

Method 1 (row ratios). Write $\boldsymbol{M} = \boldsymbol{u}\boldsymbol{v}^\top$ with $\boldsymbol{u} \in \mathbb{R}^2, \boldsymbol{v} \in \mathbb{R}^2$. From row 1: $u_1 v_1 = 2$ and $u_1 v_2 = 4$, so $v_2/v_1 = 2$. From row 2: $u_2 v_1 = 3$, so the missing $M_{22} = u_2 v_2 = u_2 v_1 \cdot (v_2/v_1) = 3 \cdot 2 = 6$.

Method 2 (determinant test). Rank ≤ 1 iff $\det \boldsymbol{M} = 0$:

$$\det \begin{pmatrix} 2 & 4 \\ 3 & ? \end{pmatrix} = 2 \cdot ? - 4 \cdot 3 = 0 \implies ? = 12/2 = 6.$$

Method 3 (column-ratio sanity check). Column 1: $(2,3)^\top$. Column 2 must be a scalar multiple. Top entry gives the scalar $4/2 = 2$, so column 2 is $2 \cdot (2,3)^\top = (4,6)^\top$. The missing entry is 6.

The completed matrix

$$? = 6, \qquad \boldsymbol{M} = \begin{pmatrix} 2 & 4 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 1 \\ 3/2 \end{pmatrix} \begin{pmatrix} 2 & 4 \end{pmatrix}. \tag{30.19}$$

Verification: $\det = 2 \cdot 6 - 4 \cdot 3 = 12 - 12 = 0$, so $\operatorname{rank}(\boldsymbol{M}) = 1$. Singular value $\sigma_1 = \sqrt{2^2 + 4^2 + 3^2 + 6^2} = \sqrt{65} \approx 8.06$, and $\sigma_2 = 0$, consistent with rank 1.

30.3 Part III: Proofs

30.3.1 Question 1: The ℓ_1–ℓ_0 uniqueness chain

Problem statement

Let $\boldsymbol{x}^\star$ be k-sparse with $\boldsymbol{A}\boldsymbol{x}^\star = \boldsymbol{y}$.

(a) Suppose $\operatorname{spark}(\boldsymbol{A}) > 2k$. Prove *by contradiction* that $\boldsymbol{x}^\star$ is the unique sparsest solution (unique ℓ_0 minimizer).

(b) Suppose additionally $\boldsymbol{A}$ satisfies NSP of order k: for every $\boldsymbol{v} \in \ker \boldsymbol{A} \setminus \{\boldsymbol{0}\}$ and every $|S| \le k$, $\|\boldsymbol{v}_S\|_1 < \|\boldsymbol{v}_{S^c}\|_1$. Prove *by contradiction* that $\boldsymbol{x}^\star$ is also the unique ℓ_1 minimizer. *Hint: assume $\hat{\boldsymbol{x}} \neq \boldsymbol{x}^\star$ with $\|\hat{\boldsymbol{x}}\|_1 \le \|\boldsymbol{x}^\star\|_1$, set $\boldsymbol{h} = \hat{\boldsymbol{x}} - \boldsymbol{x}^\star \in \ker \boldsymbol{A}$, $S = \operatorname{supp}(\boldsymbol{x}^\star)$, and use the reverse triangle inequality.*

(a) Spark $> 2k$ implies unique ℓ_0 minimizer

Setup. Assume, for contradiction, that there exists $\boldsymbol{z}^\star \neq \boldsymbol{x}^\star$ with $\boldsymbol{A}\boldsymbol{z}^\star = \boldsymbol{y}$ and $\|\boldsymbol{z}^\star\|_0 \le k$ (so $\boldsymbol{z}^\star$ is another solution at least as sparse as $\boldsymbol{x}^\star$). We will derive a contradiction with the spark hypothesis.

Step 1: form the difference. Define

$$\boldsymbol{v} := \boldsymbol{x}^\star - \boldsymbol{z}^\star. \tag{30.20}$$

Then $\boldsymbol{v} \neq \boldsymbol{0}$ (since $\boldsymbol{x}^\star \neq \boldsymbol{z}^\star$). Apply $\boldsymbol{A}$:

$$\boldsymbol{A}\boldsymbol{v} = \boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{A}\boldsymbol{z}^\star = \boldsymbol{y} - \boldsymbol{y} = \boldsymbol{0}, \tag{30.21}$$

so $\boldsymbol{v} \in \ker(\boldsymbol{A}) \setminus \{\boldsymbol{0}\}$.

Step 2: bound the sparsity of $\boldsymbol{v}$ from above. $\operatorname{supp}(\boldsymbol{v}) \subseteq \operatorname{supp}(\boldsymbol{x}^\star) \cup \operatorname{supp}(\boldsymbol{z}^\star)$ because a coordinate of $\boldsymbol{v}$ can be nonzero only where $\boldsymbol{x}^\star$ or $\boldsymbol{z}^\star$ has a nonzero. Since support sizes add under union (with possible overcount on overlap),

$$\|\boldsymbol{v}\|_0 \le \|\boldsymbol{x}^\star\|_0 + \|\boldsymbol{z}^\star\|_0 \le k + k = 2k. \tag{30.22}$$

Step 3: bound the sparsity of $\boldsymbol{v}$ from below. By the definition of spark, *every* nonzero vector in $\ker(\boldsymbol{A})$ has at least $\operatorname{spark}(\boldsymbol{A})$ nonzeros. So

$$\|\boldsymbol{v}\|_0 \;\ge\; \operatorname{spark}(\boldsymbol{A}) \;>\; 2k. \tag{30.23}$$

Step 4: collide the bounds. Equation (30.22) gives $\|\boldsymbol{v}\|_0 \le 2k$; Eq. (30.23) gives $\|\boldsymbol{v}\|_0 > 2k$. These are contradictory.

Conclusion. No such $\boldsymbol{z}^\star$ can exist. Hence $\boldsymbol{x}^\star$ is the *unique* solution of $\min \|\boldsymbol{z}\|_0$ subject to $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{y}$. ■

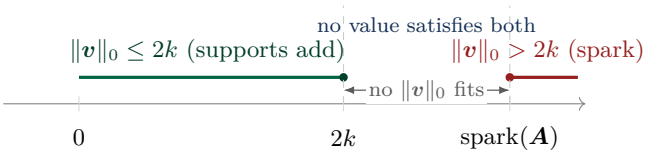


Figure 30.5. The spark contradiction. Any difference of two k-sparse solutions has at most $2k$ nonzeros (top, supports add) but at least $\mathrm{spark}(\boldsymbol{A}) > 2k$ nonzeros (bottom, spark forbids small kernel vectors). The bounds cannot both hold, so no second k-sparse solution exists.

(b) NSP implies unique ℓ_1 minimizer for $x^\star$ k-sparse

Assume, for contradiction, that there is a feasible $\hat{\boldsymbol{x}} \neq \boldsymbol{x}^\star$ (so $\boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{y}$) with

$$\|\hat{\boldsymbol{x}}\|_1 \le \|\boldsymbol{x}^\star\|_1. \tag{30.24}$$

We will show Eq. (30.24) is impossible. Let

$$\boldsymbol{h} := \hat{\boldsymbol{x}} - \boldsymbol{x}^\star, \qquad S := \mathrm{supp}(\boldsymbol{x}^\star), \quad |S| \le k. \tag{30.25}$$

Step 1: h is a nonzero kernel vector. $\hat{\boldsymbol{x}} \neq \boldsymbol{x}^\star$ gives $\boldsymbol{h} \neq \boldsymbol{0}$. Both are feasible, so

$$\boldsymbol{A}\boldsymbol{h} = \boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{A}\boldsymbol{x}^\star = \boldsymbol{y} - \boldsymbol{y} = \boldsymbol{0}, \tag{30.26}$$

i.e. $\boldsymbol{h} \in \ker(\boldsymbol{A}) \setminus \{\boldsymbol{0}\}$.

Step 2: split $\|\hat{\boldsymbol{x}}\|_1$ by S. Because the ℓ_1 norm is separable over disjoint index sets,

$$\|\hat{\boldsymbol{x}}\|_1 = \|\hat{\boldsymbol{x}}_S\|_1 + \|\hat{\boldsymbol{x}}_{S^c}\|_1. \tag{30.27}$$

On S: $\hat{\boldsymbol{x}}_S = \boldsymbol{x}^\star_S + \boldsymbol{h}_S = \boldsymbol{x}^\star + \boldsymbol{h}_S$, since $\boldsymbol{x}^\star$ is supported on S. On S^c: $\boldsymbol{x}^\star_{S^c} = \boldsymbol{0}$, so $\hat{\boldsymbol{x}}_{S^c} = \boldsymbol{h}_{S^c}$. Substituting,

$$\|\hat{\boldsymbol{x}}\|_1 = \|\boldsymbol{x}^\star + \boldsymbol{h}_S\|_1 + \|\boldsymbol{h}_{S^c}\|_1. \tag{30.28}$$

Step 3: reverse triangle on the S block. For any two vectors $\boldsymbol{a}, \boldsymbol{b}$, the triangle inequality $\|\boldsymbol{a}+\boldsymbol{b}\|_1 \ge \|\boldsymbol{a}\|_1 - \|\boldsymbol{b}\|_1$ (the *reverse* triangle) holds in every norm. Apply it to $\boldsymbol{a} = \boldsymbol{x}^\star$, $\boldsymbol{b} = \boldsymbol{h}_S$:

$$\|\boldsymbol{x}^\star + \boldsymbol{h}_S\|_1 \;\ge\; \|\boldsymbol{x}^\star\|_1 - \|\boldsymbol{h}_S\|_1. \tag{30.29}$$

Substituting into Eq. (30.28),

$$\|\hat{\boldsymbol{x}}\|_1 \;\ge\; \|\boldsymbol{x}^\star\|_1 - \|\boldsymbol{h}_S\|_1 + \|\boldsymbol{h}_{S^c}\|_1 = \|\boldsymbol{x}^\star\|_1 + \big(\|\boldsymbol{h}_{S^c}\|_1 - \|\boldsymbol{h}_S\|_1\big). \tag{30.30}$$

Step 4: apply NSP. By NSP of order k applied to $\boldsymbol{h} \in \ker(\boldsymbol{A}) \setminus \{\boldsymbol{0}\}$ at the index set S with $|S| \le k$,

$$\|\boldsymbol{h}_S\|_1 \;<\; \|\boldsymbol{h}_{S^c}\|_1 \quad\Longrightarrow\quad \|\boldsymbol{h}_{S^c}\|_1 - \|\boldsymbol{h}_S\|_1 \;>\; 0. \tag{30.31}$$

Step 5: combine and conclude. Plug Eq. (30.31) into Eq. (30.30):

$$\|\hat{\boldsymbol{x}}\|_1 \;>\; \|\boldsymbol{x}^\star\|_1. \tag{30.32}$$

This contradicts the assumption Eq. (30.24). Therefore no such competitor $\hat{\boldsymbol{x}}$ exists, and $\boldsymbol{x}^\star$ is the unique ℓ_1 minimizer.

The NSP–recovery chain

$$\mathrm{spark}(\boldsymbol{A}) > 2k \overset{\text{(a)}}{\iff} \text{unique } \ell_0 \text{ recovery} \overset{\text{(c)}}{\Longleftarrow} \text{unique } \ell_1 \text{ recovery} \overset{\text{(b)}}{\iff} \mathrm{NSP}_k.$$

Part (a) is the spark characterization of ℓ_0 uniqueness; part (b) is the NSP characterization of ℓ_1 uniqueness. The single one-way arrow (c) is the crux: ℓ_1 uniqueness implies ℓ_0 uniqueness, but *not* conversely (Question 1 exhibits a matrix with unique ℓ_0 but not unique ℓ_1 recovery). So NSP_k is strictly stronger than $\mathrm{spark}(\boldsymbol{A}) > 2k$.

■

30.3.2 Question 2: Uncertainty principle (coherence bound)

Problem statement

Let $\boldsymbol{\Psi}, \boldsymbol{\Phi}$ be two orthonormal bases of $\mathbb{R}^n$ with mutual coherence

$$\mu(\boldsymbol{A}) = \max_{i,j} |\langle \boldsymbol{\psi}_i, \boldsymbol{\phi}_j \rangle|,$$

where $\boldsymbol{\psi}_i, \boldsymbol{\phi}_j$ are their columns. Let $\boldsymbol{b} \neq \boldsymbol{0}$ have representations $\boldsymbol{b} = \boldsymbol{\Psi}\boldsymbol{\alpha} = \boldsymbol{\Phi}\boldsymbol{\beta}$. Without loss of generality assume $\|\boldsymbol{b}\|_2 = 1$.

(a) Show that $1 = \boldsymbol{\alpha}^\top \boldsymbol{\Psi}^\top \boldsymbol{\Phi} \boldsymbol{\beta} = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \beta_j \, \boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j$.

(b) Using the definition of $\mu(\boldsymbol{A})$, prove

$$1 \;\leq\; \mu(\boldsymbol{A}) \sum_{i=1}^n \sum_{j=1}^n |\alpha_i||\beta_j| \;=\; \mu(\boldsymbol{A}) \|\boldsymbol{\alpha}\|_1 \|\boldsymbol{\beta}\|_1.$$

Hint: apply the triangle inequality, bound each $|\boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j|$, then factor the double sum.

(a) The double-sum identity

We start from $\|\boldsymbol{b}\|_2^2 = 1$ and rewrite both copies of $\boldsymbol{b}$ in different bases.

Step 1: write $\|\boldsymbol{b}\|_2^2$ as an inner product. By definition,

$$1 = \|\boldsymbol{b}\|_2^2 = \langle \boldsymbol{b}, \boldsymbol{b} \rangle = \boldsymbol{b}^\top \boldsymbol{b}. \tag{30.33}$$

Step 2: substitute the two representations. Write the left factor as $\boldsymbol{b} = \boldsymbol{\Psi}\boldsymbol{\alpha}$ and the right as $\boldsymbol{b} = \boldsymbol{\Phi}\boldsymbol{\beta}$:

$$1 = \boldsymbol{b}^\top \boldsymbol{b} = (\boldsymbol{\Psi}\boldsymbol{\alpha})^\top (\boldsymbol{\Phi}\boldsymbol{\beta}) = \boldsymbol{\alpha}^\top \, \boldsymbol{\Psi}^\top \boldsymbol{\Phi} \, \boldsymbol{\beta}. \tag{30.34}$$

This is the first form of the identity we are asked to establish.

Step 3: expand the triple product as a double sum. A matrix-vector product $(\boldsymbol{\Psi}^\top \boldsymbol{\Phi})\boldsymbol{\beta}$ has i-th entry

$$\big((\boldsymbol{\Psi}^\top \boldsymbol{\Phi})\boldsymbol{\beta}\big)_i = \sum_{j=1}^n (\boldsymbol{\Psi}^\top \boldsymbol{\Phi})_{ij} \beta_j. \tag{30.35}$$

The (i,j) entry of $\boldsymbol{\Psi}^\top \boldsymbol{\Phi}$ is the inner product of row i of $\boldsymbol{\Psi}^\top$ with column j of $\boldsymbol{\Phi}$, i.e.

$$(\boldsymbol{\Psi}^\top \boldsymbol{\Phi})_{ij} = \boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j = \langle \boldsymbol{\psi}_i, \boldsymbol{\phi}_j \rangle \,. \tag{30.36}$$

Now the outer product with $\boldsymbol{\alpha}^\top$:

$$\begin{aligned} \boldsymbol{\alpha}^\top \, \boldsymbol{\Psi}^\top \boldsymbol{\Phi} \, \boldsymbol{\beta} &= \sum_{i=1}^n \alpha_i \big((\boldsymbol{\Psi}^\top \boldsymbol{\Phi})\boldsymbol{\beta}\big)_i \\ &= \sum_{i=1}^n \alpha_i \sum_{j=1}^n (\boldsymbol{\Psi}^\top \boldsymbol{\Phi})_{ij} \beta_j \\ &= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \beta_j \, \boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j. \end{aligned} \tag{30.37}$$

Combining Eqs. (30.34) and (30.37):

The double-sum identity

$$\boxed{1 = \boldsymbol{\alpha}^\top \boldsymbol{\Psi}^\top \boldsymbol{\Phi}\, \boldsymbol{\beta} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\, \beta_j\, \boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j.} \tag{30.38}$$

Sanity check. If $\boldsymbol{\Phi} = \boldsymbol{\Psi}$, then $\boldsymbol{\Psi}^\top\boldsymbol{\Phi} = \boldsymbol{\Psi}^\top\boldsymbol{\Psi} = \boldsymbol{I}_n$ (orthonormality), so $\boldsymbol{\alpha}^\top \boldsymbol{I}\boldsymbol{\beta} = \boldsymbol{\alpha}^\top\boldsymbol{\beta}$. And $\boldsymbol{\alpha} = \boldsymbol{\beta}$ (the unique representation in one basis), so the sum equals $\|\boldsymbol{\alpha}\|_2^2 = \|\boldsymbol{b}\|_2^2 = 1$. ✓

(b) The coherence bound

We prove

$$1 \;\le\; \mu(\boldsymbol{A}) \sum_{i,j} |\alpha_i||\beta_j| = \mu(\boldsymbol{A}) \|\boldsymbol{\alpha}\|_1 \|\boldsymbol{\beta}\|_1.$$

Start from Eq. (30.38) and apply four basic inequalities in sequence.

Step 1: take absolute values. Since the left side of Eq. (30.38) is 1, certainly $1 = |1|$:

$$1 = |1| = \left| \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\beta_j\, \boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j \right|. \tag{30.39}$$

Step 2: triangle inequality on the double sum. The absolute value of a sum is at most the sum of absolute values, $|\sum_k z_k| \le \sum_k |z_k|$ (the triangle inequality, repeated once for each index). Applied to both summation indices:

$$1 \;\le\; \sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_i\beta_j\, \boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j| = \sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_i||\beta_j||\boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j|, \tag{30.40}$$

using $|ab| = |a||b|$ for real scalars.

Step 3: bound each cross inner product by $\mu(\boldsymbol{A})$. By the definition of mutual coherence between the two bases,

$$|\boldsymbol{\psi}_i^\top \boldsymbol{\phi}_j| = |\langle \boldsymbol{\psi}_i,\, \boldsymbol{\phi}_j\rangle| \le \mu(\boldsymbol{A}) \quad \text{for every } i, j \in \{1, \dots, n\}. \tag{30.41}$$

Substituting this uniform bound into Eq. (30.40):

$$1 \;\le\; \sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_i||\beta_j| \cdot \mu(\boldsymbol{A}) = \mu(\boldsymbol{A}) \sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_i||\beta_j|. \tag{30.42}$$

Step 4: factor the double sum. The double sum $\sum_{i,j}|\alpha_i||\beta_j|$ separates because each term is a product of an i-only factor and a j-only factor (Fubini, or just the distributive law):

$$\sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_i||\beta_j| = \left(\sum_{i=1}^{n} |\alpha_i|\right)\left(\sum_{j=1}^{n} |\beta_j|\right) = \|\boldsymbol{\alpha}\|_1\, \|\boldsymbol{\beta}\|_1. \tag{30.43}$$

Conclusion. Combining Eq. (30.42) and Eq. (30.43),

The coherence inequality

$$1 \;\le\; \mu(\boldsymbol{A}) \sum_{i=1}^{n} \sum_{j=1}^{n} |\alpha_i||\beta_j| = \mu(\boldsymbol{A})\, \|\boldsymbol{\alpha}\|_1\, \|\boldsymbol{\beta}\|_1. \tag{30.44}$$

■

Why this is an uncertainty principle. Rearranging Eq. (30.44),

$$\|\boldsymbol{\alpha}\|_1\, \|\boldsymbol{\beta}\|_1 \;\ge\; \frac{1}{\mu(\boldsymbol{A})}. \tag{30.45}$$

A nonzero signal cannot be simultaneously ℓ_1-small in both bases when those bases are incoherent (small μ). The argument extends through Cauchy–Schwarz on the support: $\|\boldsymbol{\alpha}\|_1 \le \sqrt{\|\boldsymbol{\alpha}\|_0}\|\boldsymbol{\alpha}\|_2 = \sqrt{\|\boldsymbol{\alpha}\|_0}$ (since $\|\boldsymbol{b}\|_2 = 1$ and $\boldsymbol{\Psi}$ is orthonormal), giving the ℓ_0 form $\|\boldsymbol{\alpha}\|_0\|\boldsymbol{\beta}\|_0 \ge 1/\mu(\boldsymbol{A})^2$ and by AM–GM, $\|\boldsymbol{\alpha}\|_0 + \|\boldsymbol{\beta}\|_0 \ge 2/\mu(\boldsymbol{A})$ (the Donoho–Stark uncertainty principle).

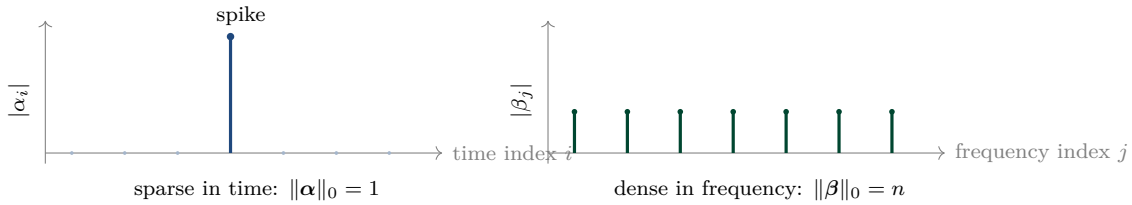


Figure 30.6. The uncertainty principle in pictures. A single spike in the time basis (left) spreads to all n frequencies (right). The product $\|\boldsymbol{\alpha}\|_0\|\boldsymbol{\beta}\|_0 \ge 1/\mu^2$ is met with equality for the spike–Fourier pair where $\mu = 1/\sqrt{n}$, giving $\|\boldsymbol{\alpha}\|_0\|\boldsymbol{\beta}\|_0 \ge n$. The signal cannot hide from random measurements in either basis at once.

PART IX

Appendices

APPENDIX A

Key Results at a Glance

This appendix collects the book's central theorems in one place, each stated tersely with its hypothesis and a pointer to its full statement and proof. It is a lookup card: when you need to recall *which* condition a result requires, or *where* it lives, start here. Notation follows Appendix G; $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ senses an s-sparse vector, and $\mu, \operatorname{spark}, \delta_s$ are its coherence, spark, and restricted isometry constant.

A.1 Uniqueness of the sparse solution

ℓ_0 is NP-hard.
Finding the sparsest solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ is NP-hard by reduction from exact cover (Chapter 3). This is the obstacle the whole book routes around.

Spark uniqueness.
An s-sparse $\boldsymbol{x}$ is the unique solution of (P_0) for every $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ if and only if $2s < \operatorname{spark}(\boldsymbol{A})$, equivalently $s < \operatorname{spark}(\boldsymbol{A})/2$ (Theorem 4.1).

Coherence lower-bounds spark.
$\operatorname{spark}(\boldsymbol{A}) \geq 1 + 1/\mu(\boldsymbol{A})$, so $\mu < 1/(2s-1)$ certifies s-sparse uniqueness (Proposition 4.6). Checkable in $O(N^2 m)$.

Welch bound.
For unit-norm columns and $N > m$, $\mu(\boldsymbol{A}) \geq \sqrt{(N-m)/(m(N-1))} \approx 1/\sqrt{m}$ (Proposition 8.14): coherence cannot beat $1/\sqrt{m}$, which forces the quadratic bottleneck $m \gtrsim s^2$.

A.2 When the convex relaxation is exact

Null space property.
(P_1) recovers every s-sparse vector if and only if every nonzero $\boldsymbol{v} \in \ker(\boldsymbol{A})$ satisfies $\|\boldsymbol{v}_S\|_1 < \|\boldsymbol{v}_{S^c}\|_1$ for all $|S| \leq s$ (Theorem 4.10). The exact characterization of ℓ_1 recovery.

ℓ_1 minimizers are sparse.
A unique (P_1) solution has at most $\operatorname{rank}(\boldsymbol{A}) \leq m$ nonzeros (Corollary 5.6): sparsity falls out of the geometry, unasked.

Restricted isometry property.
If $\delta_{2s}(\boldsymbol{A}) < \sqrt{2} - 1$ (looser versions use $\frac{1}{3}$), then (P_1) recovers every s-sparse vector, stably under noise and robustly under near-sparsity (Theorem 8.4). The workhorse sufficient condition.

Coherence bounds the RIP constant.
$\delta_s(\boldsymbol{A}) \leq (s-1)\mu(\boldsymbol{A})$ (Theorem 8.13): a small coherence gives a usable restricted isometry constant, the bridge between the checkable and the powerful.

Stable and robust recovery.
If $\delta_{2s} < \sqrt{2} - 1$, the noise-aware program obeys $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2 \leq C_0\, \sigma_s(\boldsymbol{x})_1/\sqrt{s} + C_1 \eta$ (Theorem 8.8): exact for sparse noiseless signals, error linear in the noise η, and graceful for compressible signals. The bound the field actually uses.

A.3 Random and structured sensing

Random matrices satisfy the RIP.
A Gaussian or sub-Gaussian $\boldsymbol{A}$ has $\delta_{2s} < \frac{1}{3}$ with overwhelming probability once $m \gtrsim s \log(N/s)$ (Chapter 9), the optimal measurement count.

Johnson–Lindenstrauss.
A random projection to $m \sim \varepsilon^{-2} \log n$ dimensions preserves all pairwise distances among n points to within $1 \pm \varepsilon$, independent of the ambient dimension (Theorem 9.3).

Sparse uncertainty principle.
For coherence μ between two bases, any nonzero signal with coordinates $\boldsymbol{\alpha}, \boldsymbol{\beta}$ obeys $\|\boldsymbol{\alpha}\|_0 \|\boldsymbol{\beta}\|_0 \geq 1/\mu^2$ and $\|\boldsymbol{\alpha}\|_0 + \|\boldsymbol{\beta}\|_0 \geq 2/\mu$ (Theorem 10.2): a signal cannot be sparse in two incoherent bases at once.

Bounded orthonormal systems.
A structured random matrix from a K-bounded orthonormal system (Fourier, Walsh–Hadamard, $K = 1$) satisfies the RIP once $m \gtrsim K^2 s \log^4 N$ (Chapter 10), nearly optimal but with a fast transform.

A.4 Low-rank and matrix models

Eckart–Young.
The best rank-k approximation of a matrix is its truncated SVD, with Frobenius error $(\sum_{i>k} \sigma_i^2)^{1/2}$ and spectral error σ_{k+1} (Theorem D.3). The matrix analogue of best s-term approximation.

Matrix null space property.
Nuclear-norm minimization recovers every rank-r matrix if and only if every nonzero $\boldsymbol{M} \in \ker(\mathcal{A})$ has $\sum_{i \leq r} \sigma_i(\boldsymbol{M}) < \sum_{i>r} \sigma_i(\boldsymbol{M})$ (Theorem 11.2): the vector NSP with the SVD split in place of the support split.

Candès–Recht completion.
A rank-r incoherent $N_1 \times N_2$ matrix is recovered exactly by nuclear-norm minimization from $m \gtrsim (N^\star)^{5/4} r \log N^\star$ random entries, sharpened later to $rN^\star$ polylog (Theorem 11.7).

Robust PCA (CLMW).
Principal component pursuit $\min \|\boldsymbol{X}\|_* + \lambda \|\boldsymbol{E}\|_1$ with $\lambda = 1/\sqrt{N^\star}$ separates an incoherent low-rank $\boldsymbol{X}$ from a sparse $\boldsymbol{E}$ exactly, even when $\boldsymbol{E}$ corrupts a constant fraction of entries (Theorem 12.4).

A.5 Factorization and joint models

Lee–Seung convergence.
The nonnegative multiplicative updates decrease the Frobenius objective monotonically (a majorization-minimization scheme), converging to a stationary point (Theorem 14.3).

Joint-sparse uniqueness.
An S-row-sparse $\boldsymbol{X}$ is the unique row-sparse solution of $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}$ if and only if $S < (\operatorname{spark}(\boldsymbol{A}) - 1 + \operatorname{rank}(\boldsymbol{Y}))/2$ (Theorem 15.5): the bound *improves* by the rank of the measurements.

SOMP exact recovery.
If $\max_{j \notin S} \|\boldsymbol{A}_S^\dagger \boldsymbol{a}_j\|_1 < 1$, then simultaneous OMP recovers the shared support exactly, independent of the inner norm used in selection (Theorem 15.8). The same exact-recovery condition as single-vector OMP.

Every entry above is one precise form of the book's single sentence (Chapter 19): low-dimensional structure makes high-dimensional recovery possible from few measurements, provided the sensing operator meets a condition that random matrices satisfy with overwhelming probability.

APPENDIX B

Linear Algebra Reference

This appendix is a self-contained reference for the linear algebra the book uses. Unlike a bare formula sheet, every concept here comes with a worked numerical example and, where geometry helps, a picture. Chapter 2 develops the same material in the flow of the main text; this appendix is the place to look up a definition, recompute a quantity, or refresh a proof. Throughout, $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ has columns $\boldsymbol{a}_1, \dots, \boldsymbol{a}_N$, singular values $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$ with $n = \min(m, N)$, and rank r.

B.1 Vectors, norms, and geometry

A **norm** measures the size of a vector. The book uses four, listed in Table B.1, and the choice of norm is never cosmetic: the entire subject turns on the difference between the ℓ_0 count, the ℓ_1 sum, and the ℓ_2 length.

Norm	Formula	Role
$\|\boldsymbol{x}\|_0$	$\|\{i : x_i \neq 0\}\|$	sparsity count; not a true norm (fails homogeneity)
$\|\boldsymbol{x}\|_1$	$\sum_i \|x_i\|$	convex surrogate for ℓ_0; promotes sparsity
$\|\boldsymbol{x}\|_2$	$\left(\sum_i \|x_i\|^2\right)^{1/2}$	Euclidean length; energy
$\|\boldsymbol{x}\|_\infty$	$\max_i \|x_i\|$	largest coordinate

Table B.1. The four vector measures used in the book.

Example B.1 (All four norms of one vector). Take $\boldsymbol{x} = (3, -4, 0, 1)^\top$. Then $\|\boldsymbol{x}\|_0 = 3$ (three nonzeros), $\|\boldsymbol{x}\|_1 = 3 + 4 + 0 + 1 = 8$, $\|\boldsymbol{x}\|_2 = \sqrt{9 + 16 + 0 + 1} = \sqrt{26} \approx 5.10$, and $\|\boldsymbol{x}\|_\infty = 4$. They sit in the order $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1$, here $4 \leq 5.10 \leq 8$, which holds for every vector. ◊

Two inequalities relate the norms on a *sparse* vector and recur throughout the recovery proofs. If $\boldsymbol{x}$ has at most s nonzeros,

$$\|\boldsymbol{x}\|_1 \leq \sqrt{s}\,\|\boldsymbol{x}\|_2, \qquad \|\boldsymbol{x}\|_2 \leq \sqrt{s}\,\|\boldsymbol{x}\|_\infty,$$

both from Cauchy–Schwarz applied to the support. For Example B.1 with $s = 3$ the first reads $8 \leq \sqrt{3}\cdot\sqrt{26} = \sqrt{78} \approx 8.83$, true and nearly tight because the nonzeros are comparable in size. The general bounds are Cauchy–Schwarz $|\langle \boldsymbol{u}, \boldsymbol{v}\rangle| \leq \|\boldsymbol{u}\|_2\|\boldsymbol{v}\|_2$ and the triangle inequality $\|\boldsymbol{u} + \boldsymbol{v}\| \leq \|\boldsymbol{u}\| + \|\boldsymbol{v}\|$.

The **inner product** $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \sum_i \overline{u_i} v_i$ gives length $\|\boldsymbol{v}\|_2 = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v}\rangle}$ and angle $\cos\theta = \langle \boldsymbol{u}, \boldsymbol{v}\rangle / (\|\boldsymbol{u}\|_2\|\boldsymbol{v}\|_2)$; vectors are **orthogonal** when $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = 0$. For $\boldsymbol{u} = (1, 2, 2)^\top$ and $\boldsymbol{v} = (2, 1, -2)^\top$, $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = 2 + 2 - 4 = 0$, so they are orthogonal, and each has length 3.

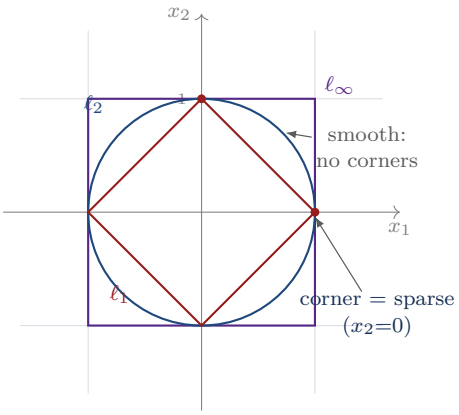


Figure B.1. The unit balls $\{\|\boldsymbol{x}\|_p \le 1\}$ for $p = 1$ (diamond), $p = 2$ (circle), $p = \infty$ (square) in $\mathbb{R}^2$. The ℓ_1 ball has its corners on the coordinate axes, where a vector has a zero coordinate and so is sparse; the ℓ_2 ball is smooth and favors no direction. A linear constraint sliding toward the origin first touches the ℓ_1 ball at a corner, which is why ℓ_1 minimization returns sparse solutions and ℓ_2 minimization does not.

B.2 Matrices as operators

A matrix $\boldsymbol{A}$ acts on a vector by forming a linear combination of its columns, $\boldsymbol{A}\boldsymbol{x} = \sum_j x_j \boldsymbol{a}_j$. Reading the product this way, rather than row by row, is the habit the whole book relies on: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ says $\boldsymbol{y}$ is a combination of s columns when $\boldsymbol{x}$ is s-sparse.

Example B.2 (Matrix-vector product as a column combination). With $\boldsymbol{A} = \left(\begin{smallmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \end{smallmatrix}\right)$ and $\boldsymbol{x} = (3, 4, 0)^\top$, the product $\boldsymbol{A}\boldsymbol{x} = 3\binom{1}{0} + 4\binom{0}{1} + 0\binom{2}{1} = \binom{3}{4}$ uses only the first two columns, because $\boldsymbol{x}$ is supported on $\{1, 2\}$. ◊

Matrices have their own norms, summarized in Table B.2. The most important is the **spectral norm** $\|\boldsymbol{A}\|_{2\to 2} = \sigma_1$, the largest factor by which $\boldsymbol{A}$ can stretch a unit vector. Geometrically $\boldsymbol{A}$ maps the unit sphere to an ellipsoid whose semi-axes are the singular values (Fig. B.2); the longest axis is σ_1 and the shortest is σ_n.

Norm	Formula	In terms of σ_i
Frobenius	$\left(\sum_{i,j} \lvert A_{ij}\rvert^2\right)^{1/2} = \sqrt{\operatorname{tr}(\boldsymbol{A}^*\boldsymbol{A})}$	$\left(\sum_i \sigma_i^2\right)^{1/2}$
Spectral $(2\to 2)$	$\max_{\|\boldsymbol{x}\|_2=1} \|\boldsymbol{A}\boldsymbol{x}\|_2$	σ_1
Nuclear	$\|\boldsymbol{A}\|_*$	$\sum_i \sigma_i$
Induced $1\to 1$	max absolute column sum	—
Induced $\infty\to\infty$	max absolute row sum	—

Table B.2. Matrix norms and their singular-value expressions. Ordering $\|\boldsymbol{A}\|_{2\to 2} \le \|\boldsymbol{A}\|_F \le \|\boldsymbol{A}\|_*$.

Example B.3 (Three matrix norms of one matrix). Let $\boldsymbol{A} = \left(\begin{smallmatrix} 3 & 0 \\ 0 & 2 \\ 0 & 0 \end{smallmatrix}\right)$. Its singular values are 3 and 2 (it is already diagonal), so $\|\boldsymbol{A}\|_{2\to 2} = 3$, $\|\boldsymbol{A}\|_F = \sqrt{9+4} = \sqrt{13} \approx 3.61$, and $\|\boldsymbol{A}\|_* = 3 + 2 = 5$. The ordering $3 \le 3.61 \le 5$ holds, as it must. The induced $1\to 1$ norm is the largest column sum, $\max(3, 2) = 3$; the $\infty\to\infty$ norm is the largest row sum, also 3. ◊

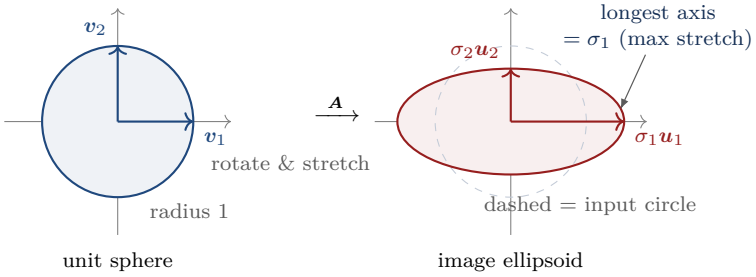


Figure B.2. A matrix maps the unit sphere (navy) to an ellipsoid (red). The dashed circle is the input radius-1 sphere copied over, so the semi-axes show the stretch factors directly. The right singular vectors $\boldsymbol{v}_i$ are the input axes that map to the ellipsoid's axes $\sigma_i \boldsymbol{u}_i$. The spectral norm σ_1 is the longest semi-axis (maximum stretch); σ_n is the shortest.

B.3 The four fundamental subspaces

Every matrix $\boldsymbol{A} \in \mathbb{C}^{m\times N}$ of rank r defines four subspaces: the **column space** (or range) $\mathcal{R}(\boldsymbol{A}) \subseteq \mathbb{C}^m$, the **row space** $\mathcal{R}(\boldsymbol{A}^*) \subseteq \mathbb{C}^N$, the **null space** $\mathcal{N}(\boldsymbol{A}) \subseteq \mathbb{C}^N$, and the **left null space** $\mathcal{N}(\boldsymbol{A}^*) \subseteq \mathbb{C}^m$. Their dimensions are r, r, $N-r$, and $m-r$, and the **rank–nullity** theorem $\operatorname{rank}(\boldsymbol{A}) + \dim \mathcal{N}(\boldsymbol{A}) = N$ ties the first column to the third. The row space and null space are orthogonal complements in $\mathbb{C}^N$, and the column space and left null space are orthogonal complements in $\mathbb{C}^m$ (Fig. B.3).

Example B.4 (The four subspaces of a rank-one matrix). Take $\boldsymbol{A} = \left(\begin{smallmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{smallmatrix}\right)$, whose second row is twice the first, so $r = 1$. The *column space* is spanned by $(1,2)^\top$, since every column is a multiple of it; dimension 1. The *row space* is spanned by $(1,2,3)^\top$; dimension 1. The *null space* solves $(1,2,3)\boldsymbol{x} = 0$, giving $x_1 = -2x_2 - 3x_3$, so it is spanned by $(-2,1,0)^\top$ and $(-3,0,1)^\top$; dimension 2, and indeed $r + \dim \mathcal{N} = 1+2 = 3 = N$. The *left null space* solves $\boldsymbol{y}^\top \boldsymbol{A} = \boldsymbol{0}$, giving $y_1 + 2y_2 = 0$, spanned by $(-2,1)^\top$; dimension $m - r = 1$. Note $(1,2,3)\cdot(-2,1,0) = -2+2+0 = 0$: the row space is orthogonal to the null space. ◊

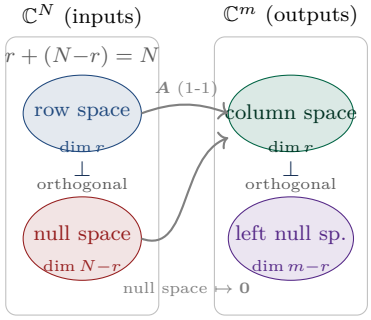


Figure B.3. The four fundamental subspaces. $\boldsymbol{A}$ maps the row space one-to-one onto the column space (green) and collapses the null space (red) to $\boldsymbol{0}$. Within each space the two subspaces are orthogonal complements ($\perp$): row space $\perp$ null space in the input, column space $\perp$ left null space in the output. The input dimensions add to N (rank–nullity).

B.4 Eigenvalues and eigenvectors

A nonzero $\boldsymbol{v}$ is an **eigenvector** of a square $\boldsymbol{B}$ with **eigenvalue** λ if $\boldsymbol{B}\boldsymbol{v} = \lambda\boldsymbol{v}$: the operator only rescales $\boldsymbol{v}$. The eigenvalues are the roots of the characteristic polynomial $\det(\boldsymbol{B} - \lambda\boldsymbol{I}) = 0$.

Example B.5 (A 2×2 eigendecomposition). Let $\boldsymbol{B} = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$. The characteristic polynomial is $(2-\lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = (\lambda - 1)(\lambda - 3)$, so the eigenvalues are $\lambda_1 = 3$ and $\lambda_2 = 1$. For $\lambda_1 = 3$, solving $(\boldsymbol{B} - 3\boldsymbol{I})\boldsymbol{v} = \boldsymbol{0}$ gives $\left(\begin{smallmatrix} -1 & 1 \\ 1 & -1 \end{smallmatrix}\right)\boldsymbol{v} = \boldsymbol{0}$, so $\boldsymbol{v}_1 \propto (1, 1)^\top$. For $\lambda_2 = 1$, $\left(\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right)\boldsymbol{v} = \boldsymbol{0}$ gives $\boldsymbol{v}_2 \propto (1, -1)^\top$. The eigenvectors are orthogonal, as the spectral theorem guarantees for a symmetric matrix. ◊

Theorem B.6 (Spectral theorem). A Hermitian matrix $\boldsymbol{B} = \boldsymbol{B}^*$ has real eigenvalues and an orthonormal eigenbasis: $\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^*$ with $\boldsymbol{Q}$ unitary and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ real.

This diagonalization is the workhorse behind the SVD (Appendix D), principal component analysis (Chapter 12), and positive-semidefinite tests. For Example B.5, $\boldsymbol{Q} = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$ and $\boldsymbol{\Lambda} = \operatorname{diag}(3, 1)$, and one checks $\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top = \boldsymbol{B}$. A **quadratic form** $\boldsymbol{x}^\top\boldsymbol{B}\boldsymbol{x}$ then becomes $\sum_i \lambda_i z_i^2$ in the eigenbasis $\boldsymbol{z} = \boldsymbol{Q}^\top\boldsymbol{x}$, so its sign is governed entirely by the signs of the eigenvalues.

B.5 Positive semidefinite matrices

A Hermitian $\boldsymbol{M}$ is **positive semidefinite** ($\boldsymbol{M} \succeq 0$) if $\boldsymbol{x}^*\boldsymbol{M}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$, equivalently if all its eigenvalues are nonnegative. It is **positive definite** ($\boldsymbol{M} \succ 0$) if the inequality is strict for $\boldsymbol{x} \neq \boldsymbol{0}$. The **Gram matrix** $\boldsymbol{A}^*\boldsymbol{A}$ is always positive semidefinite, since $\boldsymbol{x}^*\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{x} = \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \geq 0$. For a 2×2 symmetric matrix there is a quick test:

$$\begin{pmatrix} a & b \\ b & d \end{pmatrix} \succeq 0 \iff a \geq 0,\ d \geq 0,\ ad - b^2 \geq 0.$$

Example B.7 (A positive-definite check). For $\boldsymbol{M} = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$, the test gives $a = 2 \geq 0$, $d = 2 \geq 0$, and $ad - b^2 = 4 - 1 = 3 > 0$, so $\boldsymbol{M} \succ 0$. This agrees with Example B.5, whose eigenvalues 3 and 1 are both positive. ◊

B.6 Orthogonal projection

The **orthogonal projection** of $\boldsymbol{b}$ onto a subspace $\mathcal{V} = \mathcal{R}(\boldsymbol{A})$ is the closest point of $\mathcal{V}$ to $\boldsymbol{b}$; the residual $\boldsymbol{b} - \operatorname{proj}_{\mathcal{V}}\boldsymbol{b}$ is orthogonal to $\mathcal{V}$. When $\boldsymbol{A}$ has full column rank the projector is

$$\boldsymbol{P} = \boldsymbol{A}(\boldsymbol{A}^*\boldsymbol{A})^{-1}\boldsymbol{A}^*,$$

which satisfies $\boldsymbol{P}^2 = \boldsymbol{P} = \boldsymbol{P}^*$ (idempotent and self-adjoint), the defining algebra of a projector.

Example B.8 (Projection onto a plane). Project $\boldsymbol{b} = (2, 3, 4)^\top$ onto the xy-plane, the column space of $\boldsymbol{A} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{smallmatrix}\right)$. Here $\boldsymbol{A}^*\boldsymbol{A} = \boldsymbol{I}_2$, so $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{A}^* = \operatorname{diag}(1, 1, 0)$ and $\boldsymbol{P}\boldsymbol{b} = (2, 3, 0)^\top$. The residual $\boldsymbol{b} - \boldsymbol{P}\boldsymbol{b} = (0, 0, 4)^\top$ points straight out of the plane, orthogonal to it. ◊

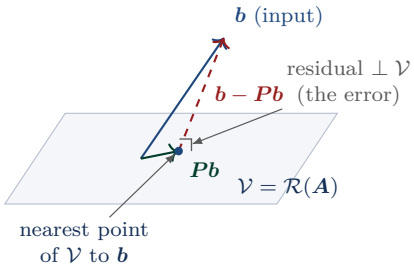


Figure B.4. Orthogonal projection, color-coded input (navy), image (green), residual (red). $\boldsymbol{Pb}$ is the point of $\mathcal{V}$ nearest $\boldsymbol{b}$; the residual $\boldsymbol{b} - \boldsymbol{Pb}$ leaves the foot at a right angle, perpendicular to $\mathcal{V}$. Least squares is exactly this projection: the shortest path from $\boldsymbol{b}$ down to the subspace.

Case	$\boldsymbol{A}^\dagger$	Property
tall, full column rank	$(\boldsymbol{A}^*\boldsymbol{A})^{-1}\boldsymbol{A}^*$	$\boldsymbol{A}^\dagger\boldsymbol{A} = \boldsymbol{I}$, least squares
wide, full row rank	$\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{A}^*)^{-1}$	$\boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{I}$, min-norm solution
general	$\boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^*$	invert nonzero σ_i

Table B.3. The pseudoinverse in three cases. The general form inverts the nonzero singular values and is computed from the SVD (Appendix D).

B.7 Least squares and the pseudoinverse

The least-squares problem $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$ is solved by projecting $\boldsymbol{y}$ onto $\mathcal{R}(\boldsymbol{A})$. Setting the gradient to zero gives the **normal equations** $\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^*\boldsymbol{y}$, whose solution is $\hat{\boldsymbol{x}} = \boldsymbol{A}^\dagger\boldsymbol{y}$ with the **pseudoinverse** of Table B.3.

Example B.9 (A pseudoinverse by hand). Let $\boldsymbol{A} = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, tall with full column rank. Then $\boldsymbol{A}^\top\boldsymbol{A} = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$ with inverse $\frac{1}{3}\left(\begin{smallmatrix} 2 & -1 \\ -1 & 2 \end{smallmatrix}\right)$, so

$$\boldsymbol{A}^\dagger = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top = \tfrac{1}{3}\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} = \tfrac{1}{3}\begin{pmatrix} 1 & -1 & 2 \\ 1 & 2 & -1 \end{pmatrix}.$$

One checks $\boldsymbol{A}^\dagger\boldsymbol{A} = \frac{1}{3}\left(\begin{smallmatrix} 3 & 0 \\ 0 & 3 \end{smallmatrix}\right) = \boldsymbol{I}_2$, confirming the left-inverse property. ◊

For a wide, full-row-rank $\boldsymbol{A}$ the formula $\boldsymbol{A}^\dagger = \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{A}^*)^{-1}$ gives the *minimum-norm* solution of the underdetermined system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, the ℓ_2 analogue of basis pursuit, which is why the book contrasts it so often with the ℓ_1 solution.

B.8 Orthonormalization and conditioning

Gram–Schmidt turns a set of independent vectors into an orthonormal basis for the same span by subtracting off projections.

Example B.10 (Gram–Schmidt on two vectors). Orthonormalize $\boldsymbol{a}_1 = (1,1,0)^\top$, $\boldsymbol{a}_2 = (1,0,1)^\top$. First $\boldsymbol{q}_1 = \boldsymbol{a}_1/\sqrt{2} = (1,1,0)^\top/\sqrt{2}$. Then remove the component of $\boldsymbol{a}_2$ along $\boldsymbol{q}_1$: $\langle \boldsymbol{a}_2, \boldsymbol{q}_1\rangle = 1/\sqrt{2}$, so $\boldsymbol{v}_2 = \boldsymbol{a}_2 - \frac{1}{\sqrt{2}}\boldsymbol{q}_1 = (\frac{1}{2}, -\frac{1}{2}, 1)^\top$, and $\boldsymbol{q}_2 = \boldsymbol{v}_2/\|\boldsymbol{v}_2\| = (1,-1,2)^\top/\sqrt{6}$. Check $\langle \boldsymbol{q}_1, \boldsymbol{q}_2\rangle = (1-1+0)/\sqrt{12} = 0$: orthonormal. ◊

Collecting the $\boldsymbol{q}_i$ as columns of $\boldsymbol{Q}$ and the projection coefficients in an upper-triangular $\boldsymbol{R}$ gives the **QR factorization** $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$, the numerically stable way to solve least squares.

The **condition number** $\kappa(\boldsymbol{A}) = \sigma_1/\sigma_n$ measures how much $\boldsymbol{A}$ can amplify relative errors: solving $\boldsymbol{Ax} = \boldsymbol{y}$ can magnify a relative perturbation in $\boldsymbol{y}$ by up to $\kappa(\boldsymbol{A})$. For $\boldsymbol{A} = \operatorname{diag}(10, 0.1)$, $\kappa = 10/0.1 = 100$, so a third-digit error in $\boldsymbol{y}$ can corrupt the first digit of $\boldsymbol{x}$. A large condition number is exactly the near-rank-deficiency that small σ_n signals, and it is why recovery guarantees demand well-conditioned (incoherent, RIP) submatrices.

B.9 Trace, the Frobenius inner product, and gradients

The **trace** $\operatorname{tr}(\boldsymbol{A}) = \sum_i A_{ii} = \sum_i \lambda_i$ is linear and cyclic, $\operatorname{tr}(\boldsymbol{ABC}) = \operatorname{tr}(\boldsymbol{CAB})$, and gives the Frobenius norm through $\operatorname{tr}(\boldsymbol{A}^*\boldsymbol{A}) = \|\boldsymbol{A}\|_F^2$ and the **Frobenius inner product** $\langle \boldsymbol{A}, \boldsymbol{B}\rangle_F = \operatorname{tr}(\boldsymbol{A}^*\boldsymbol{B})$. The gradients the book uses most are

$$\nabla_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{Ax} - \boldsymbol{y}\|_2^2 = \boldsymbol{A}^*(\boldsymbol{Ax} - \boldsymbol{y}), \qquad \nabla_{\boldsymbol{x}}\, \boldsymbol{x}^\top \boldsymbol{Qx} = 2\boldsymbol{Qx} \;\; (\boldsymbol{Q} = \boldsymbol{Q}^\top).$$

The first is the gradient that the normal equations set to zero, and the second underlies every quadratic step in Chapter 7. For nonsmooth terms, the **subdifferential** of the absolute value is $\partial|x| = \{\operatorname{sgn} x\}$ for $x \neq 0$ and the interval $[-1, 1]$ at $x = 0$; this is what makes the soft-threshold the proximal map of the ℓ_1 norm. Every norm is convex, so for a convex objective every local minimum is global, and strict convexity forces the minimizer to be unique.

Example B.11 (The least-squares gradient by hand). For $\boldsymbol{A} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{smallmatrix}\right)$, $\boldsymbol{y} = (1, 1, 0)^\top$, the gradient at $\boldsymbol{x} = \boldsymbol{0}$ is $\boldsymbol{A}^\top(\boldsymbol{A} \cdot \boldsymbol{0} - \boldsymbol{y}) = -\boldsymbol{A}^\top \boldsymbol{y} = -(1, 1)^\top$. A gradient step from $\boldsymbol{0}$ therefore moves toward $(1, 1)^\top$, decreasing the residual, exactly as the descent methods of Chapter 7 prescribe. ◊

APPENDIX C

Probability and Concentration Reference

This appendix collects the probabilistic tools behind the random-matrix theory of Chapter 9, with a worked example for every inequality. The guiding theme is *concentration of measure*: a sum of many independent random variables is sharply concentrated around its mean, with a failure probability that decays exponentially in the number of terms. That exponential decay is what lets a random matrix satisfy the restricted isometry property with overwhelming probability, and it is the engine of the $m \sim s \log(N/s)$ measurement count.

C.1 Expectation, variance, and the basic tail bounds

The **expectation** $\mathbb{E}[X]$ is the mean and the **variance** $\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}X)^2] = \mathbb{E}[X^2] - (\mathbb{E}X)^2$ measures spread. Two elementary inequalities turn a known mean or variance into a tail bound.

Markov.
For $X \geq 0$ and $a > 0$, $\quad \mathbb{P}[X \geq a] \leq \dfrac{\mathbb{E}[X]}{a}$.

Chebyshev.
$\mathbb{P}[\,|X - \mathbb{E}X| \geq a\,] \leq \dfrac{\mathrm{Var}(X)}{a^2}$, obtained by applying Markov to $(X - \mathbb{E}X)^2$.

Example C.1 (Markov and Chebyshev in numbers). Let $X \geq 0$ have $\mathbb{E}[X] = 2$. Markov bounds $\mathbb{P}[X \geq 10] \leq 2/10 = 0.2$, with no further information needed. If in addition $\mathrm{Var}(X) = 4$, then Chebyshev bounds $\mathbb{P}[\,|X - 2| \geq 6\,] \leq 4/36 = 1/9 \approx 0.11$, a tighter statement because it uses the spread, not just the mean. Both are *distribution-free*: they hold for every nonnegative X with those moments, which is also why they are loose for any specific distribution. ◊

These polynomial-decay bounds are too weak for the union bounds the book needs. The fix is the exponential moment.

C.2 The Chernoff method

The **Chernoff bound** applies Markov to $e^{\lambda X}$ and then optimizes the free parameter $\lambda > 0$:

$$\mathbb{P}[X \geq a] = \mathbb{P}[e^{\lambda X} \geq e^{\lambda a}] \leq e^{-\lambda a}\,\mathbb{E}[e^{\lambda X}], \qquad \text{then minimize over } \lambda.$$

The quantity $\mathbb{E}[e^{\lambda X}]$ is the **moment generating function** (MGF). Its one magical property is that for *independent* variables it factorizes,

$$\mathbb{E}\Big[e^{\lambda \sum_i X_i}\Big] = \prod_i \mathbb{E}[e^{\lambda X_i}],$$

turning a sum in the exponent into a product of MGFs. Each factor contributes, and the product of m factors below one gives the decay e^{-cm} that powers every concentration result in Chapter 9.

Example C.2 (Chernoff for a sum of Bernoullis). Let $X = \sum_{i=1}^{n} B_i$ with independent $B_i \sim \text{Bernoulli}(p)$, mean $\mu = np$. Each MGF is $\mathbb{E}[e^{\lambda B_i}] = 1 - p + pe^{\lambda} = 1 + p(e^{\lambda} - 1) \leq \exp\big(p(e^{\lambda} - 1)\big)$, using $1 + u \leq e^{u}$. Multiplying, $\mathbb{E}[e^{\lambda X}] \leq \exp\big(\mu(e^{\lambda} - 1)\big)$, so $\mathbb{P}[X \geq (1+\delta)\mu] \leq \exp\big(\mu(e^{\lambda} - 1) - \lambda(1+\delta)\mu\big)$. Minimizing over λ at $\lambda = \ln(1+\delta)$ yields the standard multiplicative bound

$$\mathbb{P}[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \leq e^{-\mu\delta^2/3}, \quad 0 < \delta \leq 1.$$

The decay is exponential in the mean μ, not polynomial: this is the qualitative jump over Example C.1. ◊

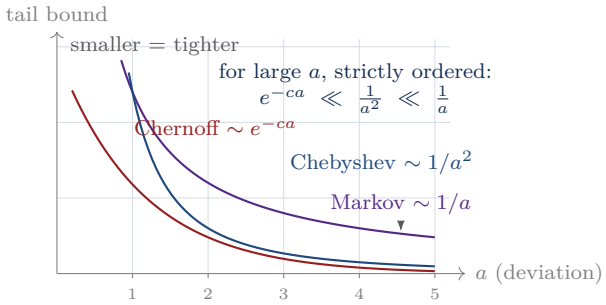


Figure C.1. Three tail bounds on the same deviation. Markov decays like $1/a$, Chebyshev like $1/a^2$, and the Chernoff/exponential bound like e^{-ca}. In the tail the order is strict, $e^{-ca} \ll 1/a^2 \ll 1/a$: the exponential bound hugs the axis. Only that exponential decay survives a union bound over exponentially many events, which is why the random-matrix proofs use the moment generating function.

C.3 Gaussian and chi-squared concentration

The Gaussian is the canonical light-tailed variable. The single computation the book leans on is the MGF of a squared standard Gaussian.

Proposition C.3 (Chi-squared MGF). If $z \sim \mathcal{N}(0,1)$ then $\mathbb{E}[e^{\lambda z^2}] = (1-2\lambda)^{-1/2}$ for $\lambda < \frac{1}{2}$.

Proof. Complete the square in the Gaussian integral,

$$\mathbb{E}[e^{\lambda z^2}] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} e^{\lambda z^2}\, dz = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-(1-2\lambda)z^2/2}\, dz = \frac{1}{\sqrt{1-2\lambda}},$$

the last step rescaling z to recognize a Gaussian of variance $(1-2\lambda)^{-1}$. The condition $\lambda < \frac{1}{2}$ keeps the integral finite, which is the signature of a *sub-exponential* (not sub-Gaussian) tail. ■

Summing m independent squares, $\chi^2 = \sum_{i=1}^{m} z_i^2$ is chi-squared with m degrees of freedom and mean m. Applying the Chernoff method with Proposition C.3 gives the two-sided concentration used in the random-matrix point estimate of Lemma 9.2,

$$\mathbb{P}\big[\,|\chi^2 - m| > \varepsilon m\,\big] \;\le\; 2\,e^{-(\varepsilon^2-\varepsilon^3)m/4}, \qquad \varepsilon \in (0,1).$$

Example C.4 (How sharp is the χ^2 bound?). Take $m = 100$ and $\varepsilon = 0.2$. The bound gives $2e^{-(0.04-0.008)\cdot 100/4} = 2e^{-0.8} \approx 0.90$, weak at this size. At $m = 1000$ it is $2e^{-8} \approx 7 \times 10^{-4}$, and at $m = 4000$ it is $2e^{-32} \approx 2 \times 10^{-14}$. The failure probability collapses as m grows, which is exactly the regime ($m \gtrsim s \log N$) the recovery theorems live in. ◊

C.4 Sub-Gaussian and sub-exponential variables

A mean-zero X is **sub-Gaussian** with parameter K if its tails are no heavier than a Gaussian's, equivalently $\mathbb{E}[e^{\lambda X}] \le e^{K^2\lambda^2/2}$ for all λ. Gaussian and bounded variables (such as the ± 1 Bernoulli of a sign matrix) are sub-Gaussian. The *square* of a sub-Gaussian variable is **sub-exponential**, with a heavier tail controlled only for small λ, exactly the $\lambda < \frac{1}{2}$ restriction of Proposition C.3. Figure C.2 contrasts the two tail shapes.

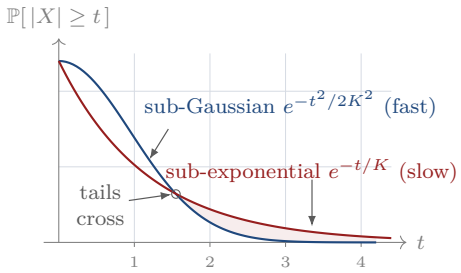


Figure C.2. Sub-Gaussian tails fall off like e^{-t^2} (fast); sub-exponential tails, the kind produced by squaring, fall off only like e^{-t} (slower, but still exponential). The curves cross near $t \approx 1.5$; past it the sub-exponential lies above (shaded), the heavier tail that Bernstein's inequality is built to handle.

Hoeffding.
For independent $X_i \in [a_i, b_i]$, $\mathbb{P}\big[\,|\sum_i (X_i - \mathbb{E}X_i)| \ge t\,\big] \le 2\exp\big(-2t^2/\sum_i (b_i - a_i)^2\big)$.

Bernstein.
For independent mean-zero sub-exponential X_i, $\mathbb{P}\big[\,|\sum_i X_i| \ge t\,\big] \le 2\exp\big(-c\min(t^2/\sigma^2,\, t/M)\big)$, with a Gaussian (small-deviation) regime t^2/σ^2 and an exponential (large-deviation) regime t/M. Bernstein extends the Gaussian point estimate of Chapter 9 to general sub-Gaussian ensembles, because the relevant quantities are sums of squared sub-Gaussians.

Example C.5 (Hoeffding on coin flips). Flip a fair coin $n = 200$ times and let $\bar{X}$ be the fraction of heads, mean $\frac{1}{2}$. With each $X_i \in [0,1]$, Hoeffding for the average gives $\mathbb{P}[\,|\bar{X} - \frac{1}{2}| \ge t\,] \le 2e^{-2nt^2}$. At $t = 0.1$ this is $2e^{-2\cdot 200\cdot 0.01} = 2e^{-4} \approx 0.037$: at most a 3.7% chance the empirical rate strays by more than ten points. Doubling n to 400 squares the bound to $2e^{-8} \approx 7 \times 10^{-4}$. ◊

C.5 The union bound and covering numbers

The **union bound** $\mathbb{P}[\bigcup_i E_i] \le \sum_i \mathbb{P}[E_i]$ turns a guarantee at one point into a guarantee over a whole set, at the cost of a multiplicative count of the points. The art is keeping that count small, which is what *covering numbers* do.

Proposition C.6 (Covering number of a ball). The unit ball of $\mathbb{R}^s$ admits a ρ-net (a finite set within distance ρ of every point) of size at most $(1 + 2/\rho)^s$.

Example C.7 (Why the count is benign). For $s = 5$ and $\rho = \frac{1}{2}$, the net has size at most $(1 + 4)^5 = 5^5 = 3125$. Its logarithm is $5 \ln 5 \approx 8.0$, *linear* in the dimension s. When the union bound multiplies an e^{-cm} point estimate by e^8 net points, the product is still e^{8-cm}, tiny once $m \gtrsim s$. This linear-in-s logarithm is precisely the source of the $s \log(N/s)$ measurement count: covering the s-sparse signals costs $\binom{N}{s} \leq (eN/s)^s$ supports times $(1+2/\rho)^s$ directions, and the logarithm of that is $O(s \log(N/s))$. ◊

The **Johnson–Lindenstrauss lemma** (Theorem 9.3) is the union bound applied to the $\binom{n}{2}$ pairwise differences of a finite point set: a random projection to dimension $m \sim \varepsilon^{-2} \log n$ preserves all pairwise distances to within $1 \pm \varepsilon$, with the target dimension independent of the ambient dimension. It is the cleanest illustration of the whole method: a point estimate from the moment generating function, amplified across a controlled number of events by the union bound.

APPENDIX D

The Singular Value Decomposition

The singular value decomposition underlies half the book, so it gets a focused treatment: existence, a worked computation by hand, the geometry, the four subspaces it exposes, and the Eckart–Young theorem that makes truncation the best low-rank approximation. Every abstract statement is paired with a concrete computation.

D.1 Existence

Theorem D.1 (SVD). Every $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ factors as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ with $\boldsymbol{U} \in \mathbb{R}^{m\times m}$, $\boldsymbol{V} \in \mathbb{R}^{N\times N}$ orthogonal and $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times N}$ diagonal with entries $\sigma_1 \geq \cdots \geq 0$.

Proof. The matrix $\boldsymbol{A}^\top\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is symmetric and positive semidefinite, so by the spectral theorem (Theorem B.6) it has an orthonormal eigenbasis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ with real eigenvalues $\lambda_1 \geq \cdots \geq \lambda_N \geq 0$. Set $\sigma_i = \sqrt{\lambda_i}$. For the indices with $\sigma_i > 0$ define $\boldsymbol{u}_i = \boldsymbol{A}\boldsymbol{v}_i/\sigma_i$; these are orthonormal, since $\boldsymbol{u}_i^\top\boldsymbol{u}_j = \boldsymbol{v}_i^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{v}_j/(\sigma_i\sigma_j) = \lambda_j\,\boldsymbol{v}_i^\top\boldsymbol{v}_j/(\sigma_i\sigma_j) = \delta_{ij}$. Extend them to an orthonormal basis $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m$ of $\mathbb{R}^m$. Then $\boldsymbol{A}\boldsymbol{v}_i = \sigma_i\boldsymbol{u}_i$ for every i (both sides vanish when $\sigma_i = 0$, because $\|\boldsymbol{A}\boldsymbol{v}_i\|_2^2 = \lambda_i = 0$), which is exactly $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Sigma}$, i.e. $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. ∎

The columns of $\boldsymbol{V}$ are the *right* singular vectors (eigenvectors of $\boldsymbol{A}^\top\boldsymbol{A}$), the columns of $\boldsymbol{U}$ are the *left* singular vectors (eigenvectors of $\boldsymbol{A}\boldsymbol{A}^\top$), and the singular values are the square roots of the shared eigenvalues. The proof is also the recipe, which we now run on a concrete matrix.

Example D.2 (An SVD computed by hand). Take $\boldsymbol{A} = \left(\begin{smallmatrix} 3 & 0 \\ 4 & 5 \end{smallmatrix}\right)$. Form $\boldsymbol{A}^\top\boldsymbol{A} = \left(\begin{smallmatrix} 25 & 20 \\ 20 & 25 \end{smallmatrix}\right)$, whose characteristic polynomial $(25-\lambda)^2 - 400 = 0$ gives $25-\lambda = \pm 20$, so $\lambda = 45$ and $\lambda = 5$. The singular values are $\sigma_1 = \sqrt{45} = 3\sqrt{5} \approx 6.71$ and $\sigma_2 = \sqrt{5} \approx 2.24$. Two sanity checks pass at once: $\sigma_1\sigma_2 = 3\sqrt{5}\cdot\sqrt{5} = 15 = |\det \boldsymbol{A}|$ and $\sigma_1^2 + \sigma_2^2 = 45 + 5 = 50 = 3^2 + 4^2 + 5^2 = \|\boldsymbol{A}\|_F^2$.

The right singular vectors solve $(\boldsymbol{A}^\top\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{v} = \boldsymbol{0}$. For $\lambda = 45$, $-20v_1 + 20v_2 = 0$ gives $\boldsymbol{v}_1 = \frac{1}{\sqrt{2}}(1,1)^\top$; for $\lambda = 5$, $20v_1 + 20v_2 = 0$ gives $\boldsymbol{v}_2 = \frac{1}{\sqrt{2}}(1,-1)^\top$. The left singular vectors are $\boldsymbol{u}_i = \boldsymbol{A}\boldsymbol{v}_i/\sigma_i$:

$$\boldsymbol{u}_1 = \frac{\boldsymbol{A}(1,1)^\top/\sqrt{2}}{3\sqrt{5}} = \frac{(3,9)^\top}{3\sqrt{10}} = \frac{1}{\sqrt{10}}(1,3)^\top, \qquad \boldsymbol{u}_2 = \frac{\boldsymbol{A}(1,-1)^\top/\sqrt{2}}{\sqrt{5}} = \frac{1}{\sqrt{10}}(3,-1)^\top,$$

which are orthonormal ($\boldsymbol{u}_1^\top\boldsymbol{u}_2 = (3-3)/10 = 0$). Assembled,

$$\boldsymbol{A} = \underbrace{\frac{1}{\sqrt{10}}\begin{pmatrix} 1 & 3 \\ 3 & -1 \end{pmatrix}}_{\boldsymbol{U}} \underbrace{\begin{pmatrix} 3\sqrt{5} & 0 \\ 0 & \sqrt{5} \end{pmatrix}}_{\boldsymbol{\Sigma}} \underbrace{\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}}_{\boldsymbol{V}^\top}. \diamondsuit$$

D.2 Geometry: rotate, stretch, rotate

Because $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal (rotations or reflections) and $\boldsymbol{\Sigma}$ is diagonal (axis-aligned scaling), $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ reads as three steps: rotate the input by $\boldsymbol{V}^\top$, stretch each axis by σ_i, rotate the result by $\boldsymbol{U}$. Every linear map is a rotation, a scaling, and another rotation (Fig. D.1). The unit sphere becomes an ellipsoid with semi-axes $\sigma_i \boldsymbol{u}_i$, which is the picture behind the spectral norm σ_1 and the condition number σ_1/σ_n of Appendix B.

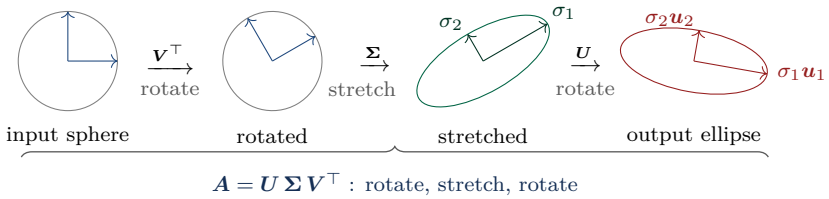


Figure D.1. The SVD as rotate ($\boldsymbol{V}^\top$), stretch ($\boldsymbol{\Sigma}$), rotate ($\boldsymbol{U}$). The stretch step $\boldsymbol{\Sigma}$ is where the singular values σ_i enter as the semi-axis lengths; the two rotations only reorient the frame. The right singular vectors are the input axes and the left singular vectors $\boldsymbol{u}_i$ the output axes.

The rank-one-sum form makes the four subspaces explicit:

$$\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i\, \boldsymbol{u}_i \boldsymbol{v}_i^\top, \qquad r = \operatorname{rank}(\boldsymbol{A}).$$

The first r columns of $\boldsymbol{U}$ span the column space, the remaining $m - r$ span the left null space; the first r columns of $\boldsymbol{V}$ span the row space, the remaining $N - r$ span the null space. The SVD is the cleanest route to every quantity in Appendix B.3. The **polar decomposition** $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{P}$ with $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{V}^\top$ orthogonal and $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^\top \succeq 0$ follows by inserting $\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}$; it is the matrix analogue of writing a complex number as (phase)×(magnitude).

D.3 Eckart–Young: truncation is optimal

Theorem D.3 (Eckart–Young, 1936). The best rank-k approximation of $\boldsymbol{A}$ in the Frobenius and spectral norms is the truncated SVD $\boldsymbol{A}_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$, with errors

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_{2\to 2} = \sigma_{k+1}, \qquad \|\boldsymbol{A} - \boldsymbol{A}_k\|_F = \Big(\sum_{i>k} \sigma_i^2\Big)^{1/2}.$$

Proof of the rank-one case. We prove $k = 1$; the general case is identical with the top-k subspace in place of the top vector. Any rank-one matrix can be written $\sigma \boldsymbol{u}\boldsymbol{v}^\top$ with unit $\boldsymbol{u}, \boldsymbol{v}$. Expanding the squared Frobenius distance and using $\|\boldsymbol{u}\boldsymbol{v}^\top\|_F = 1$,

$$\|\boldsymbol{A} - \sigma \boldsymbol{u}\boldsymbol{v}^\top\|_F^2 = \|\boldsymbol{A}\|_F^2 - 2\sigma\, \boldsymbol{u}^\top \boldsymbol{A}\boldsymbol{v} + \sigma^2,$$

which is minimized over σ at $\sigma = \boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v}$, giving residual $\|\boldsymbol{A}\|_F^2 - (\boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v})^2$. This is smallest when $\boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v}$ is largest, and the maximum of $\boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v}$ over unit vectors is σ_1, attained at the top singular pair $(\boldsymbol{u}_1, \boldsymbol{v}_1)$. So the best rank-one approximation is $\sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^\top$, with Frobenius error $\sqrt{\|\boldsymbol{A}\|_F^2 - \sigma_1^2} = \big(\sum_{i\ge 2} \sigma_i^2\big)^{1/2}$ and spectral error σ_2. ∎

Example D.4 (Low-rank approximation in numbers). Suppose $\boldsymbol{A}$ is 3×3 with singular values $10, 1, 0.1$. The best rank-one fit keeps $\sigma_1 = 10$ and discards the rest, with Frobenius error $\sqrt{1^2 + 0.1^2} = \sqrt{1.01} \approx 1.005$ against a total $\|\boldsymbol{A}\|_F = \sqrt{100 + 1 + 0.01} \approx 10.05$, a relative error of about 10%. The *energy* captured is $\sigma_1^2 / \sum_i \sigma_i^2 = 100/101.01 \approx 99.0\%$: one component already explains nearly all of the matrix. This is why a slowly decaying singular spectrum compresses well, and it is exactly the variance-capture computation of principal component analysis (Chapter 12). ◊

Truncation is the matrix analogue of best s-term approximation: keep the largest singular values, drop the rest. It is the engine of principal component analysis (Chapter 12), the per-atom update of K-SVD (Chapter 13), and the singular-value thresholding that solves nuclear-norm problems (Chapters 11 and 12).

D.4 SVD versus eigendecomposition

The two factorizations are cousins but not the same, and the distinctions matter in practice (Table D.1). The SVD exists for every matrix, even rectangular ones, with nonnegative singular values; the eigendecomposition needs a square matrix and may have complex or repeated eigenvalues and non-orthogonal eigenvectors. They coincide exactly when $\boldsymbol{A}$ is symmetric positive semidefinite, where $\sigma_i = \lambda_i$ and the singular vectors are the eigenvectors. In general, the singular values of $\boldsymbol{A}$ are the square roots of the eigenvalues of $\boldsymbol{A}^\top \boldsymbol{A}$.

	Eigendecomposition	**SVD**
Applies to	square matrices	any $m \times N$ matrix
Form	$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$	$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$
Scalars	eigenvalues (can be complex/negative)	singular values ≥ 0
Vectors	eigenvectors (orthogonal iff normal)	always two orthonormal sets

Table D.1. The SVD is the universal factorization; the eigendecomposition is its special case for symmetric positive-semidefinite matrices, where the two agree.

- $\sum_i \sigma_i^2 = \|\boldsymbol{A}\|_F^2$; for square $\boldsymbol{A}$, $\prod_i \sigma_i = |\det \boldsymbol{A}|$.
- $\sigma_1 = \|\boldsymbol{A}\|_{2\to 2}$ (largest stretch); σ_n (smallest) measures how close $\boldsymbol{A}$ is to rank-deficient, and σ_1/σ_n is the condition number.
- Pseudoinverse $\boldsymbol{A}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^\top$, inverting the nonzero singular values.
- Rank $= \|\boldsymbol{\sigma}\|_0$; nuclear norm $= \|\boldsymbol{\sigma}\|_1$; this is the vector–matrix dictionary of Chapter 11.
- Singular-value perturbation (Lemma 11.3): per index (Weyl) $|\sigma_i(\boldsymbol{A}) - \sigma_i(\boldsymbol{B})| \leq \sigma_1(\boldsymbol{A} - \boldsymbol{B}) = \|\boldsymbol{A} - \boldsymbol{B}\|_2$; summed (Mirsky) $\sum_i |\sigma_i(\boldsymbol{A}) - \sigma_i(\boldsymbol{B})| \leq \sum_i \sigma_i(\boldsymbol{A} - \boldsymbol{B}) = \|\boldsymbol{A} - \boldsymbol{B}\|_*$.

APPENDIX E

Convex Optimization Reference

Every recovery program in the book is a convex optimization problem, and the reason they are solvable at all is the structure convexity provides. This appendix collects that structure: convex sets and functions, the linear program that ℓ_1 minimization becomes, the simplex and interior-point methods that solve it, and the duality and optimality conditions that tie the basis pursuit family together. As in the other appendices, every idea is paired with a worked example or a picture.

E.1 Convex sets and functions

A set C is **convex** if it contains the segment between any two of its points: $\boldsymbol{x}, \boldsymbol{y} \in C$ and $\theta \in [0,1]$ imply $\theta\boldsymbol{x}+(1-\theta)\boldsymbol{y} \in C$. A function f is **convex** if its graph lies below its chords,

$$f\big(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}\big) \le \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y}),$$

equivalently (for twice-differentiable f) if its Hessian is positive semidefinite. Every norm is convex, every affine set is convex, and the intersection of convex sets is convex.

Example E.1 (Checking convexity). The function $f(x) = x^2$ has $f''(x) = 2 > 0$, so it is convex. Jensen's inequality is then visible at a point: with $\boldsymbol{x} = 0$, $\boldsymbol{y} = 2$, $\theta = \frac{1}{2}$,

$$f(\tfrac{1}{2}\cdot 0 + \tfrac{1}{2}\cdot 2) = f(1) = 1 \;\le\; \tfrac{1}{2}f(0) + \tfrac{1}{2}f(2) = \tfrac{1}{2}\cdot 0 + \tfrac{1}{2}\cdot 4 = 2,$$

the midpoint of the graph sits below the midpoint of the chord. The ℓ_1 ball is convex because it is the unit ball of a norm; the feasible set $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}\}$ is convex because it is affine. ◊

Theorem E.2 (Local minima are global). For a convex f over a convex set C, every local minimum is a global minimum. If f is strictly convex, the minimizer is unique.

Proof. Suppose $\boldsymbol{x}^\star$ is a local minimum but some $\boldsymbol{z} \in C$ has $f(\boldsymbol{z}) < f(\boldsymbol{x}^\star)$. By convexity, for small $\theta > 0$ the point $\boldsymbol{x}_\theta = (1-\theta)\boldsymbol{x}^\star + \theta\boldsymbol{z} \in C$ satisfies $f(\boldsymbol{x}_\theta) \le (1-\theta) f(\boldsymbol{x}^\star) + \theta f(\boldsymbol{z}) < f(\boldsymbol{x}^\star)$, and $\boldsymbol{x}_\theta \to \boldsymbol{x}^\star$ as $\theta \to 0$, contradicting local minimality. ■

This is the entire reason the relaxations of Chapter 5 are useful: a solver that finds *a* minimum has found *the* minimum, with no false valleys to escape. The combinatorial (P_0) is nonconvex (its feasible structure is a union of coordinate subspaces), which is exactly why it is hard.

E.2 Linear programming and standard form

A **linear program** (LP) minimizes a linear objective over a polyhedron. The standard form is

$$\min_{\boldsymbol{z}} \; \boldsymbol{c}^\top \boldsymbol{z} \quad \text{subject to} \quad \boldsymbol{B}\boldsymbol{z} = \boldsymbol{y},\; \boldsymbol{z} \ge \boldsymbol{0}.$$

Basis pursuit becomes one through the positive–negative split of Definition 5.2: writing $\boldsymbol{x} = \boldsymbol{u} - \boldsymbol{v}$ with $\boldsymbol{u}, \boldsymbol{v} \ge 0$ turns $\|\boldsymbol{x}\|_1$ into the linear $\mathbf{1}^\top\boldsymbol{u} + \mathbf{1}^\top\boldsymbol{v}$ and the constraint into $[\boldsymbol{A} \;\; -\boldsymbol{A}][\boldsymbol{u};\boldsymbol{v}] = \boldsymbol{y}$, so $\boldsymbol{c} = \mathbf{1}$, $\boldsymbol{B} = [\boldsymbol{A} \;\; -\boldsymbol{A}]$, $\boldsymbol{z} = [\boldsymbol{u};\boldsymbol{v}]$.

Example E.3 (A tiny LP solved by hand). Let $\boldsymbol{A} = (1, -1, 1)$ and $\boldsymbol{y} = 2$, so $\boldsymbol{B} = (1, -1, 1, -1, 1, -1)$ and the LP is $\min \mathbf{1}^\top \boldsymbol{z}$ subject to $z_1 - z_2 + z_3 - z_4 + z_5 - z_6 = 2$, $\boldsymbol{z} \geq \mathbf{0}$. To satisfy the constraint with the least total mass, load a single $+1$-coefficient variable: $z_1 = 2$, the rest zero, objective 2, giving $\boldsymbol{x} = (2, 0, 0)^\top$. The choice $z_3 = 2$ is equally optimal, giving $\boldsymbol{x} = (0, 0, 2)^\top$; both are 1-sparse. The optimum is a *vertex* of the feasible polyhedron, and every vertex here is a 1-sparse point. ◊

E.3 Vertices and the simplex method

The geometry of Example E.3 is general: a linear objective over a polyhedron attains its minimum at a **vertex** (an extreme point), so one never has to search the interior, only the corners. A vertex of the standard-form polyhedron has at most m nonzero coordinates, which is the linear-programming proof of the sparsity bound $\|\boldsymbol{x}^\star\|_0 \leq m$ of Corollary 5.6.

Dantzig's **simplex method** (1947) exploits this by walking from vertex to vertex along the edges of the polyhedron, each step moving to an adjacent vertex of lower objective, until no neighbor improves (Fig. E.1). It is exact and fast in practice, though its worst case is exponential.

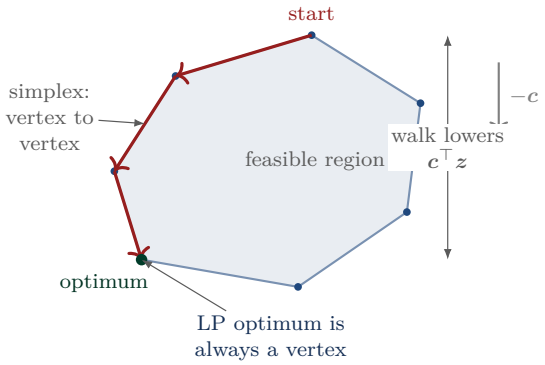


Figure E.1. The simplex method walks the edges of the feasible polyhedron from a starting vertex to the optimal one, lowering the objective $\boldsymbol{c}^\top \boldsymbol{z}$ at each step (the dimension bar on the right shows the total drop). The minimum of a linear objective is always at a vertex, so only the corners matter.

E.4 Interior-point methods and the central path

Karmarkar's **interior-point method** (1984) takes the opposite route: rather than crawl the boundary, it cuts through the interior along a smooth *central path* toward the optimum, taking Newton steps on a barrier-regularized objective $\boldsymbol{c}^\top \boldsymbol{z} - \frac{1}{t} \sum_j \log z_j$ that blows up near the boundary and keeps the iterates strictly feasible (Fig. E.2). As the barrier weight $1/t$ shrinks, the path bends toward the optimal vertex. Interior-point methods solve an LP in provably polynomial time, roughly $O(N^3)$ per solve. It was the first *practical* polynomial-time method for linear programming, after Khachiyan's ellipsoid method (1979) had established that the problem is polynomial-time solvable in principle.

Both methods are exact, but both cost roughly $O(N^3)$ and solve a dense linear system per iteration, which is too slow for the million-dimensional problems compressed sensing produces. That is why the book turns to the first-order proximal methods of Chapter 7, whose per-iteration cost is a cheap matrix-vector product with $\boldsymbol{A}$ rather than a dense solve.

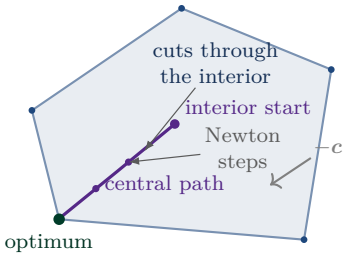


Figure E.2. An interior-point method follows the central path from inside the polyhedron to the optimal vertex (dots mark the Newton iterates), staying strictly feasible by a logarithmic barrier so the path never touches the boundary. It reaches the optimum in polynomial time, unlike the simplex walk along the edges.

E.5 Duality and the KKT conditions

Every convex program has a **dual**, and the two optima coincide (*strong duality*) under mild conditions. For the standard LP the dual is

$$\max_{\boldsymbol{\nu}} \ \boldsymbol{y}^\top \boldsymbol{\nu} \quad \text{subject to} \quad \boldsymbol{B}^\top \boldsymbol{\nu} \le \boldsymbol{c},$$

and weak duality, $\boldsymbol{y}^\top \boldsymbol{\nu} \le \boldsymbol{c}^\top \boldsymbol{z}$ for any feasible pair, is one line: $\boldsymbol{y}^\top \boldsymbol{\nu} = (\boldsymbol{B}\boldsymbol{z})^\top \boldsymbol{\nu} = \boldsymbol{z}^\top (\boldsymbol{B}^\top \boldsymbol{\nu}) \le \boldsymbol{z}^\top \boldsymbol{c}$, using $\boldsymbol{z} \ge \boldsymbol{0}$ and $\boldsymbol{B}^\top \boldsymbol{\nu} \le \boldsymbol{c}$.

Example E.4 (The dual of the tiny LP). For Example E.3, $\boldsymbol{B}^\top \boldsymbol{\nu} \le \boldsymbol{1}$ with $\boldsymbol{B} = (1, -1, 1, -1, 1, -1)$ reads $\nu \le 1$ and $-\nu \le 1$, that is $\nu \in [-1, 1]$, and the dual maximizes 2ν. The dual optimum is $\nu = 1$ with value 2, equal to the primal optimum of 2: strong duality holds, and the matching values certify both solutions at once. ◊

For differentiable problems, optimality is captured by the **Karush–Kuhn–Tucker (KKT)** conditions: stationarity (the gradient of the Lagrangian vanishes), primal and dual feasibility, and complementary slackness. The KKT conditions are what make the basis pursuit family of Section 5.4 equivalent: BPDN is the Lagrangian relaxation of quadratically constrained basis pursuit, with λ the multiplier on the residual constraint, and the LASSO is the constrained form whose budget corresponds monotonically to λ.

Example E.5 (KKT gives soft thresholding). The one-variable BPDN problem $\min_x \frac{1}{2}(x - y)^2 + \lambda |x|$ has the stationarity condition $0 \in (x - y) + \lambda\, \partial|x|$. For $x > 0$ this is $x = y - \lambda$ (valid when $y > \lambda$); for $x < 0$, $x = y + \lambda$ (valid when $y < -\lambda$); and for $|y| \le \lambda$, $x = 0$. Collecting the cases is exactly $x = \mathcal{S}_\lambda(y)$, the soft-thresholding prox of Chapter 7. The optimality conditions of convex analysis are where the proximal operators come from. ◊

E.6 From linear programs to first-order methods

The thread of the book's algorithms is a steady retreat from exactness toward scalability. The ℓ_1 problem is an LP, solvable exactly by simplex or interior-point, but at $O(N^3)$ those are too slow when N is in the millions. Moving to the unconstrained LASSO/BPDN form and solving it by proximal gradient descent (ISTA, accelerated to FISTA) replaces the dense linear solve with a matrix-vector product against $\boldsymbol{A}$, the only operation cheap enough at scale, and the structured transforms of Chapter 10 make even that product fast. The greedy methods of Chapter 6 take a different scalability route, building the support one atom at a time. Table E.1 places the solvers on the accuracy-versus-scale spectrum the book traverses.

Method	Cost	Niche
Simplex	exponential worst case	small exact LPs
Interior-point	$O(N^3)$, polynomial	medium problems, high accuracy
ISTA / FISTA	$O(N \log N)$ per step (structured)	large-scale, the practical default
Greedy (OMP/CoSaMP)	$O(s\,mN)$	fast approximate recovery, known sparsity

Table E.1. The solver spectrum, from exact and small (simplex) to scalable and approximate (first-order and greedy). Compressed sensing lives at the scalable end.

APPENDIX F

A Timeline of Compressed Sensing

The ideas in this book accumulated over nearly a century. This appendix tells the story in order, era by era, then collects the dates in a reference table. The arc is one idea sharpened over time: real signals have few degrees of freedom, and a few good measurements plus a convex program suffice to recover them.

F.1 The classical barrier (1928–1948)

The story starts with a wall. In 1928 Harry Nyquist quantified how fast a bandlimited signal must be sampled to be reconstructed, and in 1948 Claude Shannon, in the same papers that founded information theory, made it a theorem: to capture every frequency up to B, sample at rate $2B$. The **Nyquist–Shannon sampling theorem** became the unquestioned law of signal acquisition. It says nothing about *sparsity*, though, because it assumes the worst case in which every frequency up to the band limit might be present. A signal that is sparse in frequency carries far less information than the band limit suggests, and the classical theorem leaves that slack unused. Compressed sensing is the discovery that the slack is enormous.

F.2 The seeds (1936–1995)

Several tools were forged long before anyone assembled them. In 1936 Carl Eckart and Gale Young proved that the truncated singular value decomposition is the best low-rank approximation, the result that makes the matrix half of this book possible. In 1974 Lloyd Welch proved the lower bound on the coherence of a redundant system, the limit on how incoherent a dictionary can be. In 1984 William Johnson and Joram Lindenstrauss showed that a random projection preserves pairwise distances, the lemma that random sensing matrices turn out to satisfy. By the early 1990s the greedy idea was in the air: Mallat and Zhang's **matching pursuit** (1993) and the **orthogonal matching pursuit** of Pati, Rezaiifar, and Krishnaprasad (1993) built sparse approximations one atom at a time. In 1995 Natarajan proved the bad news that made the good news surprising: finding the sparsest exact representation, the ℓ_0 problem, is NP-hard.

F.3 Convex relaxation (1996–2004)

If the combinatorial problem is intractable, relax it. In 1996 Robert Tibshirani introduced the **LASSO** in statistics, fitting regressions with an ℓ_1 penalty that drives coefficients to exactly zero. In 1998 Chen, Donoho, and Saunders introduced **basis pursuit**, the constrained ℓ_1 program, and showed it routinely found sparse representations where greedy methods stalled. The same years saw the algorithmic engine assembled: Daubechies, Defrise, and De Mol gave the iterative soft-thresholding algorithm (2004), and Tropp's analysis of orthogonal matching pursuit ("greed is good," 2004) put the greedy methods on rigorous footing. Donoho and Elad introduced the **spark** and the uncertainty principle for two bases (2003), the first clean uniqueness condition. The pieces were on the table; what was missing was the theorem that ℓ_1 *always* works under the right conditions.

F.4 The breakthrough (2004–2006)

It arrived in a burst. Emmanuel Candès, Justin Romberg, and Terence Tao, and independently David Donoho, proved that an s-sparse signal in $\mathbb{R}^N$ can be recovered *exactly* from only $O(s \log(N/s))$ random linear measurements by ℓ_1 minimization, far below the Nyquist rate. The enabling concept was the **restricted isometry property**: a random matrix nearly preserves the length of every sparse vector, and that alone guarantees the convex program returns the true signal. The demonstration that made the field was visual. Reconstructing a standard test phantom from a random subset of its Fourier coefficients, well under the Nyquist count, produced an image *indistinguishable* from the original, the recovered and true signals lying exactly on top of each other. A picture that classical theory said could not be reconstructed was reconstructed perfectly. The reaction in the community was that it looked like magic, and the rest of the field is the explanation of why it is not.

F.5 Algorithms and applications (2006–2009)

Once the theory was secure, the applications and faster solvers came quickly. Aharon, Elad, and Bruckstein introduced **K-SVD** for learning dictionaries (2006). Duarte and collaborators built the **single-pixel camera** (2006), acquiring images through a single photodiode and random masks. Lustig, Donoho, and Pauly brought compressed sensing to **MRI** (2007), where shorter scans matter most for children who cannot hold still. Baraniuk, Davenport, DeVore, and Wakin gave a short concentration proof of the restricted isometry property (2008). On the algorithm side, Needell and Tropp's **CoSaMP** and Dai and Milenkovic's **subspace pursuit** (2009) made greedy recovery provably as good as ℓ_1, and Beck and Teboulle's **FISTA** (2009) accelerated iterative thresholding to the optimal first-order rate.

F.6 The matrix era (2009–2011)

The same logic then lifted from vectors to matrices. Candès and Recht proved that a low-rank matrix can be completed exactly from a random sample of its entries (2009), the mathematics behind the Netflix Prize. Recht, Fazel, and Parrilo proved rank recovery from generic measurements (2010), and Cai, Candès, and Shen gave singular-value thresholding (2010), the matrix soft-threshold. The capstone was robust PCA: Candès, Li, Ma, and Wright proved that a convex program separates a low-rank matrix from sparse gross corruption exactly (2011), a result whose paper title ends in a question mark because the authors found it hard to believe. By 2011 the entire dictionary, sparse vectors to low-rank matrices, ℓ_1 to nuclear norm, was complete.

F.7 The legacy

The ideas did not stop. Sparse coding reappeared in deep learning as LISTA, an unrolled thresholding network, and as the sparse autoencoders now used to interpret large language models. The geometry of covering numbers connects to sphere packing, the subject for which Maryna Viazovska won the Fields Medal in 2022. The thread is unbroken: exploit structure, measure little, recover by a tractable program. Table F.1 is the timeline at a glance.

Year	Result
1928	Nyquist states the sampling rate needed to reconstruct a bandlimited signal.
1936	Eckart and Young prove the truncated SVD is the best low-rank approximation.
1948	Shannon founds information theory and the sampling theorem, setting the classical barrier.
1974	Welch proves the lower bound on the coherence of a redundant system.
1984	Johnson and Lindenstrauss prove that random projections preserve distances.
1993	Mallat and Zhang introduce matching pursuit; Pati, Rezaiifar, and Krishnaprasad introduce orthogonal matching pursuit.
1995	Natarajan proves sparse approximation is NP-hard.
1996	Tibshirani introduces the LASSO.
1998	Chen, Donoho, and Saunders introduce basis pursuit, the ℓ_1 program.
1999–2001	Lee and Seung introduce nonnegative matrix factorization and the multiplicative updates.
2003	Donoho and Elad introduce the spark and the uncertainty principle for two bases.
2004	Daubechies, Defrise, and De Mol give ISTA; Tropp analyzes OMP ("greed is good").
2005–2006	Candès, Romberg, and Tao, Candès and Tao, and Donoho found compressed sensing: exact recovery from $O(s \log(N/s))$ random measurements, the RIP, and ℓ_1 minimization.
2006	Aharon, Elad, and Bruckstein introduce K-SVD; Duarte et al. build the single-pixel camera.
2007	Lustig, Donoho, and Pauly bring compressed sensing to MRI.
2008	Baraniuk, Davenport, DeVore, and Wakin give the simple concentration proof of the RIP.
2009	Candès and Recht prove exact matrix completion; Needell and Tropp give CoSaMP; Beck and Teboulle give FISTA; Dai and Milenkovic give subspace pursuit.
2010	Recht, Fazel, and Parrilo prove rank recovery; Cai, Candès, and Shen give singular-value thresholding.
2011	Candès, Li, Ma, and Wright prove robust PCA separates low rank from sparse corruption.
2022	Viazovska wins the Fields Medal for sphere packing, the geometry behind covering numbers.

Table F.1. The milestones the course draws on, in the order they appeared.

APPENDIX G

Notation and Symbols

A complete glossary of the symbols used in the book, with a one-line meaning for each. The typographic conventions come first, then the symbols grouped by role.

G.1 Typographic conventions

The book follows a fixed set of conventions so that the type of an object is visible at a glance.

- **Scalars** are lowercase italic: x, λ, σ, ε.
- **Vectors** are lowercase bold: $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{a}_j$. A vector is a column by default; $\boldsymbol{x}^\top$ is a row.
- **Matrices** are uppercase bold: $\boldsymbol{A}$, $\boldsymbol{X}$, $\boldsymbol{\Psi}$. Entry (i, j) is A_{ij}, column j is $\boldsymbol{a}_j$, row i is $\boldsymbol{a}^i$ or $\boldsymbol{A}_{i,:}$.
- **Linear operators** on matrices are script: $\mathcal{A}$, $\mathcal{P}_\Omega$.
- **Index sets** are uppercase italic: S, T, Ω; the restriction of a vector to S is $\boldsymbol{x}_S$, of a matrix's columns to S is $\boldsymbol{A}_S$.
- **Estimates** carry a hat: $\hat{\boldsymbol{x}}$ is the recovered signal, $\hat{\imath}$ the predicted class.

G.2 Objects

$x, \lambda, \alpha, \sigma$	scalars (real or complex numbers)
$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v}, \boldsymbol{a}_j$	vectors (bold lowercase); $\boldsymbol{a}_j$ is column j of $\boldsymbol{A}$
$\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{F}, \boldsymbol{\Psi}$	matrices (bold uppercase); $\boldsymbol{F}$ DFT, $\boldsymbol{\Psi}$ sparsity basis
$\mathcal{A}, \mathcal{P}_\Omega$	linear operators on matrices; $\mathcal{P}_\Omega$ samples entries on Ω
$\boldsymbol{A}^\top, \boldsymbol{A}^*, \boldsymbol{A}^\dagger$	transpose, conjugate (Hermitian) transpose, Moore–Penrose pseudoinverse
$\boldsymbol{A}^{-1}, \det \boldsymbol{A}$	inverse, determinant (square $\boldsymbol{A}$)
$\boldsymbol{I}, \mathbf{1}, \mathbf{0}$	identity matrix, all-ones vector, zero vector or matrix
$\boldsymbol{e}_i$	i-th standard basis vector (a spike)
$\mathbb{R}, \mathbb{C}, \mathbb{Z}, \mathbb{N}, \mathbb{R}_{\geq 0}$	reals, complexes, integers, naturals, nonnegative reals
$[N]$	the index set $\{1, 2, \ldots, N\}$

G.3 Dimensions and structure

N	ambient signal dimension (length of $\boldsymbol{x}$)
m	number of measurements, $m \ll N$ in the interesting regime
s	sparsity, the number of nonzeros; r the rank for matrices
S, T, Ω	index sets and supports; S^c the complement of S
$\boldsymbol{x}_S, \boldsymbol{A}_S$	restriction of a vector to S, column submatrix on S
Σ_s	the set of s-sparse vectors $\{\boldsymbol{x} : \|\boldsymbol{x}\|_0 \le s\}$
$\operatorname{supp}(\boldsymbol{x})$	support, the index set $\{i : x_i \neq 0\}$
$\mathcal{N}(\boldsymbol{A}), \mathcal{R}(\boldsymbol{A})$	null space (kernel), range (column space)
$\dim, \operatorname{rank}(\boldsymbol{A})$	dimension of a subspace, rank of a matrix
$V_i,\ \operatorname{cone}(\cdot)$	a class subspace; the conic hull of a set

G.4 Norms and operators

$\|\boldsymbol{x}\|_p,\ \|\boldsymbol{x}\|_0$	ℓ_p norm ($p = 1, 2, \infty$); ℓ_0 nonzero count
$\|\boldsymbol{A}\|_F, \|\boldsymbol{A}\|_{2\to 2}, \|\boldsymbol{A}\|_*$	Frobenius, spectral, nuclear norm
$\|\boldsymbol{X}\|_{p,q}, \|\boldsymbol{X}\|_{\text{row},0}$	mixed $\ell_{p,q}$ norm; nonzero-row count (joint sparsity)
$\langle \boldsymbol{x},\, \boldsymbol{y} \rangle, \langle \boldsymbol{A},\, \boldsymbol{B} \rangle_F$	vector inner product; Frobenius (trace) inner product
$\operatorname{spark}(\boldsymbol{A}), \mu(\boldsymbol{A}), \delta_s$	spark, mutual coherence, restricted isometry constant
$\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi})$	mutual coherence between two bases
$\sigma_i(\boldsymbol{A}), \lambda_i, \operatorname{rank}(\boldsymbol{A}), \operatorname{tr}$	singular values, eigenvalues, rank, trace
$\kappa(\boldsymbol{A})$	condition number σ_1/σ_n
$\mathcal{S}_\tau, \mathcal{H}_s, \mathcal{D}_\tau$	soft threshold, hard threshold, singular-value threshold
$\operatorname{prox}_{\lambda f}, \operatorname{proj}_C$	proximal operator of λf, projection onto C
$\nabla f, \partial f$	gradient, subdifferential
$\mathbb{E}, \mathbb{P}, \operatorname{Var}$	expectation, probability, variance
$\preceq, \succeq$	positive-semidefinite ordering of Hermitian matrices

G.5 The canonical problems

(P_0)	$\min \|\boldsymbol{x}\|_0$ s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ (NP-hard)
(P_1)	$\min \|\boldsymbol{x}\|_1$ s.t. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ (basis pursuit, an LP)
$(\mathrm{P}_{1,\varepsilon})$	$\min \|\boldsymbol{x}\|_1$ s.t. $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2 \le \varepsilon$ (noisy)
$(\mathrm{PM}_0), (\mathrm{PM}_1)$	rank minimization and its nuclear-norm relaxation (matrices)
(PCP)	$\min \|\boldsymbol{X}\|_* + \lambda \|\boldsymbol{E}\|_1$ s.t. $\boldsymbol{X} + \boldsymbol{E} = \boldsymbol{Y}$ (robust PCA)
$(\mathrm{P}_{R0}), (\mathrm{P}_{1,2})$	row-sparse recovery and its group-lasso relaxation (joint sparsity)

G.6 Greek letters and recurring constants

σ_i	singular value; also σ a noise level
λ	eigenvalue; also a regularization weight or Lagrange multiplier
μ	mutual coherence; also a penalty parameter in augmented Lagrangians
δ_s	restricted isometry constant of order s
ε	error tolerance, or a small positive number
τ	threshold level (soft/hard/singular-value thresholding)
ρ, γ, ν	a net radius / fraction; a separation margin; a noise bound
Ω	the set of observed entries in matrix completion
$\mathbf{\Phi}, \mathbf{\Psi}$	sensing basis and sparsity basis
$\mathbf{\Sigma}, \mathbf{\Lambda}$	diagonal singular-value / eigenvalue matrix; also a dual variable

APPENDIX H

Annotated Further Reading

An opinionated guide to where to go next, organized so that each entry says what the source is good for and which part of this book it extends. Full citations are in the bibliography.

H.1 Textbooks

Foucart and Rauhut, *A Mathematical Introduction to Compressive Sensing* (2013).
The definitive reference and the book this course follows most closely. Complete and rigorous: the null space property, the restricted isometry property, coherence, random and structured random matrices, and the recovery guarantees, all proved in full. Start here to go deeper on any theorem in Parts II–IV. Its Chapter 9 (NSP and coherence), Chapter 6 (RIP), and Chapter 12 (structured random matrices) map directly onto Chapters 4, 8 and 10.

Elad, *Sparse and Redundant Representations* (2010).
The applied counterpart. Strong on dictionaries, K-SVD, and image processing, with the analysis-versus-synthesis viewpoint and the algorithmic side developed at length. The reference for Chapters 6 and 13, and the source for the patch-based denoising of Problem Set 8.

Wright and Ma, *High-Dimensional Data Analysis with Low-Dimensional Models* (2022).
The modern, unified treatment of the matrix half of the book: completion, robust PCA, and the low-dimensional-models philosophy, with deep-learning connections. The reference for Part V (Chapters 11 to 14) and for the union-of-subspaces view behind Chapter 16.

Vershynin, *High-Dimensional Probability* (2018).
The probabilistic toolkit. Sub-Gaussian and sub-exponential variables, concentration, covering numbers, and random matrices, exactly the machinery of Chapter 9 and Appendix C. The clearest modern source for the concentration inequalities.

Boyd and Vandenberghe, *Convex Optimization* (2004).
The optimization background: convexity, duality, the simplex and interior-point methods, and the proximal viewpoint behind Chapters 5 and 7. Freely available, and the standard reference for why ℓ_1 minimization is a tractable convex program.

Golub and Van Loan, *Matrix Computations* (2013).
The reference for the linear algebra of Appendices B and D: how the SVD, QR, and least squares are actually computed, and what conditioning means numerically.

H.2 Founding papers

The papers that started the field are short and readable. For the vector theory, Candès, Romberg, and Tao's "Robust uncertainty principles" and Candès and Tao's "Near-optimal signal recovery" established exact recovery and the restricted isometry property, while Donoho's "Compressed Sensing" gave the geometric, independent account. Candès and Wakin's "An Introduction to Compressive Sampling" is the gentlest overview and the best first paper to read. Tropp and Gilbert's analysis of orthogonal matching pursuit is the readable source for the greedy guarantees of Chapter 6.

For the matrix story, Candès and Recht's "Exact matrix completion via convex optimization" is the foundational completion paper, and Candès, Li, Ma, and Wright's "Robust principal component analysis?" is the robust-PCA paper whose title's question mark captures how surprising the exact-separation result was. Recht, Fazel, and Parrilo connect rank minimization to the RIP for matrices.

H.3 By topic, where to go deeper

Greedy algorithms.
Beyond Chapter 6: Needell and Tropp on CoSaMP, Dai and Milenkovic on subspace pursuit, and Blumensath and Davies on iterative hard thresholding give the provable greedy methods with RIP guarantees.

Proximal and first-order methods.
Beyond Chapter 7: Beck and Teboulle's FISTA paper for the accelerated rate, and Parikh and Boyd's monograph *Proximal Algorithms* for the operator-splitting view (ISTA, ADMM, Douglas–Rachford).

Structured random matrices.
Beyond Chapter 10: Rudelson and Vershynin on the RIP for partial Fourier matrices, and Ailon and Chazelle on the fast Johnson–Lindenstrauss transform.

Matrix completion and robust PCA.
Beyond Chapters 11 and 12: Cai, Candès, and Shen on singular-value thresholding, and Koren, Bell, and Volinsky's "Matrix factorization techniques" for the methods that won the Netflix Prize.

Dictionaries and classification.
Beyond Chapters 13 and 16: Aharon, Elad, and Bruckstein's K-SVD paper, and Wright et al. on sparse representation-based classification.

H.4 Where the field went

Beyond this book, the same ideas drive structured and group sparsity, phase retrieval (recovery from magnitude-only measurements, as in X-ray crystallography), tensor recovery (the higher-order analogue of low-rank matrices), and the low-dimensional-models view of deep learning, where sparse coding reappears as LISTA, an unrolled thresholding network, and as the sparse autoencoders now used to interpret large language models. The connection to sphere packing, the geometry behind covering numbers, links the subject to one of the deepest results in recent mathematics. The thread is unbroken across all of it: exploit structure, measure little, recover by a tractable program.

Bibliography

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[2] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 557–563, 2006.

[3] Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[5] Rajendra Bhatia. *Matrix Analysis*, volume 169 of *Graduate Texts in Mathematics*. Springer, New York, 1997.

[6] Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[8] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[9] Emmanuel J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, 2008.

[10] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[11] Emmanuel J. Candès and Terence Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[12] Emmanuel J. Candès and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[13] Emmanuel J. Candès and Michael B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[14] Emmanuel J. Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[15] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):1–37, 2011.

[16] Scott S. Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[17] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009.

[18] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[19] Chris Ding, Xiaofeng He, and Horst D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 606–610, 2005.

[20] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[21] David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[22] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean-Luc Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 58(2): 1094–1121, 2012.

[23] Marco F. Duarte, Mark A. Davenport, Dharmpal Takhar, Jason N. Laska, Ting Sun, Kevin F. Kelly, and Richard G. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2): 83–91, 2008.

[24] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3): 211–218, 1936.

[25] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[26] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

[27] Kjersti Engan, Sven Ole Aase, and John Håkon Husøy. Method of optimal directions for frame design. In *Proceedings of IEEE ICASSP*, volume 5, pages 2443–2446, 1999.

[28] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser, 2013.

[29] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[30] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 399–406, 2010.

[31] Rémi Gribonval and Morten Nielsen. Sparse representations in unions of bases. *IEEE Transactions on Information Theory*, 49(12):3320–3325, 2003.

[32] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[33] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[34] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[35] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems (NIPS) 13*, pages 556–562, 2001.

[36] Michael Lustig, David Donoho, and John M. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[37] Stéphane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[38] Leon Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960.

[39] Balas K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[40] Deanna Needell and Joel A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[41] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the AIEE*, 47(2):617–644, 1928.

[42] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[43] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.

[44] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[45] Mark Rudelson and Roman Vershynin. On sparse reconstruction from Fourier and Gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008.

[46] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[47] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

[48] Joel A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[49] Joel A. Tropp and Anna C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[50] Joel A. Tropp, Anna C. Gilbert, and Martin J. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.

[51] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

[52] Lloyd R. Welch. Lower bounds on the maximum cross correlation of signals. *IEEE Transactions on Information Theory*, 20(3):397–399, 1974.

[53] John Wright and Yi Ma. *High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications*. Cambridge University Press, 2022.

[54] John Wright, Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.