

Linear Algebra for Data Science, Machine Learning, and Signal Processing

JEFFREY A. FESSLER
RAJ RAO NADAKUDITI

"The authors provide a comprehensive contemporary presentation of linear algebra, demonstrating its foundational and intrinsic value to modern subjects, such as machine/deep learning, data science, and signal processing. The presentation is fun, exciting, topic-diverse, classroom tested, and addresses practical implementation in ways that jump start students' use."

Christ D. Richmond, Duke University

"This is an excellent and timely text that addresses the specific needs of data science (DS), machine learning (ML), and signal processing (SP). Its nicely crafted coverage is designed to prepare students in the areas of DS/ML/SP, in particular, by drawing thoughtful examples from these fields. With increasing demands from data-based sciences, there is a pressing need for a book in 'the new linear algebra,' and this text fills this gap."

Yousef Saad, University of Minnesota

"With the emergence of Graphics Processing Units (GPUs), the importance of linear algebra for machine learning cannot be overstated. This is a thoughtful and timely work on the topic of linear algebra for machine learning, which I anticipate will be one of the definitive textbooks in this field."

Vahid Tarokh, Duke University

"To see the spirit of this book, just look at pages 1 and 2. A painting is deblurred by linear algebra. Great ideas and how to use them in real time – all on display!"

Gilbert Strang, Massachusetts Institute of Technology

Linear Algebra for Data Science, Machine Learning, and Signal Processing

JEFFREY A. FESSLER

University of Michigan, Ann Arbor

RAJ RAO NADAKUDITI

University of Michigan, Ann Arbor



CAMBRIDGE
UNIVERSITY PRESS



Shaftesbury Road, Cambridge CB2 8EA, United Kingdom
One Liberty Plaza, 20th Floor, New York, NY 10006, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India
103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment,
a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of
education, learning and research at the highest international levels of excellence.

www.cambridge.org

Information on this title: www.cambridge.org/highereducation/isbn/9781009418140

DOI: 10.1017/9781009418164

© Jeff Fessler and Raj Rao Nadakuditi 2024

This publication is in copyright. Subject to statutory exception and to the provisions
of relevant collective licensing agreements, no reproduction of any part may take
place without the written permission of Cambridge University Press & Assessment.

First published 2024

Printed in Great Britain by CPI Group (UK) Ltd, Croydon CR0 4YY

A catalogue record for this publication is available from the British Library

Library of Congress Cataloging-in-Publication data

Names: Fessler, Jeffrey A., 1964- author. | Nadakuditi, Raj Rao, 1977- author.

Title: Linear algebra for data science, machine learning, and signal

processing / Jeffrey A. Fessler, Raj Rao Nadakuditi.

Description: New York, NY : Cambridge University Press, 2024. | Includes
bibliographical references and index.

Identifiers: LCCN 2023050878 (print) | LCCN 2023050879 (ebook) |

ISBN 9781009418140 (hardback) | ISBN 9781009418164 (epub)

Subjects: LCSH: Algebras, Linear—Textbooks. | Matrices—Textbooks. |

Machine learning—Mathematics—Textbooks. | Signal

processing—Mathematics—Textbooks. | Julia (Computer program
language)—Textbooks.

Classification: LCC QA184.2 .F47 2024 (print) | LCC QA184.2 (ebook) |

DDC 512/.5—dc23/eng/20231117

LC record available at <https://lccn.loc.gov/2023050878>

LC ebook record available at <https://lccn.loc.gov/2023050879>

ISBN 978-1-009-41814-0 Hardback

Additional resources for this publication at www.cambridge.org/FesslerNadakuditi

Cambridge University Press & Assessment has no responsibility for the persistence
or accuracy of URLs for external or third-party internet websites referred to in this
publication and does not guarantee that any content on such websites is, or will
remain, accurate or appropriate.

To Sue

To Tata, Bama, Dad, Reizo and Ms. Gracie

Contents

| | |
|--|----------------|
| <i>Preface</i> | <i>page</i> xv |
| <i>Acknowledgments</i> | xix |
| 1 Getting Started | 1 |
| 1.1 Introduction | 1 |
| 1.2 Example Applications | 1 |
| 1.2.1 Signal Processing Example: Image Deblurring | 1 |
| 1.2.2 Computer Vision Applications | 2 |
| 1.2.3 Machine Learning Example: Handwritten Digit Recognition | 3 |
| 1.3 Formatting | 3 |
| 1.4 Notation Preview | 4 |
| 1.4.1 What the \mathbb{F} Means | 6 |
| 1.5 Julia | 7 |
| 1.6 Fields, Vector Spaces, Linear Maps | 7 |
| 1.6.1 Field of Scalars | 8 |
| 1.6.2 Vector Spaces | 9 |
| 1.6.3 Linear Maps and Linear Operators | 11 |
| 2 Introduction to Matrices | 12 |
| 2.1 Introduction | 12 |
| 2.2 Basics of Vectors and Matrices | 12 |
| 2.3 Matrix Structures | 17 |
| 2.3.1 Common Matrix Shapes and Types | 17 |
| 2.3.2 Matrix Transpose and Symmetry | 20 |
| 2.4 Multiplication | 23 |
| 2.4.1 Vector–Vector Multiplication | 23 |
| 2.4.2 Matrix–Vector Multiplication | 25 |
| 2.4.3 Matrix–Matrix Multiplication | 29 |
| 2.4.4 Matrix Multiplication Properties | 30 |
| 2.4.5 Kronecker and Hadamard Products, and the vec Operator | 34 |
| 2.4.6 Using Matrix–Vector Operations | 35 |
| 2.4.7 Invertibility | 40 |
| 2.5 Orthogonality and the Euclidean Norm | 42 |
| 2.5.1 Orthogonal Vectors | 42 |
| 2.5.2 Euclidean Norm | 42 |

| | | |
|----------|---|----|
| 2.5.3 | Cauchy–Schwarz Inequality | 43 |
| 2.5.4 | Orthogonal Matrices | 43 |
| 2.6 | Determinant of a Matrix | 45 |
| 2.6.1 | Determinant Properties | 46 |
| 2.6.2 | Matrices with Units | 47 |
| 2.6.3 | Laplace’s Determinant Formula | 48 |
| 2.6.4 | Avoiding Computation | 48 |
| 2.6.5 | Small Matrices | 49 |
| 2.7 | Eigenvalues | 50 |
| 2.7.1 | Eigenvectors | 51 |
| 2.7.2 | Practical Implementation | 51 |
| 2.7.3 | Properties of Eigenvalues | 52 |
| 2.8 | Trace | 54 |
| 2.9 | Summary | 55 |
| 3 | Matrix Factorization: Eigendecomposition and SVD | 63 |
| 3.1 | Introduction | 63 |
| 3.1.1 | Matrix Factorizations | 63 |
| 3.1.2 | Square Matrices | 64 |
| 3.2 | Spectral Theorem for Symmetric Matrices | 65 |
| 3.2.1 | Normal Matrices | 66 |
| 3.2.2 | Square Asymmetric and Nonnormal Matrices | 68 |
| 3.2.3 | Geometry of Matrix Diagonalization | 70 |
| 3.2.4 | Matrix Powers and Matrix Exponential | 73 |
| 3.3 | Singular Value Decomposition | 74 |
| 3.3.1 | Singular Values and Singular Vectors | 74 |
| 3.3.2 | Existence of SVD | 74 |
| 3.3.3 | Geometry | 75 |
| 3.3.4 | Practical Implementation in JULIA | 76 |
| 3.3.5 | SVD Basic Properties | 77 |
| 3.4 | The Matrix 2-Norm or Spectral Norm | 78 |
| 3.4.1 | Optimization: \min versus $\arg \min$ | 79 |
| 3.4.2 | Eigenvalues as Optimization Problems | 80 |
| 3.4.3 | Smallest Singular Value | 81 |
| 3.5 | Relating SVDs and Eigendecompositions | 82 |
| 3.5.1 | When Does $U = V$? | 84 |
| 3.5.2 | SVD Computation Using Eigendecomposition | 85 |
| 3.5.3 | SVD Nonuniqueness Revisited | 87 |
| 3.6 | Positive Semidefinite Matrices | 88 |
| 3.6.1 | Relating Positive (Semi)Definiteness to Eigenvalues | 89 |
| 3.7 | Summary | 89 |
| 4 | Subspaces, Rank, and Nearest-Subspace Classification | 96 |
| 4.1 | Introduction | 96 |
| 4.2 | Subspaces | 96 |
| 4.2.1 | Span | 98 |

| | | |
|----------|--|-----|
| 4.2.2 | Linear Independence | 99 |
| 4.2.3 | Basis | 101 |
| 4.2.4 | Dimension | 103 |
| 4.2.5 | Sums and Intersections of Subspaces | 105 |
| 4.2.6 | Direct Sum of Subspaces | 106 |
| 4.2.7 | Dimensions of Sums of Subspaces | 106 |
| 4.2.8 | Orthogonal Complement of a Subspace | 107 |
| 4.2.9 | Linear Transforms | 108 |
| 4.2.10 | Range of a Matrix | 109 |
| 4.3 | Rank of a Matrix | 111 |
| 4.3.1 | Practical Use in JULIA | 112 |
| 4.3.2 | Rank of a Matrix Product | 112 |
| 4.3.3 | Other Rank Properties | 113 |
| 4.3.4 | Spark | 114 |
| 4.3.5 | Unitary Invariance of Rank | 114 |
| 4.4 | Nullspace of a Matrix | 115 |
| 4.4.1 | Nullspace or Kernel | 116 |
| 4.4.2 | Properties of Null Space | 116 |
| 4.4.3 | Columns of Unitary Matrices | 118 |
| 4.5 | The Four Fundamental Spaces | 119 |
| 4.5.1 | Anatomy of the SVD | 121 |
| 4.5.2 | SVD of Finite Differences | 123 |
| 4.5.3 | Synthesis View of Matrix Decomposition | 125 |
| 4.6 | Orthogonal Bases | 125 |
| 4.6.1 | Finding Coordinates in an Orthogonal Basis | 126 |
| 4.6.2 | Stiefel Manifold of Orthogonal Bases | 127 |
| 4.7 | Spotting Decompositions | 128 |
| 4.7.1 | Matrix–Vector Products and the SVD | 130 |
| 4.8 | Application: Signal Classification | 130 |
| 4.8.1 | Projection onto a Set | 130 |
| 4.8.2 | Nearest Point in a Subspace | 131 |
| 4.8.3 | Signal Classification by Nearest Subspace | 133 |
| 4.9 | Optimization Preview | 134 |
| 4.9.1 | Convex Sets | 135 |
| 4.9.2 | Convex Functions | 136 |
| 4.10 | Summary | 137 |
| 5 | Linear Least-Squares Regression and Binary Classification | 143 |
| 5.1 | Introduction | 143 |
| 5.2 | Introduction to Linear Equations | 143 |
| 5.2.1 | Solving $\mathbf{A}\mathbf{x} = \mathbf{y}$ | 144 |
| 5.2.2 | Linear Regression and Machine Learning | 144 |
| 5.2.3 | Lifting for Nonlinear Regression | 145 |
| 5.3 | Linear Least-Squares Estimation | 145 |
| 5.3.1 | Minimization and Gradients | 148 |
| 5.3.2 | Solving LLS Using the Normal Equations | 150 |

| | | |
|----------|--|-----|
| 5.3.3 | Solving LLS Problems Using the Compact SVD | 151 |
| 5.3.4 | Uniqueness of LLS Solution | 154 |
| 5.4 | Moore–Penrose Pseudoinverse | 155 |
| 5.4.1 | Pseudoinverse and Matrix Products | 155 |
| 5.4.2 | Pseudoinverse and SVD | 156 |
| 5.5 | LLS: Under-Determined Case | 158 |
| 5.5.1 | Orthogonality Principle | 160 |
| 5.5.2 | Minimum-Norm LS Solution via Pseudoinverse | 162 |
| 5.6 | Truncated SVD Solution | 164 |
| 5.6.1 | Condition Number | 164 |
| 5.6.2 | Practical Implementation of Truncated SVD Solution | 165 |
| 5.6.3 | Low-Rank Approximation Interpretation of Truncated SVD | 165 |
| 5.6.4 | Noise Effects and Perturbations | 166 |
| 5.6.5 | Tikhonov Regularization, or Ridge Regression | 167 |
| 5.7 | Summary of LLS Solution Methods in Terms of SVD | 168 |
| 5.8 | Frames and Tight Frames | 168 |
| 5.8.1 | Properties of a Frame | 170 |
| 5.8.2 | Tight Frame | 170 |
| 5.8.3 | Parseval Tight Frame | 171 |
| 5.8.4 | Properties of Parseval Tight Frames | 171 |
| 5.8.5 | Frame Summary | 173 |
| 5.9 | Projection and Orthogonal Projection | 174 |
| 5.9.1 | Idempotent Matrix | 174 |
| 5.9.2 | Orthogonal Projection Matrix | 176 |
| 5.9.3 | Projection onto a Subspace | 177 |
| 5.9.4 | Binary Classifier Design Using Least Squares | 182 |
| 5.9.5 | Empirical Risk Minimization | 183 |
| 5.10 | Recursive Least Squares | 184 |
| 5.10.1 | RLS with a Forgetting Factor | 185 |
| 5.11 | Summary | 186 |
| 6 | Norms and Procrustes Problems | 197 |
| 6.1 | Introduction | 197 |
| 6.2 | Vector Norms | 197 |
| 6.2.1 | Examples of Vector Norms | 198 |
| 6.2.2 | Practical Implementation | 199 |
| 6.2.3 | Properties of Vector Norms | 200 |
| 6.2.4 | Norm Notation | 201 |
| 6.2.5 | Robust Regression Application | 201 |
| 6.2.6 | Unitarily Invariant Vector Norms | 202 |
| 6.3 | Inner Products | 203 |
| 6.3.1 | Examples of Inner Products | 203 |
| 6.3.2 | Properties of Inner Products | 204 |
| 6.3.3 | More Inner Product Inequalities | 205 |
| 6.3.4 | Angle Between Vectors | 205 |
| 6.3.5 | Angle Between Subspaces | 206 |

| | | |
|------------|--|-----|
| 6.4 | Matrix Norms and Operator Norms | 206 |
| 6.4.1 | Examples of Matrix Norms | 207 |
| 6.4.2 | Induced Matrix Norms | 208 |
| 6.4.3 | Norms Defined in Terms of Singular Values | 210 |
| 6.4.4 | Practical Implementation | 214 |
| 6.4.5 | Properties of Matrix Norms | 214 |
| 6.4.6 | Spectral Radius | 217 |
| 6.4.7 | Practical Step Size for Gradient Descent | 218 |
| 6.5 | Convergence of Sequences of Vectors and Matrices | 219 |
| 6.6 | Generalized Inverse of a Matrix | 221 |
| 6.6.1 | Minimum Frobenius Norm Generalized Inverse | 221 |
| 6.7 | Procrustes Analysis | 222 |
| 6.7.1 | Sanity Check and Scale Invariance | 224 |
| 6.7.2 | Procrustes Generalizations | 225 |
| 6.7.3 | Subspace/Span Comparisons | 228 |
| 6.7.4 | Weighted Procrustes Problems | 228 |
| 6.7.5 | Practical Implementation | 229 |
| 6.8 | Summary | 229 |
| 7 | Low-Rank Approximation and Multidimensional Scaling | 238 |
| 7.1 | Introduction | 238 |
| 7.2 | Low-Rank Approximation via Frobenius Norm | 238 |
| 7.2.1 | Eckart–Young–Mirsky Theorem | 239 |
| 7.2.2 | Subspace Approximation Perspective | 240 |
| 7.2.3 | Implementation | 240 |
| 7.2.4 | Choosing Rank via Permutation | 242 |
| 7.2.5 | Nonuniqueness of SVD and Low-Rank Approximation | 243 |
| 7.2.6 | One-Dimensional Example | 244 |
| 7.2.7 | Generalization to Other Norms | 245 |
| 7.2.8 | Bases for $\mathbb{F}^{M \times N}$ | 247 |
| 7.2.9 | Low-Rank Approximation Summary | 248 |
| 7.2.10 | Rank and Stability | 249 |
| 7.2.11 | Example: Photometric Stereo | 250 |
| 7.3 | Sensor Localization Application: Multidimensional Scaling | 250 |
| 7.3.1 | Derivation (Analysis) | 251 |
| 7.3.2 | MDS Method | 254 |
| 7.3.3 | Practical Implementation | 255 |
| 7.3.4 | Extensions | 256 |
| 7.4 | Proximal Operators | 257 |
| 7.4.1 | Soft Thresholding | 257 |
| 7.4.2 | Hard Thresholding | 259 |
| 7.5 | Alternative Low-Rank Approximation Formulations | 260 |
| 7.5.1 | Unconstrained/Regularized Formulation | 260 |
| 7.5.2 | General Unitarily Invariant Formulations | 260 |
| 7.5.3 | Singular Value Hard Thresholding | 261 |
| 7.5.4 | Singular Value Soft Thresholding | 262 |

| | | |
|----------|---|-----|
| 7.5.5 | Other Extensions of Low-Rank Approximation | 263 |
| 7.6 | Choosing the Rank or Regularization Parameter | 263 |
| 7.6.1 | Stein's Unbiased Risk Estimate | 264 |
| 7.6.2 | OptShrink | 265 |
| 7.7 | Related Methods: Autoencoders and PCA | 269 |
| 7.7.1 | Relation to Autoencoder with Linear Layers | 269 |
| 7.7.2 | Relation to Principal Component Analysis | 270 |
| 7.8 | Subspace Learning for Classification | 275 |
| 7.8.1 | Subspace Clustering | 277 |
| 7.9 | Subspace Tracking and Streaming PCA | 277 |
| 7.9.1 | Incremental SVD | 278 |
| 7.9.2 | Streaming PCA | 278 |
| 7.10 | Summary | 279 |
| 8 | Special Matrices, Markov Chains, and PageRank | 283 |
| 8.1 | Introduction | 283 |
| 8.2 | Companion Matrices | 283 |
| 8.2.1 | Practical Implementation | 285 |
| 8.2.2 | Polynomial Matrix Functions | 286 |
| 8.2.3 | Eigenvectors of Companion Matrices | 287 |
| 8.2.4 | Vandermonde Matrices | 288 |
| 8.2.5 | Kronecker Sum and Polynomial Roots | 289 |
| 8.3 | Circulant Matrices | 290 |
| 8.3.1 | Relationship to DFT Properties from DSP | 293 |
| 8.3.2 | Practical Implementation | 293 |
| 8.3.3 | Spectral Properties of Circulant Matrices | 294 |
| 8.3.4 | Inverting a Circulant Matrix | 294 |
| 8.4 | Toeplitz Matrices | 294 |
| 8.4.1 | Toeplitz Matrix Multiplication with a Vector | 295 |
| 8.4.2 | Inverting a Toeplitz Matrix | 295 |
| 8.4.3 | Factoring a Toeplitz Matrix | 295 |
| 8.5 | Power Iteration | 296 |
| 8.5.1 | Convergence of the Power Iteration | 297 |
| 8.5.2 | Geršgorin Disk Theorem | 298 |
| 8.6 | Nonnegative Matrices and Graphs | 301 |
| 8.6.1 | Primitive Matrices | 301 |
| 8.6.2 | Weighted Directed Graphs | 303 |
| 8.6.3 | Strongly Connected Graphs | 305 |
| 8.6.4 | Irreducible Matrix | 306 |
| 8.6.5 | Matrix Period | 307 |
| 8.7 | Nonnegative Matrices and Perron–Frobenius Theorems | 309 |
| 8.7.1 | Perron–Frobenius for Square Nonnegative Matrices | 309 |
| 8.7.2 | Perron–Frobenius for Nonnegative Irreducible Matrices | 311 |
| 8.7.3 | Perron–Frobenius for Primitive Matrices | 312 |
| 8.7.4 | Perron–Frobenius for Stochastic Matrices | 312 |
| 8.8 | Markov Chains | 313 |

| | | |
|-----------|--|-----|
| 8.8.1 | Equilibrium Distribution(s) of a Markov Chain | 315 |
| 8.8.2 | Limiting Distribution(s) of a Markov Chain | 316 |
| 8.8.3 | Markov Chains with Strongly Connected Graphs | 318 |
| 8.8.4 | Google's PageRank Method | 319 |
| 8.9 | Graph Laplacian and Spectral Clustering | 322 |
| 8.9.1 | Clustering | 322 |
| 8.9.2 | Weighted Graph Based on Similarity | 323 |
| 8.9.3 | Connected Components | 324 |
| 8.9.4 | Graph Laplacian | 324 |
| 8.9.5 | Spectral Clustering Algorithm | 325 |
| 8.9.6 | Laplacian Eigenmaps | 326 |
| 8.10 | Summary | 327 |
| 9 | Optimization Basics and Logistic Regression | 335 |
| 9.1 | Introduction | 335 |
| 9.2 | Preconditioned Gradient Descent for LS | 335 |
| 9.2.1 | Tool: Matrix Square Root | 336 |
| 9.2.2 | Convergence Rate Analysis of PGD: First Steps | 338 |
| 9.2.3 | Classical GD: Step Size Bounds | 339 |
| 9.2.4 | Optimal Step Size for GD | 340 |
| 9.2.5 | Ideal Preconditioner for PGD | 340 |
| 9.2.6 | Tool: Positive (Semi)Definiteness Properties | 341 |
| 9.2.7 | General Preconditioners for PGD | 342 |
| 9.2.8 | Diagonal Majorizer | 342 |
| 9.2.9 | Preconditioning Illustration/Demo | 344 |
| 9.2.10 | Convergence Rates | 345 |
| 9.2.11 | Tool: Commuting (Square) Matrices | 346 |
| 9.2.12 | Monotonicity | 347 |
| 9.3 | Preconditioned Steepest Descent | 349 |
| 9.4 | Gradient Descent for Smooth Convex Functions | 349 |
| 9.4.1 | Lipschitz Continuity | 350 |
| 9.4.2 | Convexity and Hessian | 351 |
| 9.4.3 | GD Convergence Theorem | 352 |
| 9.4.4 | Nesterov's Fast Gradient Method | 353 |
| 9.4.5 | Optimized Gradient Method | 354 |
| 9.4.6 | Gradient Projection Method | 355 |
| 9.5 | Machine Learning via Logistic Regression for Binary Classification | 356 |
| 9.5.1 | Practical Implementation of Logistic Regression | 360 |
| 9.5.2 | Numerical Results: Logistic Regression | 360 |
| 9.6 | Stochastic Gradient Descent | 360 |
| 9.7 | Summary | 361 |
| 10 | Matrix Completion and Recommender Systems | 365 |
| 10.1 | Introduction | 365 |
| 10.2 | Measurement Model | 366 |
| 10.2.1 | Practical Implementation | 366 |
| 10.2.2 | Sampling Conditions for LRMC | 366 |

| | |
|--|-----|
| 10.2.3 Sampling Mask | 367 |
| 10.3 LRMC: Noiseless Case | 368 |
| 10.3.1 Noiseless Problem Statement | 368 |
| 10.3.2 Alternating Projection Approach to LRMC | 368 |
| 10.4 LRMC: Noisy Case | 371 |
| 10.4.1 Noisy Problem Statement | 371 |
| 10.4.2 Majorize–Minimize (MM) Iterations | 372 |
| 10.4.3 MM Methods for LRMC | 373 |
| 10.4.4 LRMC by Iterative Low-Rank Approximation | 373 |
| 10.4.5 LRMC by Iterative Singular Value Hard Thresholding | 374 |
| 10.4.6 LRMC by Iterative Singular Value Soft Thresholding | 374 |
| 10.4.7 Iterative Soft-Thresholding Algorithm | 375 |
| 10.4.8 Debiasing the Nuclear Norm Effects | 376 |
| 10.4.9 Factorization Approaches | 377 |
| 10.4.10 Demo | 378 |
| 10.5 Robust PCA and Video Foreground/Background Separation | 378 |
| 10.5.1 Robust PCA | 378 |
| 10.5.2 Video Foreground/Background Separation | 378 |
| 10.6 Nonnegative Matrix Factorization | 379 |
| 10.7 Summary | 380 |
| | |
| 11 Neural Network Models | 381 |
| 11.1 Introduction | 381 |
| 11.2 The Importance of Nonlinearity | 381 |
| 11.3 Fully Connected NN Models | 383 |
| 11.3.1 Perceptron Model | 383 |
| 11.3.2 Multilayer Perceptron NN Models | 384 |
| 11.3.3 Model Expressiveness | 385 |
| 11.4 Training NN Models | 385 |
| 11.4.1 Weight Regularization | 386 |
| 11.5 CNN Models | 387 |
| 11.5.1 Matrix Representations | 388 |
| 11.5.2 CNN Architectures | 388 |
| 11.6 Summary | 389 |
| | |
| 12 Random Matrix Theory, Signal + Noise Matrices, and Phase Transitions | 390 |
| 12.1 Introduction | 390 |
| 12.1.1 Perturbation Bounds | 390 |
| 12.2 Roundoff Error | 391 |
| 12.2.1 RMT for Roundoff Analysis | 393 |
| 12.3 Additive Noise | 396 |
| 12.4 Outliers | 400 |
| 12.5 Matrix Completion | 401 |
| 12.6 Summary | 403 |
| | |
| <i>References</i> | 405 |
| <i>Index</i> | 423 |

Overview

Modern methods in data science, machine learning, and signal processing (DS–ML–SP) all build extensively on matrix methods and linear algebra. Often students who are interested in DS–ML–SP are advised to “go take a linear algebra course” with the promise that the material learned there will be useful later in more advanced courses. The content in this book is designed to teach important linear algebra ideas in an integrated way with computational methods in the context of DS–ML–SP applications. The focus here is on using matrix methods to pose and solve DS–ML–SP problems, rather than to provide rigorous proofs of linear algebra theorems. Traditional linear algebra and numerical linear algebra courses spend considerable time focusing on solving $Ax = b$. Solving systems of equations is essential for physics-based applications described by partial differential equations, whereas modern DS–ML–SP applications are data-driven and rarely reduce to solving $Ax = b$. Thus, this book treats the topic of solving linear systems only very briefly so that we can get to “the good stuff” like regularized least squares regression (Chapter 5), multidimensional scaling (Chapter 6), low-rank matrix approximation and Procrustes analysis (Chapter 7), Markov chains and the PageRank method (Chapter 8), logistic regression for binary classification, (Chapter 9), and matrix completion (Chapter 10). Put another way, after establishing a foundation in Chapters 1–3, every chapter that follows has further mathematical methods and models, along with one or more compelling applications to motivate and illustrate them. The goal of this book is to provide mathematical foundations for subsequent DS–ML–SP courses while also introducing matrix-based DS–ML–SP methods and applications that are useful in their own right.

Software

Nearly every mathematical concept in this book has corresponding operations in software, and this book describes those operations using the JULIA language [1]. This relatively new language has many benefits. JULIA is designed in a way that allows the code to look very similar to the math, facilitating the translation of algorithm ideas

to working software. JULIA uses dynamic typing so it is suitable for interactive and educational use, yet it is very fast because it is compiled. JULIA is open source and its git-based package manager greatly facilitates reproducible research. Readers do need to know JULIA to begin using this book; the language borrows a lot of ideas from MATLAB and Python (among others), so readers familiar with those tools will be able to follow the examples easily. Readers can view the JULIA code examples as pseudocode even if they prefer to use other languages, learning some JULIA along the way. There are many tutorials online as well as other books based on JULIA that provide useful references, for example, [2]. Every figure in this book was generated using JULIA.

Textbook Use

The content in this book has been used in two first-year graduate courses (EECS 505 and EECS 551) at the University of Michigan since at least 2016, taken by several thousand students over that time. Since 2017, those courses have used JULIA as the primary (505) or only (551) language for illustrating and implementing the ideas. Senior-level undergraduates with mathematical maturity have also taken these courses. The courses include weekly discussion sections where students apply the techniques to real data (like handwritten digit classification) using Jupyter notebooks. (The Ju in Jupyter is for JULIA.)

Several methods in the book are illustrated in JULIA demos at the website <https://github.com/JeffFessler/book-la-demo>. These demos were created using the convenient `Literate.jl` and `Documenter.jl` tools in JULIA. Those tools generate HTML output that is easily viewed in a browser, as well as Jupyter notebooks that students can use and modify.

A prior undergraduate-level course in linear algebra is likely to be helpful as background for getting the most out of this book. A prior undergraduate-level course in digital signal processing is helpful for understanding a few of the examples, but is not essential for most of the book.

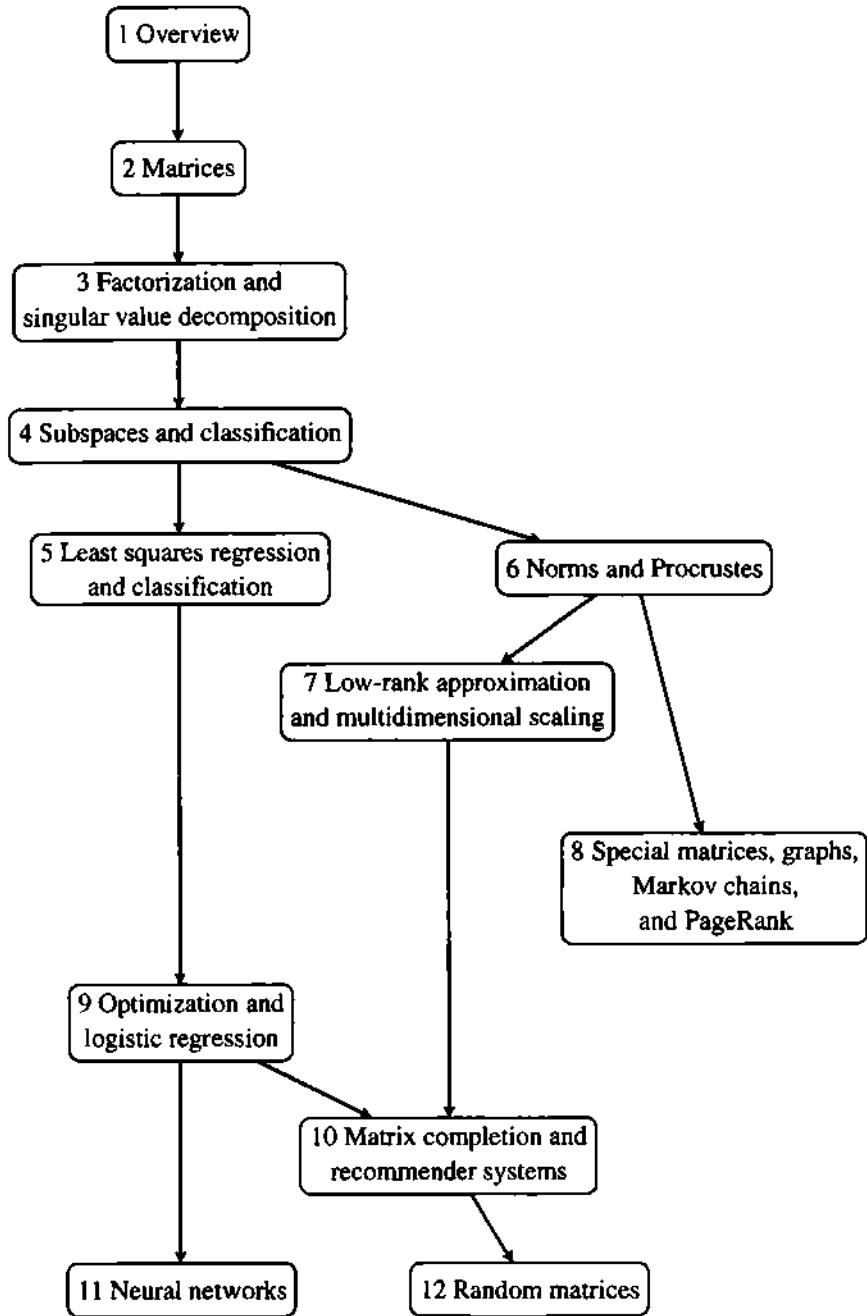
Instructor Resources

Embedded in the chapters are over 200 multiple-choice questions that instructors can use for in-class active learning exercises, or for self study by readers.

There are over 150 exercises at the ends of the chapters. Typeset solutions to these problems are available for instructors on the book's web page <https://doi.org/10.1017/9781009418164>. Also available there are slides of the material for classroom use. One version of the slides is a skeleton format with key equations omitted that an instructor can complete interactively during a lecture. (This is how the first author teaches this material.)

Organization

The following diagram illustrates how the book chapters are related. The first few chapters provide a foundation that should be read in sequence. There is more flexibility in the ordering of the subsequent chapters.



Related Books

Books we used as references when preparing this material include [3], [4], [5], and [6]. None of those books use JULIA to illustrate the ideas.

Other books that provide useful linear algebra background, also using JULIA, are [2], [7], and [8]. Those books have less depth in DS–ML–SP applications. [9] describes JULIA and uses it for some ML applications with less matrix fundamentals. Other books using JULIA for related topics include [10], [11], and [12].

There are many graduate-level books on DS–ML–SP topics that give a brief review of linear algebra concepts before delving into more advanced material. The material in this book will provide the reader with a more thorough foundation in preparation for more advanced DS–ML–SP courses.

Acknowledgments

The authors donate a portion of their royalties to organizations that empower groups that have been historically disadvantaged in science, technology, engineering, and mathematics fields, including oserm.org.

From JF

Thanks to Prof. Zhongming Liu, Prof. Yong Long, graduate student instructors Caroline Crockett, David Hong, Steven Whitaker, Haowei Xiang, and postdocs Rodrigo Lobos, Greg Ongie, and Dan Weller, and numerous past students, including Yongli He, Winston Wang, Emma Shedd, Zicheng Jin, Ege Taga and Yixuan (Isaac) Jia, for many corrections and suggestions. Special thanks to Matt Raymond for particularly detailed and insightful suggestions that refined and clarified the content of each chapter. Thanks to Raj for starting me on this journey with his handwritten notes. The best way to learn is to teach from a colleague's insightful lecture notes!

From RN

Special thanks to Jonas Kersulis and Brian Moore for helping create, edit, and test the many computationally centered homework problems.

This book would not be possible without them and the various graduate student instructors (Arvind Prasad, David Hong, Hao Wu, Rishi Sonthalia, Dipak Narayan, Yash Sanjay Bhagat, and Raj Tejas Suryaprakash) who helped edit, test, and refine the many homework problems, often right before they were about to go live to hundreds of students. Thanks to Simon Danisch for helping start this journey in 2017 by porting my MATLAB demos to JULIA.

Thanks in particular to Gil Strang for his encouragement, feedback, and support, and for his inspiration during the very special semester of Spring 2017 when we launched and taught 18.065 at MIT using some of the material in this book.

Multiple thanks to Alan Edelman for years of encouragement and inspiration, and for teaching me so much (including JULIA). A reader might sometimes recognize Alan and Gil's style in the way the math and code are presented. That's no accident. This book is infused with their DNA and years of me soaking in their thoughts and ideas on so many matters, particularly on how elegant math produces elegant code and vice versa. All they taught me about how to see math and linear algebra makes me love it, and to want to share it with you in this book, even more.

1 Getting Started

1.1 Introduction

This chapter provides introductory material, including visual examples in Section 1.2, that help motivate the rest of the book. Section 1.3 explains the book formatting and Section 1.4 previews the notation. Section 1.5 provides pointers for getting started with JULIA. Section 1.6 briefly reviews fields and vector spaces.

1.2 Example Applications

This section gives a preview of the kinds of applications that are developed in this book; these are just a few of countless applications of the material.

1.2.1 Signal Processing Example: Image Deblurring

One visually compelling example application of the methods discussed in this book is image deblurring, where the goal is to recover a sharp image \hat{x} from a blurry image y . This problem has numerous applications, including in science, such as correcting for optical imperfections in the Hubble space telescope [13], and in forensics, as seen in television crime dramas using low-quality security cameras. Figure 1.1 shows an example of an out-of-focus image of a painting (a) and the recovered sharp image (b).

Each of the images in Fig. 1.1 is an array of 1271×948 (width \times height) pixels, where each pixel has a color represented by red–green–blue (RGB) values called color channels. Here, instead of thinking of each color channel as a 1271×948 matrix, we “vectorize” all of the red pixels into a vector of length $1271 \cdot 948$; see (2.18). Likewise for the green and blue channels. For the example shown here, we process each channel separately.

The first step in any such data-processing problem is to formulate a model that relates the known data to the unknown or latent parameters. The image y in Fig. 1.1 has 1271×948 pixels, so a typical model is

$$\underbrace{y}_{\mathbb{R}^{1271 \cdot 948}} = A \underbrace{x}_{\mathbb{R}^{1271 \cdot 948}} + \varepsilon, \quad (1.1)$$



www.nga.gov/collection/art-object-page.166491.html

Figure 1.1 *Glass and Checkerboard* by Juan Gris before (a) and after (b) deblurring.

where A is a matrix of size $1\,204\,908 \times 1\,204\,908$ describing the optical blur and ϵ represents sensor noise. Under some simplifications, a solution for recovering the image x is

$$\hat{x} = \underset{x \in \mathbb{R}^{1271 \times 948}}{\arg \min} \|Ax - y\|_2^2 = A^+y = V\Sigma^+U'y. \quad (1.2)$$

This formulation and solution involves the majority of the chapters in this book!

- The problem and solution is expressed using matrices and vectors (Chapter 2).
- The solution is expressed using a singular value decomposition (Chapter 3).
- The formulation is a least-squares problem and the solution involves a matrix pseudo inverse (Chapter 5).
- The formulation here uses the Euclidean norm for vectors (Chapter 6).
- The “ $\arg \min$ ” represents an optimization problem (Section 4.9, Chapter 9).
- One way to solve this problem efficiently is to use fast Fourier transform (FFT) operations because the matrix A here has a special structure called block circulant with circulant blocks (Section 8.3).

State-of-the-art methods for image restoration use even more advanced methods such as 1-norm regularizers (Chapter 6) and deep neural networks (Chapter 11). A foundation in the matrix methods of this book is helpful for reading the literature on those methods.

1.2.2 Computer Vision Applications

Another imaging example, shown in Section 7.2.11, is photometric stereo; see Fig. 7.6 and Demo 7.2. Section 10.5.1 illustrates foreground/background separation in video; see Fig. 10.4 and Demo 10.3.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |

Figure 1.2 Labeled training data for handwritten digit recognition from the MNIST dataset [14].

1.2.3 Machine Learning Example: Handwritten Digit Recognition

One machine learning application that is a recurring theme in this book is handwritten digit recognition. Given labeled training data, like the images of handwritten digits shown in Fig. 1.2, how does one design a classifier that can automatically predict the digit seen in a test image? See Sections 4.8.3, 5.9.4, and 9.5.

1.3 Formatting

Here is an overview of how material is formatted in this book.

- JULIA code snippets are shown like this.
- ❖ A “dangerous bend” symbol in the margin warns of tricky material.
- ❖ References to homework problems look like: Problem 2.1.
- ♦♦ A double diamond symbol in the margin is “experts only” material that is included for reference but is generally not needed subsequently in the book.

Definition Key definitions are shown like this.

Fact 1.1 Important facts are shown like this.

Particularly important topics are shown like this.

- ➊ Demo 1.1 provides a JULIA overview. Demos corresponding to online examples containing both code and results are shown like this and are available at the URL shown in the preface.

Example 1.1 Examples are shown like this.

```
# Julia code files are shown like this.
</> 1.1 @assert exp(im * π) ≈ -1
```

Such JULIA code files are also available on the code website, arranged by chapter with the numbers shown in the margin.

Explore 1.1 In-line exploration problems are shown like this, with answers at the end of the chapter. The number is a hyperlink in the e-version; a click on it should cause the viewer to jump to the solution.

Q1.1 Questions designed for interactive classroom use are formatted like this.

(Answers are available to instructors through the publisher.)

A: True B: False

1.4 Notation Preview

Here is a preview of notation that is used throughout. Most of these are standard in math and linear algebra texts. For the less familiar ones, the definitions are given later in the text.

| Symbol | What | Where |
|------------------|--|--------|
| Numbers | | |
| \mathbb{N} | natural numbers $1, 2, \dots$ | |
| \mathbb{Z} | integers | |
| \mathbb{Q} | rational numbers | p. 8 |
| \mathbb{R} | real numbers | |
| \mathbb{R}_+ | nonnegative real numbers | p. 135 |
| \mathbb{C} | complex numbers | |
| \mathbb{F} | field of scalars, typically \mathbb{R} or \mathbb{C} | p. 8 |
| i | $\sqrt{-1}$ imaginary unit | p. 21 |
| z^* | complex conjugate of $z \in \mathbb{C}$ | p. 21 |
| $ z $ | complex modulus or absolute value | p. 21 |
| $\angle z$ | angle of complex number | p. 21 |
| $\text{sign}(z)$ | sign of (possibly complex) number z : $\text{sign}(z e^{i\angle z}) = e^{i\angle z}$ | p. 82 |
| Logic | | |
| \implies | logical implication | |
| \iff | if and only if | |
| \forall | for all | |
| \exists | exists | |

| Symbol | What | Where |
|-------------------------------------|---|--------|
| Sets | | |
| $\{a, b, \dots\}$ | set (unordered collection) | |
| (a, b, \dots) | tuple (ordered collection) | |
| \emptyset | empty set | p. 99 |
| \in | is a member of (a set) | |
| \subset | subset of | |
| \cap | set intersection | |
| \cup | set union | |
| $\mathcal{A}, \mathcal{B}, \dots$ | upper case calligraphic denotes sets | |
| $\mathcal{A} \setminus \mathcal{B}$ | set difference (relative complement) | p. 53 |
| Calculus and functions | | |
| \mapsto | maps to | p. 11 |
| f' | derivative of function $f: \mathbb{R} \mapsto \mathbb{R}$ | p. 148 |
| ∇f | gradient of function $f: \mathbb{F}^N \mapsto \mathbb{R}$ | p. 148 |
| $g \circ f$ | function composition | p. 131 |
| $\mathbb{I}_{\{\cdot\}}$ | indicator function (1 or 0) | p. 16 |
| $\cos .(x)$ | broadcast operation (elementwise) | p. 36 |
| $[x]_+$ | $\max(x, 0)$ | p. 259 |
| Matrices and linear algebra | | |
| \mathcal{S}, \mathcal{V} | subspace, vector space | p. 96 |
| \mathcal{S}^\perp | subspace orthogonal complement | p. 107 |
| P^\perp | project onto orthogonal complement | p. 133 |
| \subseteq | subspace of | p. 96 |
| \oplus | subspace direct sum | p. 106 |
| \otimes | Kronecker product | p. 34 |
| \odot | Hadamard product (elementwise) | p. 34 |
| A, \dots, Z | matrices: bold upper-case letters | |
| I | identity matrix | p. 18 |
| a, \dots, z | vectors: bold lower-case letters | |
| x_n | n th element of vector x | |
| a_{mn} | element in row m , column n of matrix A | |
| A^\top | transpose of matrix A | p. 20 |
| A' | Hermitian transpose of matrix A | p. 20 |
| A^+ | Moore–Penrose pseudoinverse of matrix A | p. 155 |
| λ_k | eigenvalues of a square matrix | p. 50 |
| σ_k | singular values of a matrix | p. 74 |
| $\langle x, y \rangle$ | inner product | p. 203 |
| $x \perp y$ | orthogonality | p. 42 |
| $\ x\ $ | vector norm | p. 42 |
| $\ A\ $ | submultiplicative matrix norm | p. 206 |

| Symbol | What | Where |
|-----------------------------|---|--------|
| $\mathbf{0}$ | vector (or matrix) of all zeros | p. 9 |
| $\mathbf{1}$ | vector of all ones | p. 36 |
| $A \succeq \mathbf{0}$ | A is positive semidefinite | p. 88 |
| $A > \mathbf{0}$ | A is positive definite | p. 88 |
| $A \succeq B$ | $A - B$ is positive semidefinite | p. 341 |
| $A > B$ | $A - B$ is positive definite | p. 341 |
| $\text{Diag}(x)$ | diagonal matrix from elements of vector x | p. 18 |
| $\mathcal{R}(A)$ | range of matrix A | p. 109 |
| $\mathcal{N}(A)$ | nullspace of matrix A | p. 116 |
| $\kappa(A)$ | condition number of matrix A | p. 164 |
| $\rho(A)$ | spectral radius of matrix A | p. 217 |
| e_i | i th unit vector (standard basis) | p. 24 |
| Vectors and matrices | | |
| \mathbb{R}^N | N -tuples of real numbers | p. 9 |
| \mathbb{C}^N | N -tuples of complex numbers | p. 9 |
| \mathbb{F}^N | either \mathbb{R}^N or \mathbb{C}^N | |
| $\mathbb{R}^{M \times N}$ | real $M \times N$ matrices | |
| $\mathbb{C}^{M \times N}$ | complex $M \times N$ matrices | |
| $\mathbb{F}^{M \times N}$ | either $\mathbb{R}^{M \times N}$ or $\mathbb{C}^{M \times N}$ | |

1.4.1 What the \mathbb{F} Means

The notation \mathbb{F} for a scalar field used here will typically be \mathbb{R} or \mathbb{C} . We will use this symbol frequently to discuss properties that hold for both real and complex cases. Because $\mathbb{R}^{M \times N} \subset \mathbb{C}^{M \times N}$, whenever you see the symbol $\mathbb{F}^{M \times N}$ you can just think of it as $\mathbb{C}^{M \times N}$, but it means the properties being discussed also hold for $\mathbb{R}^{M \times N}$.

To explain the convenience of using the \mathbb{F} notation, consider the following logical implication statements about vector addition, that is, $z = x + y$, all of which are true:

- (i) $x \in \mathbb{R}^N, y \in \mathbb{R}^N \implies z \in \mathbb{R}^N$.
- (ii) $x \in \mathbb{C}^N, y \in \mathbb{C}^N \implies z \in \mathbb{C}^N$.
- (iii) $x \in \mathbb{R}^N, y \in \mathbb{C}^N \implies z \in \mathbb{C}^N$.
- (iv) $x \in \mathbb{C}^N, y \in \mathbb{R}^N \implies z \in \mathbb{C}^N$.
- (v) $x \in \mathbb{R}^N, y \in \mathbb{R}^N \implies z \in \mathbb{C}^N$.

Because $\mathbb{R} \subset \mathbb{C}$, statement (ii) implies (iii), (iv), and (v). However, (ii) does *not* imply (i)! Of course (i) is true, but simply stating or proving (ii) by itself is insufficient to conclude that (i) is true. So to thoroughly describe vector addition one would need to prove (or state as fact) both (i) and (ii) separately. Instead, this book uses statements like this:

$$x \in \mathbb{F}^N, y \in \mathbb{F}^N \implies z = x + y \in \mathbb{F}^N,$$

which is essentially shorthand for stating both (i) and (ii) above. Proving a statement like that may require separately proving (i) and (ii).

1.5 Julia

All code examples in this book use the open-source JULIA programming language, which is designed for numerical computing [1]. It is interactive (like Python and MATLAB), yet has fast execution because it is compiled. The examples in this book were developed using JULIA 1.9, but should work with JULIA 1.6 or higher. We recommend using the latest 1.x version because JULIA speed tends to improve each version while retaining backward compatibility. To get started with JULIA, install the free application from <https://julialang.org>. For additional tips about editors, tutorials, and so forth, see <https://jefffessler.github.io/book-la-demo>.

Table 1.1 briefly compares three interactive languages. The JULIA column assumes you have typed `using LinearAlgebra` to use the `dot` function.

1.6 Fields, Vector Spaces, Linear Maps



There are multiple different meanings of the term vector in the literature.

- In MATLAB, a vector is simply a column of a two-dimensional (2D) matrix. In other words, a MATLAB (column) vector is an $N \times 1$ array of numbers, that is, a 2D array where the second dimension is 1.

Table 1.1. Syntax comparison.

| Operation | MATLAB | JULIA | Python <code>import numpy as np</code> |
|-----------------------|--|-------------------------------------|--|
| Dot product | <code>dot(x,y)</code> | <code>dot(x,y)</code> | <code>np.dot(x,y)</code> |
| Matrix multiplication | <code>A * B</code> | <code>A * B</code> | <code>A @ B</code> |
| Elementwise | <code>A .* B</code> | <code>A .* B</code> | <code>A * B</code> |
| Scaling | <code>3 * A</code> | <code>3A</code> or <code>3*A</code> | <code>3 * A</code> |
| Matrix power | <code>A^2</code> | <code>A^2</code> | <code>np.linalg.matrix_power(A,2)</code> |
| Elementwise | <code>A.^2</code> | <code>A.^2</code> | <code>A**2</code> |
| Inverse | <code>inv(A)</code> | <code>inv(A)</code> | <code>np.linalg.inv(A)</code> |
| Inverse | <code>A^(-1)</code> | <code>A^(-1)</code> | <code>np.linalg.inv(A)</code> |
| Indexing | <code>A(i,j)</code> | <code>A[i,j]</code> | <code>A[i-1,j-1]</code> |
| Range | <code>1:9</code> | <code>1:9</code> | <code>np.arange(1,10,1)</code> |
| Range | <code>linspace(0,4,9)</code> | <code>range(0,4,9)</code> | <code>np.arange(0,4.01,0.5)</code> |
| Strings | <code>'text'</code> or <code>"text"</code> | <code>"text"</code> | <code>'text'</code> or <code>"text"</code> |
| Slicing | <code>v(1:end)</code> | <code>v[1:end]</code> | <code>v[0:]</code> |
| Inline function | <code>f = @(x,y) x+y</code> | <code>f(x,y) = x+y</code> | <code>f = lambda x,y : x+y</code> |
| Increment | <code>A = A + B</code> | <code>A += B</code> | <code>A += B</code> |
| Hermitian transpose | <code>A'</code> | <code>A'</code> | <code>A.conj().T</code> |
| | $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ | <code>[1 2; 3 4]</code> | <code>np.array([[1, 2], [3, 4]])</code> |

- In JULIA, a `Vector` is a 1D array of N values, typically numbers. This convention is more consistent with numerical linear algebra. More precisely, a variable of type `Vector{<:Number}` is a 1D array of numbers. The JULIA `Vector` type is general enough to hold any elements.
- In general mathematics, for example, linear algebra and functional analysis, a vector belongs to a vector space.

Although this book mostly uses the numerical methods perspective, readers who want a thorough understanding should also be familiar with the more general notion of a vector space.

-  Demo 1.2 provides more illustrations of how vectors work in JULIA.

This section reviews vector spaces and linear operators defined on vector spaces. Although the definitions in this section are quite general and thus might appear somewhat abstract, the ideas are important even for the topics of an undergraduate signals and systems course. For example, analog systems like passive RLC networks are linear systems that are represented mathematically by linear maps from one (infinite-dimensional) vector space to another.

The definition of a vector space uses the concept of a field of scalars, so we first review that.

1.6.1 Field of Scalars

A field or field of scalars \mathbb{F} is a collection of elements $\alpha, \beta, \gamma, \dots$ along with an “addition” and a “multiplication” operator [15].

For every pair of scalars α, β in \mathbb{F} , there must correspond a scalar $\alpha + \beta$ in \mathbb{F} , called the sum of α and β , such that:

- Addition is commutative: $\alpha + \beta = \beta + \alpha$.
- Addition is associative: $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$.
- There exists a unique element $0 \in \mathbb{F}$, called zero, for which $\alpha + 0 = \alpha, \forall \alpha \in \mathbb{F}$.
- For every $\alpha \in \mathbb{F}$, there corresponds a unique scalar $(-\alpha) \in \mathbb{F}$ for which $\alpha + (-\alpha) = 0$.

For every pair of scalars α, β in \mathbb{F} , there must correspond a scalar $\alpha\beta$ in \mathbb{F} , called the product of α and β , such that:

- Multiplication is commutative: $\alpha\beta = \beta\alpha$.
- Multiplication is associative: $\alpha(\beta\gamma) = (\alpha\beta)\gamma$.
- Multiplication distributes over addition: $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$.
- There exists a unique element $1 \in \mathbb{F}$, called one, or unity, or the identity element, for which $1\alpha = \alpha, \forall \alpha \in \mathbb{F}$.
- For every nonzero $\alpha \in \mathbb{F}$, there corresponds a unique scalar $\alpha^{-1} \in \mathbb{F}$, called the inverse of α , for which $\alpha\alpha^{-1} = 1$.

Example 1.2 The set \mathbb{Q} of rational numbers is a field (with the usual definitions of addition and multiplication).

The primary fields needed in this book are the field of real numbers \mathbb{R} and the field of complex numbers \mathbb{C} .

1.6.2 Vector Spaces

A vector space or linear space consists of:

- a field \mathbb{F} of scalars;
- a set \mathcal{V} of entities called vectors;
- an operation called vector addition that associates a sum $x + y \in \mathcal{V}$ with each pair of vectors $x, y \in \mathcal{V}$ such that:
 - addition is commutative: $x + y = y + x$;
 - addition is associative: $x + (y + z) = (x + y) + z$;
 - there exists a unique element $\mathbf{0} \in \mathcal{V}$, called the zero vector, for which $x + \mathbf{0} = x$, $\forall x \in \mathcal{V}$;
 - For every $x \in \mathcal{V}$, there corresponds a unique vector $(-x) \in \mathcal{V}$ for which $x + (-x) = \mathbf{0}$;
- an operation called multiplication by a scalar that associates with each scalar $\alpha \in \mathbb{F}$ and vector $x \in \mathcal{V}$ a vector $\alpha x \in \mathcal{V}$, called the product of α and x , with the following properties:
 - associative: $\alpha(\beta x) = (\alpha\beta)x$;
 - distributive $\alpha(x + y) = \alpha x + \alpha y$;
 - distributive $(\alpha + \beta)x = \alpha x + \beta x$;
 - if 1 is the identity element of \mathbb{F} , then $1x = x$, $\forall x \in \mathcal{V}$.

(We do not require other operations to be defined, such as multiplying two vectors or adding a vector and a scalar.)

We describe such a space as an \mathbb{F} -vector space or a vector space over \mathbb{F} .

Example 1.3 An example of a vector space is the real coordinate space of dimension n , or Euclidean n -dimensional space, or n -tuple space: $\mathcal{V} = \mathbb{R}^n$. If $x \in \mathcal{V}$, then $x = (x_1, x_2, \dots, x_n)$ where $x_i \in \mathbb{R}$. The field of scalars is $\mathbb{F} = \mathbb{R}$. Of course, $x + y = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n)$, and $\alpha x = (\alpha x_1, \dots, \alpha x_n)$.



This space is very closely related to the notion of a vector being a column of a matrix. They are so close that much of the literature does not distinguish them (nor does MATLAB). However, to be rigorous, \mathbb{R}^n is not the same as a column of a matrix because strictly speaking there is no inherent definition of the transpose of a vector in \mathbb{R}^n , whereas transpose is well defined for any matrix including an $n \times 1$ matrix. See Section 2.3.2.

Vectors in \mathbb{F}^N are typically thought of as column vectors here. When needed, row vectors are written as x^\top or x' , where $x \in \mathbb{R}^N$ or $x \in \mathbb{C}^N$.

Example 1.4 The complex coordinate space of dimension n or complex Euclidean n -dimensional space: $\mathcal{V} = \mathbb{C}^n$. If $x \in \mathcal{V}$, then $x = (x_1, x_2, \dots, x_n)$ where $x_i \in \mathbb{C}$. The field of scalars is $\mathbb{F} = \mathbb{C}$, $x + y = (x_1 + y_1, \dots, x_n + y_n)$, and $\alpha x = (\alpha x_1, \dots, \alpha x_n)$.

Example 1.5 The set of functions $f: \mathbb{R} \mapsto \mathbb{C}$ that are continuous. Addition and scalar multiplication are defined in the natural way. This is a vector space because linear combinations of continuous functions are continuous.

Example 1.6 The set of functions on the plane \mathbb{R}^2 that are zero outside of the unit square.

Example 1.7 The set of solutions to a homogeneous linear system of equations $Ax = 0$.

Example 1.8 The set \mathcal{V} of diagonal matrices whose diagonal elements are rational numbers, over the field \mathbb{Q} .

- If A and B are both in \mathcal{V} , then their sum $A + B$ is also a diagonal matrix whose elements are rational numbers.
- If $A \in \mathcal{V}$ and $\alpha \in \mathbb{Q}$ is a rational number, then αA is again a diagonal matrix whose elements are rational numbers.

The other properties are easily verified.



In general, the vectors in a vector space need not consist of elements that are taken from the field \mathbb{F} .

Example 1.9 The set \mathcal{V} of $M \times N$ matrices whose elements are complex numbers, over the field \mathbb{R} .

- If A and B are both in \mathcal{V} , then their sum $A + B$ is also an $M \times N$ matrix whose elements are complex numbers.
- If $A \in \mathcal{V}$ and $\alpha \in \mathbb{R}$ is a real number, then αA is again an $M \times N$ matrix whose elements are complex numbers.

The other properties are easily verified.

The preceding two examples serve as a reminder that the concept of vector is quite general, and even includes matrices!

1.6.3 Linear Maps and Linear Operators

The concept of linearity in linear algebra is quite general.

Definition Let \mathcal{U} and \mathcal{V} be two vector spaces over a common field \mathbb{F} . A function $\mathcal{A}: \mathcal{U} \rightarrow \mathcal{V}$ is called a linear transform or linear map from \mathcal{U} into \mathcal{V} iff, $\forall u_1, u_2 \in \mathcal{U}$ and all scalars $\alpha, \beta \in \mathbb{F}$,

$$\mathcal{A}(\alpha u_1 + \beta u_2) = \alpha \mathcal{A}(u_1) + \beta \mathcal{A}(u_2). \quad (1.3)$$

Example 1.10 An $M \times N$ matrix A with elements in the field \mathbb{F} is naturally affiliated with the linear map from \mathbb{F}^N to \mathbb{F}^M defined by $f(x) = Ax$. This affiliation is so strong that we often refer to the matrix and the corresponding linear map interchangeably.

Example 1.11 Let $\mathbb{F} = \mathbb{R}$ and let \mathcal{V} be the space of continuous functions on \mathbb{R} .

Define the linear map \mathcal{A} by: if $F = \mathcal{A}(f)$ then $F(x) = \int_0^x f(t) dt$. Thus, integration (with suitable limits) is linear.

If \mathcal{A} is a linear map from \mathcal{V} into \mathcal{V} , then we say \mathcal{A} is a linear operator. However, the terminology distinguishing linear maps from linear operators is not universal, and the two terms are often used interchangeably.

A simple fact for linear maps: $\mathcal{A}(\mathbf{0}) = \mathbf{0}$. Proof: $\mathcal{A}(\mathbf{0}) = \mathcal{A}(0\mathbf{0}) = 0\mathcal{A}(\mathbf{0}) = \mathbf{0}$. This is called the “zero in, zero out” property.



The function $x \mapsto y = ax + b$ defines the equation for a line, but it is not a linear function. It is an affine function.



It follows that $\mathcal{A}\left(\sum_{i=1}^n \alpha_i u_i\right) = \sum_{i=1}^n \alpha_i \mathcal{A}(u_i)$ for any finite n , by induction [16]. But the above *does not* imply in general that linearity holds for infinite summations or integrals. Further assumptions about “smoothness” or “regularity” or “continuity” of \mathcal{A} are needed for that.

Solutions to Explorations

Explore 1.1 The number here is also a hyperlink in the e-version; a click on it should cause the viewer to jump back to the problem statement.

2 Introduction to Matrices

2.1 Introduction

This chapter reviews vectors and matrices, and basic *properties* like shape, orthogonality, determinant, eigenvalues, and trace. It also reviews some *operations* like multiplication and transpose. These operations will be used throughout the book and are pervasive in the literature. In short, arranging data into vectors and matrices allows one to apply powerful data analysis techniques over a wide spectrum of applications. Throughout, this chapter (and book) illustrates how the ideas are implemented in practice in JULIA.

2.2 Basics of Vectors and Matrices

Viewing data as vectors is ubiquitous in DS–ML–SP. Sets of vectors are often organized into matrices for data analysis or are acted on by matrix operators.

Example 2.1 Personal attributes:

$$\begin{bmatrix} \text{age} \\ \text{height} \\ \text{weight} \\ \text{eye color} \\ \vdots \end{bmatrix}$$

Example 2.2 Digital grayscale image.

- Each such image is a vector in the vector space of 2D arrays.
- We often think of a digital image as a vector rather than as a “matrix!”

(See Section 1.6 about general vector spaces.)

A matrix is a 2D array of numbers. There are two primary matrix uses:

- organizing data as a 2D array;
- representing a linear operation, otherwise known as a linear map or linear transform.

Some matrix methods are used primarily for data matrices, such as principal component analysis (PCA, Section 7.7.2), whereas others are used primarily for operator matrices, such as the eigendecomposition of circulant matrices (Section 8.3).

Example 2.3 An $M \times N$ data matrix for a group of N people with M attributes each:

$$X = \begin{bmatrix} \text{age}_1 & \cdots & \text{age}_N \\ \text{height}_1 & \cdots & \text{height}_N \\ \text{weight}_1 & \cdots & \text{weight}_N \\ \text{eye color}_1 & \cdots & \text{eye color}_N \\ \vdots & & \vdots \end{bmatrix}.$$

Each column of this $M \times N$ matrix X is a feature vector for one person.

Example 2.4 A classic use of matrices and vectors is solving a system of linear equations with N equations in N unknowns:

$$\begin{aligned} ax_1 + bx_2 + cx_3 &= u \\ dx_1 + ex_2 + fx_3 &= v \\ gx_1 + hx_2 + ix_3 &= w \end{aligned} \implies Ax = b, \text{ with } A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, b = \begin{bmatrix} u \\ v \\ w \end{bmatrix}.$$

Certainly the matrix–vector notation $Ax = b$ is more concise (and general) than writing out each equation individually. A traditional linear algebra course would focus extensively on solving $Ax = b$. This topic will not be a major focus here!

Often, dynamical systems are characterized using matrices.

Example 2.5 The sequence of Fibonacci numbers $0, 1, 1, 2, 3, 5, \dots$ represented by f_0, f_1, f_2, \dots satisfies the following recursion:

$$\begin{bmatrix} f_i \\ f_{i-1} \end{bmatrix} = A \begin{bmatrix} f_{i-1} \\ f_{i-2} \end{bmatrix}, \quad A \triangleq \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

One of the eigenvalues (see Section 2.7) of A is the golden ratio $(1 + \sqrt{5})/2$. Here, A describes the mapping from one state to the next.

Example 2.6 A term–document matrix is a *data matrix* used in information retrieval. For example:

Document1: EECSS51 meets on Tuesdays and Thursdays

Document2: Tuesday is the most exciting day of the week

Document3: Let us meet next Thursday.

Keywords (terms): EECSS51 meet Tuesday Thursday exciting day week next

(In extracting the keywords, we use the stem (no “s” for plurals), and ignore generic words like “the,” “and.”) See Fig. 2.1.

The entries in a (mostly) numerical table like this are naturally represented by an 8×3 matrix T (because each column is a vector in \mathbb{R}^8 and there are three columns), as seen in Fig. 2.1.

| Term | Doc1 | Doc2 | Doc3 | |
|----------|------|------|------|--|
| EECS551 | 1 | 0 | 0 | |
| meet | 1 | 0 | 1 | |
| Tuesday | 1 | 1 | 0 | |
| Thursday | 1 | 0 | 1 | |
| exciting | 0 | 1 | 0 | |
| day | 0 | 1 | 0 | |
| week | 0 | 1 | 0 | |
| next | 0 | 0 | 1 | |

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad q = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 2.1 Term-document array (with labeled rows and columns), (binary) matrix T , and example query vector q .

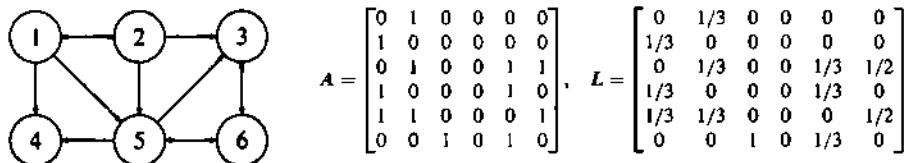


Figure 2.2 Directed graph for a network, and its adjacency matrix A and link graph matrix L .

Explore 2.1 Why “mostly?”

With this representation, a query becomes a matrix–vector operation. To find all documents relevant to the query “exciting days of the week,” we use the example query vector q in Fig. 2.1. In matrix terms, find columns of T that are “close” (defined precisely in Section 6.2) to query vector q . This is an information retrieval application expressed using matrix/vector operations.

Example 2.7 Consider a set of six web pages that are related via web links (outlinks and inlinks) according to the directed graph in Fig. 2.2 [5, Example 1.3, p. 7]. The 6×6 adjacency matrix A for this graph has a nonzero value (unity) in element A_{ij} if there is a link from node (web page) j to node i . The link graph matrix L is found from the adjacency matrix by normalizing (each column) with respect to the number of outlinks so that each column sums to unity.

Section 8.8.4 explains that Google’s PageRank algorithm [17] for quantifying the importance of web pages is related to an eigenvector of L . (There is a left eigenvector with $\lambda = 1$ so there must also be a right eigenvector with that eigenvalue, and that is the one of interest.) Representing relationships using a matrix and computing properties of that matrix is useful in many domains, even some that might seem quite unexpected at first.

In the preceding two examples, the matrix was a 2D array of *data*. Recall that there were two “whys” for matrices: *data* and *linear operations*. Now we turn to an example where the matrix is associated with a linear operation that we apply to vectors.

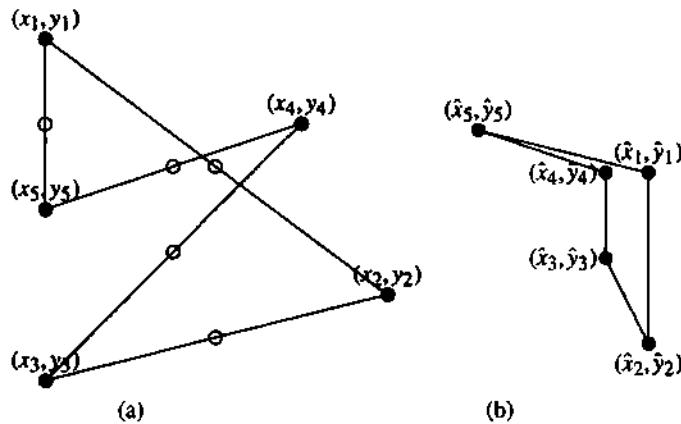


Figure 2.3 Polygon transformation operation from P_1 (a) to P_2 (b).

Example 2.8 Consider a polygon P_1 defined by N vertex points $(x_1, y_1), \dots, (x_N, y_N)$, connected in that order, and where the last point is also connected to the first point so there are N edges. See Fig. 2.3. Define a new polygon P_2 where each vertex is the *average* of the two points along an edge of polygon P_1 . Mathematically, the first new vertex point is linearly related to the old points as: $\hat{x}_1 = (x_1 + x_2)/2$, $\hat{y}_1 = (y_1 + y_2)/2$. In general, the n th new vertex point is related to the old vertex points by the following formula:

$$\hat{x}_n = \frac{x_n + x_{1+\text{mod}(n,N)}}{2}, \quad \hat{y}_n = \frac{y_n + y_{1+\text{mod}(n,N)}}{2}, \quad n = 1, \dots, N.$$

(The $1 + \text{mod}(n, N)$ closes the polygon because the modulo operation has the property that $1 + \text{mod}(N, N) = 1$.) It is convenient to collect all of these relationships into length- N vectors as follows:

$$\begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_{N-1} \\ \hat{x}_N \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_1 + x_2 \\ \vdots \\ x_{N-1} + x_N \\ x_N + x_1 \end{bmatrix}, \quad \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_{N-1} \\ \hat{y}_N \end{bmatrix} = \frac{1}{2} \begin{bmatrix} y_1 + y_2 \\ \vdots \\ y_{N-1} + y_N \\ y_N + y_1 \end{bmatrix}. \quad (2.1)$$

This is a linear operation so we can write it concisely using matrix–vector multiplication:

$$\hat{x} = Ax, \quad \hat{y} = Ay, \quad A \triangleq \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}. \quad (2.2)$$

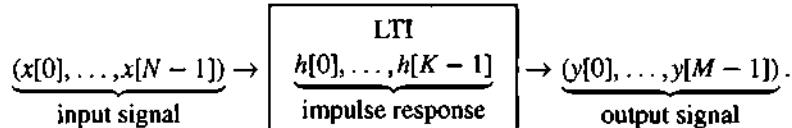
The matrix form might not seem more illuminating than the expressions in (2.1) at first. But now suppose we ask the following question: What happens to the polygon shape if we perform the process repeatedly (assuming we do some appropriate normalization)? The answer is not obvious from (2.1) but in matrix form the answer is related to the power iteration discussed in Section 8.5 for computing a principal eigenvector of A [18].

Example 2.9 Important linear operations in signal processing include the 1D and 2D discrete Fourier transform (DFT). They can both be expressed as matrix–vector products. The N -point 1D DFT has the form

$$\underbrace{X}_{\text{spectrum in } \mathbb{C}^N} = \underbrace{W}_{N \times N \text{ DFT}} \underbrace{x}_{\text{signal in } \mathbb{C}^N},$$

where $W_{kn} = e^{-j2\pi(k-1)(n-1)/N}$, $k, n = 1, \dots, N$. One can also write the 2D case as $X = Wx$ for a suitable matrix W .

Example 2.10 An important linear operation is convolution in digital signal processing (DSP) for representing a linear time-invariant (LTI) discrete-time system:



Here we consider the case of modeling a causal system, like one would use in audio signal processing, where the output at any given time depends only on the current and past inputs. For such systems, the impulse response $h[n]$ is zero for $n < 0$ and the convolution formula is

$$y[m] = (h * x)[m] \triangleq \sum_{k=\max(0, m-N+1)}^{\min(K-1, m)} h[k]x[m-k]. \quad (2.3)$$

Here, the output $y[m]$ depends only on the inputs $x[m], x[m-1], \dots, x[m - \min(K-1, m)]$. Because convolution is a linear operation, when $N, K \in \mathbb{N}$ it has the matrix–vector representation $\mathbf{y} = \mathbf{H}\mathbf{x}$, where \mathbf{x} is a length- N vector, \mathbf{y} is a length- M vector, and \mathbf{H} is an $M \times N$ matrix with elements

$$H_{mn} = h[m-n] \mathbb{I}_{\{0 \leq m-n < K\}}, \quad m = 1, \dots, M, \quad n = 1, \dots, N, \quad (2.4)$$

where $\mathbb{I}_{\{\cdot\}}$ denotes the indicator function that is 1 when the argument is true and 0 otherwise.

Q2.1 What is M in terms of N and K ? (Choose the best answer.)

- A: $\max(N, K) - 1$ B: $\max(N, K)$ C: $N + K$ D: $N + K + 1$ E: None of these.

The convolution operation is the key component of any convolutional neural network (CNN; see [19] or [20, p. 514]). Other matrices represent other important linear operations: wavelet transform, discrete cosine transformation, 2D DFT, ...

2.3 Matrix Structures

2.3.1 Common Matrix Shapes and Types

For each matrix shape, the following table gives one of many examples of why that shape is useful in practice. These shapes are defined by where nonzero values may be; that is, the values indicated by \times can be anything, including zero. A matrix of all zeros is in all of these categories!

| | | |
|------------------|--|--|
| Diagonal | $\begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}$ | Covariance matrix of a random vector with uncorrelated elements. Easiest to invert! Definition: $a_{ij} = 0$ if $i \neq j$. |
| Upper triangular | $\begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}$ | Arises in Gaussian elimination for solving systems of equations. Easy to invert. Definition: $a_{ij} = 0$ if $j < i$. |
| Lower triangular | $\begin{bmatrix} \times & 0 & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times \end{bmatrix}$ | Used in the Cholesky decomposition. |
| Tridiagonal | $\begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 \\ 0 & \times & \times & \times & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$ | For finite difference approximation of a second derivative. Arises in numerical solutions to differential equations. |
| Pentadiagonal | $\begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$ | Gram matrix for discrete approximation to second derivative. |
| Upper Hessenberg | $\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$ | Arises in eigenvalue algorithms. Fairly easy to invert. |
| Lower Hessenberg | $\begin{bmatrix} \times & \times & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}$ | Ditto. |

These are all square matrix shapes! There is also a rectangular diagonal matrix shape that will be useful in Section 3.3 for the singular value decomposition:

$$\text{tall: } \left[\begin{array}{c|ccccc} \sigma_1 & & & & & \\ \hline & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \sigma_r \\ \hline & \mathbf{0} & & & & \end{array} \right] \quad \text{wide: } \left[\begin{array}{c|ccccc} \sigma_1 & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \sigma_r \\ \hline & & & & & \mathbf{0} \end{array} \right].$$

Often these are just called “diagonal matrices” too.

2.3.1.1 Important Matrix Classes

- A full matrix or dense matrix: most entries are nonzero.
- A sparse matrix: many entries are zero or few entries are nonzero. These arise in many fields. (Obviously, terms like “most,” “many,” and “few” are qualitative.)
 - To store an $M \times N$ dense matrix in memory, we must store the MN values and also the size M, N .
 - To store a sparse matrix with K nonzero elements, we must store those values and also store the *index* of each value in some sparse storage format. Typically this is about $2K$ values.
- An $M \times N$ matrix is tall if $M > N$ and is wide if $M < N$.
- We often define a diagonal matrix in terms of a vector whose elements correspond to the diagonal elements of the matrix.

Example 2.11 For the vector $x = \begin{bmatrix} 9 \\ 8 \\ 7 \end{bmatrix}$, we write

$$\text{Diag}(x) = \text{Diag}(9, 8, 7) = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 7 \end{bmatrix}.$$

In JULIA: `x = [9, 8, 7]; D = Diagonal(x)`

- An especially important diagonal matrix is the identity matrix, denoted I , that has ones along its diagonal and zeros elsewhere. We often use a subscript to denote its size, for example, $I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. In JULIA: `I(3)`.
- Toeplitz: elements are constant along each diagonal (not necessarily square). These arise in any application that has time invariance or shift invariance. (In mathematics, one says that the translation operation is equivariant for such applications, but the term invariance is commonly used in signal processing.)

Fact 2.1 Matrix A is Toeplitz iff its elements a_{ij} have the form $a_{ij} = f(i - j)$ for some function $f: \mathbb{Z} \mapsto \mathbb{F}$.

- Circulant: each row is a shifted (circularly to the right by one column) version of the previous row. Matrices in this class are always square, and arise when considering periodic boundary conditions, for example, with the DFT.

Fact 2.2 An $N \times N$ matrix A is circulant iff its elements a_{ij} have the form $a_{ij} = f(\text{mod}(i - j, N))$ for some function $f: \{0, \dots, N - 1\} \mapsto \mathbb{F}$.

Example 2.12

| | | |
|--|---|--|
| Toeplitz matrix: $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 1 & 2 & 3 \\ 6 & 5 & 1 & 2 \end{bmatrix}$. | . | Circulant matrix: $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}$. |
|--|---|--|

Q2.2 Which statement is correct?

- A: All Toeplitz matrices are circulant.
- B: All circulant matrices are Toeplitz.
- C: Both are true.
- D: Neither statement is true.

Q2.3 Which statement is correct?

- A: All Toeplitz matrices are full.
- B: All sparse matrices are Toeplitz.
- C: There are no sparse Toeplitz matrices.
- D: None of these statements is true.

Q2.4 What kind of matrix is A in (2.2)? Choose the most specific correct answer.

- A: Diagonal
- B: Toeplitz
- C: Circulant
- D: Upper Hessenberg
- E: None

For the next two questions, recall from (2.4) that the 1D convolution matrix H has elements $H_{mn} = h[m - n] \mathbb{I}_{\{0 \leq m-n < K\}}$.

Q2.5 What kind of matrix is the convolution matrix H in (2.4)? Choose the most specific correct answer.

- A: Diagonal
- B: Toeplitz
- C: Circulant
- D: Upper Hessenberg
- E: None

Q2.6 What kind of matrix is the top $N \times N$ block of the convolution matrix H in (2.4)? Choose the most specific correct answer.

- A: Upper triangular
- B: Lower triangular
- C: Circulant
- D: Upper Hessenberg
- E: Lower Hessenberg

2.3.1.2 Convolution as a Toeplitz Matrix



Demo 2.1 illustrates convolution matrices, including (2.4) and variants for other boundary conditions. Specifically, Fig. 2.4 illustrates an impulse response for a noncausal filter $h[k]$ with $K = 5$, like one would use for processing images with a CNN. Figure 2.5 shows the corresponding H for $N = 10$ with various end conditions described in the demo. In each case here H is Toeplitz, and also circulant in one case.

2.3.1.3 Block Matrix Classes

The above definitions of shapes and classes were defined in terms of the scalar entries of a matrix. We generalize these definitions by replacing scalars with matrices, leading to block matrix shapes. We again use square brackets to denote the construction of a block matrix.

Example 2.13 A block diagonal matrix

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix},$$

where A_1 and A_2 are arbitrary (possibly full) matrices and each 0 denotes an all-zero matrix of the appropriate size. (Wikipedia says that A_1 and A_2 should be square, but that restriction is unnecessary.)

Q2.7 If A_1 is 6×8 and A_2 is 7×9 , what is the size of the 0 matrix in the upper right?
 A: 6×7 B: 6×9 C: 7×8 D: 8×7 E: 8×9

Example 2.14 A block circulant matrix:

$$M = \begin{bmatrix} A & B & C \\ C & A & B \\ B & C & A \end{bmatrix}.$$

Combinations of block shapes and the shape of each block are also important.

Example 2.15 A matrix form of the 2D DFT is block circulant with circulant blocks (BCCB).

2.3.2 Matrix Transpose and Symmetry

Definition The transpose of an $M \times N$ matrix A is denoted A^T and is the $N \times M$ matrix whose (j, i) th entry is the (i, j) th entry of A . In notation:

$$[A^T]_{j,i} = [A]_{i,j}, \quad i = 1, \dots, M, j = 1, \dots, N.$$

Definition If $A \in \mathbb{C}^{M \times N}$ then its Hermitian transpose (or conjugate transpose or adjoint) is the $N \times M$ matrix whose (i, j) th entry is the complex conjugate of the (j, i) th

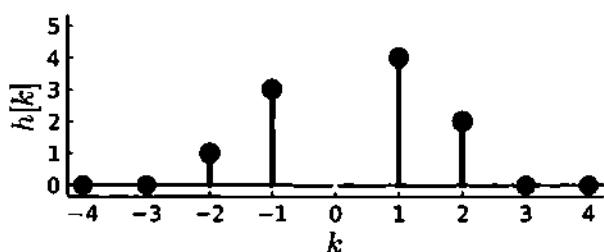


Figure 2.4 Impulse response $h[k]$.

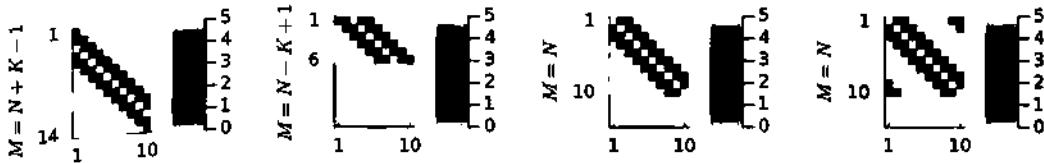


Figure 2.5 Convolution matrices for various boundary conditions.

entry of A :

$$[A']_{j,i} = [A]_{i,j}^*, \quad i = 1, \dots, M, j = 1, \dots, N.$$

Common notations for Hermitian transpose are A' , A^H , and A^* . This book will mostly use A' because that notation matches JULIA. If A is real, then $A' = A^\top$, so for simplicity of notation this book will mostly use A' regardless of whether A is real or complex.

Definition A (square) matrix A is called symmetric iff $A = A^\top$.

Definition A (square) matrix A is called Hermitian or Hermitian symmetric iff $A = A'$.

Example 2.16 The Hermitian transpose of $A = \begin{bmatrix} 1 & 2+3i \\ e^{i\theta} & 4i \end{bmatrix}$ is $A' = \begin{bmatrix} 1 & e^{-i\theta} \\ 2-3i & -4i \end{bmatrix}$ for $\theta \in \mathbb{R}$. This matrix is neither symmetric nor Hermitian.

Recall that a complex number can be written as $z = a + ib$ where $a, b \in \mathbb{R}$ and $i = \sqrt{-1}$. The complex conjugate of this number is $z^* = a - ib$. The modulus of this number is $|z| = \sqrt{a^2 + b^2} = \sqrt{zz^*}$. Another way of writing a complex number is in polar form like $z = r e^{i\phi}$ for $r = \sqrt{a^2 + b^2} \in \mathbb{R}$ and $\phi = \angle z = \arctan(b/a)$ where, by Euler's formula, $z = r \cos(\phi) + ir \sin(\phi)$, and \arctan denotes the two-argument arctangent, named `atan` in JULIA. Thus the complex conjugate in polar form is $z^* = (r e^{i\phi})^* = (r \cos(\phi) + ir \sin(\phi))^* = r \cos(\phi) - ir \sin(\phi) = r \cos(-\phi) + ir \sin(-\phi) = r e^{-i\phi}$. The preceding example used both of these forms of complex conjugate.

Example 2.17 $\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$ is symmetric and Hermitian.

Example 2.18 $\begin{bmatrix} 1 & 2-3i \\ 2-3i & 4 \end{bmatrix}$ is symmetric but not Hermitian.

Example 2.19 $\begin{bmatrix} 1 & 2-3i \\ 2+3i & 4 \end{bmatrix}$ is Hermitian (but not symmetric).

Example 2.20 For $\theta \in \mathbb{R}$, the Hermitian transpose of the vector $x = \begin{bmatrix} 1 \\ 2 - 3i \\ e^{i\theta} \end{bmatrix}$ is perhaps most naturally thought of as the 1×3 array $x' = [1 \ 2 + 3i \ e^{-i\theta}]$.

In JULIA try `[2,4,6]'` and note the returned type.

2.3.2.1 Properties of the (Hermitian) Transpose

Four key operations with matrices are scalar multiplication, matrix multiplication, addition, and stacking. For each matrix property presented, we will strive to analyze how these four operations affect that property.

- Involution: $(A')' = A$.
- Scaling: $(\alpha A)' = \alpha^* A'$ for $\alpha \in \mathbb{F}$.
- Addition: $(A + B)' = A' + B'$ (if A and B have same size so that matrix addition is well defined).
- Multiplication: $(AB)' = B'A'$ if $A \in \mathbb{F}^{M \times N}$ and $B \in \mathbb{F}^{N \times K}$, that is, if the inner dimensions match. (See matrix multiplication in Section 2.4.3.)
- Stacking: $\begin{bmatrix} A & B \\ C & D \end{bmatrix}' = \begin{bmatrix} A' & C' \\ B' & D' \end{bmatrix}$.

2.3.2.2 Practical Implementation



Caution: in JULIA, `x'` denotes the Hermitian transpose of a (possibly complex) vector or matrix `x`, whereas `x.'` in MATLAB or `transpose(x)` in JULIA denotes the transpose. In JULIA, the Hermitian transpose syntax `A'` calls `adjoint(A)`. To see this, try `@which ones(2)'`.

When performing mathematical operations with complex data, we usually need the Hermitian transpose. However, when simply rearranging how data is stored, sometimes we need only the transpose. Many hours of MATLAB software debugging time are spent on missing or extra periods (i.e., `x'` vs. `x.'`) for a transpose! To help avoid this problem, JULIA 1.0 and beyond has discontinued `x.'`, so use `transpose(x)` in the rare cases where we have complex data but need a transpose rather than a Hermitian transpose.

The transpose or Hermitian transpose of a vector takes negligible time in a modern language like JULIA because the array elements stay in the same order in memory; one just needs to modify the variable type from `Vector` to an `Adjoint` type. See Demo 1.2.

In many computing languages, the transpose of a (large) matrix requires considerable shuffling of values in memory and should be avoided when possible to save compute time. JULIA uses a clever approach called a lazy transpose or lazy adjoint that avoids memory shuffling. Conceptually, to compute `A'y`, JULIA essentially uses the equivalent form `A'y = (y'A)'` that does not require a matrix transpose.

In MATLAB there are no 1D vectors, just arrays of size $N \times 1$ or $1 \times N$. The command `x = [1;2;3]` produces a 3×1 array. One can verify that `ndims(x)` is 2 and `size(x)` is the 1×2 array `[3 1]`. Applying the adjoint operation `y = x'` produces a 1×3 array. Taking another adjoint `z = y'` returns a 3×1 array. These are all 2D arrays and the absence of 1D arrays is an impediment to writing universal code that works for arrays of any dimension.

Julia has proper 1D vectors; typing `x = [1;2;3]` produces a `Vector` type for which `ndims(x)` is 1 and `size(x)` is the one-element tuple `(3,)`. Applying the adjoint operation `y = x'` produces a special kind of 1×3 array having an `Adjoint` type. Here, `ndims(y)` is 2 and `size(y)` is the tuple `(1,3)`. Taking another adjoint `z = y'` returns an ordinary `Vector` again, with `size(z)` being `(3,)` and `ndims(z)` being 1. In fact, one can verify that `x == z` is `true`, which means that the variable `z` points to the exact same data as the original vector `x`. For memory efficiency, JULIA avoids making duplicate copies of the vector elements when applying a `transpose` or `adjoint`.

Example 2.21 To compute $x'A'y$ it may be faster (for large data, in some languages) to use $(A * x)' * y$ instead of $x' * A' * y$ because the latter may require a transpose operation (unless the compiler is smart enough to avoid it, like in JULIA). For complex data, `conj(y' * A * x)` is another alternative.

2.4 Multiplication

The defining operations of a vector space are vector addition and scalar multiplication. More generally, linear algebra deals with addition and multiplication of vectors and matrices. Addition is trivial so we focus on multiplication.

2.4.1 Vector–Vector Multiplication

Definition The dot product or inner product of two vectors $x, y \in \mathbb{F}^N$ is

$$\langle x, y \rangle = y'x = [y_1^* \cdots y_N^*] \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \sum_{i=1}^N y_i^* x_i. \quad (\text{scalar}) \quad (2.5)$$

With this definition,¹ the inner product is linear in the first argument: $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle \quad \forall \alpha \in \mathbb{F}$.

Explore 2.2 Is the inner product $\langle x, y \rangle$ linear in the second argument?

¹ Some books like [4] use the less common definition $\langle x, y \rangle = x'y$.

The dot product is central to linear discriminant analysis (LDA), a topic in pattern recognition and machine learning for two-class classification. In signal processing terms, it is at the heart of the matched filter for signal detection. The dot product is central to convolution and to neural networks [21], for example, the perceptron [22]. Section 6.3 discusses many inner product properties.

Definition In contrast, the outer product of vector $x \in \mathbb{C}^M$ with a vector of possibly different length $y \in \mathbb{C}^N$ is the following $M \times N$ matrix:

$$\mathbf{xy}' = \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix}}_{M \times 1} \underbrace{\begin{bmatrix} y_1^* & \cdots & y_N^* \end{bmatrix}}_{1 \times N} = \underbrace{\begin{bmatrix} x_1 y_1^* & x_1 y_2^* & \cdots & x_1 y_N^* \\ \vdots & & & \vdots \\ x_M y_1^* & x_M y_2^* & \cdots & x_M y_N^* \end{bmatrix}}_{M \times N}. \quad (2.6)$$

Later we will see that this outer product is a rank-1 matrix (unless either x or y are 0). Outer products are central to matrix decompositions like the singular value decomposition discussed soon.

Another vector product that appears in physics is the cross product that is defined for two vectors in \mathbb{R}^3 . This product is not a focus of general linear algebra because it does not readily generalize to higher dimensions.

Definition The n th unit vector in \mathbb{F}^N is a vector of all zeros except that the n th element is 1.

Example 2.22 In \mathbb{R}^4 ,

$$e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

The inner product of the n th unit vector in \mathbb{F}^N with a vector x in \mathbb{F}^N returns the n th element of x : $\langle x, e_n \rangle = x_n$.

2.4.1.1 Practical Implementation

For the inner product, in JULIA, $y' * x$ and $*(y', x)$ and $y'x$ and $\text{dot}(y, x)$ and $\cdot(y, x)$ all perform $y'x = \langle x, y \rangle$. The form $y'x$ is remarkably similar to the math. Caution: $y' x$ (extra space) does not work. For the form with \cdot , first do using `LinearAlgebra`, then type `\cdot` and hit tab. Another (slower) way that matches the summation expression above is `sum(conj(y) .* x)`.

In JULIA, the result of an inner product $y' * x$ is a scalar, consistent with the mathematical definition. In contrast, in MATLAB there are no scalars and the result of $y' * x$ is a 1×1 array of dimension 2.

To define unit vectors in JULIA, use `e(n) = I(N)[:,n]`.

Explore 2.3 Determine the output of the following JULIA code.

```
</> 2.1 using LinearAlgebra: I, dot
x = [10, 20, 30, 40]
N = length(x)
e(n) = I(N)[:,n]
dot(x, e(2))
```

For the outer product, in JULIA `x * y'` is the clearest syntax, although `*(x, y')` also works. In fact, `a * b` is translated by the compiler to `*(a, b)`.

- ◻ Demo 2.2 provides more details about efficient dot product computation.
- ◻ Demo 2.3 explores methods for computing outer products.

2.4.2 Matrix–Vector Multiplication

Next we consider multiplication with a matrix. First we need some notation because matrix multiplication uses rows and/or columns of a matrix.

For $A \in \mathbb{F}^{M \times N}$ (see (2.7)):

- $A_{:,1} = Ae_1$ denotes the first column of A (a vector of length M), exactly like $A[:,1]$ in JULIA, where e_1 denotes the first unit vector in \mathbb{R}^N . Similarly, let \tilde{e}_1 denote the first unit vector in \mathbb{R}^M .
- $A_{1,:} = \tilde{e}_1'A$ denotes the first row of A (a row vector of length N). Caution: $A[1,:]$ in JULIA returns a *vector*, not a “ $1 \times N$ ” array like MATLAB.

Using this colon notation we have the following two ways to think about a matrix.

Column partition of an $M \times N$ matrix: $A = [A_{:,1} \ \cdots \ A_{:,N}]$,

Row partition of an $M \times N$ matrix: $A = \begin{bmatrix} A_{1,:} \\ \vdots \\ A_{M,:} \end{bmatrix}$.

Of course, we also have the elementwise way of writing out an $M \times N$ matrix:

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \cdots & a_{M,N} \end{bmatrix},$$

but for multiplication operations we often think about the column or row partitions above instead.

If $A \in \mathbb{F}^{M \times N}$ and $x \in \mathbb{F}^N$, then the matrix–vector product $y = Ax$ is a vector of length M defined by

$$y = Ax \iff \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} a_{1,1}x_1 + \cdots + a_{1,N}x_N \\ a_{2,1}x_1 + \cdots + a_{2,N}x_N \\ \vdots \\ a_{M,1}x_1 + \cdots + a_{M,N}x_N \end{bmatrix}, \text{ i.e., } y_i = \sum_{j=1}^N a_{ij}x_j. \quad (2.7)$$

A matrix–vector product requires MN floating-point multiplies and $M(N - 1)$ floating-point additions, or a total of $2MN - M$ floating-point operations (FLOPs).

Often when considering the computational costs of some method, our primary interest is how the computation increases as the problem size grows. So constant scaling factors like the “2” above are of secondary importance, and the “ $-M$ ” term is small compared to the product MN as the problem size grows. Thus, as a concise summary using big O notation, we say that matrix–vector multiplication requires $O(MN)$ FLOPs.

Why is matrix–vector multiplication important?

- If A is a data matrix, then Ax is a linear combination of its columns, and we often use such linear combinations for modeling and prediction.
- If A corresponds to a (linear) operation, then we can think of A as representing a linear system and the operation $x \mapsto Ax$ describes the output of the system when the input is x .

2.4.2.1 Practical Implementation

In JULIA (or MATLAB) we simply write $y = A*x$ to perform the matrix–vector multiplication $y = Ax$. The terrific similarity between this syntax and the mathematical expression is a key benefit of such high-level languages! In contrast, for low-level languages (like C and Fortran), one must either call a function in a numerical linear algebra library or write a double loop, like the following, that looks nothing like the math.

```
function mv_mm(A, x) # A * x
    (M,N) = size(A)
    y = similar(x, M) # preallocate!
    for m in 1:M
        inprod = 0 # accumulator
        for n in 1:N
            inprod += A[m,n] * x[n]
        end
        y[m] = inprod
    end
    return y
end
```



For efficiency reasons and for better readability, in JULIA, unlike in MATLAB, one must preallocate space for the result vector so that memory use does not grow with the iteration over m .

The matrix–vector product (2.7) has two mathematically (but not practically) equivalent vector forms:

$$\begin{array}{l} A \in \mathbb{F}^{M \times N} \\ x \in \mathbb{F}^N \end{array} \implies Ax = \underbrace{\begin{bmatrix} A_{1,:}x \\ \vdots \\ A_{M,:}x \end{bmatrix}}_{M \text{ "inner products"}} = \underbrace{\sum_{n=1}^N A_{:,n}x_n}_{\text{linear combination of columns}} = A_{:,1}x_1 + \cdots + A_{:,N}x_N. \quad (2.8)$$

In JULIA, instead of using $y = A * x$ we could instead implement matrix–vector multiplication using either of the following two loops, corresponding to the two vector forms in (2.8):

```
function mv_row(A, x)
    M = size(A,1)
    y = similar(x, M) # preallocate
    for m in 1:M
        y[m] = transpose(A[m,:]) * x
    end
    return y
end

function mv_col(A, x)
    (M,N) = size(A)
    y = zeros(eltype(x), M) # init 0
    for n in 1:N
        y += A[:,n] * x[n]
    end
    return y
end
```



Demo 2.4 compares the compute efficiency of these implementations and optimized variants.

2.4.2.2 Matrix–Vector Multiplication with Transpose

If A is an $M \times N$ matrix and $y \in \mathbb{F}^M$ is a vector, then the (Hermitian) transposed matrix–vector product is

$$A'y = \underbrace{\begin{bmatrix} (A_{:,1})'y \\ \vdots \\ (A_{:,N})'y \end{bmatrix}}_{N \text{ inner products}} = \underbrace{\sum_{m=1}^M (A_{m,:})'y_m}_{\text{linear combination}}. \quad (2.9)$$



In JULIA (and MATLAB, but not numpy), a standard 2D `Array`, in other words a `Matrix`, is stored in memory with consecutive elements of a column being adjacent, called column-major order. (JULIA has other types like `StridedArray` and `AbstractArray` with other storage orders.)

Example 2.23 If $A = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$ then $A[::]$ or $\text{vec}(A)$ both reveal the memory

storage order: $\begin{bmatrix} 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}$. Computing $y = Ax$ via inner products means

$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 5x_1 + 7x_2 \\ 6x_1 + 8x_2 \end{bmatrix}$, where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Here, the memory access order is

nonsequential: the top row uses 5 and 7, which are not adjacent in memory. For large arrays, nonsequential access can lead to slower execution time because of poor memory cache use.

In contrast, computing $x = A'y$ via inner products means

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5y_1 + 6y_2 \\ 7y_1 + 8y_2 \end{bmatrix}$. Here, the elements of A are accessed sequentially, improving cache use.

The following test compares matrix–vector multiplication versus transposed-matrix–vector multiplication. Both Ax and $A'y$ require $O(MN)$ FLOPs. The comments show the times for experiments on a 2017 iMac Pro, 4.2 GHz four-core CPU, macOS 12.6, for JULIA 1.8.2.

```
</> 2.2 using BenchmarkTools: @btime
N = 2^11; M = N+1; A = rand(M,N); x = rand(N); y = rand(M)
f1(y,A,x) = A * x
f2(y,A,x) = A'y
f3(y,A,x) = A' * y
f4(y,A,x) = transpose(A) * y
@btime f1($y,$A,$x); # A*x           975 us
@btime f2($y,$A,$x); # A'y           930 us
@btime f3($y,$A,$x); # A'*y          930 us
@btime f4($y,$A,$x); # transpose(A)*y 930 us
```

- A'^*y is a bit faster than $A*x$ for the reasons discussed above.
- Similarly, in MATLAB R2021a, the same test used 1035 μ s for $A*x$ and 960 μ s for A'^*y .
- For timing code, it is better to use `BenchmarkTools.@btime` than to use `@time`.
- For JULIA 0.7 and above, `transpose(A)` and `A'` and `adjoint(A)` do not reorder memory, but rather set an internal header flag so that it can perform A'^*y and `transpose(A)*y` efficiently.

Try: `A = ones(2,3); B = A'; B[4] = 5; display(A)`.

Q2.8 In light of these considerations, which of the following weighted inner product calculations $y'Ax$ is likely to run fastest?

A: $y'^*(A*x)$

B: $(y'^*A)*x$

C: Roughly the same time

Consider the following very similar computation of the weighted inner product $x' A' y$.

```
</> 2.3 using BenchmarkTools: @btime
N = 1000; M = N+1; A = rand(M,N); x = rand(N); y = rand(M);
f1(x,A,y) = x'* (A'*y)
f2(x,A,y) = (x'*A')*y
f3(x,A,y) = x'*A'*y
f4(x,A,y) = (x'*transpose(A))*y
@assert f1(x,A,y) ≈ f2(x,A,y) ≈ f3(x,A,y) ≈ f4(x,A,y) # check
@btime f1($x,$A,$y) # x'* (A'*y)           # 61 us
@btime f2($x,$A,$y) # (x'*A')*y          # 71 us
@btime f3($x,$A,$y) # x'*A'*y            # 71 us
@btime f4($x,$A,$y) # (x'*transpose(A))*y # 71 us
```

- Here, the compiler is *not* smart enough to determine the best execution order. (Apparently it parses code left to right.)
- Here, using appropriate parentheses can accelerate the code!
- A minute of thinking and a few seconds of typing (parentheses) can save compute time (and energy). Training large language models costs millions of dollars [23], so efficiency is paramount.

Now examine $y'Ax$.

```
</> 2.4 using BenchmarkTools: @btime
N = 1000; M = N+1; A = rand(M,N); x = rand(N); y = rand(M);
f1(y,A,x) = y'* (A*x)
f2(y,A,x) = (y'*A)*x
f3(y,A,x) = y'*A*x
f4(y,A,x) = transpose(y)*A*x
@assert f1(y,A,x) ≈ f2(y,A,x) ≈ f3(y,A,x) ≈ f4(y,A,x) # check
@btime f1($y,$A,$x) # y'* (A*x)           # 72 us
@btime f2($y,$A,$x) # (y'*A)*x          # 52 us
@btime f3($y,$A,$x) # y'*A*x            # 52 us
@btime f4($y,$A,$x) # transpose(y)*A*x # 52 us
```

- Here, the left-to-right parsing happens to produce the faster result, so parentheses were not needed.
- Vector transpose takes negligible time because no memory shuffling is needed.

2.4.3 Matrix–Matrix Multiplication

For $A \in \mathbb{F}^{M \times K}$ and $B \in \mathbb{F}^{K \times N}$, the result of the matrix–matrix product is $C = AB \in \mathbb{F}^{M \times N}$.

Definition The standard definition for matrix multiplication uses the rows of A and the columns of B for the inner loop:

$$C_{ij} = \sum_{k=1}^K A_{ik}B_{kj} \quad \text{for } i = 1, \dots, M, j = 1, \dots, N. \quad (2.10)$$

The “inner dimension” (K here) must match for valid matrix–matrix multiplication. Matrices satisfying such suitable dimensions are called conformable.

Matrix–matrix multiplication is important for representing the cascade of two linear systems (when A and B both represent operations), and matrix decomposition relies on matrix products.



Equation (2.10) includes matrix–vector multiplication as a special case, if we abuse terminology and treat a vector in \mathbb{F}^N as an $N \times 1$ matrix (like MATLAB does). In this book we will avoid that abuse and treat vectors separately from matrices (like JULIA does).

2.4.3.1 Computation

Using the formula (2.10), we must compute K scalar multiplications for each of MN elements of C , for a total of MKN multiplications and roughly that number of additions. In particular, if A and B are both $N \times N$ matrices, then the natural implementation of (2.10) requires $O(N^3)$ operations. Recent work [24] provides a “laser method” that reduces it to $O(N^{2.37286})$ operations.

2.4.4 Matrix Multiplication Properties



- The distributive property of matrix multiplication (if the sizes match): $A(B + C) = AB + AC$.
- The associative property (if the sizes match): $A(BC) = (AB)C$.
- There is no general commutative property! In general, $AB \neq BA$ even if the sizes match.
- Multiplication by an (appropriately sized) identity matrix has no effect: $IA = AI = A$. If A is $M \times N$, then we sometimes write $I_M A = A I_N = A$.
- Matrix concatenation: if $A \in \mathbb{F}^{M \times N}$, $B \in \mathbb{F}^{N \times K}$, and $C \in \mathbb{F}^{N \times L}$, then

$$A [B \quad C] = [AB \quad AC]. \quad (2.11)$$

Explore 2.4 Find the corresponding property for vertical matrix concatenation.

2.4.4.1 Practical Consideration: Parentheses Matter!

The associative property is a simple math equality, but choice of parentheses can greatly affect code speed!

Example 2.24 Suppose x, y, z are all vectors in \mathbb{R}^N . How many FLOPs are needed for the following very simple code: $x' * y' * z$? The answer depends on where the compiler puts the parentheses.

- Left to right evaluation $(x * y') * z$ requires $O(N^2)$ FLOPs.
- Right to left evaluation $x * (y' * z)$ requires $O(N)$ FLOPs.

Bottom line: Put in parentheses yourself to ensure efficiency!

- ◆◆ For long chains of products use the `MatrixChainMultiply.jl` package.

Explore 2.5 How many multiply operations are needed to multiply two 4×4 matrices?

2.4.4.2 Four Views of Matrix Multiplication

Besides (2.10), there are at least four (!) ways of viewing (and implementing) matrix-matrix products. Why would you need four different versions?

- Some versions are preferable for big data and distributed computing using parallel processing [26].
- Different versions are useful for mathematical manipulations.

2.4.4.3 Version 1 (Dot or Inner Product Form)

Using the row partition of A and the column partition of B , one can verify that

$$C = AB = \underbrace{\begin{bmatrix} A_{1,:} \\ \vdots \\ A_{M,:} \end{bmatrix}}_{M \times K} \underbrace{\begin{bmatrix} B_{:,1} & \cdots & B_{:,N} \end{bmatrix}}_{K \times N} = \underbrace{\begin{bmatrix} A_{1,:}B_{:,1} & \cdots & A_{1,:}B_{:,N} \\ \vdots & & \vdots \\ A_{M,:}B_{:,1} & \cdots & A_{M,:}B_{:,N} \end{bmatrix}}_{M \times N}.$$

Specifically, the elements of C are given by the following (conjugate-less) inner products:

$$C_{ij} = \underbrace{A_{i,:}}_{i \times K} \underbrace{B_{:,j}}_{\in \mathbb{F}^K}. \quad (2.12)$$

In JULIA code:

- Simple but opaque: $C = A * B$.
- Triple loop in terms of scalars per (2.10):

```
C = similar(A,M,N)
for m in 1:M, n in 1:N
    inprod = 0 # accumulator
    for k in 1:K
        inprod += A[m,k] * B[k,n]
    end
    C[m,n] = inprod
end
```

- Double loop of vector “inner products”:

```
C = similar(A,M,N)
for m in 1:M, n in 1:N # nested loop
    C[m,n] = transpose(A[m,:]) * B[:,n]
end
```

The transpose `transpose(A)` (not Hermitian transpose A') is crucial for complex matrices! Instead of `C[m,n] = transpose(A[m,:]) * B[:,n]` you could use `C[m,n] = dot(conj(A[m,:]), B[:,n])` or `C[m,n] = sum(A[m,:].*B[:,n])`.



Reminder: For a $K \times N$ matrix B , in JULIA `B[k,:]` returns a 1D vector of length N , not an “ $N \times 1$ ” array nor a “row vector” (a $1 \times N$ array, like in MATLAB). See Demo 1.2.

2.4.4.4 Version 2 (Columnwise Accumulation)

Using the column partition of A and the elements of B :

$$\begin{aligned} C = AB &= [A_{:,1} \cdots A_{:,K}] \begin{bmatrix} B_{1,1} & \cdots & B_{1,N} \\ \vdots & & \vdots \\ B_{K,1} & \cdots & B_{K,N} \end{bmatrix} \\ \implies C_{:,n} &= [A_{:,1} \cdots A_{:,K}] \begin{bmatrix} B_{1,n} \\ \vdots \\ B_{K,n} \end{bmatrix} = \sum_{k=1}^K A_{:,k} B_{k,n}. \end{aligned} \quad (2.13)$$

In JULIA code we would loop over the columns of A , performing vector–scalar multiplication:

```
</> 2.5 C = zeros(eltype(A),M,N) # must preallocate
      for n in 1:N, k in 1:K # nested loop
          C[:,n] += A[:,k] * B[k,n]
      end
```

The `+=` accumulation operation in this code is akin to the equivalent C language construct.

2.4.4.5 Version 3 (Matrix–Vector Products)

Using the column partition of B , one can verify this “horizontal concatenation” property:

$$\begin{aligned} C = AB &= \underbrace{A}_{M \times K} \underbrace{\begin{bmatrix} B_{:,1} & \cdots & B_{:,N} \end{bmatrix}}_{K \times N} = \underbrace{\begin{bmatrix} AB_{:,1} & \cdots & AB_{:,N} \end{bmatrix}}_{M \times N} \\
 \implies C_{:,j} &= AB_{:,j}. \end{aligned} \quad (2.14)$$

JULIA code for this using a single loop over columns of B of matrix–vector products:

```
</> 2.6 C = similar(A,M,N)
    for n in 1:N
        C[:,n] = A * B[:,n]
    end
```

As a practical tip, by default, using $C = zeros(M,N)$ would make a `Float64` array. That is fine for small problems and when very good numerical precision is needed. To save memory for large problems (at the price of reduced precision) use

`zeros(Float32, M, N)`.

If you are desperate for memory savings, consider `zeros(Float16, M, N)`. Here we use $C = similar(A,M,N)$ so that C matches the precision of A . Professional code would consider the element types of both arrays:

◆◆ `T = promote_type(eltype(A),eltype(B)); C = similar(T, M, N)`

This book often uses the matrix–vector operation view of (2.14).

2.4.4.6 Version 4 (Sum of Outer Products)

Using the column partition of A and the row partition of B :

$$C = AB = \underbrace{[A_{:,1} \cdots A_{:,K}]}_{M \times K} \underbrace{\begin{bmatrix} B_{1,:} \\ \vdots \\ B_{K,:} \end{bmatrix}}_{K \times N} = \sum_{k=1}^K \underbrace{A_{:,k}}_{\in \mathbb{R}^M} \underbrace{B_{k,:}}_{1 \times N}. \quad (2.15)$$

For proof, see [4, Theorem 1.3].

JULIA code using a single loop (over the inner dimension) of outer products:

```
</> 2.7 C = zeros(eltype(A),M,N)
    for k in 1:K
        C .+= A[:,k] * transpose(B[k,:]) # column times row!
    end
```

This code may be slow due to the irregular memory access order, but we will use the mathematical “sum of outer product” representation (2.15) extensively. See Problem 2.9 for two more versions.

2.4.4.7 Block Matrix Multiplication

The inner operation in (2.15) is an example of block matrix multiplication; the definition for such operations is basically the same as (2.10) as long as all the matrix dimensions involved match properly; see Problem 2.7.

2.4.5 Kronecker and Hadamard Products, and the vec Operator

The formula in (2.10) is the standard form of matrix multiplication. There are at least two other types of matrix multiplication that are important and have their own names.

The Kronecker product is usually denoted $A \otimes B$. If $A \in \mathbb{F}^{m \times n}$ and $B \in \mathbb{F}^{k \times l}$, then

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ \vdots & \ddots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{F}^{(mk) \times (nl)}. \quad (2.16)$$

In JULIA it is `kron(A, B)`. The following properties of the Kronecker product follow directly from its definition:

$$\begin{aligned} \alpha(X \otimes Y) &= (\alpha X) \otimes Y = X \otimes (\alpha Y), & \alpha \in \mathbb{F}, \\ (X \otimes Y) \otimes Z &= X \otimes (Y \otimes Z), \\ (X \otimes Y)' &= X' \otimes Y', \\ (X \otimes Y)^{-1} &= X^{-1} \otimes Y^{-1} & \text{if invertible,} \\ (X \otimes Y) + (X \otimes Z) &= X \otimes (Y + Z) & \text{if sizes match,} \\ (X \otimes Y)(W \otimes Z) &= (XW) \otimes (YZ) & \text{if sizes match.} \end{aligned} \quad (2.17)$$

The Hadamard product is the elementwise multiplication of two *same-sized* matrices. It is usually denoted $A \odot B$. In JULIA it is `A .* B`.

♦♦ Other matrix products include the Tracy–Singh and Khatri–Rao products.

2.4.5.1 The vec Operator and Matrix Multiplication

The vec operator converts a matrix to a vector. It is often written as

$$\text{vec}(A) = \underbrace{\text{vec}([A_{:,1} \cdots A_{:,N}])}_{M \times N} = \underbrace{\begin{bmatrix} A_{:,1} \\ \vdots \\ A_{:,N} \end{bmatrix}}_{\text{vector in } \mathbb{F}^{MN}}. \quad (2.18)$$

In JULIA there are at least three ways to perform this operation, all of which stack the columns:

- `vec(A)` (this is the most memory efficient because it uses a JULIA `view`).
- `A[:, :]`.
- `reshape(A, length(A))`. Note that `reshape(A, length(A), 1)` is *not* the same thing because the result in this case is a 2D array of size $MN \times 1$, which is a different variable type than a 1D vector of length MN .

The following vec trick property [27, 28] is particularly important (Problem 2.5):

$$\text{vec}(AXB) = (B^T \otimes A) \text{vec}(X). \quad (2.19)$$

 In this formula B^T must be a regular transpose, not a Hermitian transpose, even if B is complex!

2.4.5.2 Units for Matrix–Vector Multiplication

As discussed earlier in the chapter, a matrix can simply be an array of data, or it can represent a linear operation. In many applications, data has units. If we think of a data matrix simply as a 2D array, then the units of each element can be arbitrary. But if we want to perform the elementary linear algebra operation of matrix–vector multiplication, then to have consistent units we must be more careful.

Fact 2.3 For matrix–vector multiplication to be well defined for matrices with elements having units, the units of the matrix elements must be row–column separable, that is, for an $M \times N$ matrix A , there must exist units $\alpha_1, \dots, \alpha_M$ and β_1, \dots, β_N such that the units of a_{ij} are the product $\alpha_i\beta_j$. Put another way, the units of each row of the matrix must be same, up to column-dependent scale factors that themselves can have units. In other words, matrix–vector multiplication with a matrix A is possible iff we can write A in the form

$$A = \text{Diag}(\alpha_i) \tilde{A} \text{Diag}(\beta_j), \quad (2.20)$$

where the $M \times N$ matrix \tilde{A} is unitless.

Example 2.25 This matrix is suitable for matrix–vector multiplication:

$$A = \begin{bmatrix} 1\text{m} & 2\text{s} \\ 3\text{g m/s} & 4\text{g} \end{bmatrix} = \begin{bmatrix} 1\text{s} & 0 \\ 0 & 1\text{g} \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1\text{m/s} & 0 \\ 0 & 1 \end{bmatrix}. \quad (2.21)$$

Example 2.26 JULIA’s `Unitful.jl` package is helpful for data with units.

```
</> 2.8 using Unitful: m, s, g, rad # some useful units
A = [1m 2s; 3g*m/s 4g] # 2nd row units are g/s × 1st row units
x = [5s, 6m] * 1rad # units compatible for multiplying by A
A * x
y = [5g, 6s] * 1rad # units compatible for multiplying by A'
A' * y
A * y # fails appropriately due to unit dimension incompatibility
```

Explore 2.6 Does the matrix A in the preceding example belong to a vector space?

2.4.6 Using Matrix–Vector Operations

Particularly in high-level, array-oriented languages like JULIA, matrix and vector operations abound. One should be on the lookout for opportunities to “parallelize” (vectorize) using such operations.

Example 2.27 Suppose we want to visualize the 2D Gaussian bump function $f(x, y) = e^{-(x^2+30y^2)}$. An elementary implementation with a double loop follows.

```
</> 2.9 using Plots
M, N = 101, 103
x = range(-2, 2, M)
y = range(-1.1, 1.1, N)
F = zeros(M,N)
for m in 1:M, n in 1:N # nested loop
    F[m,n] = exp(-(x[m]^2 + 30 * y[n]^2))
end
heatmap(x, y, F', color=:grays, aspect_ratio=:equal)
```



How do we leverage matrix–vector concepts to write this more concisely? (Concise code is often easier to read and maintain, and often looks more like the mathematical expressions.) We define a matrix A such that $A_{ij} = x_i^2 + 30y_j^2$, and then use an elementwise exponential operation: $F = \exp.(-A)$. In JULIA, `exp.(A)` applies exponentiation elementwise to an array. This is called a broadcast operation. In JULIA, `exp(A)` is meaningless for a matrix; use `expm(A)` for a matrix exponential.

How do we define A where $A_{ij} = x_i^2 + 30y_j^2$ without writing a double loop? One way is to use outer products. If $u \in \mathbb{R}^M$ has elements $u_i = x_i^2$ and $v \in \mathbb{R}^N$ has elements $v_j = y_j^2$, then the following sum of outer products yields an $M \times N$ matrix:

$$\begin{aligned} A &= u\mathbf{1}'_N + 30\mathbf{1}_M v' = \underbrace{\begin{bmatrix} x_1^2 \\ \vdots \\ x_M^2 \end{bmatrix}}_{\text{repeated column}} \underbrace{\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}}_{1 \times N} + 30 \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{\text{length } M} \begin{bmatrix} y_1^2 & \cdots & y_N^2 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} x_1^2 & \cdots & x_1^2 \\ \vdots & \ddots & \vdots \\ x_M^2 & \cdots & x_M^2 \end{bmatrix}}_{\text{repeated column}} + 30 \underbrace{\begin{bmatrix} y_1^2 & \cdots & y_N^2 \\ \vdots & \ddots & \vdots \\ y_1^2 & \cdots & y_N^2 \end{bmatrix}}_{\text{repeated row}}, \end{aligned} \quad (2.22)$$

where $\mathbf{1}_N$ denotes the vector of all ones, that is, `ones(N)` in JULIA or `ones(N, 1)` in MATLAB.

The key line of the following JULIA code looks reasonably close to the above outer-product form.

```
</> 2.10 using Plots
M, N = 101, 103
x = range(-2, 2, M)
y = range(-1.1, 1.1, N)
A = (x.^2) * ones(N)' + 30 * ones(M) * (y.^2)'
F = exp.(-A)
heatmap(x, y, F', color=:grays, aspect_ratio=:equal)
```

This version spares the developer from having to write a double loop. Of course, JULIA itself must do loops internally when evaluating `exp.(-A)` and similar expressions.

The outer-product approach has the benefit of avoiding writing double loops. However, this type of operation arises so frequently that JULIA provides functions that avoid the unnecessary operations of multiplying by the `ones` vector.

In JULIA there is automatic broadcast of singleton dimensions by the scalar addition operator: `A = x.^2 .+ 30 * (y.^2)'`. There are two somewhat distinct uses of `.` here:

- `.^2` : elementwise power;
- `.+` : normally elementwise addition of arrays, but here with automatic broadcast like in (2.22).

An even better approach is to use JULIA's `@.` broadcast macro. This shorthand for `@__dot__` fuses all the elementwise operations more efficiently; it is used like so: `A = @. x.^2 + 30 * (y.^2)'`. This broadcast feature leads to the following concise code that looks a lot like the math.

```
</> 2.11 using Plots
x = range(-2, 2, 101)
y = range(-1.1, 1.1, 103)
A = @. x.^2 + 30 * (y.^2)' # a lot happens here!
F = exp.(-A)
heatmap(x, y, F', color=:grays, aspect_ratio=:equal)
```

Finally, if your goal is merely to make a picture of the function $f(x,y)$, without illustrating any matrix properties, the following is a “JULIA way” to do it using its multiple dispatch feature. This version is the shortest of all so it also includes a few labeling commands; you can see the results in Fig. 2.6.

```
</> 2.12 using Plots, LaTeXStrings
x = range(-2, 2, 101)
y = range(-1.1, 1.1, 103)
f = (x,y) -> exp.(-(x.^2 + 30y.^2)) # look, no "*" !
p4 = heatmap(x, y, f,
    title=L"f(x,y)", color=:grays, aspect_ratio=:equal,
    xlabel=(L"x", (-2,2), -2:2), ylabel=(L"y", (-1.1,1.1), -1:1))
```

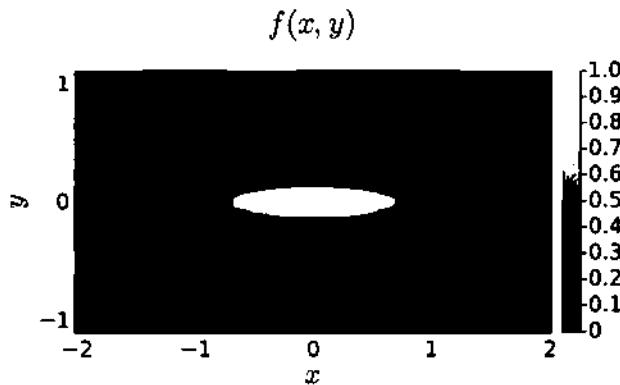


Figure 2.6 Image output from `heatmap`.

Example 2.28 One can perform 1D numerical integration using a vector operation (a dot product). To compute the area under a curve:

$$\text{Area} = \int_a^b f(x) dx \approx \sum_{m=1}^M \underbrace{(x_m - x_{m-1})}_{\text{base}} \underbrace{f(x_m)}_{\text{height}} = w'f,$$

$$w = \begin{bmatrix} x_1 - x_0 \\ \vdots \\ x_M - x_{M-1} \end{bmatrix}, \quad f = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_M) \end{bmatrix}, \quad (2.23)$$

where the (possibly nonuniformly spaced) sampled points satisfy $a = x_0 < x_1 < \dots < x_M = b$. This summation is a dot product between two vectors in \mathbb{R}^M and is straightforward in JULIA.

```
</> 2.13 f(x) = x^2 # parabola
      x = range(0,3,2000) # sample points
      w = diff(x) # "widths" of rectangles
      Area = w' * f.(x[2:end])
```

Explore 2.7 What should the exact value be for the area in this example?

Explore 2.8 Why `x[2:end]` instead of `x[1:end]` or simply `x`?

Demo 2.5 illustrates (2.23) and generates Fig. 2.6.

Example 2.29 Consider the problem of computing numerically the volume under a 2D function $f(x, y)$:

$$V = \int_a^b \int_c^d f(x, y) dy dx.$$

We can also use matrix–vector products for this operation. First, approximate the integral:

$$V = \int_a^b \int_c^d f(x, y) dy dx \approx S \triangleq \sum_{m=1}^M \sum_{n=1}^N (x_m - x_{m-1})(y_n - y_{n-1})f(x_m, y_n),$$

where $c = y_0 < y_1 < \dots < y_N = d$.

Define a vector $w \in \mathbb{R}^M$ as in the 1D example above, along with $u \in \mathbb{R}^N$ and $F \in \mathbb{R}^{M \times N}$:

$$w = \begin{bmatrix} x_1 - x_0 \\ \vdots \\ x_M - x_{M-1} \end{bmatrix}, \quad u = \begin{bmatrix} y_1 - y_0 \\ \vdots \\ y_N - y_{N-1} \end{bmatrix}, \quad F = \begin{bmatrix} f(x_1, y_1) & \dots & f(x_1, y_N) \\ \vdots & \ddots & \vdots \\ f(x_M, y_1) & \dots & f(x_M, y_N) \end{bmatrix}.$$

Then the above double sum is the following product in matrix–vector form (Problem 2.2): $V \approx S = w' Fu$.

Proof.

$$\begin{aligned} w' Fu &= [w_1 \ \dots \ w_M] \begin{bmatrix} f(x_1, y_1) & \dots & f(x_1, y_N) \\ \vdots & \ddots & \vdots \\ f(x_M, y_1) & \dots & f(x_M, y_N) \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix} \\ &= [w_1 \ \dots \ w_M] \begin{bmatrix} \sum_{n=1}^N u_n f(x_1, y_n) \\ \vdots \\ \sum_{n=1}^N u_n f(x_M, y_n) \end{bmatrix} = \sum_{m=1}^M w_m \sum_{n=1}^N u_n f(x_m, y_n) = S. \quad \square \end{aligned}$$

Again, this computation is easy to code in a high-level language like JULIA.

```
</> 2.14 f(x,y) = exp(-(x^2 + 3*y^2)) # gaussian bump function
      x = range(0,3,2000) # sample points
      y = range(0,2,1000) # sample points
      w = diff(x) # "widths" of rectangles in x
      u = diff(y) # "widths" of rectangles in y
      F = f.(x[2:end], y[2:end]') # automatic broadcasting again!
      S = w' * F * u # summation approximation to volume
```

(It would be fine to write it as a double loop, albeit with more typing and possibly slower in some languages.) In this case it is possible to determine the analytical value, but that might take more time than writing and running this code.

2.4.7 Invertibility

For a nonzero scalar x , we have $x(1/x) = 1$. Matrix inversion generalizes this identity.

Definition

$B \in \mathbb{F}^{N \times M}$ is a left inverse of $A \in \mathbb{F}^{M \times N}$ iff $BA = I_{N \times N}$.

$C \in \mathbb{F}^{N \times M}$ is a right inverse of $A \in \mathbb{F}^{M \times N}$ iff $AC = I_{M \times M}$.

A square matrix A is called invertible iff there exists a matrix X of the same size such that $AX = XA = I$. Another name for invertibility is nonsingular.

Fact 2.4 For nonsquare matrices, at most only one of a left or right inverse can exist, and is not unique.

Fact 2.5 For square matrices, a left inverse exists iff a right inverse exists and the two are identical, due to the invertible matrix theorem.

2.4.7.1 Basic Matrix Inversion Properties

If A is an invertible matrix, then:

- $(A^{-1})^{-1} = A$.
- $(A^k)^{-1} = (A^{-1})^k \triangleq A^{-k}$, $\forall k \in \mathbb{N}$ where $A^k = \underbrace{AA \cdots A}_{k \text{ times}}$ (Problem 2.10).
- $c \neq 0 \implies (cA)^{-1} = (1/c)A^{-1}$.
- A' is also invertible and $(A')^{-1} = (A^{-1})'$; thus the inverse of a symmetric matrix is symmetric.
- If B is also invertible and has the same size as A , then their product is invertible:



$$(AB)^{-1} = B^{-1}A^{-1}. \quad (2.24)$$

Proof. If $X = AB$ then $B^{-1}A^{-1}X = B^{-1}A^{-1}AB = I$ so $X^{-1} = B^{-1}A^{-1}$. \square

- The equation $Ax = \mathbf{0}$ has only the trivial solution $x = \mathbf{0}$ iff A is invertible.

2.4.7.2 Trying To Avoid Matrix Inversion

Typical methods for inverting an $N \times N$ matrix use $O(N^3)$ operations, which is very expensive for large matrices, so we often seek matrix identities to reduce computation when possible.

The following inverse of 2×2 block matrices holds if A and B are invertible (and if the sizes of D and C are appropriate):

$$\begin{bmatrix} A & D \\ C & B \end{bmatrix}^{-1} = \begin{bmatrix} (A - DB^{-1}C)^{-1} & -A^{-1}DB^{-1} \\ -\Delta^{-1}CA^{-1} & \Delta^{-1} \end{bmatrix} \quad (2.25)$$

if $\Delta \triangleq B - CA^{-1}D$, which denotes the Schur complement of A for this matrix, is invertible.

The inverse of a matrix product is relatively easy, as seen in (2.24). In contrast, the inverse of a matrix sum is nontrivial. Using the associative property and some rearranging yields the Sherman–Morrison–Woodbury identity, also known as the matrix inversion lemma:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}, \quad (2.26)$$

provided the matrices have compatible sizes and A and C are invertible.

A special case that is useful if we have previously computed the inverse of A and now need the inverse of the rank-1 update $A + xy'$, if it is invertible, is the Sherman–Morrison formula [29, 30, 31]:

$$(A + xy')^{-1} = A^{-1} - \frac{1}{1 + y'A^{-1}x}A^{-1}xy'A^{-1}. \quad (2.27)$$

A more general special case of (2.26) is when U and V are $N \times K$ matrices, leading to the following rank- K update formula:

$$(A + UV')^{-1} = A^{-1} - A^{-1}U(I_K + V'A^{-1}U)^{-1}V'A^{-1}, \quad (2.28)$$

provided $I_K + V'A^{-1}U$ is nonsingular.

Multiplying (2.26) on the right by B and simplifying yields the following useful related equality, sometimes called the push-through identity:

$$(A + BCD)^{-1}B = A^{-1}B(DA^{-1}B + C^{-1})^{-1}C^{-1}. \quad (2.29)$$

In particular, if $C = I_N$ and $A = aI_M$ then

$$(aI_M + BD)^{-1}B = B(aI_N + DB)^{-1}, \quad (2.30)$$

which is especially useful if one of N or M is much smaller than the other.

2.4.7.3 More Invertible Matrix Properties

The following properties of any invertible matrix A relate to topics discussed later in the book:

- A has linearly independent columns;
- A has full rank, that is, all of its eigenvalues are nonzero (see Section 4.3);
- if A is unitary then $A^{-1} = A'$;
- the determinant of A is nonzero and $\det(A^{-1}) = 1/\det(A)$.

2.4.7.4 Practical Implementation

To compute the inverse of a square matrix A in JULIA, use `inv(A)`. It is somewhat rare to need this operation in DS–ML–SP problems. More often, one needs to compute $A^{-1}b$, the product of the inverse of a matrix times a vector. The efficient way to do this in JULIA is to use the backslash function `A \ b`, or equivalently `\(A, b)`, also known as left division. Typically JULIA uses a lower–upper (LU) decomposition for this operation, which is a matrix representation of Gaussian elimination. In fact, `inv(A)`

is implemented in JULIA using left division equivalent to $\backslash(A, I)$. See Section 5.3.2.1 for more details.

Q2.9 If B is an $N \times N$ matrix, $XB = I_N$, and $BY = I_N$, then $X = Y$.

For consideration of units, see Problem 2.14.

2.5 Orthogonality and the Euclidean Norm

For ordinary scalars, $0 \cdot x = 0$. And if $xy = 0$, then at least one of x, y must be zero. For vectors and matrices, there are more interesting ways to multiply and end up with zero!

2.5.1 Orthogonal Vectors

Definition Two vectors x and y (of same length) are orthogonal (or perpendicular) iff their inner product is zero: $\langle x, y \rangle = y'x = 0$. In such cases we write $x \perp y$. If, in addition, $x'x = y'y = 1$ (that is, both have unit norm), then we call them orthonormal vectors.

Definition A collection of vectors that are pairwise orthogonal is called an orthogonal set. A collection of unit-norm vectors that are pairwise orthogonal is called an orthonormal set.

Example 2.30 In \mathbb{R}^3 , the vectors $v_1 = (4, 2, 0)$ and $v_2 = (-1, 2, 0)$ are orthogonal (but not orthonormal).

2.5.2 Euclidean Norm

Definition The (Euclidean) norm of a vector $x \in \mathbb{F}^N$, also known as the 2-norm of x , is defined by

$$\|x\|_2 = \sqrt{\langle x, x \rangle} = \sqrt{x'x} = \sqrt{\sum_{n=1}^N |x_n|^2}. \quad (2.31)$$

Often we simply write $\|x\|$ as a shorthand for $\|x\|_2$, because the Euclidean norm is particularly prevalent.

2.5.2.1 Properties of the Euclidean Norm

- $$\begin{aligned} \text{Homogeneity: } & \|ax\| = |\alpha| \|x\| \text{ for } \alpha \in \mathbb{F}. \\ \text{Triangle inequality: } & \|u + v\| \leq \|u\| + \|v\|. \\ \text{Quadratic expansion: } & \|u + v\|_2^2 = \|u\|_2^2 + 2 \operatorname{real}\{u'v\} + \|v\|_2^2. \end{aligned} \quad (2.32)$$

Proof.

$$\begin{aligned}\|u + v\|_2^2 &= \sum_{i=1}^n |u_i + v_i|^2 = \sum_{i=1}^n (u_i + v_i)(u_i + v_i)^* \\ &= \sum_{i=1}^n (|u_i|^2 + 2 \operatorname{real}\{u_i^* v_i\} + |v_i|^2).\end{aligned}$$

□

Pythagorean theorem: $u \perp v \iff \|u + v\|_2^2 = \|u\|_2^2 + \|v\|_2^2$.

Stacking property: $u \in \mathbb{F}^M, v \in \mathbb{F}^N \implies \left\| \begin{bmatrix} u \\ v \end{bmatrix} \right\|^2 = \|u\|_2^2 + \|v\|_2^2 = \sum_{i=1}^M |u_i|^2 + \sum_{j=1}^N |v_j|^2. \quad (2.33)$

2.5.3 Cauchy–Schwarz Inequality

Fact 2.6 The Cauchy–Schwarz inequality (or Schwarz or Cauchy–Bunyakovsky–Schwarz inequality) relates inner products and Euclidean norms as follows:

$$|\langle x, y \rangle| \leq \|x\|_2 \|y\|_2 = \sqrt{\langle x, x \rangle} \sqrt{\langle y, y \rangle} \quad \forall x, y \in \mathcal{V} \subset \mathbb{F}^N, \quad (2.34)$$

where $\|\cdot\|_2$ denotes the Euclidean norm on \mathbb{F}^N .

For a proof, see Section 6.3.

2.5.3.1 Angle Between Vectors

Definition The angle θ between two nonzero vectors $x, y \in \mathcal{V}$ is defined by

$$\cos \theta = \frac{|\langle x, y \rangle|}{\|x\|_2 \|y\|_2} \implies \theta \in [0, \pi/2]. \quad (2.35)$$

The Cauchy–Schwarz inequality is equivalent to the statement $|\cos \theta| \leq 1$.

Q2.10 What is the angle between two orthogonal vectors?

- A: 0 B: 1 C: $\pi/2$ D: π E: None of these

♦♦ A less familiar formula that is more accurate numerically for $x, y \in \mathbb{R}^N$ is [32]:

$$|\theta| = 2 \arctan \left(\frac{\|x/\|x\|_2 - y/\|y\|_2\|_2}{\|x/\|x\|_2 + y/\|y\|_2\|_2} \right) \in [0, \pi/2],$$

as implemented in the JULIA package `AngleBetweenVectors.jl`.

2.5.4 Orthogonal Matrices

Definition We say a (square) matrix $Q \in \mathbb{R}^{N \times N}$ is an orthogonal matrix iff $Q^\top Q = QQ^\top = I$.

(Arguably the term “orthonormal matrix” seems more appropriate, but sadly is used less often.)

Definition We say a (usually complex) square matrix $Q \in \mathbb{C}^{N \times N}$ is a unitary matrix iff $Q'Q = QQ' = I$.

-  The set of columns of a unitary matrix (or an orthogonal matrix) is an orthonormal set. (So is the set of rows.)

Unitary and orthogonal matrices are key building blocks for matrix decompositions. A generalization discussed in Section 5.8 is a tight frame, where only one of the two conditions holds [33, 34].

Q2.11 If a (possibly complex) matrix A has orthonormal columns, then it is unitary.

A: True **B: False**

Q2.12 If a square, real matrix A has orthogonal columns, then it is an orthogonal matrix.

- ◆ Every orthogonal matrix can be decomposed into a product of Givens rotation matrices and Householder reflection matrices. So intuitively, an $N \times N$ orthogonal matrix generalizes rotation and reflection operations.

2.5.4.1 Invertibility of Unitary Matrices

Fact 2.7 A key property of orthogonal and unitary matrices is that their inverse is simply their (Hermitian) transpose:

$$Q^{-1} = Q'. \quad (2.36)$$

Proof. Recall that Q is unitary iff

$$Q'Q = QQ' = I, \quad (2.37)$$

Thus (2.36) follows from the definition of matrix inverse.

Thus, the easiest (nondiagonal) matrices to invert are unitary matrices. Because of the equivalence of the left and right inverses for invertible square matrices, we really only need one of the two conditions in (2.37) to conclude the other and (2.36). So, people often just write $Q^*Q = I$ when mentioning that a (square) matrix Q is unitary.

2.5.4.2 Norm Invariance to Rotations

Fact 2.8 Orthogonal/unitary matrices act somewhat like rotation operations. An important property of $N \times N$ orthogonal or unitary matrices is that they do not change the Euclidean norm of a vector:

$$Q \text{ orthogonal or unitary} \implies \|Qx\|_2 = \|x\|_2 \quad \forall x \in \mathbb{C}^N. \quad (2.38)$$

Proof. $\|Qx\|_2 = \sqrt{(Qx)'(Qx)} = \sqrt{x'Q'Qx} = \sqrt{x'Ix} = \sqrt{x'x} = \|x\|_2$, using the orthogonality of Q . \square

This fact is related to Parseval's identity, and is also called unitary invariance of the Euclidean norm.

Definition A real $N \times K$ matrix Q with orthonormal columns is called a semiorthogonal matrix. A matrix $Q \in \mathbb{C}^{N \times K}$ with orthonormal columns is called a semiunitary matrix. The columns of Q are called an orthonormal K -frame.

Results from Section 4.3 will show that we must have $1 \leq K \leq N$, that is, Q must be tall (or square) for it to have orthogonal columns.

Q2.13 If U is an $M \times N$ semiunitary matrix, then

$$\|Ux\|_2 = \|x\|_2 \quad \forall x \in \mathbb{C}^N. \quad (2.39)$$

A: True

B: False

Q2.14 If u_1, \dots, u_K is a set of orthogonal vectors for which $\langle u_k, u_l \rangle = 0 \forall l \neq k$, then $U = [u_1 \ \cdots \ u_K]$ is a semiunitary matrix.

A: True

B: False

Q2.15 If U is an $M \times N$ semiunitary matrix, then $U'U = I_N$ and $UU' = I_M$.

A: True

B: False

Fact 2.9 The Euclidean inner product is unitarily invariant. If U is an $N \times N$ unitary matrix, then $\langle Ux, Uy \rangle = \langle x, y \rangle \ \forall x, y \in \mathbb{F}^N$.

Explore 2.9 Generalize that fact to the case where U is semiunitary or provide a counterexample.

2.6 Determinant of a Matrix

Now we begin discussing important matrix properties. The matrix determinant is a property that is defined only for square matrices. Data matrices are usually nonsquare, with graph data being a notable exception (see Section 8.6). So we usually do not examine the determinant of a data matrix directly! But if X is an $M \times N$ data matrix, we will often work with the $N \times N$ Gram matrix $X'X$ (e.g., when solving least-squares problems) and a Gram matrix is always square. Many operator matrices (like DFT) are also square.

There are many ways to introduce the determinant of a matrix because it has many properties. Here we consider the following four “axioms” expressed in terms of matrices.

Definition If $A \in \mathbb{F}^{N \times N}$ then the determinant of A is defined so that:

D1: If A is upper triangular and $N \times N$ then $\det(A) = a_{11} \cdot a_{22} \cdots \cdots a_{NN}$.

- D2: If $A, B \in \mathbb{F}^{N \times N}$ then $\det(AB) = \det(A)\det(B)$.
 D3: $\det(A') = \det(A)^*$, where z^* denotes the complex conjugate of z .
 D4: If $P_{i,j} \in \mathbb{R}^{N \times N}$ denotes the matrix that swaps the i th and j th rows, $i \neq j$, then $\det(P_{i,j}) = -1$.

Geometrically, the determinant of a matrix is the (signed) volume of the parallelepiped defined by its column vectors. The determinant of a Jacobian matrix arises in multivariate calculus for integration by substitution.

2.6.1 Determinant Properties

Fact 2.10 A matrix A is invertible iff its determinant is nonzero.

So a linear system of N equations with N unknowns, for example, $Ax = b$, has a unique solution iff the determinant corresponding to the coefficients is nonzero. This is the historical reason for the term determinant because it “determines” the uniqueness of the solution to such a set of equations.

Fact 2.11 From D2, the following determinant commutative property follows immediately:

$$A, B \in \mathbb{F}^{N \times N} \implies \det(AB) = \det(BA). \quad (2.40)$$

Explore 2.10 Can (2.40) be generalized to the (rectangular) case where $A \in \mathbb{F}^{M \times N}$ and $B \in \mathbb{F}^{N \times M}$?

Fact 2.12 $\det(A^{-1}) = 1/\det(A)$.

Explore 2.11 Use two of the above “axioms” to prove the preceding property.

A concise formula for the row-swapping matrix $P_{i,j}$ (a special permutation matrix) used in D4 is

$$P_{i,j} = I - e_j e'_j - e_i e'_i + e_j e'_i + e_i e'_j \quad (2.41)$$

for $i, j \in \{1, \dots, N\}$, where e_i denotes the i th unit vector.

Example 2.31 For $N = 5$:

$$P_{1,4} = P_{4,1} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ so } P_{1,4} \begin{bmatrix} A_{1,:} \\ A_{2,:} \\ A_{3,:} \\ A_{4,:} \\ A_{5,:} \end{bmatrix} = \begin{bmatrix} A_{4,:} \\ A_{2,:} \\ A_{3,:} \\ A_{1,:} \\ A_{5,:} \end{bmatrix}.$$

In words, left multiplying a matrix by $P_{i,j}$ swaps the i th and j th rows of the matrix.

A consequence of D2 and D4 is that swapping any two rows of a matrix negates the sign of the determinant: $i \neq j \implies \det(P_{i,j}A) = -\det(A)$. By D3, the same property holds for swapping two columns.

We can use the above four axioms to prove that multiplying a column (or row) of A by a scalar gives a new matrix whose determinant scales by that scalar factor:

$$B = [a_1 \cdots a_{n-1} \ b a_n \ a_{n+1} \cdots a_N] \implies \det(B) = b \det(A).$$

Proof. Simply write $B = AD$, where $D = \text{Diag}\{1, \dots, 1, b, 1, \dots, 1\}$, so $\det(B) = \det(A) \det(D) = b \det(A)$. \square

Note the strategy of writing (linear) operations in terms of matrix products so we can apply D2.

The transpose property is

$$\det(A^\top) = \det(A). \quad (2.42)$$

The scaling property is

$$\det(cA) = c^N \det(A) \quad \forall A \in \mathbb{F}^{N \times N}, c \in \mathbb{F}. \quad (2.43)$$

Multiplying a row (or column) of a matrix by a scalar and adding it to a different row (or column) does not change the determinant. So, for $k \neq m$:

$$\text{if } A = \begin{bmatrix} A_{1,:} \\ \vdots \\ A_{M,:} \end{bmatrix} \text{ and } B = \begin{bmatrix} A_{1,:} \\ \vdots \\ A_{m-1,:} \\ bA_{k,:} + A_{m,:} \\ A_{m+1,:} \\ \vdots \\ A_{M,:} \end{bmatrix}, \text{ then } \det(B) = \det(A).$$

Proof. $B = A + b e_m e_k' A = (I + b e_m e_k') A \implies \det(B) = \det(I + b e_m e_k') \det(A) = \det(A)$. Note that $e_k' A = A_{k,:}$, and think about the outer product $e_m(e_k' A) = e_m A_{k,:}$. \square

Explore 2.12 Where did this proof use the assumption that $k \neq m$?

Explore 2.13 What is $\det(B)$ if $k = m$?

Q2.16 An $M \times N$ matrix X has orthonormal columns; the determinant of the Gram matrix $X'X$ is:

- A: 0 B: 1 C: N D: M E: None of these

2.6.2

Matrices with Units

The determinant of A indicates whether the system of equations $Ax = b$ has a unique solution, per Fact 2.10. Thus, for a matrix with units, the determinant is relevant only

if it satisfies property (2.20) in Section 2.4.5.2. The determinant of such a matrix is:

$$\det(A) = \left(\prod_{i=1}^M \alpha_i \right) \left(\prod_{j=1}^N \beta_j \right) \det(\check{A}). \quad (2.44)$$

Example 2.32 For the matrix A in (2.21), $\det(A) = -2 \text{ g m}$.

2.6.3 Laplace's Determinant Formula

A general rule, called Laplace expansion, for an $N \times N$ matrix A is:

$$\det(A) = \sum_{n=1}^N (-1)^{m+n} a_{m,n} \det\{A_{m,n}\} \quad (2.45)$$

for any $m \in \{1, \dots, N\}$, where here $A_{m,n}$ denotes the submatrix of A formed by deleting the m th row and n th column; $\det\{A_{m,n}\}$ is called a minor of A .

2.6.4 Avoiding Computation

Naive methods for computing the determinant of an $N \times N$ matrix would require $O(N!)$ operations, but more efficient decomposition methods require $O(N^3)$ operations (or less). This is still expensive for large N , so we often seek properties to use to reduce computation when possible.

Generalizing property D2 in Section 2.6, if A is block upper triangular (or block lower triangular), with square blocks, then its determinant is the product of the determinants of its diagonal blocks:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots \\ 0 & A_{22} & A_{23} & \cdots \\ \vdots & \ddots & & \cdots \\ 0 & & 0 & A_{KK} \end{bmatrix} \implies \det(A) = \prod_{k=1}^K \det(A_{kk}). \quad (2.46)$$

Using block matrix triangularization and (2.46), if A is invertible then we can find the determinant of a large block matrix in terms of determinants of combinations of its blocks [35]:

$$\begin{aligned} \begin{bmatrix} A & B \\ C & D \end{bmatrix} &= \begin{bmatrix} A & 0 \\ C & I \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ 0 & D - CA^{-1}B \end{bmatrix} \\ &\implies \det\left\{\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right\} = \det\{D - CA^{-1}B\} \det(A). \end{aligned} \quad (2.47)$$

Similarly, if D is invertible:

$$\begin{aligned} \begin{bmatrix} A & B \\ C & D \end{bmatrix} &= \begin{bmatrix} I & B \\ 0 & D \end{bmatrix} \begin{bmatrix} A - BD^{-1}C & 0 \\ D^{-1}C & I \end{bmatrix} \\ &\implies \det\left\{\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right\} = \det\{A - BD^{-1}C\} \det(D). \end{aligned} \quad (2.48)$$

The matrix determinant lemma says that if A is invertible and if x and y have the appropriate sizes then the determinant of the rank-1 update is

$$\det\{A + xy'\} = (1 + y'A^{-1}x) \det\{A\}. \quad (2.49)$$

This property can be useful if the inverse and determinant of A are already known.

Example 2.33 Find the determinant of

$$B = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

$$B = I + 11' \implies \det\{B\} = 1 + 1'I1 = 4.$$

Check: $\det([2 \ 1 \ 1; 1 \ 2 \ 1; 1 \ 1 \ 2])$.

Sylvester's determinant theorem, or the Weinstein–Aronszajn identity, can be useful for rectangular matrices $A, B \in \mathbb{F}^{M \times N}$ with $N \ll M$:

$$\det\{I_M + AB'\} = \det\{I_N + B'A\}. \quad (2.50)$$

More generally, if $B, C \in \mathbb{F}^{M \times N}$, and if $A \in \mathbb{F}^{N \times N}$ and $D \in \mathbb{F}^{M \times M}$ are invertible, then

$$\det\{D + CA^{-1}B'\} \det\{A\} = \det\{A + B'D^{-1}C\} \det\{D\}. \quad (2.51)$$

The proof follows from (2.47) and (2.48) (see Problem 2.21).

For practical use of the determinant in JULIA use `det(A)`, having specified `using LinearAlgebra` or using `LinearAlgebra: det`.

2.6.5 Small Matrices

We can use the above properties to show that the determinant of a 2×2 matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is $ad - bc$.

Proof. If a is nonzero, then multiplying the first row by $-c/a$ and adding to the second row yields $B = \begin{bmatrix} a & b \\ 0 & d - bc/a \end{bmatrix}$, which is upper triangular so has determinant $a(d - bc/a) = ad - bc$.

If a is zero, then swapping the first and second rows yields $C = \begin{bmatrix} c & d \\ 0 & b \end{bmatrix}$, which again is upper triangular and the determinant is cb , so the determinant of A in this case is $-cb = 0d - cb = ad - cb$. \square

Example 2.34 $\det\left\{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right\} = 0 \cdot 0 - 1 \cdot 1 = -1$, consistent with D4.

If $A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ then $\det(A) = a\det\begin{bmatrix} e & f \\ h & i \end{bmatrix} - b\det\begin{bmatrix} d & f \\ g & i \end{bmatrix} + c\det\begin{bmatrix} d & e \\ g & h \end{bmatrix}$.

One can verify this equality using the properties.

2.7 Eigenvalues

Definition An important property of any square matrix $A \in \mathbb{F}^{N \times N}$ is its eigenvalues, often denoted $\lambda_1, \dots, \lambda_N$, defined as the set of solutions of the characteristic equation

$$\det(zI - A) = 0, \quad (2.52)$$

where I denotes the identity matrix of the same size as A .

Viewed as a function of z , we call $\det(zI - A)$ the characteristic polynomial, and the eigenvalues of A are its roots.

When $A \in \mathbb{F}^{N \times N}$, it follows from (2.45) that the characteristic polynomial has degree N and thus N (possibly complex) roots by the fundamental theorem of algebra. Thus, by the definition in (2.52), A has N eigenvalues (but they are not necessarily distinct). If $A \in \mathbb{R}^{N \times N}$, then the characteristic polynomial has real coefficients, so any nonreal eigenvalues appear in complex-conjugate pairs. The characteristic polynomial is a monic polynomial that can always be factored as a product of monomials:

$$\det(zI - A) = \prod_{n=1}^N (z - \lambda_n). \quad (2.53)$$



The literature can be inconsistent about how many eigenvalues an $N \times N$ matrix has. For example, the characteristic polynomial of the 2×2 matrix $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ is $\det(zI - A) = z^2 = (z - 0)(z - 0)$, which has a repeated root at zero and is sometimes said to have “only one eigenvalue.” We will not use that terminology; we will say that the matrix has two eigenvalues, both of which are zero.

Explore 2.14 Determine the eigenvalues of $A = \begin{bmatrix} 2 & -1 \\ 3 & 6 \end{bmatrix}$.

Example 2.35 Determine the eigenvalues of $A = I_N + yy'$ for $y \in \mathbb{F}^N$. Using (2.49):

$$\begin{aligned} \det(zI_N - A) &= \det\{zI_N - I_N - yy'\} = \det\{(z - 1)I_N - yy'\} \\ &= \left(1 - y'((z - 1)I_N)^{-1}y\right) \det\{(z - 1)I_N\} \\ &= \left(1 - \|y\|_2^2/(z - 1)\right) (z - 1)^N = (z - 1)^{N-1} \left(z - (1 + \|y\|_2^2)\right), \end{aligned} \quad (2.54)$$

which has $N - 1$ roots (eigenvalues) at 1 and one root (eigenvalue) at $1 + \|y\|_2^2$.

As mentioned in Section 2.6, data matrices are usually nonsquare, except for graph adjacency matrices, so we will rarely need to examine the eigenvalues of a data matrix directly. But given $X \in \mathbb{F}^{M \times N}$, we will often consider the eigenvalues of the square $N \times N$ Gram matrix $X'X$ or of the square $M \times M$ outer-product matrix XX' that arises when estimating a covariance matrix or a scatter matrix.

2.7.1 Eigenvectors

Fact 2.13 If z is an eigenvalue of A , then (2.52) implies that $zI - A$ is a singular matrix (not invertible), so there exists a nonzero vector v such that $(zI - A)v = 0$, or, equivalently,

$$Av = zv. \quad (2.55)$$

Conversely, if (2.55) holds for a nonzero vector v , then z is an eigenvalue of A .

Definition Any nonzero vector v that satisfies (2.55) is called an eigenvector of A .

Example 2.36 For the matrix $A = I_N + yy'$ for $y \in \mathbb{F}^N$, one eigenvector is y because $Ay = (I_N + yy')y = (y + yy'y) = (1 + \|y\|_2^2)y$. Any vector of the form αy for nonzero $\alpha \in \mathbb{F}$ is also an eigenvector. All other eigenvectors are orthogonal to y because here $x \perp y$ iff $Ax = x$.

For an $N \times N$ matrix A , let $\lambda_1, \dots, \lambda_N$ denote its N eigenvalues. For each eigenvalue λ_i , there is a corresponding eigenvector v_i . So we have N equations of the form (2.55): $Av_1 = \lambda_1 v_1, \dots, Av_N = \lambda_N v_N$. It is convenient to collect these N matrix–vector equations into a single important matrix–matrix equation:

$$AV = V\Lambda, \quad V \triangleq [v_1 \ \dots \ v_N], \quad \Lambda \triangleq \text{Diag}\{\lambda_1, \dots, \lambda_N\}. \quad (2.56)$$

The $N \times N$ matrix V has the N eigenvectors as its columns, the $N \times N$ matrix Λ has the N eigenvalues along its diagonal, and their product has the scaled eigenvectors as its columns:

$$V\Lambda = [\lambda_1 v_1 \ \dots \ \lambda_N v_N]. \quad (2.57)$$

2.7.2 Practical Implementation

To find a set of eigenvalues and eigenvectors of a square matrix A in JULIA:

```
using LinearAlgebra
z,V = eigen(A)
```

To obtain just the eigenvalues, use any of `eigvals(A)`, `eigen(A).values`, or `z,_ = eigen(A)`. The underscore `_` output is discarded.

To obtain just a set of eigenvectors, use any of `eigvecs(A)`, `eigen(A).vectors`, or `_,V = eigen(A)`.



The usual notation for an eigenvalue is λ and when the eigenvalues are all real, by convention we often choose to order them such that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$. But the `eigvals` and `eigen` commands return the eigenvalues in essentially arbitrary order (depending on the matrix type), usually not decreasing; see JULIA's `eigen` documentation.

Example 2.37 Find the eigenvalues and eigenvectors of $A = \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix}$.

The characteristic polynomial is

$$\det(zI - A) = \det\left(\begin{bmatrix} z-2 & -1 \\ 1 & z-2 \end{bmatrix}\right) = (z-2)(z-2) + 1 = z^2 - 4z + 5,$$

which has roots $z = 2 \pm i$, so the two eigenvalues of A are $\lambda_1 = 2 + i$ and $\lambda_2 = 2 - i$, where $i = \sqrt{-1}$.

To find the eigenvectors, note that, by inspection,

$$(A - \lambda_1 I) v_1 = \begin{bmatrix} -i & 1 \\ -1 & -i \end{bmatrix} v_1 = \mathbf{0} \quad \text{when } v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ i \end{bmatrix},$$

$$(A - \lambda_2 I) v_2 = \begin{bmatrix} i & 1 \\ -1 & i \end{bmatrix} v_2 = \mathbf{0} \quad \text{when } v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -i \end{bmatrix}.$$

JULIA check: `eigen([2 1; -1 2])`.

2.7.3 Properties of Eigenvalues

The defining determinant properties in Section 2.6 lead to useful eigenvalue properties.

D1 \implies If A is upper triangular, then the eigenvalues of A are its diagonal elements.

D3 \implies The eigenvalues of A' are the complex conjugates of the eigenvalues of A .

D2 \implies The eigenvalues of αA are α times the eigenvalues of A for $\alpha \in \mathbb{F}$.

D2 \implies The eigenvalues of A are invariant to similarity transforms [4, Theorem 9.19].

Fact 2.14 If A is $N \times N$ and T is an $N \times N$ invertible matrix then TAT^{-1} has the same eigenvalues as A .

Fact 2.15 As a special case, if Q is an $N \times N$ orthogonal (or unitary in the complex case) matrix, then QAQ' has the same eigenvalues as A .

Proof.

$$\begin{aligned} \det(zI - TAT^{-1}) &= \det(zTT^{-1} - TAT^{-1}) = \det(T(zI - A)T^{-1}) \\ &= \det((zI - A)T^{-1}T) = \det((zI - A)I) = \det(zI - A). \end{aligned}$$

So TAT^{-1} and A have the same characteristic polynomial. Here we used the distributive property of matrix multiplication. \square

Fact 2.16 The determinant of any square matrix is the product of its eigenvalues [4, Theorem 9.25]:

$$A \in \mathbb{F}^{N \times N} \implies \det(A) = \prod_{i=1}^N \lambda_i(A). \quad (2.58)$$

A is invertible iff all of its eigenvalues are nonzero.

2.7.3.1 Eigenvalue Properties for Matrix Products

Fact 2.17 If A has eigenvalues $\{\lambda_1, \dots, \lambda_N\}$, then A^2 has eigenvalues $\{\lambda_1^2, \dots, \lambda_N^2\}$. More generally, for $k \in \mathbb{N}$ the matrix power A^k has eigenvalues $\{\lambda_1^k, \dots, \lambda_N^k\}$.

Proof. $AV = V\Lambda \implies A^2V = A(AV) = A(V\Lambda) = (AV)\Lambda = (V\Lambda)\Lambda = V\Lambda^2$. \square

Definition Denote the set difference or relative complement of set X in Y by

$$Y \setminus X \triangleq \{x \in Y : x \notin X\}. \quad (2.59)$$

Fact 2.18 Suppose $A \in \mathbb{F}^{M \times N}$ and $B \in \mathbb{F}^{N \times M}$. If z is a *nonzero* eigenvalue of AB , then z is also a *nonzero* eigenvalue of BA . We summarize this concisely as the following commutative property for eigenvalues:

$$\text{eig}(AB) \setminus \{0\} = \text{eig}(BA) \setminus \{0\}, \quad (2.60)$$

that is, the *nonzero* elements of each set of eigenvalues are the same. (Sylvester [36] noted this in 1883!)

Proof. If $ABv = zv$ for some nonzero v , then clearly $u \triangleq Bv$ is also nonzero. Multiplying by B yields $BABv = zBv$, so $BAu = zu$ and hence z is a nonzero eigenvalue of BA . \square

Q2.17 What is the set of nonzero eigenvalues of the outer product matrix

$$A = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} [1 \ 1 \ 1]?$$

- A: {1, 2, 3} B: {1, 2} C: {1} D: {2} E: {6}

Here is another simple proof of (2.60) that goes further by showing that the nonzero eigenvalues of CB and BC have the same multiplicity for $C \in \mathbb{F}^{N \times M}$ and $B \in \mathbb{F}^{M \times N}$.

Proof. Let $A = \sqrt{\lambda}I_M$ and $D = \sqrt{\lambda}I_N$ in (2.47) and (2.48), so that

$$\begin{aligned} \det(A) \det(D - CA^{-1}B) &= \det(D) \det(A - BD^{-1}C) \\ \implies \lambda^{M/2} \det(\sqrt{\lambda}I_N - \lambda^{-1/2}CB) &= \lambda^{N/2} \det(\sqrt{\lambda}I_M - \lambda^{-1/2}BC) \\ \implies \lambda^{(M-N)/2} \det(\lambda I_N - CB) &= \lambda^{(N-M)/2} \det(\lambda I_M - BC). \end{aligned} \quad (2.61)$$

Writing the two determinants as products of monomials, we see that all the nonzero eigenvalues \mathbf{CB} and \mathbf{BC} are the same and have the same multiplicities. \square

We often use (2.60) to look for the smallest and or largest eigenvalue, for which multiplicity is unimportant.

Explore 2.15 Find the eigenvalues of $\mathbf{AB} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -2 & 2 & -2 & 2 \end{bmatrix}$.

2.7.3.2 Units of Eigenvalues

Data matrices involving scientific quantities can have elements with units, and it is useful to consider how the matrix element's units affect the units of the eigenvalues themselves.

From the eigenvalue definition in (2.52), for matrices whose elements have units, eigenvalues are well defined only if all of the diagonal elements of the matrix have the same units. In such cases, the eigenvalues must have the same units as those diagonal elements. Otherwise $a_{ii} - \lambda$ would not be well defined. Furthermore, since definition (2.52) involves a determinant, eigenvalues are defined only if (2.44) is applicable, that is, when the row-column property in (2.20) holds. Combining these conditions, $\alpha_i\beta_i$ must have the same units, say γ , for $i = 1, \dots, N$. Thus, the factorization in (2.20) has the form

$$\mathbf{A} = \text{Diag}\{\alpha_i\} \check{\mathbf{A}} \text{Diag}\{\gamma/\alpha_i\}. \quad (2.62)$$

In applications where all of the matrix elements have the same units, the eigenvalues have those same units.

Explore 2.16 Is it necessary for all matrix elements to have the same units for eigenvalues to be well defined?

Explore 2.17 Do eigenvectors have units?

2.8 Trace

Another important matrix property is its trace.

Definition The trace of a square matrix, denoted $\text{trace}[\mathbf{A}]$, is the sum of its diagonal elements [4, p. 6].

Some of the following properties of trace are shown as part of Problem 2.26.

- $\text{trace}\{\cdot\}$ is a linear function:

$$\text{trace}\{\alpha\mathbf{A} + \beta\mathbf{B}\} = \alpha \text{trace}\{\mathbf{A}\} + \beta \text{trace}\{\mathbf{B}\} \quad \forall \alpha, \beta \in \mathbb{F}.$$

- A cyclic commutative property:

$$A \in \mathbb{F}^{M \times N}, B \in \mathbb{F}^{N \times M} \implies \text{trace}\{AB\} = \text{trace}\{BA\}. \quad (2.63)$$

- Why cyclic? Because (let $A = X$ and $B = YZ$):

$$\text{trace}\{XYZ\} = \text{trace}\{YZX\} = \text{trace}\{ZXY\} \neq \text{trace}\{XZY\}.$$

- $\text{trace}\{A\}$ is the sum of the eigenvalues of A [4, Theorem 9.25]. (The proof involves the Jordan normal form of Section 3.2.2.1, a linear algebra topic of less importance in DS–ML–SP.)

Explore 2.18 Determine a trace property for stacking matrices.

For practical use in JULIA, first load the `LinearAlgebra` package with `using LinearAlgebra` or `using LinearAlgebra: tr`. Then `tr(A)` returns the trace of a square matrix A .

For consideration of units, see Problem 2.15.

2.9 Summary

This chapter has reviewed basic properties of vectors and matrices. It is notable that multiplication operations are prevalent throughout. For example, orthogonality is a property involving multiplication, and even the key properties defining a determinant (and hence eigenvalues) involve multiplication. The only property that is defined in terms of a sum is the matrix trace, but after defining it we immediately considered matrix products (and sums). Matrix and vector products will continue to be prevalent hereafter.

Solutions to Explorations

Explore 2.1 For this data matrix to be useful, one also needs column and row labels. The JULIA package `DataFrames.jl` is useful for such tabular data.

Explore 2.2 Yes, when $\mathbb{F} = \mathbb{R}$, but not when $\mathbb{F} = \mathbb{C}$ because $\langle x, \alpha y \rangle = \alpha^* \langle x, y \rangle \neq \alpha \langle x, y \rangle$ when α has a nonzero imaginary part.

Explore 2.3 The second element of x , which is 20.

Explore 2.4 For compatible matrix sizes, $\begin{bmatrix} B \\ C \end{bmatrix} A = \begin{bmatrix} BA \\ CA \end{bmatrix}$.

Explore 2.5 The $O(N^3)$ formula on p. 30 leads to the natural answer $4^3 = 64$. However, a machine learning algorithm “discovered” an approach that requires only 47 multiplies [25].

Explore 2.6 Yes. It belongs to the vector space of 2×2 matrices of the form

$$\begin{bmatrix} x_{11} \text{ m} & x_{12} \text{ s} \\ x_{21} \text{ g m/s} & x_{22} \text{ g} \end{bmatrix},$$

where the x_{ij} values all come from the same field such as \mathbb{R} . Here, the field of scalars can be (unitless) \mathbb{R} . This example is a reminder that the elements of a matrix need not come from the scalar field used for scalar multiplication.

Explore 2.7 $\int_0^3 x^2 dx = x^3/3|_{x=3} = 9$.

Explore 2.8 Because $w = \text{diff}(x)$ returns a vector without the first element “ $x_1 - x_0$ ” in (2.23).

Explore 2.9 If $U \in \mathbb{F}^{M \times N}$ is semiunitary, then $\langle Ux, Uy \rangle = (Uy)'Ux = y'U'Ux = y'x = \langle x, y \rangle \quad \forall x, y \in \mathbb{F}^N$.

Explore 2.10 No. Consider $A = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 \end{bmatrix}$.

Explore 2.11 $1 = \det\{I\} = \det\{A^{-1}A\} = \det\{A^{-1}\} \det\{A\}$.

Explore 2.12 When concluding that $\det\{I + b e_m e_k'\} = 1$. For $m = k$ this would be $1 + b$.

Explore 2.13 $(1 + b) \det\{A\}$.

Explore 2.14 $\det\{A - zI\} = \det\left(\begin{bmatrix} 2 - z & -1 \\ 3 & 6 - z \end{bmatrix}\right) = (2 - z)(6 - z) + 3 = z^2 - 8z + 15 = (z - 3)(z - 5)$, so the eigenvalues are $\{3, 5\}$.

Explore 2.15 $BA = \begin{bmatrix} 4 & 2 \\ 0 & 4 \end{bmatrix}$. So the only nonzero eigenvalue of BA is 4 (repeated) and thus the only nonzero eigenvalue of AB is also 4. Using (2.61), we have $\text{eig}\{AB\} = (4, 4, 0, 0)$.

Explore 2.16 No. Consider the 2×2 case involving roots of

$$\det\left(\begin{bmatrix} a_{11} - z & a_{12} \\ a_{21} & a_{22} - z \end{bmatrix}\right) = (a_{11} - z)(a_{22} - z) - a_{12}a_{21}.$$

Here, a_{11} and a_{22} must have the same units, but it suffices for the product $a_{12}a_{21}$ to have the same units as the square of the units of a_{11} and a_{22} . Example:

$$A = \begin{bmatrix} 2 \text{ m} & -1 \text{ m/s} \\ 3 \text{ ms} & 6 \text{ m} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \text{ s} \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 1 \text{ m} & 0 \\ 0 & 1 \text{ m/s} \end{bmatrix}$$

has eigenvalues {3 m, 5 m}.

Explore 2.17 For eigendecompositions (see Section 3.2), we usually scale eigenvectors to have unit norm, and those normalized eigenvectors must be unitless. But any (nonzero) scaled version is also an eigenvector, per (2.55), and the scaling factor could have units. For the product (2.62), if \mathbf{x} is an eigenvector of \check{A} , that is, $\check{A}\mathbf{x} = \lambda\mathbf{x}$, then $\mathbf{v} = D\mathbf{x}$ for $D = \text{Diag}\{\alpha_i\}$ is an eigenvector of A with eigenvalue $\gamma\lambda$. In general, for $N > 1$, that \mathbf{v} has elements with different units so it cannot be normalized. Nevertheless, eigendecompositions (see Section 3.2.2) are possible; if $\check{A} = V\Lambda V^{-1}$, then $A = (DV)(\gamma\Lambda)(DV)^{-1}$.

Explore 2.18 For square matrices of the same size:

$$\text{trace}\left\{\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right\} = \text{trace}(A) + \text{trace}(D).$$

Problems

Problem 2.1 Let A be an $M \times N$ matrix having elements $a_{ij} = i2^j$ for $i = 1, \dots, M$, $j = 1, \dots, N$.

- (a) Express matrix A mathematically as an outer product of two appropriately defined vectors.
- (b) Write (on paper) a one-line JULIA expression for creating A when M and N are defined already. Test your JULIA expression.

Problem 2.2 Rewrite the expression

$$y = \sum_{i=1}^N \sum_{j=1}^N x_i^* a_{ij} x_j$$

in terms of the $N \times N$ matrix A and the vector $x \in \mathbb{F}^N$ whose i th entry is x_i . Check your answer using JULIA by trying some random examples.

Hint: Consider (2.5). The left-hand side is an expression involving vectors, whereas the right-hand side is an expression involving elements of those vectors. We often need to go back and forth between these two types of expressions.

Problem 2.3 Let Σ be an $M \times N$ diagonal matrix. Let the diagonal elements of Σ be denoted by $\sigma_1, \sigma_2, \dots$, and let U and V be $M \times M$ and $N \times N$ matrices, respectively. Define $A = U\Sigma V'$.

- (a) Suppose $M < N$. Express A as a sum of outer-product matrices where each outer-product matrix is formed from the columns of U and V . How many such outer-product matrices add up to form A ?
- (b) Suppose $M > N$. Express A as a sum of outer-product matrices where each outer-product matrix is formed from the columns of U and V . How many such outer-product matrices add up to form A ?
- (c) Generalize the answer above: what is the maximum number of outer-product matrices formed from the columns of U and V , for $A = U\Sigma V'$, that can add up to form A ?

Problem 2.4 For vectors $x \in \mathbb{F}^M$ and $y \in \mathbb{F}^N$, so that $xy^\top \in \mathbb{F}^{M \times N}$, prove that

$$\text{vec}(xy^\top) = y \otimes x \in \mathbb{F}^{MN}.$$

Here, \otimes is the Kronecker product in (2.16) and $\text{vec}(\cdot)$ is in (2.18).

Verify this by using the commands `vec(x * y')` and `kron(y, x)` in JULIA for some real vectors, and using `vec(x * transpose(y))` for some complex vectors. (The equality above does not hold for $\text{vec}(xy')$ when y is complex.)

Problem 2.5

- (a) Prove the vec trick (2.19).
- (b) Suppose A , X , and B are all $N \times N$ dense matrices. Determine how many scalar multiplications are needed for the left-hand side $\text{vec}(AXB)$ and compare to the number for the right-hand side $(B^\top \otimes A)\text{vec}(X)$. Which version uses fewer?

Problem 2.6

- (a) One use of the “vec trick” (2.19) is computing a 2D discrete Fourier transform (DFT) of a 2D signal. The (1D) DFT of a vector $x \in \mathbb{C}^N$ is the vector $f_x \in \mathbb{C}^N$ with entries

$$[f_x]_k = \sum_{n=1}^N x_n \exp\left(\frac{-2\pi i(k-1)(n-1)}{N}\right), \quad k = 1, \dots, N.$$

(Here we follow linear algebra (and JULIA) where the first array index is 1, whereas DSP books usually use $0, \dots, N-1$.) We can represent the above computation in matrix–vector form as $f_x = F_N x$, where F_N is the $N \times N$ DFT matrix with entries

$$[F_N]_{k,n} = \exp\left(\frac{-2\pi i(k-1)(n-1)}{N}\right), \quad k, n = 1, \dots, N.$$

We can generate F_N in JULIA as follows.

```
</> 2.15    # N x N DFT matrix
           using FFTW: fft
           using LinearAlgebra: I
           F = fft(I(N), 1)
```

One can verify that $\mathbf{F}'_N \mathbf{F}_N = N\mathbf{I}_N$, so the (1D) inverse DFT of f_x can be computed as $x = (1/N)\mathbf{F}'_N f_x$.

We compute the 2D DFT, call it S_X , of the $M \times N$ matrix X by computing the 1D DFT of each column of X followed by the 1D DFT of each row of the result (or vice versa). Explain why the following expression computes the 2D DFT of X :

$$S_X = \mathbf{F}_M X \mathbf{F}_N^\top.$$

- (b) Write $\text{vec}(S_X)$ as the product of a matrix and $\text{vec}(X)$.
- (c) Show that the following expression computes the 2D inverse DFT of S_X :

$$X = \frac{1}{MN} \mathbf{F}'_M S_X \bar{\mathbf{F}}_N,$$

where \bar{Y} denotes the (elementwise) complex conjugate of matrix Y .

Hint: Use the fact that $\mathbf{F}'_N \mathbf{F}_N = \mathbf{F}_N \mathbf{F}'_N = N\mathbf{I}_N$.

- (d) Write $\text{vec}(X)$ as the product of a matrix and $\text{vec}(S_X)$.

Problem 2.7 Evaluate the following block matrix operations by simplifying as much as possible. (For simplicity, you may assume A, B, \dots, H are all square matrices of the same size. Most of the equalities generalize to rectangular matrices of appropriate sizes.) I_2 denotes the 2×2 identity matrix.

- (a) $[A \ B \ C] \begin{bmatrix} D \\ E \\ F \end{bmatrix}$
- (b) $\begin{bmatrix} A \\ B \end{bmatrix} [C \ D]$
- (c) $A [B \ C \ D]$
- (d) $\begin{bmatrix} A \\ B \\ C \end{bmatrix} D$
- (e) $\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix}$
- (f) $(I_2 \otimes A) \begin{bmatrix} E & F \\ G & H \end{bmatrix}$
- (g) $[A \ B] \begin{bmatrix} C & 0 \\ 0 & D \end{bmatrix} [E \ F]'$
- (h) $\text{trace}\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right)$
- (i) $\det\left(\begin{bmatrix} A & 0 \\ B & C \end{bmatrix}\right)$
- (j) $A \text{ Diag}(\lambda_1, \dots, \lambda_N)$

Problem 2.8 The vectors u , v , x , y , and z are all in \mathbb{R}^N and we want to compute $z^*u'*v*x'*y$.

- (a) Rewrite the code with parentheses to ensure that the computation is as efficient as possible.
- (b) Exactly how many scalar multiplication operations are needed? Explain.

Problem 2.9 After defining matrix–matrix multiplication in terms of the elements of those matrices, Section 2.4.4.2 describes four different versions. In fact there are two more versions. Study versions 1–4 and describe two more versions that are distinct from the definition and the given versions. Discuss any possible advantages or disadvantages of these versions.

Problem 2.10 Show that if A is an invertible matrix, then A^k is invertible and $(A^k)^{-1} = (A^{-1})^k \forall k \in \mathbb{N}$, where $A^k = \underbrace{AA \cdots A}_{k \text{ times}}$.

Problem 2.11 Generalize the vec trick (2.19) to simplify $(C \otimes B \otimes A)\text{vec}(X)$.

Problem 2.12 A matrix $A \in \mathbb{F}^{N \times N}$ is called idempotent iff $A^2 = A$ (see Section 5.9.1). Show that the matrix $A = \frac{1}{2} \begin{bmatrix} 2\cos^2(\theta) & \sin(2\theta) \\ \sin(2\theta) & 2\sin^2(\theta) \end{bmatrix}$ is idempotent for all θ .

Hint: Verify that $A = vv'$, where $v = \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}$.

Problem 2.13 We use the notation \mathbb{F} to denote either the field \mathbb{R} of real numbers or the field \mathbb{C} of complex numbers. This problem reviews what a field is (see Section 1.6).

Determine whether the following sets (with the usual senses of multiplication, addition, etc.) are fields or not. If the answer is yes then you may say so without proof. If the answer is no then give a concrete counterexample for one of the defining properties of a field that is violated.

- (a) The set of numbers that are irrational or zero, that is, the set $(\mathbb{R} - \mathbb{Q}) \cup \{0\}$.
- (b) The set of $N \times N$ diagonal matrices (where the “1” element is I_N , and the “0” element is $\mathbf{0}_{N \times N}$).
- (c) The set of $N \times N$ diagonal matrices whose diagonal elements are either all zero or all nonzero.
- (d) The set of rational functions, that is, functions of the form $P(x)/Q(x)$ where P and Q are both polynomials (over the same field \mathbb{F}) and Q is not zero.
- (e) The set of $N \times N$ invertible matrices along with the $N \times N$ zero matrix.

Problem 2.14 Determine whether the condition (2.20) is necessary and sufficient for a matrix A with units to be invertible.

Problem 2.15 For a matrix A whose values have units, find necessary and sufficient conditions for its trace to be well defined.

Problem 2.16 Let $U_1, U_2, \dots, U_k \in \mathbb{F}^{N \times N}$ denote unitary matrices. Show that the product $U_1 U_2 \cdots U_k$ is a unitary matrix.

Problem 2.17 Let v_1, v_2, \dots, v_N denote orthonormal vectors in \mathbb{R}^N . Let \mathcal{B} denote the set of vectors $\{Av_1, Av_2, \dots, Av_N\}$ for $A \in \mathbb{R}^{N \times N}$.

- (a) Show that if A is an orthogonal matrix, then \mathcal{B} is an orthonormal set.
- (b) Show the converse: if \mathcal{B} is an orthonormal set, then A is an orthogonal matrix.

Problem 2.18

- (a) Let Q denote an $M \times K$ matrix having orthonormal columns. Show that $Q'Q = I_K$. Thus, $\|Qx\|_2 = \|x\|_2$ for all $x \in \mathbb{C}^K$.
- (b) Show that the following converse is true: if A is an $M \times K$ matrix for which $\|Ax\|_2 = \|x\|_2$ for all $x \in \mathbb{C}^K$, then $A'A = I_K$. Your proof should be general enough to cover the case where A has complex elements.
Hint: Examine products with standard unit vectors e_j and combinations like $e_j + e_k$, or consider an eigendecomposition of $A'A - I$.

Problem 2.19 Use the defining determinant properties on p. 45 to solve the following problem. If A is orthogonal, what are the possible values of $\det\{A\}$?

Problem 2.20

- (a) For $x, y \in \mathbb{F}^N$, prove that $\det\{I - xy'\} = 1 - y'x$.
Hint: Use the defining determinant properties on p. 45, Fact 2.12, (2.47), and (2.48).
- (b) Express $\det\{\lambda I_N - xy'\}$ in terms of λ , $y'x$, and N .
Hint: Consider writing $\lambda I_N - xy' = (\lambda I_N)(I_N - xy'/\lambda)$ when $\lambda \neq 0$. Also consider the $\lambda = 0$ case.
- (c) Use the previous step to find all eigenvalues of the matrix xy' .
- (d) When are the eigenvalues of the matrix xy' all equal to 0 even when the matrix is not equal to zero?

Problem 2.21 Show that if $B, C \in \mathbb{F}^{M \times N}$, and if $A \in \mathbb{F}^{N \times N}$ and $D \in \mathbb{F}^{M \times M}$ are invertible, then $\det\{D + CAB'\} = \det(A^{-1} + B'D^{-1}C)\det\{A\}\det\{D\}$. You may use properties stated or shown in previous problems.

Problem 2.22

- (a) Determine by hand the eigenvalues of $A = \begin{bmatrix} 6 & 16 \\ -1 & -4 \end{bmatrix}$.
- (b) Compute the determinant and trace of this matrix A . Compute the product of the eigenvalues of A and the sum of its eigenvalues. How do these compare to the determinant and the trace?
- (c) Check your answers by invoking:
`using LinearAlgebra lambda, V = eigen(A)`
Are the eigenvectors (columns of V) orthogonal? Determine the matrix $V'V$.

- (a) Some software sorts the eigenvalues in a certain order. What order does JULIA return for symmetric matrices?

Problem 2.23 Prove how the eigenvalues and eigenvectors of $B = A - 10I$ relate to the eigenvalues and eigenvectors of A , when $A, B \in \mathbb{F}^{N \times N}$.

Problem 2.24 Let $B = T^{-1}AT$ for an invertible matrix T . Determine the relationship between the eigenvalues and eigenvectors of B and the eigenvalues and eigenvectors of A . Explain. The matrices A and B , when thus related, are called similar.

Problem 2.25 A matrix T has the property $T^3 = I$. What are its possible eigenvalues?

Problem 2.26 For $A \in \mathbb{F}^{N \times N}$, the trace of A , denoted by $\text{trace}(A)$, is defined as the sum of its diagonal elements: $\text{trace}(A) = \sum_{i=1}^N a_{ii}$.

- (a) Show that the trace is a linear function, that is, if $A, B \in \mathbb{F}^{N \times N}$ and $\alpha, \beta \in \mathbb{F}$, then $\text{trace}\{\alpha A + \beta B\} = \alpha \text{trace}\{A\} + \beta \text{trace}\{B\}$.
- (b) Show that $\text{trace}\{AB\} = \text{trace}\{BA\}$, even though in general $AB \neq BA$, when both AB and BA are square. Your work should be general enough to handle cases where A and B are rectangular.
- (c) Let $S \in \mathbb{R}^{N \times N}$ be skew-symmetric, that is, $S^\top = -S$. Show that $\text{trace}\{S\} = 0$.
- (d) Prove the converse of the previous part, or provide a counterexample.

Problem 2.27 Suppose A and B are symmetric. Show that

$$\text{trace}\{AB\} = \text{vec}^\top(A) \text{vec}(B).$$

What is the corresponding property when neither are symmetric?

3 Matrix Factorization: Eigendecomposition and SVD

3.1 Introduction

One of the main topics in the previous chapter was matrix multiplication. This chapter introduces matrix factorizations; a factorization is somewhat like a reverse of matrix multiplication. Matrix decompositions revolutionized matrix computations [37] and underpin modern numerical methods. Section 3.2 starts with the eigendecomposition of symmetric matrices, then generalizes to normal and asymmetric matrices. Section 3.3.1 introduces the basics of the singular value decomposition (SVD) of general matrices. Section 3.4 discusses a simple application of the SVD that uses the largest singular value of a matrix, posed as an optimization problem, and then describes optimization problems related to eigenvalues and the smallest singular value. The “real” SVD applications appear in subsequent chapters. Section 3.5 discusses the special situations when one can relate the eigendecomposition and an SVD of a matrix, leading to the special class of positive (semi)definite matrices in Section 3.6. Along the way there are quite a few small eigendecomposition and SVD examples.

3.1.1 Matrix Factorizations

There are many factorizations used in linear algebra and numerical linear algebra. Here are eight important ones. The first six are for square matrices only. Of these eight, only the SVD accommodates any matrix size and type.

$\mathbf{A} = \mathbf{LU}$ ($M = N$): LU decomposition by Gaussian elimination, for some (not all) square \mathbf{A} : \mathbf{L} is lower triangular; \mathbf{U} is upper triangular. For general \mathbf{A} , use pivoting.

$\mathbf{A} = \mathbf{LL}'$ ($M = N$): Cholesky decomposition (for any positive definite \mathbf{A}): \mathbf{L} is lower triangular.

$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}'$ ($M = N$): Orthogonal eigendecomposition (for any symmetric or normal \mathbf{A}), \mathbf{Q} is unitary; Λ is diagonal (and real if $\mathbf{A} = \mathbf{A}'$).

$\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1}$ ($M = N$): Diagonalization (possible only for some square \mathbf{A}): \mathbf{V} is (linearly independent) eigenvectors; Λ is eigenvalues.

$\mathbf{A} = \mathbf{P}\mathbf{J}\mathbf{P}^{-1}$ ($M = N$): Jordan normal form for *any* square \mathbf{A} : \mathbf{P} is invertible; \mathbf{J} is block diagonal.

$A = QRQ'$ ($M = N$): Schur decomposition (for any square A): Q is unitary; R is upper triangular.

$A = QR$ ($M \geq N$): QR decomposition via Gram–Schmidt orthogonalization (for any “tall” A): Q has orthonormal columns; R is upper triangular.

$A = U\Sigma V'$ (any M, N): SVD (for any A): U and V are unitary, columns are singular vectors; Σ is (rectangular) diagonal with real, nonnegative, singular values.

The LU, QR, and Cholesky decompositions are important for solving systems of equations, but are less helpful for analysis than the other forms. This chapter focuses on two particularly important matrix decompositions: the eigendecomposition and the SVD.

There are no applications in this chapter, but the tools are the foundation for most of the applications that appear in later chapters. Sometimes we use these tools for mathematical analysis (on paper only, especially for very large problems) but often we use them numerically. A short summary of this chapter is this: an eigendecomposition is usually the right tool for (square) Hermitian matrices, whereas an SVD is usually the right tool otherwise.

3.1.2 Square Matrices

Recall that any (square) matrix $A \in \mathbb{F}^{N \times N}$ has N eigenvalues $\lambda_1, \dots, \lambda_N \in \mathbb{C}$, possibly nondistinct. For each eigenvalue λ_n , the matrix $A - \lambda_n I$ is singular, so there must exist a (nonzero) vector $v_n \in \mathbb{F}^N$ (an eigenvector) such that

$$(A - \lambda_n I) v_n = \mathbf{0} \implies Av_n = \lambda_n v_n \quad \text{for } n = 1, \dots, N. \quad (3.1)$$

In general, the eigenvalues and eigenvectors can be complex, even if A is real. In matrix form:

$$AV = VA, \quad V = [v_1 \cdots v_N], \quad A = \text{Diag}(\lambda_1, \dots, \lambda_N). \quad (3.2)$$

Because each v_n is nonzero, by convention we always normalize each to have *unit norm* for decompositions. Despite this normalization, V is never unique because we can always scale each v_n by ± 1 or even $e^{i\phi}$. Put another way, although the matrix V in (3.2) has N columns, the cardinality of the set of eigenvectors of any $N \times N$ matrix A is uncountably infinite.

For a general square matrix, the equality (3.2) is about all we can say about an $N \times N$ eigenvector matrix V . However, for (Hermitian) symmetric matrices we can say much more, thanks to the spectral theorem discussed next.

3.2 Spectral Theorem for Symmetric Matrices

Fact 3.1 If $A \in \mathbb{F}^{N \times N}$ is (Hermitian) symmetric, that is, $A = A'$, then the spectral theorem says the following:

- The eigenvalues of A are all real.
- Remarkably, there is an orthonormal basis for \mathbb{F}^N consisting of eigenvectors of A . That is, there exists a V as in (3.2) that is an orthogonal (or unitary) matrix; that is, $V'V = VV' = I$, so $V^{-1} = V'$.
- Multiplying (3.2) on the right by V' yields a unitary eigendecomposition (a matrix factorization):

$$A = V\Lambda V' = \sum_{n=1}^N \lambda_n v_n v_n'. \quad (3.3)$$

Both the factored (product) form and the “sum of outer products” form are useful (see Problem 2.3).

- If $A \in \mathbb{R}^{N \times N}$ is symmetric, then there exists a *real* orthogonal matrix V satisfying (3.3). Because λ_n is real, we can find a corresponding (unit norm) real eigenvector v_n satisfying (3.1).

In words, *every symmetric/Hermitian (hence square) matrix has an orthogonal/unitary eigendecomposition*. This factorization is very useful for analysis and sometimes for computation.

As mentioned previously, a data matrix X is rarely square, but the Gram matrix $X'X$ and the outer-product matrix XX' are always square. Furthermore, the Gram matrix and the outer-product matrix are always (Hermitian) symmetric, so the spectral theorem applies. In DS–ML–SP, we usually need an eigendecomposition only for matrices of the form $X'X$ or XX' .

Proof that eigenvalues are real. If v is an eigenvector of $A = A'$ with eigenvalue λ , then $Av = \lambda v \implies v'A v = \lambda v'v \implies (v'A v)' = \lambda' v'v \implies v'A'v = \lambda' v'v \implies v'A v = \lambda' v'v \implies \lambda = \lambda' = \lambda$. \square

We can also sketch a proof of the orthogonality of eigenvectors for the case of distinct eigenvalues. Suppose v is an eigenvector of $A = A'$ with (real) eigenvalue λ , and u is an eigenvector with different (real) eigenvalue $\beta \neq \lambda$. $Av = \lambda v \implies u'A v = \lambda u'v \implies (u'A v)' = (\lambda u'v)' \implies v'A'u = \lambda v'u \implies v'Au = \lambda v'u$. $Au = \beta u \implies v'Au = \beta v'u \implies \lambda v'u = \beta v'u \implies v'u = 0$ because $\beta \neq \lambda$.

Example 3.1 Consider the symmetric matrix $A = \begin{bmatrix} 4 & 6 \\ 6 & 9 \end{bmatrix}$ for which

$\det(A - \lambda I) = (4 - \lambda)(9 - \lambda) - 6^2 = \lambda^2 - 13\lambda$, so the eigenvalues of A are $\{13, 0\}$. Now,

$$A - 13I = \begin{bmatrix} -9 & 6 \\ 6 & -4 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} [-3 \ 2].$$

so an eigenvector corresponding to eigenvalue 13 is $v_1 = \frac{1}{\sqrt{13}} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$. Similarly, an eigenvector corresponding to eigenvalue 0 is $v_2 = \frac{1}{\sqrt{13}} \begin{bmatrix} -3 \\ 2 \end{bmatrix}$. Thus, an eigendecomposition is

$$\begin{aligned} A &= \underbrace{\begin{bmatrix} 4 & 6 \\ 6 & 9 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} 2 & -3 \\ 3 & 2 \end{bmatrix}}_{\Lambda} \underbrace{\begin{bmatrix} 13 & 0 \\ 0 & 0 \end{bmatrix}}_{\Lambda'} \underbrace{\begin{bmatrix} 2 & 3 \\ -3 & 2 \end{bmatrix}}_{V'} = 13v_1v_1' + 0v_2v_2' \\ &= \underbrace{\begin{bmatrix} 2 \\ 3 \end{bmatrix}}_{\sqrt{\lambda_1}v_1} \underbrace{\begin{bmatrix} 2 & 3 \end{bmatrix}}_{\sqrt{\lambda_1}v_1'}. \end{aligned}$$

3.2.1 Normal Matrices

Hermitian symmetry is a *sufficient* but not *necessary* condition for the existence of a unitary eigendecomposition.

Definition A matrix A is a normal matrix iff $A'A = AA'$.

Clearly, any normal matrix must be square, and every Hermitian matrix is normal.

Fact 3.2 The spectral theorem says that a square matrix A is diagonalizable by a unitary matrix, that is, has a unitary eigendecomposition, in other words there exist V unitary and Λ diagonal such that $A = V\Lambda V'$, iff A is a normal matrix.

For a normal matrix Λ need not be real, whereas for a (Hermitian) symmetric matrix Λ is real.

Q3.1 Every unitary matrix U is a normal matrix? Every normal matrix is unitary?

- A: (T,T) B: (T,F) C: (F,T) D: (F,F)

3.2.1.1 Permutation Matrix

Example 3.2 An important type of normal matrix is a permutation matrix.

Definition An $N \times N$ permutation matrix has exactly one 1 in each row and one 1 in each column, and all the other elements are zero.

Fact 3.3 If P is a permutation matrix, then $P^{-1} = P'$ so P is an orthogonal matrix.

Proof 1. Write $P = [p_1 \quad \dots \quad p_N]$ so $[P'P]_{ij} = p_i'p_j = \begin{cases} 1, & i = j, \\ 0, & \text{otherwise,} \end{cases}$ because $p_i'p_i = 1$ since p_i is all zeros except for one element, and $p_i'p_j = 0$ for $i \neq j$ because p_i

and p_j must have their 1 in different locations since every row contains exactly one 1. Thus $P'P = I$. Similarly, $PP' = I$. \square

Proof 2. $PP' = \sum_{n=1}^N p_n p'_n = I$ because the outer product $p_n p'_n = \text{Diag}\{p_n\}$. \square

Because P is an orthogonal matrix, it is normal, so P has a unitary eigendecomposition. Typically its eigenvalues and eigenvectors are complex.

Example 3.3 The permutation matrix that (circularly) shifts each element of a vector in \mathbb{F}^3 by one index is

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \quad (3.4)$$

This permutation matrix happens to be a circulant matrix (see Section 8.3), so its eigenvalues are given by the three-point DFT of the first column: $\{e^{i2\pi k/3} : k = 0, 1, 2\} = \{1, e^{\pm i2\pi/3}\}$.

Example 3.4 The following rotation matrix is asymmetric unless ϕ is a multiple of π :

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

Yet this matrix must have two eigenvalues with corresponding eigenvectors. The eigenvalues satisfy $(\cos \phi - \lambda)^2 + \sin^2 \phi = 0$ leading to $\lambda_{\pm} = e^{\pm i\phi}$. The corresponding eigenvectors are $v_{\pm} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ \pm i \end{bmatrix}$ because

$$\begin{aligned} R_\phi \begin{bmatrix} 1 \\ si \end{bmatrix} &= \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} 1 \\ si \end{bmatrix} = \begin{bmatrix} \cos \phi + si \sin \phi \\ -\sin \phi + si \cos \phi \end{bmatrix} \\ &= (\cos \phi + si \sin \phi) \begin{bmatrix} 1 \\ si \end{bmatrix} = e^{si\phi} \begin{bmatrix} 1 \\ si \end{bmatrix} \end{aligned}$$

for $s \in \{\pm 1\}$. This rotation matrix is normal because $R_{-\phi} = R'_\phi = R_\phi^{-1}$; that is, multiplying by R and then R' corresponds to rotating by ϕ and then rotating back, and $R'R = RR' = I_2$. (In fact, R is unitary.) R is diagonalizable by the unitary matrix $V = [v_+ \ v_-]$ and here is a unitary eigendecomposition:

$$R_\phi = \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} = \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix} \right) \begin{bmatrix} e^{i\phi} & 0 \\ 0 & e^{-i\phi} \end{bmatrix} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -i \\ 1 & i \end{bmatrix} \right). \quad (3.5)$$

R is not diagonalizable by a real matrix V because rotation leads to a vector pointing in a different direction, unless ϕ is a multiple of π . But R is diagonalizable by the above unitary matrix V .

Q3.2 What is the best way to think about the rotation matrix R ?

- A: A data matrix. B: An operator matrix. C: Neither.

Example 3.5 The permutation example in (3.4) and the previous rotation example are unitary matrices, so clearly normal. An example of a nonunitary (and also non-Hermitian) normal matrix is

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

having eigenvalues $\{2, (1 \pm i\sqrt{3})/2\}$, for which

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

3.2.1.2 Sums and Products of Normal Matrices

If A and B are normal matrices, what about their sum $A + B$ and their product AB ?

Consider $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$. These are both unitary matrices, and hence normal. But $A + B = \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix}$ which is not normal and not even diagonalizable (see Section 3.2.2).

Now consider $X = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, which is diagonal, and hence normal. But the product $Y = AX = \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix}$ is not normal.

3.2.2 Square Asymmetric and Nonnormal Matrices

Some, but not all, square *asymmetric* matrices that are not normal matrices are diagonalizable.

Definition A square matrix is diagonalizable iff it is similar to a diagonal matrix, *i.e.*, iff there exists an invertible matrix V such that $V^{-1}AV$ is diagonal. Otherwise it is called defective.

If $A \in \mathbb{F}^{N \times N}$ has N linearly independent eigenvectors $V = [v_1 \ \dots \ v_N]$, then V^{-1} exists and

$$A = V\Lambda V^{-1}, \quad \Lambda = \text{Diag}\{\lambda_n\}. \quad (3.6)$$

- If A is asymmetric and its eigenvalues are all real, then A cannot have a unitary eigendecomposition. (If it did, then we would have $A = V\Lambda V' = A'$, contradicting asymmetry.)

- If A has N distinct eigenvalues (no repeated roots of characteristic equation), then A is diagonalizable. (But having distinct eigenvalues is not a necessary condition for being diagonalizable.)
- If A is both invertible and diagonalizable then $A^{-1} = V\Lambda^{-1}V^{-1}$.
- Being diagonalizable does not imply invertibility because some (or all) eigenvalues of a diagonalizable matrix can be 0.

Some square matrices are not diagonalizable (see the following example). Such matrices might arise when trying to solve a system of N equations in N unknowns so they are a major topic in a linear algebra course, but they are much less important in DS-ML-SP so we do not dwell on them further here.

Every matrix, even if nonsquare, has a singular value decomposition (SVD) so that will be the primary tool we use for data matrices.

Example 3.6 The matrix $A = \begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix}$ has repeated eigenvalue $\lambda = 3$, and

$A - 3I = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ so $v = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Here, $AV = V\Lambda$, where $V = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ and

$\Lambda = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$ contains both eigenvalues of A , but the columns of V are linearly dependent so A is not diagonalizable, that is, it is defective.

Readers interested in general eigendecompositions and diagonalizable matrices should study minimal polynomials and the Jordan normal form in (3.7).

Section 9.2.11 uses the following definition.

Definition We say a set of matrices A_1, A_2, \dots is simultaneously diagonalizable iff there exists an invertible matrix P such that $P^{-1}A_kP$ is diagonal for every A_k in the set.

3.2.2.1 Jordan Normal Form

Fact 3.4 Every matrix $A \in \mathbb{F}^{N \times N}$ is similar to a matrix in Jordan normal form:

$$A = PJP^{-1} \quad (3.7)$$

for an invertible matrix P , where the matrix J is called the Jordan normal form of A .

The matrix J is a block diagonal matrix where each Jordan block has the upper triangular form shown on the right:

$$J = \begin{bmatrix} J_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & J_p \end{bmatrix}, \quad J_i = \begin{bmatrix} \lambda_i & 1 & & \mathbf{0} \\ & \lambda_i & \ddots & \\ & \mathbf{0} & & 1 \\ & & & \lambda_i \end{bmatrix}.$$

Including repeats, the eigenvalues of A are the diagonal elements of all the Jordan blocks.

The same eigenvalue can appear in more than one Jordan block.

Example 3.7 $B = \begin{bmatrix} 5 & 1 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$.

A is diagonalizable iff each Jordan block is a 1×1 matrix.

Example 3.8 The matrix B in the preceding example is *not* diagonalizable.

The full story involves the algebraic multiplicity and the geometric multiplicity of the eigenvalues, and these topics are rarely important in DS–ML–SP problems because we usually work with Hermitian symmetric (hence normal) matrices like $X'X$ and XX' .

Q3.3 Every permutation matrix has (one, all) real eigenvalues.

- A: (T,T) B: (T,F) C: (F,T) D: (F,F)

Q3.4 There exists a permutation matrix with all real eigenvalues.

- A: True B: False

3.2.3 Geometry of Matrix Diagonallization

Let $A \in \mathbb{F}^{N \times N}$ be a (Hermitian) symmetric matrix, so $A = V\Lambda V'$. Consider the linear transform $x \mapsto y = Ax = V\Lambda V'x$. We can think of Ax as a cascade of three linear transforms:

$$x \xrightarrow{V'} w \xrightarrow{\Lambda} z \xrightarrow{V} y.$$

$x \mapsto w = V'x$ is a coordinate change (a rotation, in fact, possibly with a sign flip for one axis). w denotes the coefficient vector for x in the basis V , because $x = Vw$.

$w \mapsto z = \Lambda w$ is scaling of each coordinate by the diagonal elements of Λ .

$z \mapsto y = Vz$ is going back to the original coordinate system.

It is useful to understand these three operations geometrically.

Example 3.9 We illustrate for the case of a symmetric 2×2 matrix.

Fact 3.5 Every $V \in \mathbb{R}^{2 \times 2}$ with $V'V = I_2$ (i.e., orthogonal) has the following form for some rotation angle θ :

$$V = \begin{bmatrix} \cos \theta & -q \sin \theta \\ \sin \theta & q \cos \theta \end{bmatrix}, \quad q \in \{\pm 1\}. \quad (3.8)$$

Explore 3.1 Verify that $V'V = I_2$.

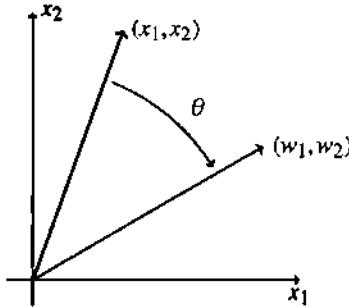


Figure 3.1 Illustrating $\mathbf{w} = V'\mathbf{x}$.

For simplicity we focus hereafter on the case of rotation matrices where $q = +1$. Consider the linear transform

$$\mathbf{x} \mapsto \mathbf{w} = V'\mathbf{x} = \begin{bmatrix} x_1 \cos \theta + x_2 \sin \theta \\ -x_1 \sin \theta + x_2 \cos \theta \end{bmatrix}.$$

As seen in Fig. 3.1, the lengths of \mathbf{x} and \mathbf{w} are the same:

$$\mathbf{w} = V'\mathbf{x} \implies \|\mathbf{w}\|_2^2 = \mathbf{w}'\mathbf{w} = (V'\mathbf{x})'V'\mathbf{x} = \mathbf{x}'VV'\mathbf{x} = \mathbf{x}'I\mathbf{x} = \mathbf{x}'\mathbf{x} = \|\mathbf{x}\|_2^2.$$

The next mapping is

$$\mathbf{w} \mapsto \mathbf{z} = \mathbf{A}\mathbf{w}, \text{ where } \mathbf{A} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \implies \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 w_1 \\ \lambda_2 w_2 \end{bmatrix}.$$

For interpretation, assume $\lambda_1, \lambda_2 \neq 0$ and suppose $\|\mathbf{x}\|_2 = 1 \implies \mathbf{x}'\mathbf{x} = 1 \implies x_1^2 + x_2^2 = 1$, which in turn implies $w_1^2 + w_2^2 = 1$, that is, \mathbf{x} and \mathbf{w} lie on the unit circle. Because $w_1 = z_1/\lambda_1$ and $w_2 = z_2/\lambda_2$, we have

$$\left(\frac{z_1}{\lambda_1}\right)^2 + \left(\frac{z_2}{\lambda_2}\right)^2 = 1,$$

that is, \mathbf{z} lies on an ellipse with axes governed by λ_1 and λ_2 . See Fig. 3.2.

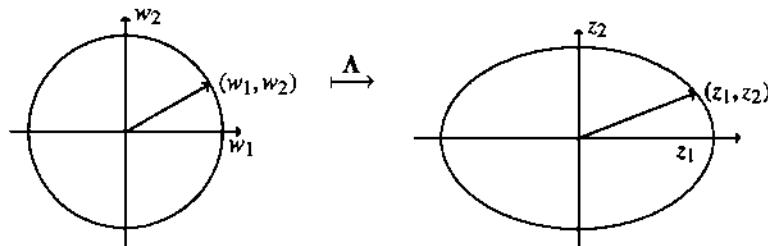


Figure 3.2 Illustrating $\mathbf{z} = \mathbf{A}\mathbf{w}$.

Typically \mathbf{z} is *not* collinear with \mathbf{w} . The exception is when $\lambda_1 = \lambda_2$, in which case $\mathbf{A} = \lambda_1 \mathbf{I}_2$, which is a trivial case.

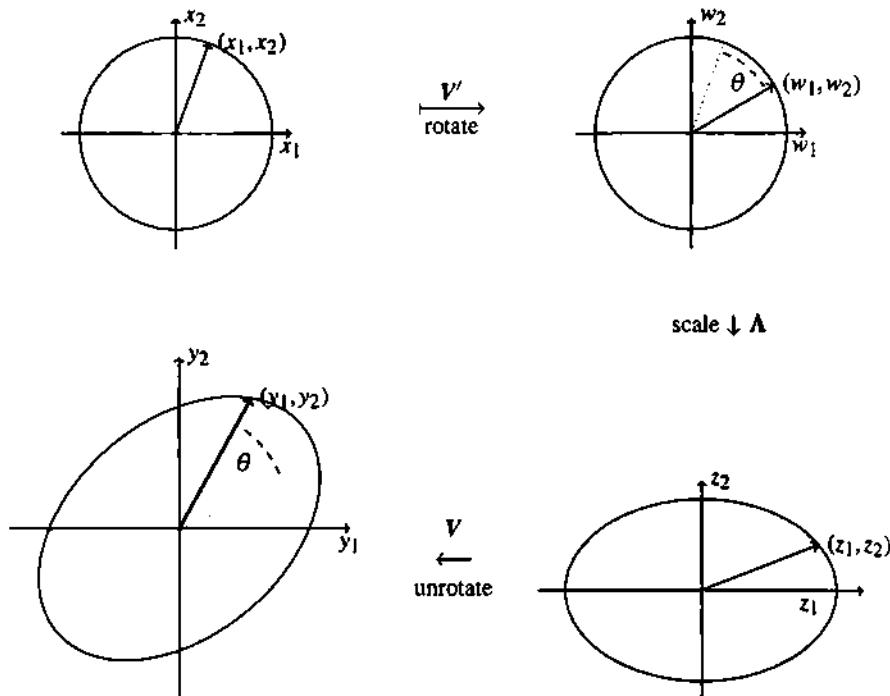


Figure 3.3 Illustrating $y = Ax = VAV^{-1}x$.

Explore 3.2 When does $y = Ax$ produce y that is collinear with x ?

For the third and final mapping, we return to geometry. If $x \xrightarrow{V'} w$ represents counterclockwise rotation, then $w \xrightarrow{V} x$ must represent clockwise rotation, because $VV' = I$ so $V(V'x) = (VV)x = Ix = x$. If V includes a sign flip, for example, $V = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, then $V^{-1} = V'$ also undoes that sign flip.

- An orthogonal matrix with determinant equal to +1 is called a rotation.
- If the determinant is -1 then it is an improper rotation.

We ignore the possibility of a sign flip in the graphical illustrations, for simplicity.

Figure 3.3 gives a graphical summary in the 2×2 case of $y = Ax = \underbrace{VAV^{-1}}_W \underbrace{x}_{V'x}$.
(The same principles apply in higher dimensions.)

Q3.5 If $A \in \mathbb{R}^{2 \times 2}$ has real eigenvalues λ_1, λ_2 , then the locus of points $\{Ax : x \in \mathbb{R}^2 \text{ where } \|x\|_2 = 1\}$ is a nondegenerate ellipse when:

- A: Always B: $\lambda_1 \lambda_2 \neq 0$ C: $\lambda_1^2 + \lambda_2^2 > 0$ D: $\lambda_1 + \lambda_2 > 0$ E: None of these

3.2.3.1 Practical Implementation in JULIA

`d = eigvals(A)` returns a 1D array of eigenvalues.

`V = eigvecs(A)` returns a matrix of eigenvectors.

`obj = eigen(A)` returns an “object” (akin to a dictionary) of type `Eigen` with components:

`d = obj.values` or `d = eigen(A).values` (vector containing eigenvalues);
`V = obj.vectors` or `V = eigen(A).vectors` (matrix with eigenvectors).

Note the use of argument/index chaining: `f(arg1).arg2` `f(arg1)[1]`.

To extract both parts in one line, use: `d, V = eigen(A)`.

If A is diagonalizable, then $A = V \text{Diag}(d) V^{-1}$, that is, $A = V * \text{Diagonal}(d) * \text{inv}(V)$.

- ◆ For work expressing eigenvector elements in terms of eigenvalues, see [38].

3.2.4 Matrix Powers and Matrix Exponential

Many analyses need matrix powers, another easy byproduct of eigendecompositions. We focus on powers of a square diagonalizable matrix $A \in \mathbb{F}^{N \times N}$:

$$A = V \Lambda V^{-1} \implies A^2 = AA = (V \Lambda V^{-1})(V \Lambda V^{-1}) = V \Lambda^2 V^{-1}. \quad (3.9)$$

We define $A^0 = I$. Proceeding by induction shows more generally that

$$A = V \Lambda V^{-1} \implies A^k = V \Lambda^k V^{-1} \quad \forall k \in \mathbb{N}. \quad (3.10)$$

Q3.6 If A is diagonalizable and invertible, then A^k is (diagonalizable, invertible) for all $k \in \mathbb{N}$?

- A: (T,T) B: (T,F) C: (F,T) D: (F,F) E: Depends

If A is a square matrix, then the matrix exponential is defined as the power series

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k. \quad (3.11)$$

This expression is useful when solving an ordinary differential equation (ODE).

If A is diagonalizable, that is, $A = V \Lambda V^{-1}$, then it follows that

$$e^A = V e^{\Lambda} V^{-1}, \quad e^{\Lambda} = \text{Diag}\{e^{\lambda_i}\}. \quad (3.12)$$

3.3 Singular Value Decomposition

3.3.1 Singular Values and Singular Vectors

Definition A nonnegative real number σ is called a singular value of a matrix $A \in \mathbb{F}^{M \times N}$ iff there exist unit norm vectors $u \in \mathbb{F}^M$ and $v \in \mathbb{F}^N$ for which

- $Av = \sigma u$, and
- $A'u = \sigma v$.

Any time this pair of relationships holds, we call u and v a pair of left and right singular vectors of A .

Fact 3.6 An $M \times N$ matrix has at most $\min(M, N)$ distinct singular values.

In contrast, the set of all possible singular vectors of A is uncountably infinite because if u and v are a pair of left and right singular vectors, then $-u$ and $-v$ are also a pair of left and right singular vectors. More generally, $e^{i\phi} u$ and $e^{i\phi} v$ are also a pair of left and right singular vectors.

Example 3.10 Consider the 1×2 matrix $A = [3 \ 4]$, and let $u = [1]$ and $v = \begin{bmatrix} 3/5 \\ 4/5 \end{bmatrix}$.

Then $Av = 5 = 5u$ and $A'u = \begin{bmatrix} 3 \\ 4 \end{bmatrix} = 5v$ so u and v are a pair of left and right singular vectors corresponding to the singular value $\sigma = 5$. This matrix has no other singular values because $\min(M, N) = 1$. This matrix is not square so it does not have an eigendecomposition.

3.3.2 Existence of SVD

Fact 3.7 If $X \in \mathbb{F}^{M \times N}$ then there exist (for proof, see [4, Theorem 5.1]) matrices U , V , Σ such that

$$X = U\Sigma V' = \sum_{k=1}^{\min(M, N)} \sigma_k u_k v'_k. \quad (3.13)$$

This factorization is called a singular value decomposition (SVD), where:

- $U \in \mathbb{F}^{M \times M}$ is unitary ($U'U = UU' = I_M$), and its columns consist of M left singular vectors of X .
- $V \in \mathbb{F}^{N \times N}$ is unitary ($V'V = VV' = I_N$), and its columns consist of N right singular vectors of X .
- If $\mathbb{F} = \mathbb{R}$, then U and V can each also be guaranteed to be a real orthogonal matrix.
- Σ is an $M \times N$ rectangular diagonal matrix containing *all* the $\min(M, N)$ singular values of X that looks like one of:

$$\Sigma = \underbrace{\begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_N \\ \hline & 0_{M-N,N} & \end{bmatrix}}_{M > N \text{ (tall)}} \quad \text{or} \quad \Sigma = \underbrace{\begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_M \\ \hline & & 0_{N-M,M} \end{bmatrix}}_{N > M \text{ (wide)}} \quad \text{or} \quad \Sigma = \underbrace{\begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_M \end{bmatrix}}_{M = N \text{ (square)}}$$

- These possible shapes of Σ are why the sum in (3.13) has the $\min(M, N)$ limit.
- The singular values $\sigma_1, \dots, \sigma_{\min(M, N)}$ are real and nonnegative.
- Every SVD of X uses the same set of singular values; that is, the set of singular values of X is unique.
- By convention we always use descending order: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(M, N)}$.
- The first r singular values are positive, where $0 \leq r \leq \min(M, N)$ is the rank of the matrix (see Section 4.3).
- We often casually call U and V “the” left and right singular vectors of X , but this wording is imprecise.
- For a history of the SVD, see [39].
- A subtle point is that the sum form (3.13) does not use all the columns of U when $M > N$, nor all the columns of V when $N > M$. More on this in Section 4.5.1 that discusses the compact SVD.

3.3.3 Geometry

Example 3.11 If A is any real 2×2 matrix, then its SVD has the form

$$A = U\Sigma V' = \begin{bmatrix} \cos \phi & -q_1 \sin \phi \\ \sin \phi & q_1 \cos \phi \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} \cos \theta & -q_2 \sin \theta \\ \sin \theta & q_2 \cos \theta \end{bmatrix}', \quad q_1, q_2 \in \{\pm 1\},$$

where $\theta \neq \phi$ in general. (In fact, when $M \neq N$, U and V even have different sizes!) Consider the case $q_1 = q_2 = +1$ for simplicity. Figure 3.4 illustrates the 2×2 geometry graphically.

What are the differences here (for SVD) from before (eigendecomposition)?

- The final rotation angle differs: $\phi \neq \theta$ in general; that is, $U \neq V$ in general.
- For nonsquare matrices, Σ is nonsquare, so the interpretation that it “scales” is incomplete.
- The SVD always exists, even for (asymmetric) 2×2 matrices that have no eigen-decomposition.

Explore 3.3 Find an SVD of the rotation matrix $R = \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix}$. (Recall that no *real* eigendecomposition exists for this important matrix in general.)

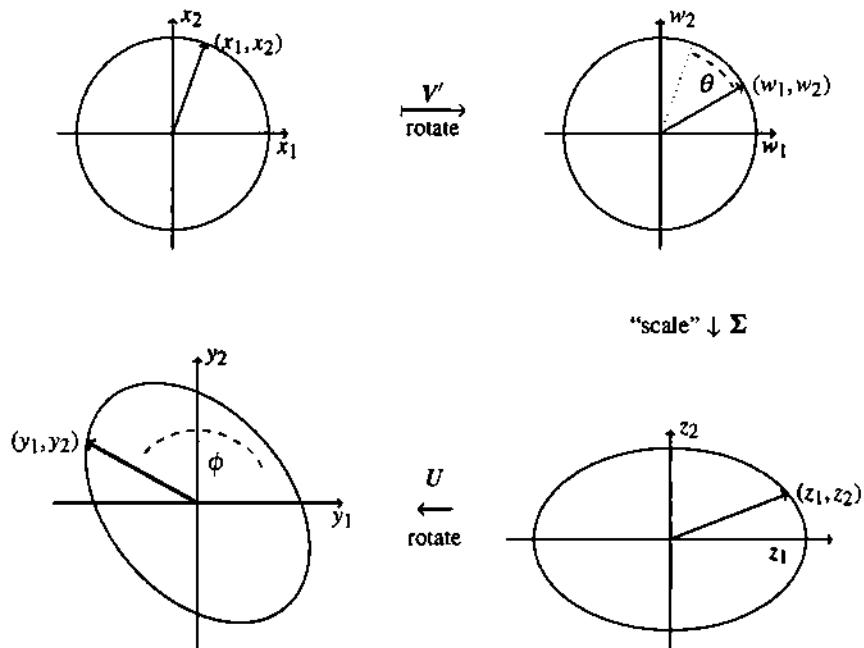


Figure 3.4 Singular value decomposition geometry for $N = M = 2$.

Q3.7 If we prevent sign flips by requiring $U_{11} \geq 0$ and $U_{12} \geq 0$, then this matrix R has a unique SVD.

A: True

B: False

For eigendecomposition we asked “when is Ax aligned with x ?”

Explore 3.4 Should we ask that question for the SVD?

Explore 3.5 What choice of vectors x and z makes Ax perpendicular to Az ?

In general, SVD maps “circles” (precisely: the set of vectors with unit norm) to “rotated ellipses.”

- The major and minor axis directions correspond to the left singular vectors in U .
- The eccentricity depends on the singular values.

3.3.4 Practical Implementation in Julia

`obj = svd(A)` returns an “object” (akin to a dictionary) of type `SVD` with components:

 $U = obj.U$ has size $M \times \min(M, N)$. Caution: U differs from $U \in \mathbb{F}^{M \times M}$ in the typical tall case where $M > N$.

$s = obj.S$ for a vector s of length $\min(M, N)$ such that $\text{Diag}(s)$ is the core of Σ .

$Vt = \text{obj}.Vt$ for V' has size $\min(M, N) \times N$.

 $V = \text{obj}.V$ (internally this is an `Adjoint` array to avoid a transpose!) Caution: V differs from $V \in \mathbb{F}^{N \times N}$ in the wide case where $M < N$.

Another option is $U, s, V = \text{svd}(A)$ or $U, s, V = (\text{svd}(A), \dots)$.

One can rebuild A (to within numerical precision) using $U * \text{Diagonal}(s) * V'$.

If one wants a full SVD where U is $M \times M$ and V is $N \times N$ then use `svd(A, full=true)`, but we rarely need that in practice! The default `svd(A)` is equivalent to `svd(A, full=false)` because this is the usual use case. The nonfull default in JULIA is equivalent to MATLAB's economy SVD option.

There is also `svd(A, 'econ')`, also sometimes called the thin SVD. Section 4.5.1 describes a compact SVD that differs further from the economy SVD and full SVD. See Fig. 4.7.

See [40] for a survey of methods for computing an SVD.

3.3.5 SVD Basic Properties

There are numerous SVD properties that will be explored in subsequent chapters and problems. Here are a few basic ones, assuming that $A = U\Sigma V'$.

- By the transpose property of matrix multiplication, $A' = V\Sigma'U'$. (Note that Σ is rectangular in general, so we do need the transpose Σ' above.)
- If v_i is the i th column of V , then $Av_i = \sigma_i u_i$ (see Problem 3.11).
- Similarly, if u_i is the i th column of U , then $A'u_i = \sigma_i v_i$.
- Scaling property: if $B = \alpha A$, then $B = U\tilde{\Sigma}V'$ with $\tilde{\Sigma} = \alpha\Sigma$ is an SVD of B .
- If $B = AQ'$, where Q is a unitary matrix, then an SVD of B is $B = U\Sigma\tilde{V}'$, where $\tilde{V}' = QV$ (see Problem 4.7).
- Pseudoinverse: $A^+ = V\Sigma^+U'$ (see Section 5.4).
- Non-uniqueness due to sign flips.

Example 3.12 Consider the following simple 2×2 diagonal matrix:

$$\begin{aligned} A &= \begin{bmatrix} 7 & 0 \\ 0 & 5 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_U \underbrace{\begin{bmatrix} 7 & 0 \\ 0 & 5 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{V'} \quad (\text{SVD version 1}) \\ &= \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}}_{\tilde{U}} \underbrace{\begin{bmatrix} 7 & 0 \\ 0 & 5 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}}_{\tilde{V}'} \quad (\text{SVD version 2}). \end{aligned}$$

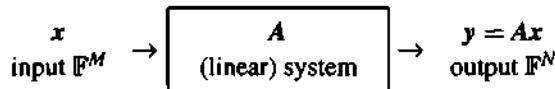
- More generally, if we have an SVD of A of the form $A = \sum_i \sigma_i u_i v_i'$ and $s_i = \pm 1$, then another SVD is

$$A = \sum_i \sigma_i \underbrace{(s_i u_i)}_{\tilde{u}_i} \underbrace{(s_i v_i)''}_{\tilde{v}_i}. \quad (3.14)$$

We can even take each s_i to be arbitrary points around the unit circle in the complex plane, $s_i = e^{i\phi_i}$, to arrive at an uncountably infinite collection of possible SVDs of any matrix, because $s_i s_i^* = 1$.

3.4 The Matrix 2-Norm or Spectral Norm

Moving towards one (of many) applications of the SVD, we now ask: What unit norm “input vector” x produces an “output vector” Ax having the “largest” output (as defined by the norm, or energy)? In other words, find a unit norm x_* such that $\|Ax_*\|_2 \geq \|Ax\|_2$ for all unit norm x . Note the “systems” language here. Very often when we talk about $y = Ax$ we also think about a system block diagram like



In this setting, we are thinking of A as an operation, not as data.

Expressing the question mathematically (an important skill to develop):

$$x_* = \arg \max_{x \in \mathbb{R}^N : \|x\|_2=1} \|Ax\|_2, \quad \text{where } \|x\|_2 = \sqrt{\langle x, x \rangle} = \sqrt{x'x}. \quad (3.15)$$

This is the first of *many* optimization problems formulated and solved in this book. The notation $\arg \max$ here means the argument that maximizes the expression to its right.

Fact 3.8 $x_* = v_1$, the first right singular vector of A (having the largest singular value σ_1).

Another maximizer is $x_* = e^{i\phi} v_1$, for any $\phi \in \mathbb{R}$, where $A = U\Sigma V'$.

Proof. Because V is orthogonal (or unitary), we can write x in terms of the V basis as $x = Vz$, where $z = V'x$ are the coefficients. Note that $\|x\|_2 = 1 \iff \|z\|_2 = 1$. Thus, $Ax = U\Sigma V'x = U\Sigma V'Vz = U\Sigma z$, so $\|Ax\|_2 = \|U\Sigma z\|_2 = \|\Sigma z\|_2 = \sqrt{\sum_{k=1}^r \sigma_k^2 |z_k|^2} \leq \sqrt{\sigma_1^2 \sum_{k=1}^r |z_k|^2} = \sigma_1 \|z\|_2 = \sigma_1$, where $r = \min(M, N)$. So $\|x\|_2 = 1 \implies \|Ax\|_2 \leq \sigma_1$. This gives an upper bound on the norm of the output. But we can achieve that upper bound by choosing $x = v_1$, because then $z = e_1$ and $Ax = \sigma_1 u_1$ so $\|Ax\|_2 = \|\sigma_1 u_1\|_2 = \sigma_1$. \square

Definition This property of a matrix is called the **matrix 2-norm**, or **spectral norm**, or **operator norm**:

$$\|A\|_2 \triangleq \max_{x: \|x\|_2=1} \|Ax\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_1. \quad (3.16)$$

◆ This is also called a variational characterization of σ_1 , because it expresses σ_1 as the solution to an optimization problem. The other singular values also have variational formulations by the Courant–Fischer min–max principle. See Problem 4.15.

This property must be important because it has multiple names and ways of writing it! By definition, the spectral norm of a matrix A gives a tight upper bound on how much the Euclidean norm of a vector can be scaled when multiplying by A :

$$\|Ax\|_2 \leq \|A\|_2 \|x\|_2 = \sigma_1 \|x\|_2 \quad \forall x \in \mathbb{F}^N. \quad (3.17)$$

It is a tight bound because there is no smaller upper bound; in fact, this upper bound is achieved when $x = e^{i\phi} v_1$. This important SVD property has practical applications for power maximization, shown next.

Fact 3.9 The solution $x_* = v_1$ is unique up to within a phase factor $e^{i\phi}$ iff $\sigma_1 > \sigma_2$.

To see why it is not unique when $\sigma_1 = \sigma_2$, see Example 3.14.

One application of (3.15) is optical imaging through materials like yogurt or human skin that scatter light so much that they seem completely opaque [41]. However, by using (3.15) to optimize the parameters of spatial light modulators, researchers have shown that light can be focused through such materials [42]. Such methods may aid noninvasive imaging of tissue inside the body in the future. Like most imaging applications, machine learning methods are also used; see, for example, [43].

3.4.1 Optimization: min versus arg min

Careful readers will have noticed that (3.15) used “arg max” whereas (3.16) used “max.” It is crucial to understand the difference.

Figure 3.5 illustrates the difference between “arg min” and “min” for an optimization problem involving $f: \mathcal{V} \rightarrow \mathbb{R}$: arg min gives the vector $\hat{x} \in \mathcal{V}$ that is the minimizer of the function $f(x)$; min gives the lowest function value (a real number). The distinction is analogous for maximization problems.

Explore 3.6 For $f(x) = 5 + |x - 3|$, determine $\arg \min_{x \in \mathbb{R}} f(x)$ and $\min_{x \in \mathbb{R}} f(x)$.

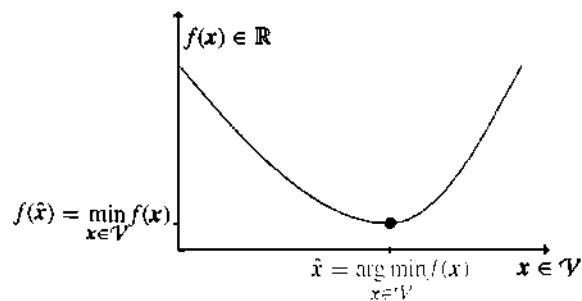


Figure 3.5 Illustrating min versus arg min.

Example 3.13 Multi-input multi-output (MIMO) communications with multi-antenna systems

Consider a system with N transmit antennas and M receive antennas, such as in 802.11n Wi-Fi. Let a_{ij} denote the (complex) gain between the j th transmit antenna and the i th receive antenna. (Matrix A depends on the amplifier and receiver properties and the multipath wave propagation between them.)

The goal is to design transmission amplitudes so that the received signal has the largest possible signal-to-noise ratio (SNR). For some signal we want to transmit, we use amplitudes $\mathbf{x} = (x_1, \dots, x_N)$ for the N transmit antennas. There are transmit power limits (amplifier hardware and allowable interference) so we constrain the input amplitudes: $\|\mathbf{x}\| \leq 1$.

After transmission, the signals received by the M antennas will have amplitudes $\mathbf{y} = A\mathbf{x}$. To maximize SNR, we want to design \mathbf{x} to make the received signal energy $\|\mathbf{y}\|$ as large as possible. (The background noise power is independent of \mathbf{x} .)

This problem has the form $\arg \max_{\mathbf{x}: \|\mathbf{x}\| \leq 1} \|\mathbf{Ax}\|$, so a solution is $\mathbf{x} = v_1$, the first right singular vector (see Problem 3.14).

In practice, the designer of a Wi-Fi system does not know the A for your place, and in fact A changes if you rearrange the furniture or relocate your computer. So the Wi-Fi router must determine A “on the fly” and this process is called channel estimation. The basic idea is that the transmitter first sends N orthonormal training waveforms $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{C}^N$, called pilot signals, to the receiver. The receiver records the corresponding outputs $y_1, \dots, y_N \in \mathbb{C}^M$, where $y_n = Ax_n$. Writing as a matrix:

$$[y_1 \ \cdots \ y_N] = A[\mathbf{x}_1 \ \cdots \ \mathbf{x}_N] \implies Y = AX.$$

Because X is unitary by design, we can estimate A as $A = YX^{-1} = YX'$. The receiver must know what pilot signals X are transmitted; this specification is part of the protocol. Once the router has determined A , it can compute its SVD and use v_1 as the best transmit amplitude vector. This process is related to the topic known as beamforming.

3.4.2 Eigenvalues as Optimization Problems

The equality in (3.16) expresses the first singular value σ_1 of an arbitrary matrix A as the solution of an optimization problem:

$$\sigma_1 = \max_{\mathbf{x}: \|\mathbf{x}\|_2=1} \|\mathbf{Ax}\|_2 = \max_{\mathbf{x}: \|\mathbf{x}\|_2=1} \sqrt{\mathbf{x}'A'A\mathbf{x}}. \quad (3.18)$$

For $\sigma_{\min(M,N)}$, see (3.23). Can we also express any eigenvalue of a square matrix as some optimization problem? The answer is yes, at least for Hermitian matrices.

Fact 3.10 If A is an $N \times N$ Hermitian matrix, with eigenvalues ordered as $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$, then

$$\lambda_1 = \max_{\mathbf{x}: \|\mathbf{x}\|_2=1} \mathbf{x}'A\mathbf{x}, \quad \lambda_N = \min_{\mathbf{x}: \|\mathbf{x}\|_2=1} \mathbf{x}'A\mathbf{x}. \quad (3.19)$$

This is sometimes called the Rayleigh–Ritz theorem, albeit perhaps dubiously [44, p. 655].

Under the same assumptions, we can rewrite (3.19) in a form called the Rayleigh quotient as follows:

$$\lambda_N \leq \frac{\mathbf{x}' \mathbf{A} \mathbf{x}}{\mathbf{x}' \mathbf{x}} \leq \lambda_1 \quad \forall \mathbf{x} \in \mathbb{F}^N. \quad (3.20)$$

Explore 3.7 Generalize (3.19) to normal matrices, or provide a counterexample.

◆ A generalization of (3.19) (under the same assumptions) is [45]:

$$\begin{aligned} \sum_{k=1}^K \lambda_k &= \max_{(\mathbf{x}_1, \dots, \mathbf{x}_K) \in \mathcal{X}_K} \sum_{k=1}^K \mathbf{x}_k' \mathbf{A} \mathbf{x}_k, \\ \sum_{k=N-K+1}^N \lambda_k &= \min_{(\mathbf{x}_1, \dots, \mathbf{x}_K) \in \mathcal{X}_K} \sum_{k=N-K+1}^N \mathbf{x}_k' \mathbf{A} \mathbf{x}_k, \end{aligned} \quad (3.21)$$

where \mathcal{X}_K denotes the collection of all possible sets of K orthonormal vectors in \mathbb{F}^N .

For an $N \times N$ Hermitian matrix \mathbf{A} with eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$ the min–max theorem expresses all eigenvalues as an optimization problem involving the Rayleigh quotient:

$$\lambda_k = \min_{\mathbf{U} \in \mathbb{F}^{N \times k}} \max_{\mathbf{x} \in \mathcal{R}(\mathbf{U}) \setminus \{\mathbf{0}\}} \frac{\mathbf{x}' \mathbf{A} \mathbf{x}}{\|\mathbf{x}\|_2^2}, \quad k = 1, \dots, N. \quad (3.22)$$

For other eigenvalue and generalized eigenvalue problems put in optimization form, see [46].

3.4.3 Smallest Singular Value

The smallest singular value of an $M \times N$ matrix \mathbf{A} also corresponds to an optimization problem when $N \leq M$:

$$\sigma_N = \min_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{A} \mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \min_{\mathbf{x}: \|\mathbf{x}\|_2=1} \|\mathbf{A} \mathbf{x}\|_2. \quad (3.23)$$

When \mathbf{A} is wide, that is, when $N > M$, the columns of \mathbf{A} are linearly dependent (see Section 4.2.2), so there is a unit norm vector \mathbf{x} for which $\mathbf{A} \mathbf{x} = \mathbf{0}$, so the above minimization problem simplifies to 0, even if $\sigma_N > 0$. Now focus on the (tall or square) case where $N \leq M$. Then

$$\begin{aligned} \|\mathbf{A} \mathbf{x}\|_2^2 &= \left\| \sum_{k=1}^N \sigma_k u_k v_k' \mathbf{x} \right\|^2 = \left\| \sum_{k=1}^N (\sigma_k v_k' \mathbf{x}) u_k \right\|^2 \\ &= \sum_{k=1}^N \|(\sigma_k v_k' \mathbf{x}) u_k\|_2^2 \quad (\text{by Pythagoras' theorem}) \\ &= \sum_{k=1}^N \sigma_k^2 |v_k' \mathbf{x}|^2 \|\mathbf{u}_k\|_2^2 = \sum_{k=1}^N \sigma_k^2 |v_k' \mathbf{x}|^2 \geq \sum_{k=1}^N \sigma_N^2 |v_k' \mathbf{x}|^2 \end{aligned}$$

$$\begin{aligned}
 &= \sigma_N^2 \sum_{k=1}^N |\nu_k' x|^2 = \sigma_N^2 \|V' x\|_2^2 \\
 &= \sigma_N^2 \|x\|_2^2 \quad \text{by unitary invariance, i.e., (2.38).}
 \end{aligned}$$

Combining leads to the inequalities

$$\|x\|_2 \cdot \begin{cases} \sigma_N & (N \leq M) \\ 0 & (M < N) \end{cases} \leq \|Ax\|_2 \leq \sigma_1 \|x\|_2 \quad \forall x \in \mathbb{F}^N. \quad (3.24)$$

When $N \leq M$, the lower bound is achieved for $x = v_N$. This solution is used to find the projective transformation that relates different camera views of a scene via a homographic transformation [47].

3.5 Relating SVDs and Eigendecompositions

The two big topics of this chapter are SVDs and eigendecompositions. Are they related? Every matrix, even if rectangular, has an SVD, whereas only some square matrices have an eigendecomposition. Nevertheless, we can find some relationships.

Fact 3.11

- Any right singular vector of A is an eigenvector of $A'A$, because $A'A v_k = A'(\sigma_k u_k) = \sigma_k^2 v_k$.
- Any left singular vector of A is an eigenvector of AA' , similarly.
- The $\min(M, N)$ singular values of a matrix $A \in \mathbb{F}^{M \times N}$ are the sorted (principal) square roots of the eigenvalues of $A'A$ or AA' , whichever is the smaller matrix.

Proof. For the case where A is tall with SVD $A = U\Sigma V'$,

$$A'A = V\Sigma'U'U\Sigma V' = V \underbrace{\Sigma' \Sigma}_{\Lambda} V' = V \underbrace{\begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_N^2 \end{bmatrix}}_{\Lambda} V',$$

which has (nonnegative) eigenvalues $\{\sigma_k^2\}$. So the square roots of those eigenvalues of $A'A$ (properly sorted) are the singular values of A .

This is about all we can say to relate SVDs and eigendecompositions for general (rectangular) matrices, so next we turn to square matrices.

In general, if A is an arbitrary square matrix, there is not too much one can say to relate its eigenvalues and its singular values. One known relationship is Weyl's inequality:

$$|\lambda_1(A) \cdots \lambda_K(A)| \leq \sigma_1(A) \cdots \sigma_K(A), \quad 1 \leq K \leq N, \quad (3.25)$$

assuming that we order the eigenvalues of A so that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_N|$ [48]. Probably the most useful inequality is the case $K = 1$, which says $|\lambda_1| \leq \sigma_1$.

In general, if A is an arbitrary diagonalizable square matrix, then its eigendecomposition $A = V\Lambda V^{-1}$ is *unrelated* to any SVD of A . However, for normal matrices, we can relate any eigendecomposition of A to an SVD of A . If A is a normal $N \times N$ matrix (e.g., a Hermitian symmetric), then it has a unitary eigendecomposition

$$A = V\Lambda V' = \sum_{n=1}^N \lambda_n v_n v_n'.$$

Without loss of generality, we can order the eigenvalues with decreasing magnitudes, that is, $|\lambda_1| \geq \cdots \geq |\lambda_N|$. Specifically, define P to be the permutation matrix that orders the eigenvalues that way. Then rewrite the eigendecomposition of A as follows:

$$A = V\Lambda V' = \underbrace{V}_{\tilde{V}} \underbrace{\Lambda}_{\tilde{\Lambda}} \underbrace{PAP'}_{\tilde{V}'} (\underbrace{V'}_{\tilde{V}'})'.$$

For the rest of this subsection, assume such reordering already took place.

Even with such reordering, $V\Lambda V'$ is still not an SVD of A in general because some of the eigenvalues λ_n can be negative or even complex, so $\Sigma \neq \Lambda$. To construct an SVD of A in terms of V and Λ , we use the fact that $\lambda_n = \text{sign}(\lambda_n) |\lambda_n|$, where $\text{sign}(z) e^{i\angle z} = e^{i\angle z}$ for $z \in \mathbb{C}$, as follows:

$$\begin{aligned} A &= V\Lambda V' = \sum_{n=1}^N \lambda_n v_n v_n' = \sum_{n=1}^N \underbrace{|\lambda_n|}_{\sigma_n} \underbrace{\text{sign}(\lambda_n) v_n}_{u_n} v_n' \\ &= \underbrace{U}_{\Sigma} \underbrace{S'\Lambda}_{V'} V', \quad S \triangleq \text{Diag}\{\text{sign}(\lambda_n)\}. \end{aligned} \quad (3.26)$$

The diagonal matrix S is unitary; thus, $U = VS$ is unitary.

So when A is normal with eigenvectors V and eigenvalues Λ with descending magnitudes, an SVD of A is:

$$A = U\Sigma V', \quad U = VS, \quad S \triangleq \text{Diag}\{\text{sign}(\lambda_n)\}, \quad \Sigma = S'\Lambda = \text{Diag}\{|\lambda_n|\}. \quad (3.27)$$

This SVD is not unique; we could have associated some or all of the sign values with V , for example.

The construction (3.26) shows that if A is normal, then for any unitary eigendecomposition of A of the form $A = V\Lambda V'$, we can construct an SVD of the form (3.27) where the matrix V of right singular vectors consists entirely of eigenvectors of A , and where the matrix U of left singular vectors also consists entirely of eigenvectors of A , because in (3.26) we have $u_n = \text{sign}(\lambda_n) v_n$ so u_n is also an eigenvector of A .

In short, if A is normal than we can construct an SVD of A using any orthonormal set of eigenvectors of A . However, it does *not* follow that all singular vectors of a normal matrix A are eigenvectors!

Example 3.14 Consider $A = \begin{bmatrix} 3 & 0 \\ 0 & -3 \end{bmatrix}$ and $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. This x is a right singular vector because $A'Ax = 9x$. However, this x is not an eigenvector of A . To elaborate, here are two different SVDs of A , one that is constructed from a set of eigenvectors of A , and one that is not:

$$\begin{aligned}
 A &= \begin{bmatrix} 3 & 0 \\ 0 & -3 \end{bmatrix} && (3.28) \\
 &= \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_V \underbrace{\begin{bmatrix} 3 & 0 \\ 0 & -3 \end{bmatrix}}_A \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{V'} && (\text{an elementary eigendecomposition}) \\
 &= \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}}_U \underbrace{\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}}_\Sigma \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{V'} && (\text{SVD v1, using } V \text{ and } u_n = \text{sign}(\lambda_n)v_n) \\
 &= \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}}_{\tilde{U}} \underbrace{\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}}_\Sigma \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}_{\tilde{V}'} && (\text{SVD v2, unrelated to eigenvectors of } A).
 \end{aligned}$$

(3.29)

Fact 3.12 If A is normal and has eigenvalues with distinct magnitudes, or if eigenvalues with equal magnitudes have equal values, then for every SVD of A , the left and right singular vectors are all eigenvectors of A . See Problem 3.17.

3.5.1 When Does $U = V$?

A question about the SVD $A = U\Sigma V'$ that sometimes arises is: When does $U = V$? This is an imprecise question because U and V are not unique. A more precise version is this: For what class of matrices does there exist an SVD in which $U = V$?

First, if $N \neq M$ then U and V have different sizes. So we focus here on the square case where $N = M$. If $A = V\Sigma V'$ then clearly A is Hermitian symmetric. But is symmetry a sufficient condition for possibly having $U = V$? No.



Recall from (3.3) that we can write any (Hermitian) symmetric matrix as $A = V\Lambda V'$ where Λ is the diagonal of the eigenvalues. This looks a lot like an SVD with $U = V$. However, for the SVD we always have $\sigma_k \geq 0$ whereas the eigenvalues of (symmetric) A are real but not necessarily nonnegative.

So, to have $U = V$ we need A to be Hermitian symmetric *and* to have nonnegative eigenvalues.

Fact 3.13 An SVD of A can have $U = V$ iff A is square, $A = A'$, and all eigenvalues of A are nonnegative.

Refer to (3.26) for the key idea.

Example 3.15 Consider the eigendecomposition of the following matrix and the following three (not unique!) SVD forms:

$$\begin{aligned}
 A &= \underbrace{\begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}}_V = \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}}_U \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}}_{\Lambda} \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}}_{V'} \quad (\text{eigendecomposition}) \\
 &= \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}}_U \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}}_{\Sigma} \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}}_{V'} \quad (\text{SVD version 1}) \\
 &= \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_{\tilde{U}} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\tilde{V}'} \quad (\text{SVD version 2}) \\
 &= \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\tilde{U}} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_{\tilde{V}'} \quad (\text{SVD version 3}).
 \end{aligned}$$

Even though A is symmetric, and we have exhibited three different SVDs for it, we cannot find an SVD here where \tilde{U} and V' are the same because A has a negative eigenvalue: $\det(A - zI) = z^2 - 4 \implies z = \pm 2$.

Explore 3.8 Which of the above forms corresponds to (3.26)?

3.5.2 SVD Computation Using Eigendecomposition

To find a (full) SVD by hand using an eigendecomposition, or if you were stuck on a desert island with a computer that had an `eigen` command but no `svd` command, here is how you could do it for the case of a tall $M \times N$ matrix with $M \geq N$ and rank N .

First, use the eigendecomposition of AA' to find \tilde{U} and Σ :

$$AA' = U\Lambda U' = U \underbrace{\Sigma \Sigma'}_{M \times M} U'. \quad (3.30)$$

Here, \tilde{U} will be the left singular vectors of A , and the $N \leq M$ singular values will be $\sigma_n = \sqrt{\lambda_n}$, $n = 1, \dots, N$. Now obtain V' as follows by multiplying A on the left by $\text{Diag}(1/\sigma_n) U[:, 1:N]'$:

$$\begin{aligned}
 \text{Diag}(1/\sigma_n) U[:, 1:N]' A &= \text{Diag}(1/\sigma_n) U[:, 1:N]' (U \Sigma V') \\
 &= \text{Diag}(1/\sigma_n) [I \ 0] \Sigma V' = \text{Diag}(1/\sigma_n) \text{Diag}(\sigma_n) V' = V'.
 \end{aligned} \quad (3.31)$$

Unfortunately, this process does not work when A is not full rank (see Section 4.3), and it requires an eigendecomposition of the “large” size $M \times M$ so it is impractical. Furthermore, it can be numerically unstable due to the $1/\sigma_n$ terms. A numerically stable approach involves a QR decomposition [49].

Example 3.16 Find an SVD of $A = \begin{bmatrix} 0 & 3 \\ 0 & 0 \end{bmatrix}$.

Recall that this (square but asymmetric) matrix does not have an orthogonal eigendecomposition.

$$A'A = \begin{bmatrix} 0 & 0 \\ 3 & 0 \end{bmatrix} \begin{bmatrix} 0 & 3 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 9 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 9 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = V\Sigma^2V'$$

$$\Rightarrow V = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{ and } \Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}.$$

$$AA' = \begin{bmatrix} 0 & 3 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 9 & 0 \\ 0 & 0 \end{bmatrix} = I_2 \begin{bmatrix} 9 & 0 \\ 0 & 0 \end{bmatrix} I_2 = U\Sigma^2U'$$

$$\implies U = I_2.$$

Note that to find V properly, we applied a permutation to have the eigenvalues of $A'A$ in descending order. And in general one would need to do more work to properly match the U and V ordering.

Thus, an SVD is

$$A = \underbrace{I_2}_U \underbrace{\begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_V.$$

Q3.8 Every permutation matrix has an SVD.

A: True

B: False

Q3.9 If $A = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 2 \end{bmatrix}$, then $\max_{x \in \mathbb{R}^3 : \|x\|=1} \|Ax\| = 5$.

Q3.10 The singular values of a permutation matrix are all equal to 1.

A: True

B: False

Q3.11 (A uniqueness question.) If $A = U\Sigma V'$ and $A = \tilde{U}\tilde{\Sigma}\tilde{V}'$ are both SVDs of A , then $\Sigma = \tilde{\Sigma}$.

A: True **B: False**

By definition, to determine if x is an eigenvector of a square matrix A , simply compute $y = Ax$ and see if $y = \alpha x$ for some $\alpha \in \mathbb{F}$. The following question explores this property further.

Q3.12 If x is an eigenvector of a normal matrix A having unitary eigendecomposition $A = V\Lambda V'$, then $x = \alpha v_j$ for some $\alpha \in \mathbb{F}$, where v_j is one of the columns of V .

A: True

B: False

For an $M \times N$ matrix A , how do we test if $v \in \mathbb{F}^N$ is a right singular vector or $u \in \mathbb{F}^M$ is a left singular vector?

- A right singular vector of A is an eigenvector of $A'A$.
- A left singular vector of A is an eigenvector of AA' .

Q3.13 If x is a right singular vector of a matrix A having SVD $A = U\Sigma V'$, then $x = \alpha v_j$ for some $\alpha \in \mathbb{F}$, where v_j is one of the columns of V .

A: True

B: False

3.5.3 SVD Nonuniqueness Revisited

Section 3.3.5 already demonstrated that an SVD is never unique because there can always be sign flips of the singular vectors. More generally, if there are any *repeated* singular values (including two or more zeros), then one can have rotations of the singular vectors, as illustrated in (3.29). Specifically, if there are any repeated singular values then there exist infinitely many nondiagonal unitary matrices Q leading to distinct SVDs of the form

$$A = U\Sigma V' = UQ\Sigma Q'V' = \tilde{U}\Sigma\tilde{V}'. \quad (3.32)$$

Example 3.17 The following diagonal matrix has numerous SVDs (and eigendecompositions):

$$\Sigma = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} = I\Sigma I = Q\Sigma Q', \quad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -q \sin \theta \\ 0 & \sin \theta & q \cos \theta \end{bmatrix}, \quad q \in \{\pm 1\}, \theta \in \mathbb{R}.$$

Explore 3.9 Prove the following converse of the above discussion. If the singular values of a matrix are distinct, then the only unitary matrices Q for which (3.32) holds are diagonal matrices Q where $|q_{ii}| = 1$, that is, “sign flips.”

Fact 3.14 If $A = U\Sigma V'$ has distinct singular values, then every right singular vector of A is a multiple of some column of V .

Proof. Suppose u, v is a singular vector pair of A . Then, by definition, $Av = \sigma_k u$ and $A'u = \sigma_k v$, where σ_k is one of the $\min(M, N)$ singular values of A . Combining, we have $\sigma_k^2 v = \sigma_k A'u = A'A v = V\Sigma'\Sigma V'v$. Thus, because V is unitary, $\sigma_k^2 V'v = \Sigma'\Sigma V'v$. Defining $z \triangleq V'v$, we have $\sigma_k^2 z = \Sigma'\Sigma z = \text{Diag}\{\sigma_1^2, \dots, \sigma_{\min(M,N)}^2\}z$. When the singular values are distinct, this equality is possible iff $z = \alpha e_k$ for some $\alpha \in \mathbb{F}$. Thus, $v = Vz = V(\alpha e_k) = \alpha v_k$. Similarly, $u = \alpha u_k$ for the same α . \square

3.6 Positive Semidefinite Matrices

In general, trying to relate eigendecompositions and SVDs is futile because usually we use the SVD for nonsquare matrices. As mentioned previously, when we do consider a square matrix, it is usually a Gram matrix $X'X$ or an outer-product matrix XX' . These matrices are not only symmetric, they are positive semidefinite.

Definition An $N \times N$ (square) Hermitian matrix A is positive semidefinite iff $x'Ax \geq 0$ for all $x \in \mathbb{C}^N$.

Definition A (square) Hermitian matrix A is positive definite iff $x'Ax > 0$ for all $x \neq 0$.

We write $A > 0$ to denote positive definite and $A \succeq 0$ to denote positive semidefinite (Loewner order).

Fact 3.15 If $A = BB'$ for any matrix B , then A is positive semidefinite.

Proof. $x'Ax = x'BB'x = \|B'x\|_2^2 \geq 0$. □

Q3.14 Which of the following statements is true?

- A: All positive semidefinite matrices are positive definite.
- B: All positive definite matrices are positive semidefinite.
- C: Neither statement is always true.

In words, any Gram matrix $X'X$ or outer-product matrix XX' is positive semidefinite; that is, $X'X \succeq 0$.

Fact 3.16 If $A = BB'$ for any matrix B , then $A = U\Sigma U'$ with $\Sigma_{ii} \geq 0$. In words, for such matrices an eigendecomposition with (real, nonnegative) eigenvalues in descending order is also an SVD.

Proof. Let $B = U\Sigma_B V'$ denote an SVD of B . Recall that Σ_B is real and nonnegative. Then $A = BB' = U\Sigma_B V'V\Sigma_B'U' = U\Sigma_B\Sigma_B'U' = U\Sigma U'$, where $\Sigma = \Sigma_B\Sigma_B'$ is diagonal with entries σ_k^2 (and some zeros if B is tall). So when $A = BB'$ the eigenvalues of A are the squares of the singular values of B (and some zeros if B is tall), and hence real and nonnegative. □

Example 3.18 Consider $B = \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ 1 & 0 \end{bmatrix} \implies A = BB' = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 9 & 0 \\ 1 & 0 & 1 \end{bmatrix}$.
`eigvals(A)` returns $(0, 2, 9)$. `svdvals(B)` returns $(3, \sqrt{2})$.



Note that `eigen` and `eigvals` do not return eigenvalues in descending magnitude order in general, whereas `svd` and `svdvals` always return singular values in descending order.

Q3.15 If A is Hermitian, then $A = BB'$ for some matrix B .

A: True

B: False

- ♦ Sylvester's criterion relates positive (semi)definiteness to properties of its principal minors [50].

3.6.1 Relating Positive (Semi)Definiteness to Eigenvalues

Let A be any $N \times N$ Hermitian matrix; then A has a unitary eigendecomposition of the form $A = V\Lambda V'$. Now, $x'Ax = x'V\Lambda V'x = z'\Lambda z = \sum_i \lambda_i |z_i|^2$, where $z = V'x$. Thus, if (Hermitian) A has all nonnegative eigenvalues, then $x'Ax \geq 0$ for all $x \in \mathbb{C}^N$, so $A \succeq 0$.

The converse is also true: If A is symmetric positive semidefinite, then A has non-negative eigenvalues.

Proof. If Hermitian matrix A has a negative eigenvalue λ_i , then let $x = Ve_i$ above and $x'Ax = \lambda_i < 0$, so A cannot be positive semidefinite. \square

Fact 3.17 Combining, a Hermitian matrix A is positive semidefinite iff all of its eigenvalues are nonnegative. Similarly, a Hermitian matrix A is positive definite iff all of its eigenvalues are positive.

To summarize, any positive semidefinite matrix has real and nonnegative eigenvalues, and it has an SVD that matches its eigendecomposition (with descending eigenvalue order), with $U = V$ and $\Sigma = \Lambda$, that is, $\sigma_n(A) = \lambda_n(A) \geq 0$. Another way of saying this is as follows.

Fact 3.18 A Hermitian matrix A is positive semidefinite iff we can write $A = BB'$ for some matrix B .

Proof. The “if” part is shown above. For the “only if” part, suppose A is $N \times N$ Hermitian and $A \succeq 0$. Then $A = V\Lambda V'$ with Λ being diagonal with nonnegative values. Now define $\Lambda^{1/2} = \text{Diag}\{\sqrt{\lambda_1}, \dots, \sqrt{\lambda_N}\}$. Then $A = V\Lambda^{1/2}\Lambda^{1/2}V' = BB'$, where $B = V\Lambda^{1/2}$. \square

Q3.16 This proof is also applicable when Λ contains real negative eigenvalues.

A: True

B: False

3.7 Summary

Letting $|\lambda_{(n)}|$ denote the n th largest magnitude eigenvalue of a square matrix, the Venn diagram in Fig. 3.6 summarizes the decompositions discussed so far. In practice, we usually end up using:

- an eigendecomposition, $A = V\Lambda V'$, when working with positive semidefinite matrices (like a Gram matrix or outer-product matrix);

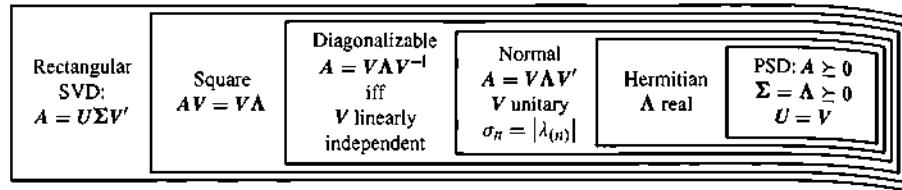


Figure 3.6 Matrix decomposition summary.

- the SVD, $A = U\Sigma V'$, for most other matrices.

A curious note about terminology:

- Columns of V are called the right eigenvectors for the eigendecomposition, because $AV = V\Lambda$, even though the eigendecomposition $A = V\Lambda V'$ has V on the left.
- Columns of U are called the left singular vectors for the SVD, because the SVD $A = U\Sigma V'$ has U on the left.

Solutions to Explorations

$$\text{Explore 3.1 } V'V = \begin{bmatrix} \cos \theta & \sin \theta \\ -q \sin \theta & q \cos \theta \end{bmatrix} \begin{bmatrix} \cos \theta & -q \sin \theta \\ \sin \theta & q \cos \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2.$$

Explore 3.2 When x is an eigenvector of A , because then $Ax = \lambda x$.

Explore 3.3 By inspection we can choose $U = R$, $\Sigma = I_2$, and $V = I_2$, for example.

Explore 3.4 No, A is nonsquare in general!

Explore 3.5 Choose the right singular vectors: $x = v_1$ and $z = v_2$.

Proof. $A = U\Sigma V' = \sigma_1 u_1 v_1' + \sigma_2 u_2 v_2'$ (a sum of outer products), so $Ax = Av_1 = \sigma_1 u_1$ and $Az = Av_2 = \sigma_2 u_2$ but $u_1 \perp u_2$, so $Ax \perp Az$. \square

Explore 3.6 For this simple case one can solve it by inspection or by plotting. We find $\arg \min_{x \in \mathbb{R}} f(x) = 3$ and $\min_{x \in \mathbb{R}} f(x) = 5$.

Explore 3.7 Normal matrices can have complex eigenvalues so they cannot be ordered like $\lambda_1 \geq \lambda_2 \geq \dots$. However, if A is normal, then $A = V\Lambda V'$ where V is unitary. Thus, by defining $x = V'z$ we have: $0 \leq |x'Ax|/x'x = |z'Az|/z'z = |\sum_i |z_i|^2 \lambda_i| / \sum_i |z_i|^2 \leq \max_i |\lambda_i|$. See (6.78).

Explore 3.8 SVD version 1, where $U = V \text{Diag}(\text{sign}(\lambda_i))$.

Explore 3.9 Multiply both sides of (3.32) by U' on the left and VQ on the right, assuming Q is unitary. Then we need to show that $\Sigma Q = Q\Sigma$ implies that Q must be diagonal with $|q_{ii}| = 1$. Looking at the (i,j) th element of both sides, we have $\sigma_i q_{ij} = \sigma_j q_{ij}$. When the singular values are distinct, this equality holds for $i \neq j$ if and only if $q_{ij} = 0$, showing that Q must be a diagonal matrix. All diagonal unitary matrices have $|q_{ii}| = 1$.

Problems

Problem 3.1 Let A be a diagonal matrix with real entries that are distinct and nonzero. Let x be a vector with all nonzero entries.

- (a) Determine how the eigenvalues of the rank-1 update $B = A + xx'$ are related to the eigenvalues of A and the vector x . Your final expression must not have any matrices in it.

Hint: They will be implicitly related (not in a closed form expression) as the solution of an equation involving the eigenvalues of A and the elements of x .

Hint: $G + H = G(I + G^{-1}H)$ if G is invertible, and see Problem 2.20.

- (b) Use your solution to the previous part to determine the eigenvalues of B when

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \text{ and } x = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}.$$

Hint: JULIA's Polynomials.jl package can be useful here.

Problem 3.2 Let $B = A + xx'$ denote a rank-1 update of an $N \times N$ matrix A , where

- A is diagonal with real entries and $a_{11} > a_{22} > \dots > a_{NN}$;
- $x_i > 0$ so that, per Problem 3.1, the eigenvalues of B differ from those of A .

- (a) Determine the eigenvectors of B in terms of x , A , and the eigenvalues of B found in Problem 3.1.

- (b) Prove that the eigenvectors are orthogonal.

Hint: A simple partial fraction expansion may be helpful.

Problem 3.3 A matrix B is antisymmetric iff $B' = -B$. Show that the eigenvalues of an antisymmetric B are purely imaginary.

Problem 3.4 Prove or disprove (by a counterexample) the following statement. If T is any matrix for which $T^k = I$ for some natural number $k > 1$, then T is normal.

Problem 3.5 When A is 3×3 and symmetric, it has six degrees of freedom (DoF; the upper triangular elements). Now $A = V\Lambda V'$, where V and Λ are both 3×3 matrices. Explain the DoF in terms of V and Λ .

Problem 3.6 Let $B = T^{-1}AT$ for an invertible matrix T ; that is, A and B are similar. If A is diagonalizable, that is, $A = V\Lambda V^{-1}$, then find an eigendecomposition of B that uses unit norm eigenvectors.

Problem 3.7 Given an $N \times N$ matrix A and initial vector $x_0 \in \mathbb{F}^N$, the power iteration uses the iterative recursion $x_{k+1} = Ax_k / \|Ax_k\|_2$. Assume A is an $N \times N$ Hermitian symmetric matrix with distinct (in magnitude) eigenvalues, ordered such that $\lambda_N < \lambda_{N-1} < \dots < \lambda_2 < \lambda_1$. (Section 8.5 considers a more general situation.) Under these assumptions, the sequence $\{x_k\}$ converges (ignoring phase factors) to v_1 , the leading eigenvector of A associated with its largest (in magnitude) eigenvalue, if $v_1'x_0 = [V'x_0]_1 \neq 0$.

- (a) Suppose that λ_1 is known. Describe how to use *one* run of the power iteration (possibly with a modified input matrix) to compute the eigenvector v_N associated with the *smallest* eigenvalue of A , namely λ_N , assuming it is unique. Here we mean smallest value, *not* smallest magnitude.

(Do not invert A because that is too expensive in large applications.)

Hint: What simple transformation of A maps its smallest eigenvalue to its largest (in magnitude)? Remember that eigenvalues can be negative so it is possible that $|\lambda_1| < |\lambda_N|$.

- (b) Describe a simple way to compute λ_N from the v_N result of part (a). The eigenvector v_N is not unique because it could be scaled by $e^{i\phi}$. Does that “sign ambiguity” affect your calculation of λ_N ?
- (c) Describe how to modify the scheme to find v_N if λ_1 is unknown.
(You may apply the power iteration more than once in this case, but do not invert A . Use as few applications of the power iteration as possible.)

Problem 3.8 Let $X \in \mathbb{F}^{M \times N}$. Using an SVD only, show that if $X'X = 0$, then $X = 0$.

Problem 3.9 The Frobenius norm of an $M \times N$ matrix A is defined as

$$\|A\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2} = \sqrt{\sum_{i=1}^M \sum_{j=1}^N |a_{ij}|^2}.$$

Express the Frobenius norm of A in terms of its singular values.

Hint: First express $\|A\|_F$ in terms of the matrix $A'A$.

Problem 3.10 Suppose that $X = U_x \Sigma_x V_x'$ and $Y = U_y \Sigma_y V_y'$ denote SVDs of the matrices $X \in \mathbb{F}^{m \times n}$ and $Y \in \mathbb{F}^{p \times q}$.

- (a) Show that $X \otimes Y = (U_x \otimes U_y)(\Sigma_x \otimes \Sigma_y)(V_x \otimes V_y)'$ is an SVD of $X \otimes Y$ (up to a permutation of the singular values). Note that the matrix $\Sigma_x \otimes \Sigma_y$ is roughly a diagonal matrix containing the pairwise products of the singular values of X and Y .

Hint: First show that the formula is correct, then argue that it is an SVD by showing that $U_x \otimes U_y$ and $V_x \otimes V_y$ are unitary matrices.

- (b) Now suppose that $A = Q_a \Lambda_a Q_a'$ and $B = Q_b \Lambda_b Q_b'$ are unitary eigendecompositions of the (square Hermitian) matrices $A \in \mathbb{F}^{N \times N}$ and $B \in \mathbb{F}^{M \times M}$. Show that

$$A \otimes B = (Q_a \otimes Q_b)(\Lambda_a \otimes \Lambda_b)(Q_a \otimes Q_b)'$$

is a unitary eigendecomposition of $A \otimes B$.

Hint: First show that the formula is correct, then argue that it is an eigendecomposition by showing that $Q_a \otimes Q_b$ is a unitary matrix.

- (c) Continuing (b), show that $A \oplus B = (Q_a \otimes Q_b)(\Lambda_a \oplus \Lambda_b)(Q_a \otimes Q_b)'$ is an eigendecomposition of the Kronecker sum $A \oplus B \triangleq (A \otimes I) + (I \otimes B)$, where you must determine the size(s) of the “ I ” in that expression. Here the matrix $\Lambda_a \oplus \Lambda_b$ is diagonal and contains the pairwise sums of the eigenvalues of A and B .

Hint: Start by writing $I_N = Q_a I_N Q_a'$ and $I_M = Q_b I_M Q_b'$ in the definition of $A \oplus B$, and then apply (b).

Problem 3.11 Let $A = U \Sigma V' = \sum_{k=1}^r \sigma_k u_k v_k'$ denote an SVD of $M \times N$ matrix A having rank r , where u_k and v_k denote the k th columns of U and V respectively, and σ_k is the (k, k) entry of Σ . For each expression below, write an equivalent expression in terms of the SVD components like U , Σ , V , u_k , v_k , σ_k , and so on.

- (a)
 - (i) Av_i
 - (ii) $(Av_i)'(Av_j)$
 - (iii) $A'u_i$ when $i \in \{1, \dots, r\}$. What if $i \in \{r+1, \dots, M\}$?
 - (iv) $(A'u_i)'(A'u_j)$
 - (v) $\|Av_i\|_2^2$
- (b)
 - (i) $Av_i v_i'$
 - (ii) $A'u_i u_i'$ when $i \in \{1, \dots, r\}$. What if $i \in \{r+1, \dots, M\}$?
 - (iii) $\|Av_i v_i'\|_F$
 - (iv) $\|A'u_i u_i'\|_F$
 - (v) $\|Av_i u_i'\|_F$
 - (vi) $\|Av_i v_j'\|_F$
 - (vii) AA'
 - (viii) $A'A$
 - (ix) $\|A'A\|_F$
 - (x) $\|AA'\|_F$
- (c)
 - (i) For $1 \leq k \leq r$, $U[:, 1:k]'A$
 - (ii) For $1 \leq k \leq r$, $AV[:, 1:k]$
 - (iii) For $1 \leq k \leq M$, $U[:, 1:k]U[:, 1:k]'A$
 - (iv) For $1 \leq k \leq N$, $AV[:, 1:k]V[:, 1:k]'$
 - (v) For $1 \leq k \leq r$, $U[:, 1:k]'AV[:, 1:k]$

Problem 3.12 Let $A \in \mathbb{C}^{N \times N} = U \Sigma V'$ have rank N , that is, N nonzero singular values.

- (a) Express A^{-1} in terms of the SVD of A without using any inverse (\cdot) $^{-1}$.
- (b) Determine an SVD for A^{-1} .

Problem 3.13 Let \mathbf{A} be an $M \times N$ matrix with spectral norm σ_1 . Show the following bounds on the Frobenius norm:

$$\sigma_1 \leq \|\mathbf{A}\|_{\text{F}} \leq \sqrt{\min(M, N)} \sigma_1.$$

The operator norm of a matrix equals its largest singular value σ_1 . The norm of a matrix can be measured many ways. The above inequality shows that if the Frobenius norm is small, then the operator norm is as well. However, the operator norm being small does not guarantee that the Frobenius norm will be as well. (Think, for example, of the setting when M and N are very large).

Optional challenge: Is the upper bound tight?

Problem 3.14 Verify the second equality in (3.16), and confirm that

$$\arg \max_{\mathbf{x}: \|\mathbf{x}\|_2=1} \|\mathbf{Ax}\|_2 = \arg \max_{\mathbf{x}: \|\mathbf{x}\|_2=1} \|\mathbf{Ax}\|_2.$$

Problem 3.15

- (a) Determine how the eigenvalues and eigenvectors of

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}' & \mathbf{0} \end{bmatrix}$$

are related to singular values and singular vectors of $\mathbf{A} \in \mathbb{F}^{N \times N}$. In other words, if $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}'$, then express eigenvalues and eigenvectors of \mathbf{B} in terms of components of \mathbf{U} , Σ , and \mathbf{V} .

Hint: Start with some numerical experiments using self-generated \mathbf{A} matrices and form a conjecture. Note that \mathbf{B} is $2N \times 2N$. Think about combining \mathbf{u}_i and \mathbf{v}_i in various ways, such as $\mathbf{u}_i \pm \mathbf{v}_i$, $\begin{bmatrix} \pm \mathbf{u}_i \\ \pm \mathbf{v}_i \end{bmatrix}$, and $\begin{bmatrix} \pm \mathbf{v}_i \\ \pm \mathbf{u}_i \end{bmatrix}$.

- (b) Optional. Write a unitary eigendecomposition of \mathbf{B} in matrix form.
(c) Optional. Generalize to the case $\mathbf{A} \in \mathbb{F}^{M \times N}$ for $M \geq N$. This problem is related to one SVD numerical approach.

Problem 3.16 Let \mathbf{A} be an $N \times N$ Hermitian matrix with unitary eigendecomposition $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}'$, where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k \geq 0$ and $0 \geq \lambda_{k+1} \geq \dots \geq \lambda_N$ for some $1 < k < N$. Determine an SVD of \mathbf{A} in terms of the given components.

Problem 3.17 Let \mathbf{A} be a normal matrix of the (unnamed) type where each eigenvalue either has a different magnitude than all other eigenvalues, or has the same value as all other eigenvalues with its magnitude. In other words, having both $|\lambda_j| = |\lambda_i|$ and $\lambda_j \neq \lambda_i$ is not allowed. For example, \mathbf{A} might have eigenvalues $(3, 4i, 4i, -5)$ but cannot have eigenvalues $(3, 4, 4i, -5)$. Prove that every right singular vector of \mathbf{A} is also an eigenvector of \mathbf{A} . This problem finishes the story on relating SVD and eigendecomposition.

Problem 3.18 Use an SVD of a square matrix \mathbf{A} to describe a factorization $\mathbf{A} = \mathbf{QS}$ where \mathbf{Q} is unitary and \mathbf{S} is positive semidefinite. Express \mathbf{Q} and \mathbf{S} in terms of the SVD

components U , Σ , and V of A . Show that your Q and S satisfy the requirements. (This matrix decomposition is analogous to the polar form $z = r e^{i\theta}$ of a complex scalar z , where r is like S and $e^{i\theta}$ is like Q .)

Hint: $V'V = I$, so $XY = XIY = X V'V Y$ for compatible matrices.

Problem 3.19 Each of the categories shown in Fig. 3.6 is a *strict superset* of the categories nested within it. Provide example matrices A_1, \dots, A_5 that belong to each of these categories but *not* to the next category nested within it. Try to provide the simplest possible example in each case. For example, the matrix $A_1 = [0 \ 1]$ is rectangular, but not square. Now you do A_2, \dots, A_5 .

Hint: All the examples can be 1×1 or 2×2 , often with simple 0 and 1 elements.

Problem 3.20 We write $A \preceq B$ iff $B - A \succeq 0$, that is, iff $B - A$ is positive semidefinite. For $a > 0$, determine

$$\left\{ x \in \mathbb{R} : x \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \preceq \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} \right\}.$$

Problem 3.21

- (a) Let A be an $N \times N$ matrix that satisfies $x'Ax = 0 \forall x \in \mathbb{R}^N$. Prove that $A = 0$ or provide a counterexample.
- (b) Let A be an $N \times N$ matrix that satisfies $x'Ax = 0 \forall x \in \mathbb{C}^N$. Prove that $A = 0$ or provide a counterexample.

Problem 3.22 A real, symmetric matrix A is positive semidefinite iff all of its eigenvalues are nonnegative, that is, iff we can write $A = U\Sigma U'$ for U orthogonal and Σ diagonal with $\sigma_i \geq 0$. We write $A \succeq 0$. A useful property is that if $A \succeq 0$ and $B \succeq 0$, then $\text{trace}(AB) \geq 0$. Let $X, K, F_k \in \mathbb{R}^{N \times N}$ for $k = 1, \dots, K$; let $c, z \in \mathbb{R}^K$. Show that $\text{trace}(KX) \geq c^T z$, assuming the following:

- $X \succeq 0$;
- $\text{trace}(F_k X) = c_k$ for $k = 1, \dots, K$;
- $K - \sum_{k=1}^K z_k F_k \succeq 0$.

This result is known as weak duality for semidefinite optimization problems.

4 Subspaces, Rank, and Nearest-Subspace Classification

4.1 Introduction

An important operation in signal processing and machine learning is dimensionality reduction. There are many such methods, but the starting point is usually *linear* methods that map data to a lower-dimensional set called a subspace. When working with matrices, the notion of *dimension* is quantified by rank. This chapter reviews subspaces, span, dimension, rank, and null space. These linear algebra concepts may seem fairly abstract initially, but they are crucial to thoroughly understanding the SVD, a primary tool for the rest of the book (and beyond). The chapter concludes with a machine learning application, signal classification by nearest subspace, that builds on all the concepts of the chapter.

4.2 Subspaces

Definition For a vector space \mathcal{V} defined on a field \mathbb{F} , a nonempty subset $S \subseteq \mathcal{V}$ is called a subspace or linear subspace of \mathcal{V} iff

- S is closed under vector addition: $u, v \in S \implies u + v \in S$;
- S is closed under scalar multiplication: $v \in S$ and $\alpha \in \mathbb{F} \implies \alpha v \in S$.

Fact 4.1 A subspace S always includes the zero vector $\mathbf{0}$.

Proof. Because a field \mathbb{F} always includes the scalar 0, and because S is nonempty, it contains some vector v . Because S is closed under scalar multiplication, S contains the vector $0v$ which is the zero vector. \square

We follow [4, p. 9] and use the notation $S \subseteq \mathcal{V}$ to indicate that S is a subspace of \mathcal{V} , even though the symbol \subseteq often denotes a subset elsewhere.

When visualizing subspaces, think of lines or planes or hyperplanes going through the origin $\mathbf{0}$ (but keep in mind that $\{\mathbf{0}\}$ and \mathcal{V} are also subspaces). See Fig. 4.1 for an example in \mathbb{R}^3 .

Example 4.1 $S = \{\mathbf{0}\}$, where $\mathbf{0} \in \mathcal{V}$ for some vector space \mathcal{V} . This is the most minimalist and uninteresting subspace.

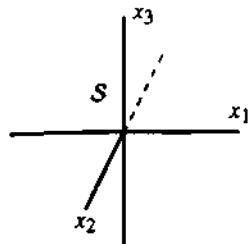


Figure 4.1 Subspace in \mathbb{R}^3 .

Example 4.2 $S = \{\alpha \mathbf{1}_N : \alpha \in \mathbb{C}\} \subseteq \mathbb{C}^N$. We will see shortly that here, $S = \text{span}(\mathbf{1}_N)$.

Example 4.3 The subspace of symmetric matrices

$S = \{A \in \mathbb{R}^{N \times N} : A \text{ is symmetric}\}$. Here, $\mathcal{V} = \mathbb{R}^{N \times N}$ and $S \subseteq \mathcal{V}$. It is easy to verify that S is closed under vector addition and scalar multiplication.

Example 4.4 The set of vectors orthogonal to some vector $v \in \mathcal{V}$, that is,

$S = \{x \in \mathcal{V} : \langle x, v \rangle = 0\}$. This set is a subspace because it is closed under vector addition and scalar multiplication:

- $x, z \in S \implies \langle x + z, v \rangle = \langle x, v \rangle + \langle z, v \rangle = 0 + 0 = 0 \implies x + z \in S$.
- $x \in S \implies \langle ax, v \rangle = a \langle x, v \rangle = a0 = 0 \implies ax \in S \forall a \in \mathbb{F}$.

Later in the chapter we will see that $S = (\text{span}(v))^\perp$.

Q4.1 Is the subset of orthogonal matrices $S = \{A \in \mathbb{R}^{N \times N} : A'A = I\}$ a subspace of $\mathbb{R}^{N \times N}$?

A: Y

B: N

C: Insufficient information

For a signal processing example, let \mathcal{V} denote the vector space (see Problem 4.3) consisting of all 1D periodic functions having period T for some given $T \neq 0$. In other words, if $f \in \mathcal{V}$, then $f(t+T) = f(t) \forall t \in \mathbb{R}$. Figure 4.2 shows two examples of vectors (signals) in this vector space.

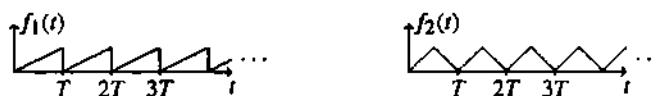


Figure 4.2 Examples of signals with period T .

Explore 4.1 Is the set of all 1D periodic functions a vector space?

The vector space \mathcal{V} above is infinite dimensional because $\forall k \in \mathbb{Z}$, $e^{i2\pi kt/T} \in \mathcal{V}$, and that set is linearly independent. In fact it is a basis, and that fact is the foundation of Fourier series.



4.2.1 Span

Definition Given a set of vectors $\{u_1, \dots, u_N\}$ in a vector space \mathcal{V} over field \mathbb{F} , the span of those vectors is

$$\text{span}(\{u_1, \dots, u_N\}) \triangleq \left\{ \sum_{n=1}^N \alpha_n u_n : \alpha_n \in \mathbb{F} \right\}. \quad (4.1)$$



This set is also called the linear span or hull. (Caution: it differs from the convex hull of a set.)

Often we will collect the vectors into a matrix $U = [u_1 \ \dots \ u_N]$ and define $\text{span}(U) \triangleq \text{span}(\{u_1, \dots, u_N\})$, although the usual mathematical definition is that the argument of $\text{span}(\cdot)$ is a set of vectors, not a matrix.

Fact 4.2 If $u_1, \dots, u_N \in \mathcal{V}$, then $\text{span}(\{u_1, \dots, u_N\})$ is a subspace of vector space \mathcal{V} (see Problem 4.4).

Example 4.5 For $\mathcal{V} = \mathbb{R}^3$, if $u_1 = (1, 0, 1)$ and $u_2 = (1, 0, -1)$ then $\text{span}(\{u_1, u_2\})$ is the entire (x, z) plane.

Example 4.6 For $\mathcal{V} = \mathbb{F}^{N \times N}$ (the vector space of $N \times N$ matrices), $\text{span}(\{e_1 e'_1, \dots, e_N e'_N\})$ is the subspace of all $N \times N$ diagonal matrices because

$$\begin{bmatrix} d_1 & \mathbf{0} \\ \mathbf{0} & d_N \end{bmatrix} = d_1 \begin{bmatrix} 1 & & \\ & 0 & \\ & & 0 \end{bmatrix} + \dots + d_N \begin{bmatrix} 0 & & \\ & 0 & \\ & & 1 \end{bmatrix} \\ = d_1 e_1 e'_1 + \dots + d_N e_N e'_N.$$

4.2.1.1 Span of an Infinite Collection of Vectors

♦♦ The definition of span in (4.1) is for a finite set of vectors. Some infinite-dimensional vector spaces are also important, such as the space of T -periodic functions above. Another example is the vector space of all polynomials. To work with such vector spaces, we use the following more general definition of span.

Definition If S is a (possibly uncountably infinite) subset of a vector space \mathcal{V} over field \mathbb{F} , the span of S consists of every (finite by definition) linear combination of elements in S :

$$\text{span}(S) \triangleq \left\{ x \in \mathcal{V} : x = \sum_{n=1}^N \alpha_n u_n, u_n \in S, \alpha_n \in \mathbb{F}, N \in \mathbb{N} \right\}. \quad (4.2)$$

Definition In a vector space \mathcal{V} , the span of the empty set \emptyset is the zero vector of that vector space:

$$\text{span}(\emptyset) \triangleq \mathbf{0} \in \mathcal{V}. \quad (4.3)$$

This definition follows from the convention that the sum of an empty collection is zero. Collectively, these definitions ensure that the span of *any* set (empty, finite, or infinite) is a subspace.

Q4.2 Let $\mathcal{V} = \mathbb{R}^3$ and consider the following (uncountably infinite) set:

$$\mathcal{S} = \{\alpha(1, 1, 1) : \alpha \in \mathbb{R}\} \cup \{(1, 0, 0)\}.$$

In words: \mathcal{S} is a line (through the origin) and another point not on that line. What is $\text{span}(\mathcal{S})$?

- A: A line B: A plane C: All of \mathbb{R}^3 D: None of these

Example 4.7 Consider the vector space \mathcal{V} of all polynomials. Now consider the (countably infinite) set of monomials with even powers: $\mathcal{S} = \{x^k : k = 0, 2, 4, \dots\}$. The span of this set \mathcal{S} is the subspace of polynomials having terms with even powers, that is, the subspace of even polynomials where $p(-x) = p(x)$.

4.2.2 Linear Independence

Definition A set of vectors u_1, \dots, u_N is linearly dependent iff there exists a tuple of coefficients $\alpha_1, \dots, \alpha_N \in \mathbb{F}$, not all of which are zero, where

$$\sum_{n=1}^N \alpha_n u_n = \mathbf{0}. \quad (4.4)$$

In words, a set of vectors is linearly dependent iff any one of the vectors is in the span of the other vectors. This latter interpretation generalizes to infinite-dimensional vector spaces. A set of vectors that is *not* linearly dependent is called a linearly independent set.

Definition A set of vectors u_1, \dots, u_N in a vector space is linearly independent iff, for any scalars $\alpha_1, \dots, \alpha_N \in \mathbb{F}$,

$$\sum_{n=1}^N \alpha_n u_n = \mathbf{0} \implies \alpha_1 = \dots = \alpha_N = 0. \quad (4.5)$$

In other words, no linear combination of the vectors is zero, except when all the coefficients are zero.

Example 4.8 In $\mathbb{R}^{N \times N}$, the set of matrices $\{e_1 e'_1, \dots, e_N e'_N\}$ is linearly independent.

- Any set of vectors containing a zero vector is a linearly dependent set.
- A set consisting of a single nonzero vector is a linearly independent set.
- Any set of orthonormal vectors is a linearly independent set.
- Any set of *nonzero* orthogonal vectors is a linearly independent set.

Example 4.9 The collection of vectors $\{v_1, v_2, v_3\}$ where

$v_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $v_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $v_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is an orthogonal set, but the set is linearly dependent because $7v_1 + 0v_2 + 0v_3 = \mathbf{0}$.

4.2.2.1 Generalization to Infinite Sets

◆◆ Recall from (2.59) that $X \setminus \{x\}$ denotes the set X with the vector x removed.

Definition A (possibly uncountably infinite) set of vectors X in a vector space is linearly dependent iff

$$\exists x \in X \text{ such that } x \in \text{span}(X \setminus \{x\}), \quad (4.6)$$

otherwise the set is called linearly independent.

Example 4.10 Consider the vector space of all polynomials. Let X denote the (countably infinite) set of all monomials, $X = \{f(x) = x^n : n = 0, 1, \dots\}$. One can show by elementary algebra that X is linearly independent. (One cannot write a monomial like x^5 as a linear combination of other monomials.)

4.2.2.2 Linear Independence and Invertibility

The following relationship between linear independence and invertibility is especially important for Chapter 5.

Fact 4.3 A set of vectors $u_1, \dots, u_N \in \mathcal{V}$ is linearly independent iff the $N \times N$ Gram matrix G is invertible, where the elements of G are all the inner products between vectors in the set: $g_{ij} = \langle u_i, u_j \rangle$, $i, j \in \{1, \dots, N\}$.

When $\mathcal{V} = \mathbb{F}^M$, then we can define the $M \times N$ matrix $A = [u_1 \ \dots \ u_N]$ and write $G = A'A$.

Proof (for the case $\mathcal{V} = \mathbb{F}^M$). If the column vectors of A are linearly dependent, then there is a nonzero vector $z \in \mathbb{F}^N$ such that $Az = \mathbf{0}$, so then $Gz = A'Az = A'\mathbf{0} = \mathbf{0}$, so G is not invertible.

Conversely, if G is not invertible, then there is a nonzero vector $z \in \mathcal{V}$ such that $\mathbf{0} = Gz$, implying that $0 = z'Gz = z'A'Az = \|Az\|_2^2$. Thus, $Az = \mathbf{0}$ and the columns of A are linearly dependent. \square

◆ **Example 4.11** Consider the vector space \mathcal{V} of polynomials defined on the interval $[0, 1]$ with the inner product in (6.19). For the (linearly independent) set of monomials $\{1, x^1, x^2, \dots, x^{N-1}\}$ in \mathcal{V} , the elements of the Gram matrix are

$$g_{ij} = \int_0^1 x^{i-1} x^{j-1} dx = \frac{1}{i+j-1}.$$

This invertible matrix is called the Hilbert matrix [51].

4.2.3 Basis

Now we use the concepts of linear independence and span to define a particularly important concept: basis.

Definition A set of vectors $\{\mathbf{b}_1, \mathbf{b}_2, \dots\}$ in a vector space \mathcal{V} is a basis for a subspace $\mathcal{S} \subseteq \mathcal{V}$ iff

- $\{\mathbf{b}_1, \mathbf{b}_2, \dots\}$ is a linearly independent set;
- $\text{span}(\{\mathbf{b}_1, \mathbf{b}_2, \dots\}) = \mathcal{S}$.

The vectors in a basis are called basis vectors. Very often we take $\mathcal{S} = \mathcal{V}$ in that definition to describe a basis for a vector space.

Example 4.12 The standard basis for \mathbb{F}^N is the set of N unit vectors $\{\mathbf{e}_1, \dots, \mathbf{e}_N\}$ because the set is linearly independent and if $\mathbf{x} \in \mathbb{F}^N$ we can write $\mathbf{x} = \sum_{n=1}^N x_n \mathbf{e}_n$, so $\text{span}(\{\mathbf{e}_1, \dots, \mathbf{e}_N\}) = \mathbb{F}^N$.

Example 4.13 The standard basis for $\mathbb{F}^{2 \times 2}$ is the following set of four 2×2 matrices:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

This set is linearly independent and we can express any 2×2 matrix as a linear combination of these four matrices, so their span is $\mathbb{F}^{2 \times 2}$.

Example 4.14 Euclidean space \mathbb{R}^N has many interesting bases used in signal processing, including those based on the DFT, the discrete cosine transform (DCT), and the orthogonal wavelet transform (OWT).

4.2.3.1 Properties of Bases

Fact 4.4 Basis vectors are always nonzero, because of the linear independence condition in the definition.

Fact 4.5 (Existence.) Every subspace in a vector space has a basis.

Proof (for $\mathcal{S} \subseteq \mathbb{F}^N$). If $\mathcal{S} = \{\mathbf{0}_N\}$ then the empty set \emptyset is a basis for \mathcal{S} because $\text{span}(\emptyset) = \mathbf{0}$ per (4.3).

Now consider a nonzero subspace \mathcal{S} . There is a nonzero vector $\mathbf{b}_1 \in \mathcal{S}$. If $\text{span}(\{\mathbf{b}_1\}) = \mathcal{S}$, then $\{\mathbf{b}_1\}$ is a basis for \mathcal{S} . Otherwise there is a nonzero vector $\mathbf{b}_2 \in \mathcal{S}$ that is not in $\text{span}(\mathbf{b}_1)$. If $\text{span}(\{\mathbf{b}_1, \mathbf{b}_2\}) = \mathcal{S}$, then $\{\mathbf{b}_1, \mathbf{b}_2\}$ is a basis for \mathcal{S} . Otherwise there is a nonzero vector $\mathbf{b}_3 \in \mathcal{S}$ that is not in $\text{span}(\{\mathbf{b}_1, \mathbf{b}_2\})$. If $\text{span}(\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}) = \mathcal{S}$, then $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ is a basis for \mathcal{S} . Otherwise continue as above. At each step, the vectors in the set are linearly independent. The process will terminate in $k \leq N$ steps, because any $N+1$ vectors in \mathbb{F}^N are linearly dependent. The resulting set $\{\mathbf{b}_1, \dots, \mathbf{b}_k\}$ is linearly independent and $\text{span}(\{\mathbf{b}_1, \dots, \mathbf{b}_k\}) = \mathcal{S}$. \square

Fact 4.6 Bases are *not unique* in general.

Nearly all vector spaces have multiple bases; the only exception is the trivial vector space $\mathcal{V} = \{\mathbf{0}\}$.

Example 4.15 If $\{\mathbf{b}_1, \mathbf{b}_2, \dots\}$ is a basis for $\mathcal{S} \subseteq \mathcal{V}$, then $\{-\mathbf{b}_1, -\mathbf{b}_2, \dots\}$ is also a basis for \mathcal{S} .

The definition of a basis ensures the following important representation property, which implies that a basis is a generalization of the usual concept of a coordinate system.

Fact 4.7 If $\{\mathbf{b}_1, \dots, \mathbf{b}_N\}$ is a basis for $\mathcal{S} \subseteq \mathcal{V}$ for $N \in \mathbb{N}$, then every vector $\mathbf{v} \in \mathcal{S}$ has a *unique* representation of the form

$$\mathbf{v} = \sum_{n=1}^N \alpha_n \mathbf{b}_n. \quad (4.7)$$

The coefficient vector $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} \in \mathbb{F}^N$ gives the coordinates of \mathbf{v} with respect to the basis.

When $\mathcal{V} = \mathbb{F}^M$, we can define the $M \times N$ basis matrix $\mathbf{B} = [\mathbf{b}_1 \ \dots \ \mathbf{b}_N]$ and write $\mathbf{v} = \mathbf{B}\boldsymbol{\alpha}$.

Proof (by contradiction) of the uniqueness of (4.7). For any $\mathbf{v} \in \mathcal{V}$, there exist some coordinates $(\alpha_1, \alpha_2, \dots)$ by the definition of span. Suppose the representation were not unique, that is, suppose there exist coefficients $(\beta_1, \beta_2, \dots)$ such that $\mathbf{v} = \sum_n \beta_n \mathbf{b}_n$, where at least one β_n differs from α_n . Then $\mathbf{0} = \mathbf{v} - \mathbf{v} = \sum_n \beta_n \mathbf{b}_n - \sum_n \alpha_n \mathbf{b}_n = \sum_n (\beta_n - \alpha_n) \mathbf{b}_n$. But because at least one coefficient in that sum is nonzero, that would imply that the set $\{\mathbf{b}_n\}$ is linearly dependent, contradicting the definition of a basis. \square

Fact 4.8 Every basis for a subspace has the same cardinality (i.e., number of elements, possibly infinite).

This fact is called the dimension theorem for vector spaces.

Example 4.16 The set $\{e^{i2\pi kt/T}, t \in \mathbb{R}: -K \leq k \leq K\}$ of complex exponential signals is a basis for all T -periodic signals that are band-limited with maximum frequency K/T . Another basis is the set $\{1, \cos(2\pi kt/T), \sin(2\pi kt/T): k = 1, \dots, K\}$. This fact about sinusoids (not proved here) is the foundation for additive synthesis musical sound generation. Both bases have cardinality $2K + 1$.

◆ The definition of basis above also generalizes to infinite-dimensional spaces. Simply replace $\{\mathbf{b}_1, \dots, \mathbf{b}_N\}$ with $\{\mathbf{b}_1, \mathbf{b}_2, \dots\}$ and allow an infinite sum in (4.7). Dealing rigorously with infinite sums requires care beyond our scope here.

Example 4.17 The set of monomials is a basis for the vector space of all polynomials.

Example 4.18 The following vector space is important in signal processing.

Q4.3 The vector space of sinusoids of frequency ν is

$$\mathcal{V} = \{A \cos(2\pi\nu t + \phi), t \in \mathbb{R}: A, \phi \in \mathbb{R}\}.$$

Is $\mathcal{S} = \{3 \cos(2\pi\nu t), 5 \sin(2\pi\nu t), 7 \cos(2\pi\nu t - \pi/4)\}$ a basis for \mathcal{V} ?

- A: Yes
 - B: No, because \mathcal{S} has linear dependence
 - C: No, because \mathcal{S} does not span \mathcal{V}
 - D: Both B and C
-

Q4.4 Let $\mathcal{V} = \mathbb{R}^3$ and $\mathcal{S} \triangleq \{(1, 1, 1), (1, 0, 0)\}$. What is $\text{span}(\mathcal{S})$?

- A: 0
- B: A line
- C: A plane
- D: All of \mathbb{R}^3
- E: None of these

Q4.5 The set of $N \times N$ upper triangular matrices is a subspace of $\mathbb{F}^{N \times N}$.

- A: True
- B: False

Q4.6 A basis for such upper triangular matrices in vector space $\mathbb{F}^{N \times N}$ contains how many vectors?

- A: N
- B: N^2
- C: $N^2/2$
- D: $N(N - 1)$
- E: None of these

4.2.4 Dimension

Definition The dimension of a subspace \mathcal{S} is the number of elements in any basis for \mathcal{S} .

This definition is well defined because every subspace has a basis, and, even though that basis is not unique in general, every basis has the same number of elements (possibly infinite), per Fact 4.8.

Example 4.19 $\dim(\mathbb{R}^N) = N$. Use the canonical or standard basis: $\{e_n : n = 1, \dots, N\}$.

Example 4.20 $\dim(\mathbb{F}^{M \times N}) = MN$. Use the canonical or standard basis:

$$\{e_m e'_n : m = 1, \dots, M, n = 1, \dots, N\}.$$

Explore 4.2 What is the dimension of the trivial vector space $\mathcal{V} = \{0\}$?

Q4.7 What is the dimension of the subspace in $\mathbb{F}^{N \times N}$ of $N \times N$ diagonal matrices?

- A: 1 B: N C: $2N$ D: N^2 E: ∞

Definition There are two types of subspaces (and vector spaces).

- A finite-dimensional (sub)space has a basis with $\dim \in \{0\} \cup \mathbb{N}$.
- Otherwise we call the (sub)space infinite dimensional.

Q4.8 What is the dimension of the vector space of polynomials with even powers?

- A: 1 B: 2 C: Infinite D: Undefined

4.2.4.1 Dimension of a Span and Bases

Fact 4.9 If $S = \text{span}(\{u_1, \dots, u_N\})$ then $\dim(S) \leq N$.

Fact 4.10 The basis reduction theorem states that either $\{u_1, \dots, u_N\}$ is a basis for $S = \text{span}(\{u_1, \dots, u_N\})$, or one can remove $N - \dim(S)$ appropriately chosen vectors from that set to form a basis for S .

Fact 4.11 The basis extension theorem [52, Theorem 5.3.7] states that every linearly independent set of N vectors in a finite-dimensional subspace S can be extended (with $\dim(S) - N$ additional vectors) to form a basis of S .

Fact 4.12 If S and T are both subspaces of a finite-dimensional vector space \mathcal{V} , then

$$S \subseteq T \implies \dim(S) \leq \dim(T). \quad (4.8)$$

For a proof of this that uses the basis extension theorem see [52, Theorem 5.4.4].

4.2.4.2 Dimensions of Vectors and Subspaces

The term “dimension” gets used for both vectors and subspaces, sometimes causing confusion at first. The following example clarifies the difference.

Example 4.21 Consider $x = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \end{bmatrix}$, $y = \begin{bmatrix} 3 \\ 6 \\ 0 \\ 0 \end{bmatrix}$. These vectors are both in the vector space $\mathcal{V} = \mathbb{R}^4$, that is, $x \in \mathcal{V}$ and $y \in \mathcal{V}$, so we say they are “four-dimensional vectors.” There is no `dim` function in JULIA, but `size(x)` returns the one-element tuple `(4,)`.

Define a set \mathcal{T} in terms of these two vectors: $\mathcal{T} = \{x, y\}$. This set is a subset of \mathcal{V} , which we write $\mathcal{T} \subset \mathcal{V}$. Because it is a set of two vectors, its cardinality is two: $|\mathcal{T}| = 2$.

Define \mathcal{S} to be the span of the set of vectors \mathcal{T} : $\mathcal{S} = \text{span}(\mathcal{T}) = \text{span}(\{x, y\})$. This \mathcal{S} is a subspace of \mathcal{V} and it is also a subset of \mathcal{V} . We write $\mathcal{S} \subseteq \mathcal{V}$.

Every vector in \mathcal{S} is an element of \mathcal{V} ; that is, every vector therein is a four-dimensional vector. However, a basis for \mathcal{S} consists of a single vector, say, x ; that is, $\mathcal{S} = \text{span}(\{x\})$. Thus, \mathcal{S} is a one-dimensional subspace, so we write $\dim(\mathcal{S}) = 1$. We must distinguish the dimension of the subspace from the dimension of the vectors within that subspace. Typically these dimensions differ; they are the same only in cases where $\mathcal{S} = \mathcal{V}$.

To further discuss dimension, we first define subspace sums.

4.2.5 Sums and Intersections of Subspaces

Definition If $\mathcal{S}, \mathcal{T} \subseteq \mathcal{V}$ then:

- the sum (or Minkowski sum) of two subspaces is defined as

$$\mathcal{S} + \mathcal{T} = \{s + t : s \in \mathcal{S}, t \in \mathcal{T}\}; \quad (4.9)$$

- the intersection of two subspaces is defined as

$$\mathcal{S} \cap \mathcal{T} = \{v \in \mathcal{V} : v \in \mathcal{S}, v \in \mathcal{T}\}. \quad (4.10)$$

Fact 4.13 If $\mathcal{S}, \mathcal{T} \subseteq \mathcal{V}$ then $\mathcal{S} + \mathcal{T}$ and $\mathcal{S} \cap \mathcal{T}$ are both subspaces of \mathcal{V} (see Problem 4.5).

Example 4.22 If \mathcal{S} is the subspace of upper Hessenberg matrices in $\mathbb{R}^{N \times N}$ and \mathcal{T} is the subspace of lower Hessenberg matrices in $\mathbb{R}^{N \times N}$ then $\mathcal{S} + \mathcal{T} = \mathbb{R}^{N \times N}$.

Q4.9 Considering the same \mathcal{S} and \mathcal{T} , what is $\mathcal{S} \cap \mathcal{T}$?

- A: \emptyset B: Diagonal matrices C: Tridiagonal matrices D: $\mathbb{R}^{N \times N}$ E: None of these



Caution: $\mathcal{S} + \mathcal{T}$ is *not* the same as the union of subspaces $\mathcal{S} \cup \mathcal{T}$.

Example 4.23 Consider $\mathcal{V} = \mathbb{R}^2$ and $\mathcal{S} = \text{span}(s)$, $s = (2, 1)$ and $\mathcal{T} = \text{span}(t)$, $t = (1, -1)$. Then the subspace sum $\mathcal{S} + \mathcal{T} = \mathbb{R}^2$ because $\{s, t\}$ spans \mathbb{R}^2 whereas the subspace union $\mathcal{S} \cup \mathcal{T}$ is just two lines, as shown in Fig. 4.3.

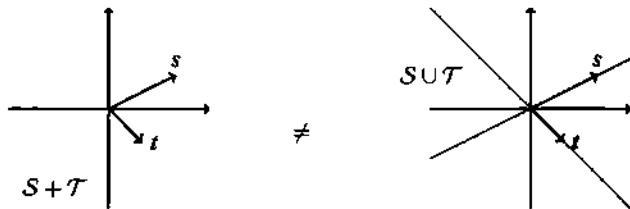


Figure 4.3 Subspace sum (left) and union (right).

Q4.10 A union of subspaces is a subspace.

- A: True B: False

4.2.6 Direct Sum of Subspaces

Now we define a particularly useful version of subspace sum.

Definition For subspaces \mathcal{S} and \mathcal{T} of vector space \mathcal{V} , we write the subspace sum $\mathcal{S} + \mathcal{T}$ as a direct sum,

$$\mathcal{S} \oplus \mathcal{T} \text{ iff } \mathcal{S} \cap \mathcal{T} = \mathbf{0}. \quad (4.11)$$

In this case we say \mathcal{S} and \mathcal{T} are complements of each other in the subspace $\mathcal{S} + \mathcal{T}$.

Example 4.24 $\mathcal{V} = \mathbb{R}^2$ and $\mathcal{S} = \text{span}(s)$, $s = (2, 1)$, and $\mathcal{T} = \text{span}(t)$, $t = (1, -1)$. Then $\mathcal{V} = \mathcal{S} \oplus \mathcal{T}$.

Q4.11 For $s, t \in \mathcal{V}$ with $s, t \neq \mathbf{0}$ and $t \neq \alpha s$ for all α , let $\mathcal{S} = \text{span}(s)$ and $\mathcal{T} = \text{span}(t)$. Is $\text{span}(\{s, t\}) = \mathcal{S} \oplus \mathcal{T}$?

- A: Always
 B: If and only if s and t are orthogonal
 C: If and only if s and t are orthonormal
 D: Never
 E: None of the above.

4.2.7 Dimensions of Sums of Subspaces

Here is yet another definition that is crucial for understanding the SVD.

Fact 4.14 If $\mathcal{U} = \mathcal{S} \oplus \mathcal{T}$ then:

- every $u \in \mathcal{U}$ can be written uniquely in the form $u = s + t$ for some $s \in \mathcal{S}$ and $t \in \mathcal{T}$;
- $\dim(\mathcal{U}) = \dim(\mathcal{S} \oplus \mathcal{T}) = \dim(\mathcal{S}) + \dim(\mathcal{T})$.

Proof of uniqueness. Suppose $u = s_1 + t_1 = s_2 + t_2$. Then

$$\underbrace{s_1 - s_2}_{\in \mathcal{S}} = \underbrace{t_2 - t_1}_{\in \mathcal{T}}.$$

But $\mathcal{S} \cap \mathcal{T} = \mathbf{0} \implies s_1 - s_2 = \mathbf{0}$, and $t_2 - t_1 = \mathbf{0} \implies$ the representation is unique. \square
This property gives us a tool to help quantify dimension.

Example 4.25 In the previous example, $\dim(\mathcal{S}) = \dim(\mathcal{T}) = 1$ and $\dim(\mathcal{S} \oplus \mathcal{T}) = \dim(\mathcal{V}) = 2$.

More generally, if we have any two subspaces in a vector space \mathcal{V} , then [4, Theorem 2.27]:

$$\dim(\mathcal{S} + \mathcal{T}) = \dim(\mathcal{S}) + \dim(\mathcal{T}) - \dim(\mathcal{S} \cap \mathcal{T}). \quad (4.12)$$

The direct sum equation above is a special case of this equality that is similar to the inclusion-exclusion principle in combinatorics. We use (4.12) on p. 246 in a proof about low-rank decomposition.

Example 4.26 In $\mathcal{V} = \mathbb{R}^3$, if \mathcal{S} is the x - y plane and \mathcal{T} is the y - z plane, then

$$\dim(\mathcal{S} + \mathcal{T}) = 3, \quad \dim(\mathcal{S}) = \dim(\mathcal{T}) = 2, \quad \dim(\mathcal{S} \cap \mathcal{T}) = 1.$$

4.2.8 Orthogonal Complement of a Subspace

Definition For a subspace \mathcal{S} of a vector space \mathcal{V} , the orthogonal complement of \mathcal{S} is the subset of vectors in \mathcal{V} that are orthogonal to every vector in \mathcal{S} :

$$\mathcal{S}^\perp = \{v \in \mathcal{V}: \langle s, v \rangle = 0 \ \forall s \in \mathcal{S}\}. \quad (4.13)$$

Example 4.27 For $\mathcal{V} = \mathbb{R}^3$, if we have $\mathcal{S} = \text{span}((1, 1, 0), (1, -1, 0))$ then $\mathcal{S}^\perp = \text{span}((0, 0, 1))$.

Example 4.28 Recall that in any vector space \mathcal{V} , the point $\{\mathbf{0}\}$ is a subspace. Clearly, $\{\mathbf{0}\}^\perp = \mathcal{V}$ and $\mathcal{V}^\perp = \{\mathbf{0}\}$.

Q4.12 In \mathbb{R}^3 , if \mathcal{S} is a line through the origin, then what geometric shape is \mathcal{S}^\perp ?

- A: Empty set B: Point C: Line D: Plane E: \mathbb{R}^3



Caution. The orthogonal complement is *not* the same as the ordinary set complement.

Fact 4.15 Key properties of orthogonal complements when \mathcal{V} is finite dimensional (like \mathbb{F}^N) [4, Theorem 3.11]:

S^\perp is itself a subspace of \mathcal{V} ;

$$(S^\perp)^\perp = S; \quad (4.14)$$

$$S \oplus S^\perp = \mathcal{V} \text{ because } S \cap S^\perp = \{\mathbf{0}\}; \quad (4.15)$$

$$\dim(S) + \dim(S^\perp) = \dim(\mathcal{V}). \quad (4.16)$$

Fact 4.16 Decomposition theorem for subspaces [4, p. 22]. If S is a subspace in \mathcal{V} , then, because $S \oplus S^\perp = \mathcal{V}$, every vector $v \in \mathcal{V}$ can be decomposed uniquely as

$$v = s + t, \quad s \in S, \quad t \in S^\perp; \quad (4.17)$$

in words, as the sum of a vector in the subspace and another in its orthogonal complement.

Example 4.29 For $\mathcal{V} = \mathbb{R}^3$, if

$$S = \text{span}\left(\left\{\begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}\right\}\right)$$

then we can write any $v \in \mathbb{R}^3$ as

$$\underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{\in \mathcal{V}} = \underbrace{\begin{bmatrix} a \\ 0 \\ c \end{bmatrix}}_{\in S} + \underbrace{\begin{bmatrix} 0 \\ b \\ 0 \end{bmatrix}}_{\in S^\perp}.$$

4.2.9 Linear Transforms

Definition Let \mathcal{V} and \mathcal{W} be vector spaces on a common field \mathbb{F} . A function $L: \mathcal{V} \mapsto \mathcal{W}$ is a linear transform or linear map [4, Definition 3.1] iff

$$L(\alpha u + \beta v) = \alpha L(u) + \beta L(v) \quad \forall \alpha, \beta \in \mathbb{F} \text{ and } \forall u, v \in \mathcal{V}. \quad (4.18)$$

Example 4.30 Consider $\mathcal{V} = \mathbb{R}^2$ and let \mathcal{W} denote the space of T -periodic functions on \mathbb{R} . Construct $L(\cdot)$ by

$$s = L([a \ b]^\top) \iff s(t) = a \cos(2\pi t/T) + b \sin(2\pi 5t/T + \pi/4).$$

Explore 4.3 Verify that $L(\cdot)$ in the previous example is a linear transform.

Example 4.31 Let $\mathcal{V} = \mathbb{C}^N$, $\mathcal{W} = \mathbb{C}^M$, and $A \in \mathbb{C}^{M \times N}$. Consider the transform defined by matrix multiplication: $y = L(x) \iff y = Ax$, that is, $x \mapsto y = Ax$. This transform is linear.

Proof. $L(ax + bz) = A(ax + bz) = aAx + bAz = \alpha L(x) + \beta L(z)$, where $\alpha, \beta \in \mathbb{C}$ are arbitrary, as are $x, z \in \mathbb{C}^N$. \square

4.2.10 Range of a Matrix

Definition The range of a matrix $A = [a_1 \cdots a_N] \in \mathbb{F}^{M \times N}$, also known as its image or column space, is the span of its columns:

$$\mathcal{R}(A) \triangleq \text{span}(\{a_1, \dots, a_N\}). \quad (4.19)$$

Equivalently, $\mathcal{R}(A) = \{Ax : x \in \mathbb{F}^N\}$, because the matrix–vector product is a linear combination of the matrix columns: $Ax = \sum_{n=1}^N a_n x_n$.

The range of a matrix in $\mathbb{F}^{M \times N}$ is a subspace of \mathbb{F}^M .

Definition The row space of a matrix A is the span of its rows: $\mathcal{R}(A')$.

Example 4.32 For $u \in \mathbb{C}^M$, $v \in \mathbb{C}^N$ with outer product $A = uv'$, we have

$$\mathcal{R}(uv') = \{(uv')x : x \in \mathbb{F}^N\} = \{u(v'x) : x \in \mathbb{F}^N\} = \begin{cases} \text{span}(u), & v \neq 0, \\ \{0\}, & v = 0. \end{cases} \quad (4.20)$$

Example 4.33 For $A = \begin{bmatrix} 1 & 3 \\ 2 & 6 \end{bmatrix}$, we have

$$\mathcal{R}(A) = \text{span}\left(\left\{\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 6 \end{bmatrix}\right\}\right) = \text{span}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = \left\{\alpha \begin{bmatrix} 1 \\ 2 \end{bmatrix} : \alpha \in \mathbb{F}\right\},$$

$$\mathcal{R}^\perp(A) = \text{span}\left(\begin{bmatrix} 2 \\ -1 \end{bmatrix}\right).$$

Q4.13 If D is a diagonal matrix in $\mathbb{R}^{N \times N}$, what is $\mathcal{R}(D)$?

- A: \emptyset B: Usually \mathbb{R}^N C: Always \mathbb{R}^N D: Usually $\mathbb{R}^{N \times N}$ E: Always $\mathbb{R}^{N \times N}$

Practical use in JULIA: JULIA has a `range` method, but it is unrelated to the range of a matrix! So there is no built-in function for $\mathcal{R}(A)$, but it is easy to write one using an SVD. See Fact 4.28 and Problem 4.14.

4.2.10.1 Range and Matrix Multiplication

Often we are interested in the range of matrix products. Clearly,

$$A \in \mathbb{F}^{M \times N}, \quad B \in \mathbb{F}^{N \times K} \implies \mathcal{R}(AB) \subseteq \mathcal{R}(A), \quad (4.21)$$

because $\mathcal{R}(AB) = \{Ax : x = Bz, z \in \mathbb{F}^K\} \subseteq \{Ax : x \in \mathbb{F}^N\}$.

Fact 4.17 If A is any $M \times N$ matrix and B is an $N \times K$ matrix with linearly independent rows, then

$$\mathcal{R}(AB) = \mathcal{R}(A). \quad (4.22)$$

Proof. Using (4.21) it suffices to show that $\mathcal{R}(A) \subseteq \mathcal{R}(AB)$. If $y \in \mathcal{R}(A)$ then there is some $x \in \mathbb{F}^N$ such that $y = Ax$. Because B has linearly independent rows, it follows from Fact 4.3 that BB' is an invertible matrix. Thus we can write $x = BB'(BB')^{-1}x = Bz$, where $z \triangleq B'(BB')^{-1}x$. Thus, $y = ABz$, so $y \in \mathcal{R}(AB)$. \square

However, linear independence of the rows of B (aka full row rank) is not a *necessary* condition in general for (4.22) to hold.

Example 4.34 If $A = 0$, then $\mathcal{R}(A) = \mathcal{R}(AB)$ for any suitably sized matrix B .

Explore 4.4 Prove the following partial converse of (4.22). If B is an $N \times K$ matrix for which $\mathcal{R}(AB) = \mathcal{R}(A)$ for every $M \times N$ matrix A , then B has linearly independent rows.

Explore 4.5 Prove or disprove the following “reverse” of (4.21). If B is invertible, then $\mathcal{R}(BA) = \mathcal{R}(A)$.

There are necessary and sufficient conditions on B , in terms of properties of A , such that $\mathcal{R}(AB) = \mathcal{R}(A)$ (see Problem 4.12).

Explore 4.6 Many of the concepts in this chapter can be sensitive to numerical precision effects. Examine the range of the matrix $A = \begin{bmatrix} 1 & 0 \\ 0 & a \end{bmatrix}$ as a function of $a \in \mathbb{C}$, especially $a \approx 0$.

Example 4.35 For $u_k \in \mathbb{C}^M$, $v_k \in \mathbb{C}^N$ for $k = 1, \dots, K$, with the sum of outer products $A = \sum_{k=1}^K u_k v'_k = UV'$, where $U = [u_1 \ \dots \ u_K]$, $V = [v_1 \ \dots \ v_K]$. Clearly, $\mathcal{R}(A) \subseteq \mathcal{R}(U) = \text{span}(\{u_1, \dots, u_K\})$, because

$$\mathcal{R}(A) = \mathcal{R}\left(\sum_{k=1}^K u_k v'_k\right) = \left\{ \sum_{k=1}^K u_k (v'_k x) : x \in \mathbb{F}^N \right\} \subseteq \text{span}(\{u_1, \dots, u_K\}). \quad (4.23)$$

If $\{v_1, \dots, v_K\}$ are linearly independent (a sufficient, not necessary, condition), then by (4.21) we have $\mathcal{R}(A) = \mathcal{R}(U)$.

4.3 Rank of a Matrix

The preceding material about subspaces applies to vectors in general vector spaces. Now we specialize to a subspace-related concept that is specific to matrices: the rank of a matrix.

Definition For any $M \times N$ matrix A :

- The column rank of $A \triangleq \dim(\mathcal{R}(A)) = \# \text{ of linearly independent columns of } A \leq N$.
- The row rank of $A \triangleq \dim(\mathcal{R}(A')) = \# \text{ of linearly independent rows of } A \leq M$.

Fact 4.18 For any $M \times N$ matrix A , its row rank equals its column rank.

Proof. Let r denote the column rank of $A = [a_1 \cdots a_N]$. Because r is the column rank, there exists a basis $V = [v_1 \cdots v_r]$ such that one can write every vector in $\mathcal{R}(A)$ as a linear combination of the columns of V . Each column of A is in $\mathcal{R}(A)$, and thus we can express each column of A as a linear combination of the columns of V , that is,

$$a_n = c_{1n}v_1 + \cdots + c_{rn}v_r, \quad n = 1, \dots, N,$$

where the c_{ij} values denote the coordinates or coefficients with respect to the basis V . In matrix form we get a sum-of-outer-products form of matrix multiplication:

$$\underbrace{\begin{matrix} A \\ M \times N \end{matrix}}_{M \times r} = \underbrace{\begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix}}_{r \times N} \underbrace{\begin{bmatrix} c_{11} & \cdots & c_{1N} \\ \vdots & & \vdots \\ c_{r1} & \cdots & c_{rN} \end{bmatrix}}_{r \times N} = \begin{bmatrix} v_1 & \cdots & v_r \end{bmatrix} \begin{bmatrix} - & c_1^\top & - \\ - & \vdots & - \\ - & c_r^\top & - \end{bmatrix} = V C = \sum_{k=1}^r v_k c_k^\top.$$

Now the m th row of A is a linear combination of the rows of C :

$$A_{m,:} = e_m' A = \sum_{k=1}^r (e_m' v_k) c_k^\top = \sum_{k=1}^r v_{mk} c_k^\top.$$

This construction holds for any row of A (or linear combinations thereof)

$$\begin{aligned} &\implies \mathcal{R}(A') = \text{row space of } A \subseteq \mathcal{R}(C^\top) \\ &\implies \text{row rank of } A \leq r = \text{column rank of } A. \end{aligned}$$

Because A was arbitrary, the same argument applies to its transpose, so:

$$\begin{aligned} &\implies \text{row rank of } A' \leq \text{column rank of } A' \\ &\implies \text{column rank of } A \leq \text{row rank of } A \\ &\implies \text{column rank of } A = \text{row rank of } A. \end{aligned}$$

□

For alternate proofs, see [4, Theorem 3.17] and the CWR factorization of [53].

Because the row rank and column rank of a matrix are always identical, we generally simply speak of the rank of a matrix without the “row” or “column” qualifier. And it suffices to define

$$\text{rank}(A) \triangleq \dim(\mathcal{R}(A)). \quad (4.24)$$

Fact 4.19

$$A \in \mathbb{F}^{M \times N} \implies 0 \leq \text{rank}(A) \leq \min(M, N). \quad (4.25)$$

Proof. $\text{rank}(A) = \text{row rank of } A \leq \# \text{ rows} = M$, and $\text{rank}(A) = \text{col rank of } A \leq \# \text{ cols} = N \implies \text{rank}(A) \leq \min(M, N)$. \square

Definition When $A \in \mathbb{F}^{M \times N}$ and $\text{rank}(A) = \min(M, N)$ we say A has full rank.

Example 4.36 For $A = \begin{bmatrix} 1 & 3 \\ 2 & 6 \end{bmatrix}$, we showed in a previous example that

$$\mathcal{R}(A) = \text{span}\left(\left\{\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 6 \end{bmatrix}\right\}\right) = \text{span}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right).$$

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

Q4.14 What is the rank of $\begin{bmatrix} 5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 7 \end{bmatrix}$?

A: 0

B: 1

C: 2

D: 3

E: 4

4.3.1 Practical Use in Julia

The JULIA function `rank(A)` uses a tolerance threshold on the singular values (Problem 4.6). A threshold is needed because finite precision effects perturb the singular values, including those that ideally are zero [54, 55, 56]. See Chapter 12.

Explore 4.7 To understand why `rank` involves a tolerance threshold, and why `rank` is a “brittle” function numerically, plot the rank of the matrix $A = \begin{bmatrix} 3 & 3 \\ 3 & a \end{bmatrix}$ as a function of $a \in \mathbb{R}$.

4.3.2 Rank of a Matrix Product

We can combine matrices in various ways and see what happens to matrix rank.

Fact 4.20 Multiplying matrices never increases rank: $A \in \mathbb{F}^{M \times N}, B \in \mathbb{F}^{N \times K} \implies$

$$0 \leq \text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)) \leq \min(K, M, N). \quad (4.26)$$

Proof.

$$\mathbf{AB} = \begin{bmatrix} | & & | \\ \mathbf{a}_1 & \cdots & \mathbf{a}_N \\ | & & | \end{bmatrix} \begin{bmatrix} - & \mathbf{b}_1^T & - \\ & \vdots & \\ - & \mathbf{b}_N^T & - \end{bmatrix} = \mathbf{a}_1\mathbf{b}_1^T + \cdots + \mathbf{a}_N\mathbf{b}_N^T$$

\implies every column of \mathbf{AB} is a linear combination of columns of \mathbf{A} (see also (2.13))

$\implies \mathcal{R}(\mathbf{AB}) \subseteq \mathcal{R}(\mathbf{A})$

$\implies \text{rank}(\mathbf{AB}) \leq \text{rank}(\mathbf{A})$ by (4.8).

Similarly, because $(\mathbf{AB})' = \mathbf{B}'\mathbf{A}'$, applying the same logic we have

$$\text{rank}(\mathbf{AB}) = \text{rank}((\mathbf{AB})') = \text{rank}(\mathbf{B}'\mathbf{A}') \leq \text{rank}(\mathbf{B}') = \text{rank}(\mathbf{B}).$$

Combining both inequalities yields the first inequality in (4.26). The second inequality follows from (4.25). \square

Caution: in general, $\text{rank}(\mathbf{AB}) \neq \text{rank}(\mathbf{BA})$, even if the sizes are compatible. Also, if \mathbf{B} has full row rank (linearly independent rows) then it follows from (4.22) that $\text{rank}(\mathbf{AB}) = \text{rank}(\mathbf{A})$.

The product \mathbf{AB} is a composition of two linear transforms. In words, (4.26) implies that composition cannot enlarge subspace dimension.

Example 4.37 A DSP analogy to (4.26): the cascade of two filters with band-limited frequency responses. See Fig. 4.4, where $[-v_3, v_3] = [-v_1, v_1] \cap [-v_2, v_2]$. Composing filters cannot recover lost frequencies.

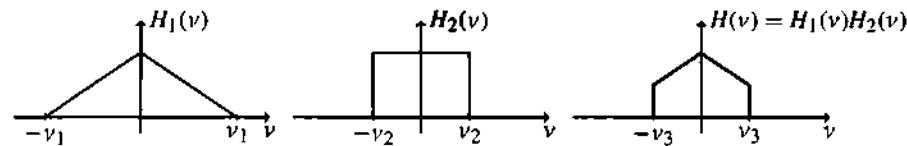


Figure 4.4 Frequency responses of some filters.

Q4.15 If \mathbf{A} has linearly independent columns, then does $\text{rank}(\mathbf{AB}) = \text{rank}(\mathbf{A})$? Does $\text{rank}(\mathbf{AB}) = \text{rank}(\mathbf{B})$?

- A: (T,T) B: (T,F) C: (F,T) D: (F,F)

Q4.16 For $\mathbf{u} \in \mathbb{C}^M$, $\mathbf{v} \in \mathbb{C}^N$, what are the minimum and maximum possible ranks of the outer product \mathbf{uv}' ?

- A: 0,1 B: 1,1 C: 0, $\min(M,N)$ D: 1, $\min(M,N)$ E: None of these

4.3.3 Other Rank Properties

There is also a lower bound for the rank of a matrix product, called Sylvester's rank inequality:

$$\mathbf{A} \in \mathbb{F}^{M \times N}, \mathbf{B} \in \mathbb{F}^{N \times K} \implies \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B}) - N \leq \text{rank}(\mathbf{AB}). \quad (4.27)$$

Furthermore, rank is subadditive:

$$A, B \in \mathbb{F}^{M \times N} \implies 0 \leq \text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B). \quad (4.28)$$

A tighter inequality is

$$A, B \in \mathbb{F}^{M \times N} \implies 0 \leq \text{rank}(A + B) \leq \min(\text{rank}(A) + \text{rank}(B), M, N). \quad (4.29)$$

Proof [57]. Using (4.12), $\text{rank}(A + B) = \dim(\mathcal{R}(A + B)) \leq \dim(\mathcal{R}(A) + \mathcal{R}(B)) = \dim(\mathcal{R}(A)) + \dim(\mathcal{R}(B)) - \dim(\mathcal{R}(A) \cap \mathcal{R}(B)) \leq \text{rank}(A) + \text{rank}(B)$.

For the lower bound, take $A = -B$.

For the upper bound, take $A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$. \square

Explore 4.8 Show the lower bound: $|\text{rank}(A) - \text{rank}(B)| \leq \text{rank}(A + B)$.

Explore 4.9 What about other matrix operations, such as:

$$\text{rank}(\alpha A), \quad \text{rank}([A \quad 0]), \quad \text{rank}([A \quad B]), \quad \text{rank}\left(\begin{bmatrix} A \\ B \end{bmatrix}\right) ?$$

4.3.4 Spark

◆◆ A concept related to rank that arises in compressed sensing theory is the spark of a matrix A , defined as the smallest integer k such that there exist k columns of A that are linearly dependent [58].

Q4.17 If A_1, A_2, \dots, A_K are sized appropriately to allow multiplication, then

$$\text{rank}(A_1 A_2 \cdots A_K) \leq \min(\text{rank}(A_1), \dots, \text{rank}(A_K)).$$

A: True

B: False

Q4.18 For $u_1, u_2 \in \mathbb{C}^M$ and $v_1, v_2 \in \mathbb{C}^N$, what are the minimum and maximum possible ranks of the $M \times N$ matrix $3u_1v_1' + 5u_2v_2'$?

A: 0, 1 B: 1, 2 C: 0, $\min(M, N)$ D: 1, $\min(M, N)$ E: None of these

Q4.19 If $\mathcal{V} = \mathbb{F}^6$ and x and y are two linearly independent vectors in \mathcal{V} , then what is the dimension of $(\text{span}(\{x, y\}))^\perp$?

A: 1 or less B: 2 C: 3 D: 4 E: 5 or more

4.3.5 Unitary Invariance of Rank

To understand rank in the context of a singular value decomposition, we examine products with unitary matrices.

Fact 4.21 If $A \in \mathbb{F}^{M \times N}$ then multiplying by a (compatible) unitary matrix on the left or right does not change the rank:

$Q \in \mathbb{F}^{M \times M}$ and Q unitary $\implies \text{rank}(QA) = \text{rank}(A)$;

$Q \in \mathbb{F}^{N \times N}$ and Q unitary $\implies \text{rank}(AQ) = \text{rank}(A)$.

4.3.5.1 Rank and Eigenvalues/Eigendecompositions

Fact 4.22 A corollary is that if A is Hermitian (or normal) with unitary eigendecomposition $A = V\Lambda V'$ then, because V is unitary (and so is V'), we have the important property that

$$\text{rank}(A) = \text{rank}(V\Lambda V') = \text{rank}(\Lambda) = \# \text{ of nonzero eigenvalues of } A. \quad (4.30)$$

Q4.20 If A is square, then $\text{rank}(A) = \# \text{ of nonzero eigenvalues of } A$.

A: True

B: False

4.3.5.2 Rank and SVD

- By unitary invariance, if A has SVD $A = U\Sigma V'$, then $\text{rank}(A) = \text{rank}(\Sigma)$.
- $\text{rank}(\Sigma) = r = \# \text{ nonzero singular values of } A$, since $\mathcal{R}(\Sigma) = \text{span}(\{e_1, \dots, e_r\})$.

Thus, the SVD expression for a matrix A having rank r where $r \leq \min(M, N)$ simplifies:

$$A \in \mathbb{F}^{M \times N} \implies A = U\Sigma V' = \sum_{k=1}^{\min(M, N)} \sigma_k u_k v'_k = \sum_{k=1}^r \sigma_k u_k v'_k. \quad (4.31)$$

When we write $\sum_{k=1}^{\min(M, N)}$, some of the σ_k values may be zero, namely $\sigma_{r+1}, \dots, \sigma_{\min(M, N)}$. When we write $\sum_{k=1}^r$, where r is the rank of A , then all the σ_k values used in the sum are nonzero. By convention, any empty sum (when $r = 0$) evaluates to zero, in this case $0_{M \times N}$.

This convention holds in JULIA as well: `sum(ones(0))` evaluates to `0` because `ones(0)` is an empty `Float64` variable (a zero-dimensional vector).

Explore 4.10 Determine the convention for an empty product.

4.4

Nullspace of a Matrix

To fully understand an SVD of a matrix, we need both its range and null space (and their orthogonal complements).

4.4.1 Nullspace or Kernel

The set of vectors that yield zero when multiplied by a matrix is often important.

Definition The null space or kernel of an $M \times N$ matrix A is

$$N(A) = \ker(A) \triangleq \{x \in \mathbb{F}^N : Ax = \mathbf{0}\} \subseteq \mathbb{F}^N. \quad (4.32)$$

Fact 4.23 Clearly we always have $\mathbf{0} \in N(A)$.

Fact 4.24 $N(A)$ is indeed a subspace (Problem 4.11). Thus, using the subspace font to write “ N ” is appropriate.

Example 4.38 $A = \begin{bmatrix} 1 & 3 \\ 2 & 6 \end{bmatrix} \implies N(A) = \text{span}\left(\begin{bmatrix} -3 \\ 1 \end{bmatrix}\right)$ and $N^\perp(A) = \text{span}\left(\begin{bmatrix} 1 \\ 3 \end{bmatrix}\right)$.

Q4.21 If $A \in \mathbb{C}^{M \times N}$, then $N(A)$ is a subspace of what vector space?

- A: \mathbb{C}^M B: \mathbb{C}^N C: \mathbb{R}^N D: \mathbb{R}^M E: None of these.

Practical use in JULIA: `nullspace(A)`. Like `rank`, this function involves a tolerance threshold on the singular values.

Explore 4.11 To understand why the `nullspace` function involves a threshold, examine the null space of the matrix $A = \begin{bmatrix} 0 & 0 \\ 0 & a \end{bmatrix}$ for $a \in \mathbb{C}$ and consider what happens for $a \approx 0$.

One use of the nullspace of a matrix is to assess whether its columns are linearly dependent. Here are some other concrete applications.

- The MUSIC algorithm for direction-of-arrival estimation uses the null space of a signal covariance matrix [59, 60, 61]. (In practice there is always noise in the data so the method uses the “noise subspace” corresponding to the minimal eigenvalues.)
- The spectral clustering method for unsupervised data clustering uses the null space of a graph Laplacian matrix. (Again, in practice it uses the k smallest eigenvalues which may not be exactly zero in the presence of noise.) See Section 8.9.5.
- Null-space vectors have a key role in reconstructing multiple-coil MRI scans [62, 63, 64, 65].

4.4.2 Properties of Null Space

- $N(A) = \{\mathbf{0}\} \iff A$ has linearly independent columns.
- $N(A) = \mathbb{F}^N \iff A = \mathbf{0}_{M \times N}$.
- $N(B) \subseteq N(AB)$

Proof. $x \in N(B) \implies Bx = \mathbf{0} \implies ABx = \mathbf{0} \implies x \in N(AB)$. □

- If $N(A) = \{\mathbf{0}\}$, that is, if A has full column rank, then $N(AB) = N(AB)$. This is not a necessary condition, but it is adequate for our purposes.

Explore 4.12 Try `nullspace(ones(3,1))` and `nullspace(zeros(3,1))` in JULIA. Do you agree with the output?

4.4.2.1 Decomposition Theorem for Matrices

Fact 4.25 If $A \in \mathbb{F}^{M \times N}$ then, from (4.15), the input and output spaces of A satisfy

$$\begin{aligned} N(A) \oplus N^\perp(A) &= \mathbb{F}^N, \\ R(A) \oplus R^\perp(A) &= \mathbb{F}^M. \end{aligned}$$

In other words, every “input” vector $x \in \mathbb{F}^N$ can be decomposed uniquely as $x = x_0 + x_1$, where $x_0 \in N(A)$, so $Ax_0 = \mathbf{0}$ and $x_1 \in N^\perp(A)$.

Likewise, every vector $y \in \mathbb{F}^M$ can be decomposed uniquely as $y = y_1 + y_0$, where $y_1 \in R(A)$, so $Ax_1 = y_1$ for some $x_1 \in \mathbb{F}^N$, and $y_0 \in R^\perp(A)$. This applies for an arbitrary $y \in \mathbb{F}^M$. If we have an “output vector” $y = Ax$, then $y \in R(A)$ by definition and $y_0 = \mathbf{0}$.

4.4.2.2 Relationships between the Null Space and Range of a Matrix

Fact 4.26 For any matrix A , its null space and range are related by:

$$N^\perp(A) = R(A'), \quad (4.33)$$

$$R^\perp(A) = N(A'). \quad (4.34)$$

Proof. If $x \in N(A)$ then $Ax = \mathbf{0}$, so $\forall y$ we have $y'(Ax) = 0 \implies x'(A'y) = 0 \implies x \in R^\perp(A')$. Conversely, if $x \in R^\perp(A')$ then $\forall y, 0 = x'(A'y)$. Take $y = Ax$ and we have $x'A'Ax = 0$ so $\|Ax\| = 0$, which implies $Ax = \mathbf{0}$, so $x \in N(A)$. Thus, using (4.14), $N(A) = R^\perp(A') \implies N^\perp(A) = R(A')$. The proof of the second equality is left to Problem 4.10. \square

Combining (4.33) and (4.34) yields the following corollary equalities:

$$\begin{aligned} \dim(N^\perp(A)) &= \dim(R(A')) = \dim(R(A)) = \text{rank}(A), \\ \dim(R^\perp(A)) &= \dim(N(A')). \end{aligned} \quad (4.35)$$

4.4.2.3 Nullity

Definition The nullity of an $M \times N$ matrix A is the dimension of its null space:

$$\text{nullity}(A) \triangleq \dim(N(A)) \in \{0, 1, \dots, N\}. \quad (4.36)$$

Fact 4.27 Rank plus nullity property. If $A \in \mathbb{F}^{M \times N}$ then

$$\underbrace{\dim(\mathcal{N}(A))}_{\text{nullity}(A)} + \underbrace{\dim(\mathcal{R}(A))}_{\text{rank}(A)} = N. \quad (4.37)$$

Proof. Because $\mathbb{F}^N = \mathcal{N}(A) \oplus \mathcal{N}^\perp(A)$, we have, from (4.15), (4.16), and (4.35), $N = \dim(\mathcal{N}(A)) + \dim(\mathcal{N}^\perp(A)) = \dim(\mathcal{N}(A)) + \text{rank}(A)$. \square

Example 4.39 The null space of the matrix $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 3 \end{bmatrix}$ is the span of the vector $\begin{bmatrix} 0 \\ -5 \\ 0 \end{bmatrix}$, so its nullity is 1.

Example 4.40 The null space of the matrix $\begin{bmatrix} 5 & 0 & 5 \\ 7 & 0 & 7 \end{bmatrix}$ is the span of the (linearly independent) vectors $\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, so its nullity is 2.

Q4.22 If $u \in \mathbb{C}^M \setminus \{0\}$ and $v \in \mathbb{C}^N \setminus \{0\}$, what is the nullity of their outer product, that is, $\text{nullity}(uv')$?

- A: 0 B: 1 C: $N - 1$ D: N E: M

4.4.3 Columns of Unitary Matrices

This section is a warm-up for the upcoming fundamental theorem of linear algebra. Let $V \in \mathbb{F}^{N \times N}$ denote a unitary matrix, and suppose we partition the columns of V into two parts like $V = [V_1 \ V_2]$, where V_1 is $N \times K$ and V_2 is $N \times N - K$, where $0 < K < N$. We now build on results from previous sections to summarize many properties of the components of this partition.

For $l = 1, 2$:

$$\begin{aligned} \mathcal{R}(V_l) \oplus \mathcal{R}^\perp(V_l) &= \mathbb{F}^N && \text{by (4.15),} \\ \mathcal{N}^\perp(V_l) &= \mathcal{R}(V'_l) && \text{by (4.33),} \\ \mathcal{R}^\perp(V_l) &= \mathcal{N}(V'_l) && \text{by (4.34).} \end{aligned}$$

Furthermore, because V is unitary, $x \in \mathbb{F}^N \implies x = VV'x = V_1(V'_1x) + V_2(V'_2x)$ where $\mathcal{R}(V_1) \perp \mathcal{R}(V_2)$, so we have

$$\mathcal{R}(V_1) \oplus \mathcal{R}(V_2) = \mathbb{F}^N. \quad (4.38)$$

Combining two of the $l = 1$ equations yields $\mathcal{R}(V_1) \oplus \mathcal{N}(V'_1) = \mathbb{F}^N$.

Combining with the subspace decomposition uniqueness theorem (4.17), it follows that

$$\mathcal{R}(V_2) = \mathcal{R}^\perp(V_1) = \mathcal{N}(V'_1) = \mathcal{N}^\perp(V'_2). \quad (4.39)$$

Explore 4.13 Determine a relationship between $N(V'_1)$ and $N(V'_2)$.

Example 4.41 For $N = 2$ and $K = 1$ with

$$V = \begin{bmatrix} \cos \phi & \sin \phi \\ \sin \phi & -\cos \phi \end{bmatrix} = [V_1 \ V_2], \quad V_1 = \begin{bmatrix} \cos \phi \\ \sin \phi \end{bmatrix}, \quad V_2 = \begin{bmatrix} \sin \phi \\ -\cos \phi \end{bmatrix},$$

we have (see Fig. 4.5):

$$\text{span}\left(\begin{bmatrix} \cos \phi \\ \sin \phi \end{bmatrix}\right) = \mathcal{R}(V_1) = \mathcal{R}^\perp(V_2) = N(V'_2) = N^\perp(V'_1),$$

$$\text{span}\left(\begin{bmatrix} \sin \phi \\ -\cos \phi \end{bmatrix}\right) = \mathcal{R}(V_2) = \mathcal{R}^\perp(V_1) = N(V'_1) = N^\perp(V'_2).$$

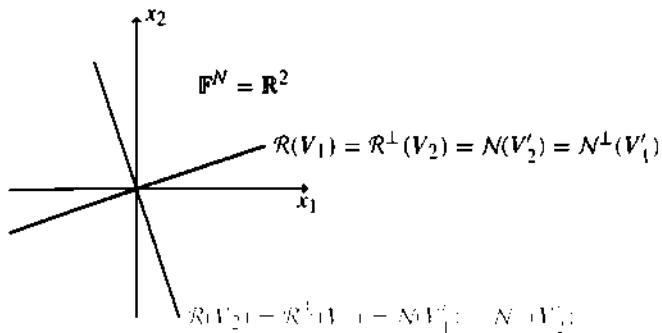


Figure 4.5 Range and null space of two parts of a unitary matrix $V = [V_1 \ V_2]$.

4.5 The Four Fundamental Spaces

Now we unify the null space and range space (and their orthogonal complements) for a general matrix A . We start by revisiting a simple 2×2 example, illustrated in Fig. 4.6.

Example 4.42 For $A = \begin{bmatrix} 1 & 3 \\ 2 & 6 \end{bmatrix}$, recall we showed previously that:

$$N(A) = \text{span}\left(\begin{bmatrix} -3 \\ 1 \end{bmatrix}\right), \quad \mathcal{R}(A) = \text{span}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right),$$

$$N^\perp(A) = \text{span}\left(\begin{bmatrix} 1 \\ 3 \end{bmatrix}\right), \quad \mathcal{R}^\perp(A) = \text{span}\left(\begin{bmatrix} 2 \\ -1 \end{bmatrix}\right).$$

The characteristics illustrated in Fig. 4.6 hold in general.

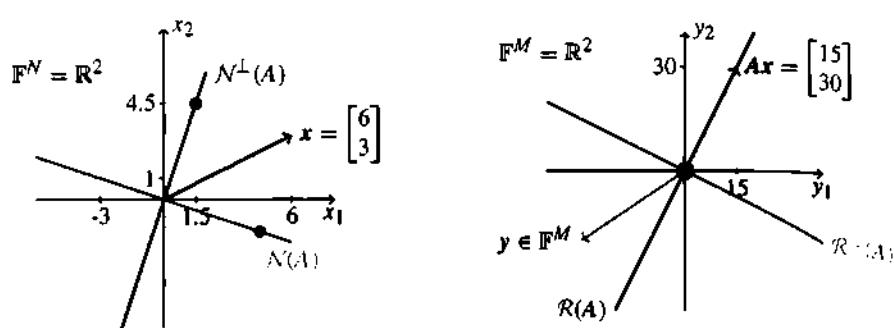


Figure 4.6 Example of the four fundamental spaces. If $y \notin R(A)$, then there is no x for which $y = Ax$ so that y is “unreachable” by A .

Fact 4.28 The following diagram summarizes the fundamental theorem of linear algebra for a matrix $A \in \mathbb{F}^{M \times N}$ having rank r , where we think of A as a mapping from \mathbb{F}^N to \mathbb{F}^M .

| dim | \mathbb{F}^N (“input”) | \mathbb{F}^M (“output”) | dim |
|---------|--------------------------|--|---------|
| r | $N^{\perp}(A) = R(V_r)$ | \xrightarrow{A} | r |
| $N - r$ | $N(A) = R(V_0)$ | $R^{\perp}(A) = R(U_0)$ ("unreachable") | $M - r$ |
| | $\oplus \{0\}$ | $\{0\} \oplus$ | |

The matrices U_r , U_0 , V_r , and V_0 are defined below. There is no standard notation for these four matrices. Some books use V_1 and V_2 ; one also sees V_{\parallel} and V_{\perp} [62].

Now we use that fundamental theorem to describe an SVD more completely.

When $A \in \mathbb{F}^{M \times N}$ has rank $0 < r < \min(M, N)$, we can partition the SVD components as follows:

$$\begin{aligned} A &= \sum_{k=1}^{\min(M,N)} \sigma_k u_k v'_k = \sum_{k=1}^r \sigma_k u_k v'_k = U \Sigma V' \\ &= \underbrace{\left[\begin{array}{c|c} U_r & U_0 \end{array} \right]}_{M \times r | M \times (M-r)} \left[\begin{array}{c|c} \Sigma_r & 0 \\ 0 & 0 \end{array} \right] \underbrace{\left[\begin{array}{c} V'_r \\ V'_0 \end{array} \right]}_{(N-r) \times N} \quad \} \quad r \times N \\ &\quad \} \quad (N-r) \times N \end{aligned}$$

where Σ_r is $r \times r$ and contains the *nonzero* singular values of A along its diagonal.

Here, $U_r = [u_1 \ \dots \ u_r]$, $U_0 = [u_{r+1} \ \dots \ u_M]$, $V_r = [v_1 \ \dots \ v_r]$, and $V_0 = [v_{r+1} \ \dots \ v_N]$.

- In the corner case where $r = 0$, we have $A = \mathbf{0}$, $\Sigma = \mathbf{0}$, $U = U_0$, $V = V_0$, and there is no U_r or V_r or Σ_r .
- In the tall case where $r = N \leq M$, we have $A = U_N \Sigma_N V'$ because $V = V_r = V_N$, and there is no V_0 . (In JULIA we can think of V_0 as an $N \times 0$ matrix.)
- In the wide case where $r = M \leq N$, we have $A = U \Sigma_M V'_M$ because $U = U_r = U_M$, and there is no U_0 .

Q4.23 What is the size of the lower right $\mathbf{0}$ above?

A: $M \times N$ B: $M \times (N - r)$ C: $(M - r) \times N$ D: $(M - r) \times (N - r)$ E: None of these

Q4.24 If $x = V_0 z$ for some $z \in \mathbb{F}^{N-r}$, then what is Ax ?

A: $\mathbf{0}$ B: $U_r \Sigma_r z$ C: $U_r V_r z$ D: $U_0 V_r z$ E: None of these

Proof of the fundamental theorem of linear algebra. To show that $\mathcal{R}(A) = \mathcal{R}(U_r)$ simply use the compact SVD to write $A = U_r \Sigma_r V'_r$. The matrix $\Sigma_r V'_r$ has linearly independent rows (because V_r has rank r), so now apply (4.22). It follows that $\mathcal{R}^\perp(A) = \mathcal{R}^\perp(U_r) = \mathcal{R}(U_0)$ because $\mathcal{R}(U_r) \oplus \mathcal{R}(U_0) = \mathbb{F}^M$ since U is an $M \times M$ unitary matrix. To show that $N^\perp(A) = \mathcal{R}(V_r)$ use (4.33) to write $N^\perp(A) = \mathcal{R}(A') = \mathcal{R}(V_r \Sigma_r U'_r) = \mathcal{R}(V_r)$ because $\Sigma_r U'_r$ has linearly independent rows. \square

4.5.1 Anatomy of the SVD

There are two cases of the above partitioning to consider in more detail: when A is tall or wide.

4.5.1.1 SVD of a Tall Matrix

When A is tall or “thin,” that is, $M > N \implies r \leq N < M$, then we can simplify:

$$\boxed{\begin{array}{c|c} A \\ \hline M \times N \end{array}} = U \Sigma V' = \left[\begin{array}{c|c} U_r & U_0 \\ \hline M \times r & M \times (M - r) \end{array} \right] \left[\begin{array}{c} \Sigma_r \\ \hline \mathbf{0} \end{array} \right] \left[\begin{array}{c} V'_r \\ \hline r \times N \end{array} \right] = \underbrace{U_r}_{M \times r} \underbrace{\Sigma_r}_{r \times r} \underbrace{V'_r}_{r \times N}. \quad (4.40)$$

Q4.25 What is the size of the lower $\mathbf{0}$ above?

A: $M \times N$ B: $M \times (N - r)$ C: $(M - r) \times N$ D: $(M - r) \times (N - r)$ E: None of these

 Caution: When we write $U \Sigma V' = U_r \Sigma_r V'_r$, we do not mean that the individual terms match (they do not!); we mean that the overall product matches.

Example 4.43 Figure 4.7 illustrates a compact SVD graphically.

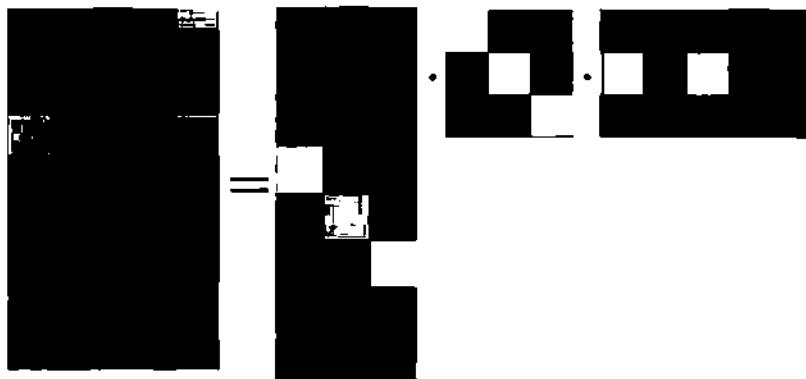


Figure 4.7 Illustration of compact SVD $\mathbf{X} = \mathbf{U}_r \boldsymbol{\Sigma}_r \mathbf{V}'_r$ for a 9×5 matrix with rank $r = 3$.

4.5.1.2 SVD of a Wide Matrix

When \mathbf{A} is wide, that is, $M < N \implies r \leq M < N$, then we can simplify:

$$\boxed{\mathbf{A}}_{M \times N} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}' = \underbrace{\mathbf{U}_r}_{M \times r} \left[\begin{array}{c|c} \boldsymbol{\Sigma}_r & \mathbf{0} \end{array} \right] \underbrace{\begin{bmatrix} \mathbf{V}'_r \\ \mathbf{V}'_0 \end{bmatrix}}_{(N-r) \times N}_{N \times N} \}^{r \times N} = \underbrace{\mathbf{U}_r}_{M \times r} \underbrace{\boldsymbol{\Sigma}_r}_{r \times r} \underbrace{\mathbf{V}'_r}_{r \times N}. \quad (4.41)$$

If $r = M$ then $\dim(\mathcal{R}(\mathbf{A})) = M$, $\mathbf{U}_r = \mathbf{U}_M = \mathbf{U}$, there is no \mathbf{U}_0 , and all of \mathbb{F}^M is reachable; that is, $\mathcal{R}(\mathbf{A}) = \mathbb{F}^M$. However, note that $\dim(\mathcal{N}(\mathbf{A})) = N - r > M - r \geq 0$, so a wide \mathbf{A} has a nontrivial null space.

Note the symmetry between the above two “compact” SVD representations for the tall and wide cases, as there must be because if \mathbf{A} is tall then \mathbf{A}' is wide.

4.5.1.3 Practical Use of SVD

The above cases are somewhat (but not exactly) related to the

$(\mathbf{U}, \mathbf{s}, \mathbf{V}) = \text{svd}(\mathbf{A}, \text{full=false})$

method in JULIA, which is the default option for `svd`.

- $(\mathbf{U}, \mathbf{s}, \mathbf{V}) = \text{svd}(\mathbf{A}, \text{full=true})$ returns the full SVD components, where \mathbf{U} is $M \times M$ and \mathbf{V} is $N \times N$ and \mathbf{s} is a vector of length $\min(M, N)$.
- $(\mathbf{U}, \mathbf{s}, \mathbf{V}) = \text{svd}(\mathbf{A}, \text{full=false})$ or just $(\mathbf{U}, \mathbf{s}, \mathbf{V}) = \text{svd}(\mathbf{A})$ returns an economy SVD, where \mathbf{U} is $M \times \min(M, N)$ and $\mathbf{U} * \text{Diagonal}(\mathbf{s}) * \mathbf{V}'$ equals \mathbf{A} to within numerical precision.

Here, when we speak of the compact SVD, we mean $\mathbf{A} = \mathbf{U}_r \boldsymbol{\Sigma}_r \mathbf{V}'_r$ as described in (4.40) and (4.41). Size (memory) comparison: compact \leq economy $=$ thin \leq full.

There is no built-in JULIA (or MATLAB) command for computing the compact SVD! Computing the compact SVD when $r > 0$ requires multiple JULIA commands, such as:

`U, s, V = svd(A)`

`r = rank(Diagonal(s))` (This useful trick avoids a redundant SVD of A .)

`Ur = U[:, 1:r]; Vr = V[:, 1:r]; sr = s[1:r]`

after which we can recover A to within numerical precision (Problem 4.13) by

`Ur * Diagonal(sr) * Vr'`

For a tall matrix A , when we use the (default) `full=false` option of JULIA's SVD, that is, `(U, s, V) = svd(A)` or `(U, s, V) = svd(A, full=false)`, then the output components correspond to:

$$\underbrace{\begin{bmatrix} U_{:,1} & \cdots & U_{:,N} \end{bmatrix}}_{M \times N} \underbrace{\begin{bmatrix} \Sigma_r & 0_{r \times (N-r)} \\ 0_{(N-r) \times r} & 0_{(N-r) \times (N-r)} \end{bmatrix}}_{N \times N} \underbrace{\begin{bmatrix} V' \\ \vdots \\ V_{:,N} \end{bmatrix}}_{N \times N}$$

$$U = U_N \quad \text{Diagonal}(s) = \Sigma \quad V'$$

where $s = (\sigma_1, \dots, \sigma_r, 0, \dots, 0)$ has length $N \leq M$ in the tall case.

 In practice, the last $N - r$ elements of the vector s will not be exactly zero, because of finite numerical precision (Problem 4.6).

 Caution: for a tall A , the output variable U is $M \times N$ whereas U is $M \times M$, and output s is an N -vector whereas Σ is $M \times N$.

Example 4.44 For a concrete example, see the rank-one case (4.46).

4.5.2 SVD of Finite Differences

Example 4.45 The derivative of $f(t) = \cos(\omega t)$ is $\dot{f}(t) = -\omega \sin(\omega t)$ and the second derivative is $\ddot{f}(t) = -\omega^2 \cos(\omega t)$. Consider an analog system whose inputs are twice differentiable signals, and whose output is the second derivative of the input:

$$f(t) \rightarrow \boxed{\text{second derivative}} \rightarrow \ddot{f}(t).$$

The eigenfunctions of this system include any signal that is of the form $A \cos(\omega t + \phi)$, $A \sin(\omega t + \phi)$, or $A e^{i(\omega t + \phi)}$, all of which have eigenvalue $-\omega^2$. We now explore the matrix–vector analog of this property.

Consider the $(N - 1) \times N$ matrix \mathbf{C} that performs a finite difference operation when multiplying a vector:

$$\mathbf{C} \triangleq \begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ & \ddots & & & & \\ 0 & \cdots & 0 & -1 & 1 & 0 \\ 0 & \cdots & 0 & 0 & -1 & 1 \end{bmatrix}, \text{ so } \mathbf{C}\mathbf{x} = \begin{bmatrix} x_2 - x_1 \\ \vdots \\ x_N - x_{N-1} \end{bmatrix} \text{ for } \mathbf{x} \in \mathbb{C}^N. \quad (4.42)$$

The $N \times N$ Gram matrix $\mathbf{C}'\mathbf{C}$ is almost Toeplitz and the related $(N - 1) \times (N - 1)$ second-difference matrix \mathbf{CC}' is exactly Toeplitz:

$$\mathbf{C}'\mathbf{C} = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ & \ddots & & & & \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}, \quad \mathbf{C}\mathbf{C}' = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ & \ddots & & & & \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix}.$$

These three important matrices arise in many signal and image processing applications, and in every field of engineering that uses a finite-element method (FEM) based on finite differences to approximate differential equations arising from physics.

One can show using trigonometric identities that for $h = \pi/N$, $\sigma_k = 2 \sin(kh/2)$, and $k = 1, \dots, N - 1$:

$$\mathbf{C}\mathbf{v}_k = \sigma_k \mathbf{u}_k, \quad \mathbf{v}_k = \sqrt{\frac{2}{N}} \begin{bmatrix} \cos(kh/2) \\ \vdots \\ \cos((2N-1)kh/2) \end{bmatrix}, \quad \mathbf{u}_k = \sqrt{\frac{2}{N}} \begin{bmatrix} -\sin(kh) \\ \vdots \\ -\sin((N-1)kh) \end{bmatrix},$$

which is a discrete analog of the fact that the derivative of \cos is $-\sin$.

It follows that an economy SVD of \mathbf{C} involves the discrete cosine transform (DCT) and discrete sine transform (DST) as follows [66]:

$$\mathbf{C} = \mathbf{U}\Sigma\mathbf{V}' = -\text{DST } \Sigma \text{ DCT}'. \quad (4.43)$$

So for this important matrix, the left and right singular vectors turn out to form matrices that are themselves important in signal processing. The facts that $\mathbf{C}'\mathbf{C} = \text{DCT } \Sigma^2 \text{ DCT}'$ and $\mathbf{CC}' = \text{DST } \Sigma^2 \text{ DST}'$ are discrete analogs of the fact that the second derivative of a sinusoid is a sinusoid.

-  Demo 4.1 visualizes this SVD.

4.5.3 Synthesis View of Matrix Decomposition

One can write both an eigendecomposition and an SVD in matrix form or summation form.

- Eigendecomposition of a square matrix (when it exists, e.g., A Hermitian):

$$A = Q\Lambda Q' = \sum_i \lambda_i q_i q_i'$$

with $Q'Q = I$, where the number of nonzero λ_i values is $r = \text{rank}(A)$.

- SVD of an $M \times N$ matrix:

$$A = U\Sigma V' = \sum_{k=1}^r \sigma_k u_k v_k'$$

where $\sigma_k > 0$ for $k = 1, \dots, r = \text{rank}(A)$, $U'U = I_M$, and $V'V = I_N$.

In both cases we can “synthesize” the matrix A using a sum of r rank-one, outer-product terms.

4.6 Orthogonal Bases

In signal processing and beyond, bases consisting of orthogonal vectors are particularly important.

Definition A collection of vectors $\{b_1, b_2, \dots\}$ in a vector space \mathcal{V} is called a **orthogonal basis** for $\mathcal{S} \subseteq \mathcal{V}$ iff

- $\{b_1, b_2, \dots\}$ is a basis for \mathcal{S} , that is,
 - $\{b_1, b_2, \dots\}$ is linearly independent,
 - $\text{span}(\{b_1, b_2, \dots\}) = \mathcal{S}$;
- the basis vectors are orthogonal, that is, $\langle b_n, b_m \rangle = 0$ for $n \neq m$.

The following property shows that the above definition *almost* has a redundancy.

Fact 4.29 (Orthogonality implies linear independence for nonzero vectors.) If $\{v_1, v_2, \dots\}$ are nonzero orthogonal vectors, then they are also linearly independent.

Proof (by contradiction). Suppose $\{v_1, v_2, \dots\}$ are orthogonal, nonzero, and also linearly dependent. Then there exist $N \in \mathbb{N}$ and $c_1, \dots, c_N \in \mathbb{F}$, not all equal to zero, such that $\mathbf{0} = \sum_{n=1}^N v_n c_n$. Pick any $m \in \{1, \dots, N\}$ and we have $0 = v_m' \mathbf{0} = v_m' \sum_{n=1}^N v_n c_n = v_m' v_m c_m = \|v_m\|_2^2 c_m \implies c_m = 0$ because v_m is nonzero. This holds for every m , contradicting the assumption that not all the c_n are zero. \square

Thus, the linear independence condition in the definition above is implied by the orthogonality condition, at least for nonzero vectors. So an equivalent definition of an orthogonal basis of \mathcal{S} is a set of *nonzero* vectors $\{b_1, b_2, \dots\}$ that are orthogonal and that $\text{span } \mathcal{S}$.

The above definition is general enough to accommodate infinite-dimensional vector spaces. In finite dimensions the situation is simpler:

Fact 4.30 The N columns of any orthogonal matrix $V \in \mathbb{R}^{N \times N}$ are an orthonormal basis for \mathbb{R}^N .

Fact 4.31 The N columns of any unitary matrix $V \in \mathbb{C}^{N \times N}$ are an orthonormal basis for \mathbb{C}^N .

Proof. The orthogonality condition ensures linear independence by the above theorem. For any vector $x \in \mathbb{F}^N$ we have

$$x = Ix = (VV')x = \underbrace{V}_{\text{basis}} \underbrace{(V'x)}_{\text{coef.}} \in \text{span}(V) \implies \text{span}(V) = \mathbb{F}^N. \quad \square$$

Example 4.46 The harmonic discrete complex exponential signals $(q_k : k = 1, \dots, N)$ are an orthonormal basis for the vector space \mathbb{C}^N , where $[q_k]_n = s(k-1, n-1)$ and $s(k, n) = (1/\sqrt{N}) e^{i2\pi kn/N}$. This basis corresponds to the (orthonormal) discrete Fourier transform (DFT) in Section 8.3. In matrix form, with $w_N = e^{i2\pi/N}$:

$$Q = [q_1, \dots, q_N] = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & \cdots & 1 & & 1 \\ \vdots & & & & \vdots \\ 1 & \cdots & w_N^{(k-1)(n-1)} & & w_N^{(N-1)(n-1)} \\ \vdots & & & & \vdots \\ 1 & \cdots & w_N^{(k-1)(N-1)} & \cdots & w_N^{(N-1)(N-1)} \end{bmatrix}. \quad (4.44)$$

Explore 4.14 Show that $\langle q_k, q_l \rangle = 0$ for $k \neq l$.

4.6.1 Finding Coordinates in an Orthogonal Basis

For $x \in \mathbb{F}^N$, its elements are coordinates in the standard basis (also known as the canonical basis):

$$x = Ix = \sum_{n=1}^N e_n x_n = \underbrace{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}}_{\text{standard basis}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}}_{\leftrightarrow \text{coordinates}}.$$

This is the most trivial orthogonal basis.

Now suppose that $Q \in \mathbb{F}^{N \times N}$ is an orthogonal matrix (or unitary matrix) and hence a basis for \mathbb{F}^N . By the definition of an orthogonal matrix (or unitary matrix) $QQ' = I$, and hence

$$\begin{aligned}
 \mathbf{x} = I\mathbf{x} &= (\mathbf{Q}\mathbf{Q}')\mathbf{x} = \underbrace{\mathbf{Q}}_{\substack{\text{basis} \\ \text{coord.}}} \underbrace{(\mathbf{Q}'\mathbf{x})}_{\substack{\text{basis} \\ \text{coord.}}} = \underbrace{\mathbf{Q}}_{\substack{\text{basis} \\ \text{coord.}}} \underbrace{\boldsymbol{\alpha}}_{\substack{\text{basis} \\ \leftrightarrow \text{coordinates}}} \quad (\text{where } \boldsymbol{\alpha} = \mathbf{Q}'\mathbf{x} \in \mathbb{F}^N) \\
 &= \sum_{n=1}^N \underbrace{\mathbf{q}_n}_{\substack{\text{basis} \\ \text{vector}}} \underbrace{a_n}_{\substack{\text{coordinate!}}}, \quad \text{where } \mathbf{Q} = [\mathbf{q}_1 \cdots \mathbf{q}_N].
 \end{aligned}$$

Alternate perspective: To write \mathbf{x} in the basis \mathbf{Q} , we want $\mathbf{x} = \mathbf{Q}\boldsymbol{\alpha}$, so we need the coordinate vector to be $\boldsymbol{\alpha} = \mathbf{Q}^{-1}\mathbf{x} = \mathbf{Q}'\mathbf{x}$ when \mathbf{Q} is unitary.

Having an orthogonal basis for \mathbb{F}^N is very convenient for finding coefficients (coordinates).

- For a general basis for \mathbb{F}^N we would need $\boldsymbol{\alpha} = \mathbf{Q}^{-1}\mathbf{x}$, requiring matrix inversion with $O(N^3)$ computation.
- For an orthogonal basis for \mathbb{F}^N , we need $\boldsymbol{\alpha} = \mathbf{Q}'\mathbf{x}$, which is simply matrix multiplication that needs $O(N^2)$ computation in general. For some bases (like DFT, DCT, OWT) it can be just $O(N \log N)$.

Explore 4.15 How do we know that the inverse \mathbf{Q}^{-1} exists for a general basis for \mathbb{F}^N ?

Note that the definition of an orthogonal basis is in terms of a set of vectors, but often we collect those vectors into a matrix and call the matrix an orthogonal basis.

One must be careful, though, about the terminology. If $\{\mathbf{b}_1, \dots, \mathbf{b}_N\}$ is an orthogonal basis for \mathbb{F}^N , the matrix $\mathbf{B} = [\mathbf{b}_1 \cdots \mathbf{b}_N]$ is not necessarily an orthogonal matrix.

Example 4.47 The vectors $\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 3 \end{bmatrix} \right\}$ form an orthogonal basis for \mathbb{R}^2 , but $\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$ is not an orthogonal matrix.

4.6.2 Stiefel Manifold of Orthogonal Bases

Orthonormal bases for K -dimensional subspaces of \mathbb{F}^M are so important that their collection has a name.

Definition The Stiefel manifold $\mathcal{V}_K(\mathbb{F}^M)$ is the set of $M \times K$ matrices having orthonormal columns

$$\mathcal{V}_K(\mathbb{F}^M) = \{\mathbf{Q} \in \mathbb{F}^{M \times K} : \mathbf{Q}'\mathbf{Q} = \mathbf{I}_K\}. \quad (4.45)$$

Clearly we must have $K \in \{1, \dots, M\}$ in this definition.

Special cases:

- $\mathcal{V}_M(\mathbb{R}^M)$ is the set of $M \times M$ orthogonal matrices;
- $\mathcal{V}_M(\mathbb{C}^M)$ is the set of $M \times M$ unitary matrices.

If $\mathbf{Q} \in \mathcal{V}_K(\mathbb{C}^M)$ then \mathbf{Q} is the first K columns of some $M \times M$ unitary matrix.

Example 4.48 When $K = 1$, $\mathcal{V}_1(\mathbb{F}^M)$ is the set of unit-norm vectors. In particular, $\mathcal{V}_1(\mathbb{R}^2)$ is essentially the unit circle and $\mathcal{V}_1(\mathbb{R}^3)$ is essentially the unit sphere.

Explore 4.16 If $Q_1, Q_2 \in \mathcal{V}_K(\mathbb{F}^M)$ with $Q_1 \neq Q_2$, then do $\mathcal{R}(Q_1)$ and $\mathcal{R}(Q_2)$ correspond to different subspaces?

Q4.26 If the $M \times N$ matrix A has SVD $A = U\Sigma V'$, then $\mathcal{R}(A) = \mathcal{R}(U)$.

A: True B: False

4.7 Spotting Decompositions

For some matrices, one can find eigenvectors “by inspection.” Recognizing these cases is a useful skill.

Example 4.49 Consider the symmetric outer product: $A = zz'$, for $z \neq \mathbf{0}$. Because

$$Az = (zz')z = z(z'z) = \underbrace{(z'z)}_{\text{scalar}} z,$$

z is an eigenvector of A with eigenvalue that is the norm squared of z : $\lambda = z'z = \|z\|_2^2$. This symmetric matrix has rank one and it has one nonzero eigenvalue (Problem 2.20).

Example 4.50 Consider the $N \times N$ matrix with $N = 2n$ where

$$A = \left[\begin{array}{cc|c} 1 & & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \\ \hline & 9 & \cdots & 9 \\ \mathbf{0} & & \vdots & \\ & 9 & \cdots & 9 \end{array} \right] = \left[\begin{array}{c} \mathbf{1}_n \\ \vdots \\ \mathbf{0}_n \end{array} \right] \left[\begin{array}{c|c} \mathbf{1}'_n & \mathbf{0}'_n \end{array} \right] + \left[\begin{array}{c} \mathbf{0}_n \\ \vdots \\ 3\mathbf{1}_n \end{array} \right] \left[\begin{array}{c|c} \mathbf{0}'_n & 3\mathbf{1}'_n \end{array} \right].$$

By inspection one can build on the previous example to see that two of the eigenvectors are:

$$\mathbf{u}_1 = \left[\begin{array}{c} \frac{1}{\sqrt{n}} \mathbf{1}_n \\ \hline \mathbf{0}_n \end{array} \right], \quad \mathbf{u}_2 = \left[\begin{array}{c} \mathbf{0}_n \\ \hline \frac{1}{\sqrt{n}} \mathbf{1}_n \end{array} \right].$$

Q4.27 What is the corresponding eigenvalue λ_1 ?

- A: 1 B: \sqrt{n} C: n D: $1/n$ E: $1/\sqrt{n}$

The second nonzero eigenvalue is nine times larger. All the other eigenvalues are zero. This symmetric matrix has rank two and it has two nonzero eigenvalues. As noted previously, in general for symmetric matrices, rank = number of (possibly nondistinct) nonzero eigenvalues.

Example 4.51 What about asymmetric matrices? $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = e_1 e_2'$ has $\lambda_1 = \lambda_2 = 0$ but $r = 1$.

For some matrices, one can find an SVD by inspection.

Example 4.52 Consider the $M \times N$ rank-one outer-product matrix $A = bc'$ with $b \neq 0_M$ and $c \neq 0_N$. Clearly,

$$\begin{aligned} A = bc' &= \underbrace{\frac{b}{\|b\|}}_{\substack{u_1 \\ "M \times 1"}}, \underbrace{\|\boldsymbol{b}\| \|\boldsymbol{c}\|}_{\sigma_1 \text{ "1 x 1"}}, \underbrace{\left(\frac{\boldsymbol{c}}{\|\boldsymbol{c}\|}\right)'_{v_1' \text{ "1 x N"}}}_{\text{compact SVD}} \\ &= \underbrace{\left[\frac{\boldsymbol{b}}{\|\boldsymbol{b}\|} \quad \boldsymbol{U}_0\right]}_{\substack{U \\ M \times M}}, \underbrace{\begin{bmatrix} \|\boldsymbol{b}\| \|\boldsymbol{c}\| & \boldsymbol{0}_{1 \times N-1} \\ \boldsymbol{0}_{M-1 \times 1} & \boldsymbol{0}_{M-1 \times N-1} \end{bmatrix}}_{\Sigma}, \underbrace{\left[\frac{\boldsymbol{c}}{\|\boldsymbol{c}\|} \quad \boldsymbol{V}_0\right]'_{\boldsymbol{V} \text{ N x N}}}_{\text{(full) SVD}}, \end{aligned} \quad (4.46)$$

where \boldsymbol{U}_0 is an $M \times (M - 1)$ matrix with orthonormal columns that span the subspace $\text{span}^\perp(b)$ and \boldsymbol{V}_0 is an $N \times (N - 1)$ matrix with orthonormal columns that span the subspace $\text{span}^\perp(c)$.

Explore 4.17 Is this SVD unique?

If A is tall ($M > N$), then the economy SVD would be similar to the full SVD except that \boldsymbol{U} involves only the first $N - 1$ columns of \boldsymbol{U}_0 and the inner matrix would be $N \times N$ and all zeros except for σ_1 in the upper left.

4.7.1 Matrix–Vector Products and the SVD

Consider the matrix–vector product

$$Ax = U\Sigma V'x = U\Sigma \underbrace{(V'x)}_{\text{coordinates of } x \text{ in term of basis } V}.$$

Define the coordinates (coefficients) to be $\alpha = V'x$ and expand the product:

$$Ax = \sum_{k=1}^r \underbrace{\sigma_k}_{\text{gain coord.}} \underbrace{\alpha_k}_{\text{ }} \underbrace{u_k}_{\text{left singular vector (basis vector)}}.$$

In particular, if $A \in \mathbb{F}^{M \times N}$ and $x = v_n$, the n th right singular vector for some $n \in \{1, \dots, N\}$, then $\alpha = V'x = e_n \implies Ax = Av_n = \sigma_n u_n$. There are numerous such properties (Problem 3.11).

4.8 Application: Signal Classification

In classification problems we have a test vector $v \in \mathcal{V}$ that we want to classify into one of C classes. One classification method first defines subsets X_1, \dots, X_C of \mathcal{V} for each of the classes (usually learned from labeled training data), and then classifies v by finding the closest subset. (This process relates to the k -nearest neighbors algorithm for $k = 1$.)

4.8.1 Projection onto a Set

The process of “finding the closest subset” involves first finding the nearest point to v in each set.

Definition The nearest point to $v \in \mathcal{V}$ in a nonempty closed set $X \subset \mathcal{V}$, also called the projection of v onto X , is

$$\hat{v} = \mathcal{P}_X(v) \triangleq \arg \min_{x \in X} \|v - x\|_2. \quad (4.47)$$

Figure 4.8 shows a two-dimensional example.

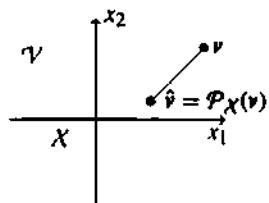


Figure 4.8 Project onto a set.

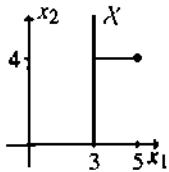


Figure 4.9 Projection example.

By default we use the Euclidean norm here, though one could also use other norms from Section 6.2. Usually we focus on convex sets; more on that in Section 4.9.1. In finite-dimensional vector spaces, a projection onto a nonempty closed set always exists [67, Theorem 2.2], but need not be unique in general.

Having defined the projection operation \mathcal{P}_X , the expression for the nearest-subset classifier is

$$\hat{c} = \hat{c}(v) = \arg \min_{c \in \{1, \dots, C\}} \|v - \mathcal{P}_{X_c}(v)\|_2. \quad (4.48)$$

Q4.28 Let $\mathcal{V} = \mathbb{R}^2$ and $X = \{x \in \mathbb{R}^2 : x_1 = 3\}$ (see Fig. 4.9). Determine $\mathcal{P}_X(v)$ when $v = (5, 4)$.

- A: (3, 3) B: (3, 4) C: (4, 4) D: (5, 4) E: (5, 3)

4.8.1.1 Properties of a Projection Operation

- For a subset $X \subset \mathcal{V}$, $\mathcal{P}_X: \mathcal{V} \mapsto X$.
- $\mathcal{P}_X(\mathcal{P}_X(v)) = \mathcal{P}_X(v)$, $\forall v \in \mathcal{V}$; that is, \mathcal{P}_X is an idempotent operation:

$$\mathcal{P}_X \circ \mathcal{P}_X = \mathcal{P}_X, \quad (4.49)$$

where \circ denotes function composition. In fact, (4.49) is the defining property of a projection.

Example 4.53 For the example shown in Fig. 4.9, $\mathcal{P}_X(3, 4) = (3, 4)$.

4.8.2 Nearest Point in a Subspace

In some classification problems, we model each class using a set S_c that is a subspace of \mathcal{V} . Thus, for classification we need to find the point in a subspace $S \subseteq \mathcal{V}$ that is nearest to a test vector $v \in \mathcal{V}$.

Fact 4.32 The projection (4.47) onto a subspace is a linear operation. (See (5.65) in Section 5.9.3; see also Problem 4.16.)

Specifically, when S is a subspace of the N -dimensional vector space $\mathcal{V} = \mathbb{R}^N$, then there is an $N \times N$ matrix P_S such that:

$$\mathcal{P}_S(v) = P_S v \quad \forall v \in \mathcal{V}. \quad (4.50)$$

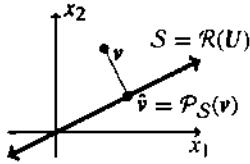


Figure 4.10 Project onto S .

Note the distinction between the projection operation \mathcal{P} and the projection matrix P .

Because \mathcal{P}_S is an idempotent operation in general, per (4.49), when S is a subspace the projection matrix P_S is an idempotent matrix (Problem 2.26):

$$\forall v \in \mathcal{V}: \mathcal{P}_S(\mathcal{P}_S(v)) = \mathcal{P}_S(v) \implies \forall v \in \mathcal{V}: P_S^2 v = P_S v \implies P_S^2 = P_S.$$

To be specific, if S is a K -dimensional subspace in an N -dimensional vector space \mathcal{V} , and if B is an $N \times K$ matrix whose K columns form a basis for S , then the $N \times N$ projection matrix, as shown in (5.68), is

$$P_S = B(B'B)^{-1}B'. \quad (4.51)$$

Instead of proving this equality here, we focus on orthonormal bases that avoid needing any matrix inverse.

4.8.2.1 Subspace with Orthonormal Basis

The process of finding the nearest point in a subspace is easy when we have an orthonormal basis for it.

Suppose $U \in \mathbb{F}^{M \times K}$ has orthonormal columns, that is, $U'U = I_K$, and define the subspace $S = \mathcal{R}(U)$, for which U is an orthonormal basis. We claim that

$$\hat{v} = \mathcal{P}_S(v) = \mathcal{P}_{\mathcal{R}(U)}(v) = U(U'v). \quad (4.52)$$

Because of this property, we define the $M \times M$ projection matrix $P_S \triangleq UU'$, so that $\mathcal{P}_{\mathcal{R}(U)}(v) = P_S v$, which is consistent with (4.51); see Fig. 4.10.

Proof. Clearly, $\hat{v} \in S$ by definition. Now consider any other point $s \in S = \mathcal{R}(U) = Uw$, for some coefficients $w \in \mathbb{F}^K$. We can define $e \triangleq s - \hat{v} \in \mathcal{R}(U) \implies e = Uz$ for some $z \in \mathbb{F}^K$. Now,

$$\begin{aligned} \|v - s\|_2^2 &= \|v - (\hat{v} + e)\|_2^2 = \|(I - UU')v - Uz\|_2^2 = \|(I - UU')v\|_2^2 + \|Uz\|_2^2 \\ &\geq \|(I - UU')v\|_2^2 = \|v - \hat{v}\|_2^2, \end{aligned}$$

because $(I - UU')v \perp Uz$, showing that \hat{v} minimizes the distance. \square

Section 5.5.1 further discusses the orthogonality principle: $(I - UU')v \perp Uz \forall z \implies v - \mathcal{P}_S(v) \perp S$.

Fact 4.33 If S is a subspace in vector space \mathcal{V} , then

$$\mathcal{P}_{S^\perp} = I - \mathcal{P}_S, \quad \text{that is, } \mathcal{P}_{S^\perp}(v) = v - \mathcal{P}_S(v) \quad \forall v \in \mathcal{V}, \quad (4.53)$$

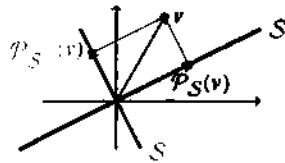


Figure 4.11 Illustrating (4.54).

where I denotes the identity function on \mathcal{V} , that is, $I(v) = v \forall v \in \mathcal{V}$.

Put another way related to (4.15) and (4.16) (see Fig. 4.11):

$$v = P_S(v) + P_{S^\perp}(v). \quad (4.54)$$

In particular, if S has orthonormal basis U then, using (4.52),

$$P_{S^\perp}(v) = v - P_S(v) = v - P_S v = v - UU'v = \underbrace{(I - UU')}_{{P_{S^\perp}}} v. \quad (4.55)$$

In this case we define the matrix analog of (4.53) to be

$$P_S^\perp \triangleq P_{S^\perp} = I - P_S = I - UU'. \quad (4.56)$$

This equation provides four notations for the projection matrix for the orthogonal complement of a subspace having orthonormal basis U . Importantly, using (4.56):

$$P_S^\perp P_S = (I - UU')(UU') = UU' - UU'UU' = UU' - UU' = 0.$$

Fact 4.34 P_S and P_S^\perp are orthogonal projection matrices, as discussed in Section 5.9 (Problem 4.17).

4.8.3 Signal Classification by Nearest Subspace

Consider an application like handwritten digit recognition where we have $C = 10$ classes of digits, and training data for each class. One method for classification is to determine subspaces S_1, \dots, S_C for each class using the SVD methods detailed in Section 7.2 (related to PCA) with corresponding orthonormal bases U_1, \dots, U_C for each class. Then we perform nearest-subspace classification of test data v by applying (4.48) with (4.52) as follows:

$$\hat{c} = \arg \min_{c \in \{1, \dots, C\}} d_c(v), \quad d_c(v) = \|v - P_{S_c} v\|_2 = \|v - U_c(U'_c v)\|_2 = \|P_{S_c}^\perp v\|_2. \quad (4.57)$$

This classification method is quite easy to implement and is illustrated in Fig. 4.12.

Q4.29 If $\mathcal{V} = \mathbb{R}^M$ and each orthonormal basis matrix U_c is $M \times K$, how many (scalar) multiplies are needed to classify one test vector $v \in \mathcal{V}$?

- A: KM B: CKM C: $2CKM$ D: $2C(KM)^2$ E: $C(2K + 1)M$

4.8.3.1 Practical Implementation of Subspace-Based Classifier

Expand (4.57) to analyze the (squared) distance of a test vector v to the subspace $\mathcal{R}(U_c)$ for the c th class:

$$\begin{aligned} d_c^2(v) &= \|v - U_c(U'_c v)\|_2^2 = \|P_{S_c^\perp} v\|_2^2 = (P_{S_c^\perp} v)' (P_{S_c^\perp} v) = v' P_{S_c^\perp} P_{S_c^\perp} v = v' P_{S_c^\perp} v \\ &= v'(I - P_{S_c})v = v'(I - U_c U'_c)v = v'v - v' U_c U'_c v = \|v\|_2^2 - \|U'_c v\|_2^2. \end{aligned} \quad (4.58)$$

Thus, in the usual case where each U_c is tall, the most efficient implementation of a subspace-based classifier is

$$\hat{c} = \arg \min_{c \in \{1, \dots, C\}} d_c^2(v) = \arg \max_{c \in \{1, \dots, C\}} \|U'_c v\|_2^2. \quad (4.59)$$

When U_c is $M \times K$, this approach requires $C(MK + K) = C(M + 1)K$ multiplies per test vector, roughly $2 \times$ faster than (4.57).

4.9 Optimization Preview

The projection operation (4.47) is one example (out of many here) of an optimization problem. Recall Fig. 3.5. In generic notation, a typical optimization problem is

$$\hat{x} = \arg \min_{x \in X} f(x), \quad (4.60)$$

where:

- X is a set of allowable values of x , for example, \mathbb{F}^N ;
- $f: \mathcal{V} \mapsto \mathbb{R}$ is a cost function that quantifies how “undesirable” any given argument x is, so we want to minimize it; and
- \hat{x} is the (or a) minimizing (best) value for x .

Often the set X is a convex set (see Section 4.9.1) and often f is a convex function. When both apply, we say (4.60) is a convex optimization problem.

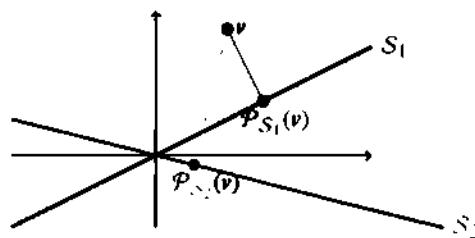


Figure 4.12 Illustration of nearest-subspace classification method.

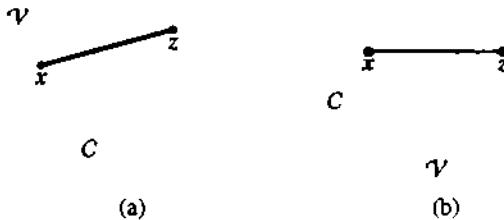


Figure 4.13 Examples of a convex set (a) and a nonconvex set (b).

If X is a strict subset of the domain of f , then we call (4.60) constrained optimization. Otherwise, we call it unconstrained optimization.

Explore 4.18 Find examples of both constrained and unconstrained optimization in Chapter 3.

4.9.1 Convex Sets

Definition A nonempty set C in a vector space V is a convex set iff, $\forall 0 \leq \alpha \leq 1$,

$$x, z \in C \implies \alpha x + (1 - \alpha)z \in C. \quad (4.61)$$

A linear combination $\alpha x + (1 - \alpha)z$ for any $0 \leq \alpha \leq 1$ is a convex combination. See Fig. 4.13.

For a convex set C , it follows by induction from (4.61) that if $x_1, \dots, x_K \in C$ and $\alpha_1, \dots, \alpha_K \geq 0$ and $\sum_{k=1}^K \alpha_k = 1$, then $\sum_{k=1}^K \alpha_k x_k \in C$. Such a sum is also called a convex combination.

Example 4.54 The nonnegative orthant $\mathbb{R}_+^N \triangleq \{x \in \mathbb{R}^N : x \geq 0\}$ is a convex set.

Example 4.55 The ball of radius $r > 0$ in \mathbb{F}^N , defined as $B_r \triangleq \{x \in \mathbb{F}^N : \|x\|_2 \leq r\}$, is a convex set (Problem 6.2).

Fact 4.35 Any subspace of a vector space is a convex set. Furthermore, any subspace of a *finite-dimensional* vector space is a closed convex set.

4.9.1.1 Projection onto Convex Sets

For a general set X , the “arg min” in the definition (4.47) of projection onto a set need not be unique. However, if C is a closed convex set (such as a subspace) in a finite-dimensional vector space V , then for any $v \in V$ the nearest point in C exists and is unique. Thus, the “arg min” in the definition (4.47) is well-defined in such cases.

Example 4.56 The projection onto the ball \mathcal{B}_r in \mathbb{F}^N defined above for $r > 0$ is given by $\mathcal{P}_{\mathcal{B}_r}(\nu) = \nu / \max(1, \|\nu\|_2/r)$.

4.9.2 Convex Functions

Definition A function $f: \mathcal{V} \mapsto \mathbb{R}$ is called a convex function on the vector space \mathcal{V} iff, for every $x, z \in \mathcal{V}$ and every $\alpha \in [0, 1]$,

$$f(\alpha x + (1 - \alpha)z) \leq \alpha f(x) + (1 - \alpha)f(z). \quad (4.62)$$

Definition A function $f: \mathcal{V} \mapsto \mathbb{R}$ is called strictly convex on the vector space \mathcal{V} iff, for every $x, z \in \mathcal{V}$ with $x \neq z$ and every $\alpha \in (0, 1)$,

$$f(\alpha x + (1 - \alpha)z) < \alpha f(x) + (1 - \alpha)f(z). \quad (4.63)$$

Example 4.57 $f(x) = |x|$ is convex and $g(x) = x^2$ is strictly convex, as shown in Fig. 4.14.

Explore 4.19 Show that the Euclidean norm function $f(x) = \|x\|_2$ on \mathbb{F}^N is a convex function.

Example 4.58 The functions $f(x) = \|Ax - b\|_2$ and $\sigma_1(X): \mathbb{F}^{M \times N} \mapsto \mathbb{R}$ are convex (Problem 4.19).

Fact 4.36 If g is a nondecreasing convex function on \mathbb{R} , and $f(x)$ is a convex function on \mathcal{V} , then $g(f(x))$ is a convex function on \mathcal{V} .

Example 4.59 Taking the nondecreasing convex function $g(r) = (\max(r, 0))^2$, it follows that $\|x\|_2^2$ and $\|Ax - y\|_2^2$ are convex functions. These two functions are particularly important in Section 5.6.5.



Figure 4.14 Convex (a) and strictly convex (b) functions.

Q4.30 When X is a subspace, are the optimization problems in (4.47) and (4.48) convex?

- A: (T,T) B: (T,F) C: (F,T) D: (F,F)

Explore 4.20 When $\mathcal{V} = \mathbb{R}$ and $X = [0, 1]$, is the projection function (4.47) a convex function?

4.10 Summary

This chapter contains many general concepts (subspace, basis, dimension, range, null space, rank). All of the definitions lead to the key “four fundamental subspaces” section that relates the null space and range of a matrix (and the orthogonal complements thereof) to components of its SVD like V_0 and U_r . One of numerous concrete applications of those SVD components is signal classification by nearest-subspace methods.

Solutions to Explorations

Explore 4.1 No, it is not closed under summation because the sum of two periodic functions with different periods whose ratio is irrational is aperiodic.

Explore 4.2 $\text{span}(\emptyset) = \mathbf{0}$ and $\dim(\emptyset) = 0$; see (4.3).

Explore 4.3 $L(\alpha u + \beta v) = (\alpha u_1 + \beta v_1) \cos(2\pi t/T) + (\alpha u_2 + \beta v_2) \sin(2\pi 5t/T + \pi/4) = \alpha L(u) + \beta L(v)$.

Explore 4.4

Suppose B has linearly dependent rows. Then there is a nonzero vector $v \in \mathbb{F}^N$ such that $v'B = \mathbf{0}$. Now define $A = \mathbf{1}_M v'$, so that $AB = \mathbf{0}$. Here, $\mathcal{R}(A) = \text{span}(\mathbf{1})$, contradicting the fact that $\mathcal{R}(AB) = \mathbf{0}$.

Explore 4.5

No. Consider $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $A = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

Explore 4.6

$\mathcal{R}(A) = \mathbb{F}^2$ when $a \neq 0$, whereas $\mathcal{R}(A) = \text{span}(\{\begin{bmatrix} 1 \\ 0 \end{bmatrix}\})$ when $a = 0$. These are two drastically different subspaces, and deciding which one to use in numerical calculations requires a tolerance threshold when $a \approx 0$.

Explore 4.7

$\text{rank}(A) = 2 - \mathbb{I}_{\{a=3\}}$, which is a discontinuous function of a .

Explore 4.8 $\text{rank}(A) = \text{rank}(A + B - B) \leq \text{rank}(A + B) + \text{rank}(B)$, so we have $\text{rank}(A) - \text{rank}(B) \leq \text{rank}(A + B)$. Reverse the roles of A and B and combine.

Explore 4.9 $\text{rank}(\alpha A) = \text{rank}(A)$ if $\alpha \neq 0$; $\text{rank}([A \ 0]) = \text{rank}\left(\begin{bmatrix} A \\ 0 \end{bmatrix}\right) = \text{rank}(A)$; $\text{rank}([A \ B]) \geq \max(\text{rank}(A), \text{rank}(B))$; $\text{rank}\left(\begin{bmatrix} A \\ B \end{bmatrix}\right) \geq \max(\text{rank}(A), \text{rank}(B))$.

Explore 4.10
`prod(ones(0))` is 1.

Explore 4.11

For $a = 0$, $N(A) = \mathbb{F}^2$, whereas for $a \neq 0$, $N(A) = \text{span}(\{\begin{bmatrix} 1 \\ 0 \end{bmatrix}\})$. The structure of the null space is a discontinuous function of a here, just as rank can be a discontinuous function of an element of a matrix.

Explore 4.12 In JULIA 1.9.2, `nullspace(ones(3,1))` produces `1x0 Matrix{Float64}`, which is essentially equivalent to \emptyset . And `nullspace(zeros(3,1))` produces `1x1 Matrix{Float64}: 1.0`, which is correct.

Explore 4.13 $N(V'_1) \oplus N(V'_2) = \mathbb{F}^N$.

Explore 4.14 $\langle q_k, q_l \rangle = \sum_{n=0}^{N-1} w_N^{(k-1)n} (w_N^{(l-1)n})^* = \sum_{n=0}^{N-1} e^{i2\pi(k-l)n/N} = 0$, $k \neq l$.

Explore 4.15 A basis is a linearly independent set, so Q is invertible for a nonempty vector space.

Explore 4.16 Not necessarily, because if $Q_2 = -Q_1$, then $Q_1 \neq Q_2$ but their spans are identical. This is just another way of saying that subspace bases are not unique.

Explore 4.17 No, because we could use $-b$ and $-c$ to define u_1 and v_1 , and there are many choices for U_0 and V_0 .

Explore 4.18 (3.15), (3.16), and (3.23) are all constrained, whereas the Explore near Fig. 3.5 is unconstrained.

Explore 4.19 Proof using the triangle inequality (6.2d). $f(\alpha x + (1 - \alpha)z) = \|\alpha x + (1 - \alpha)z\|_2 \leq \|\alpha x\|_2 + \|(1 - \alpha)z\|_2 = \alpha \|x\|_2 + (1 - \alpha) \|z\|_2 = \alpha f(x) + (1 - \alpha) f(z)$, where the middle equality uses the fact that $\alpha \in [0, 1]$.

Explore 4.20 No, it is $\mathcal{P}_{[0,1]}(v) = \min(\max(v, 0), 1)$, which is nonconvex.

Problems

Problem 4.1 Determine the eigenvalues and eigenvectors of $A = xx' + yy'$, where $x, y \in \mathbb{F}^N$. You need only find the eigenvector(s) that correspond to nonzero eigenvalue(s). Assume that $x'y = \rho \neq 0$ and focus on the case where A has its maximum possible rank.

Hint: The desired eigenvector(s) of A must be in the range of A . Hence, any eigenvector must be linearly related to x and y . Also, write A as ZZ' for some simple matrix Z . Check your answers numerically.

Problem 4.2 Let S and T denote subspaces of a vector space \mathcal{V} . Prove or disprove the following.

- (a) If S is a subset of T or vice versa, then the union of subspaces $S \cup T$ is a subspace of \mathcal{V} .
- (b) If the union of subspaces $S \cup T$ is a subspace of \mathcal{V} , then S is a subset of T or vice versa.

Problem 4.3 (a) Verify that the set of T -periodic functions \mathcal{V} illustrated in Fig. 4.2 is indeed a vector space, that is, it is closed under vector addition and scalar multiplication.
 (b) Consider the set S of one-dimensional functions having period $T > 0$ that are band-limited to maximum frequency K/T for some $K \in \mathbb{N}$. Verify that S is a subspace in \mathcal{V} , that is, $S \subseteq \mathcal{V}$.

Problem 4.4 Verify that if $u_1, \dots, u_N \in \mathcal{V}$, then $S \triangleq \text{span}(\{u_1, \dots, u_N\})$ is a subspace of the vector space \mathcal{V} .

Problem 4.5 Let $S, T \subseteq \mathcal{V}$. Show that $S + T$ and $S \cap T$ are both subspaces of \mathcal{V} .

Problem 4.6 Let $A = xy'$, where neither x nor y is $\mathbf{0}$.

- (a) How many linearly independent columns does A have? That is, of all the sets of columns from this matrix where the set is linearly independent, what is the maximum cardinality?
- (b) What is the rank of A ?
- (c) Enter this code into JULIA:

```
<> 4.1      using LinearAlgebra: svdvals
          using Plots
          using LaTeXStrings
          n = 100
          x = randn(n); y = randn(n); A = x*y'
          s = svdvals(A)
          scatter(s, yscale = :log10, label= "", 
                  title = "singular values: log scale",
                  xlabel="i", ylabel=L"\sigma_i") # to make σ,
```

- (i) How many numerically computed singular values of A are nonzero?
- (ii) What answer do you get when you type `rank(A)`?

You will have just seen that a theoretically rank-one matrix will have multiple nonzero singular values when expressed in machine precision arithmetic; this property is due to roundoff errors (finite numerical precision).

- (d) To help locate the code for the `rank` function, type this into JULIA:

```
@less rank(ones(3,2))
```

Examine the code for the `rank` function and write down the formula used to determine the threshold (or tolerance) below which, due to roundoff errors, the code considers the singular value to be “zero.”

- (e) Determine the numerical value of the threshold when $A = [9 \ 9]$.
- (f) (Optional) Check your answer to the previous part by using the JULIA debugger to step into the `rank` function and examine the tolerance.

Problem 4.7 Let $A \in \mathbb{F}^{M \times N}$.

- (a) Suppose $W \in \mathbb{F}^{M \times M}$ and $Q \in \mathbb{F}^{N \times N}$ are each unitary matrices. Show that A and $C \triangleq WAQ$ have the same singular values. Consequently, A and C have the same rank, the same Frobenius norm, and the same operator norm. This is why the Frobenius norm and the operator norm are called unitarily invariant norms. See (6.75).
- (b) Suppose that W and Q are nonsingular but not necessarily unitary matrices. Do A and $D \triangleq WAQ$ have the same rank? Prove or give a counterexample.
- (c) Continuing (b), do A and $D = WAQ$ have the same singular values? Prove or give a counterexample.

Problem 4.8 This problem examines important properties of positive semidefinite and positive definite matrices. Recall from Section 3.6 that $B'B \succeq \mathbf{0}$ for any matrix B . For each part, you may use results from previous parts if helpful.

- (a) (Optional) Show that $B'B \succ \mathbf{0}$ if B has full column rank.
- (b) (Optional) Show that $A \succ \mathbf{0}$ implies A is invertible.
- (c) (Optional) Show that $A \succeq \mathbf{0}$ and $B \succeq \mathbf{0} \implies A + B \succeq \mathbf{0}$.
- (d) Show that $A \succ \mathbf{0}$ and $B \succeq \mathbf{0} \implies A + B \succ \mathbf{0}$.
- (e) Show that $A'A + B'B$ is invertible if B has full column rank. (This property is relevant to regularized LS problems.)
- (f) Show that $A'A + B'B$ is invertible if $N(A) \cap N(B) = \{\mathbf{0}\}$, that is, if A and B have disjoint null spaces other than $\mathbf{0}$.
- (g) (Optional) Show that $A \succ \mathbf{0}$, $B \succ \mathbf{0} \implies BAB \succ \mathbf{0}$.

Problem 4.9 Let $A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$.

- (a) Determine the null space of A , denoted by $N(A)$, and the range space or column space of A , denoted by $R(A)$.

- (b) Are they equal? Does your answer hold in general? If not, provide a counterexample.

Problem 4.10 Prove (4.34), that is, that the orthogonal complement of the range of a matrix A equals the null space of A' : $\mathcal{R}^\perp(A) = \mathcal{N}(A')$.

Problem 4.11 For any matrix A , show that $\mathcal{N}(A)$ is a subspace.

Problem 4.12 This problem establishes a *necessary* and *sufficient* condition for the range of a matrix product AB to equal the range of A . Let the nonzero matrix $A \in \mathbb{F}^{M \times N}$ have compact SVD $A = U_r \Sigma_r V_r'$. Let B be a $N \times K$ matrix. Show that $\mathcal{R}(AB) = \mathcal{R}(A)$ if and only if $\mathcal{R}(V_r' B) = \mathbb{F}^K$.

Problem 4.13 For the small tall matrix $A = [4 \ 2; 2 \ 1; 0 \ 0]$, compute the compact, economy, and full SVDs and look at numerical effects by evaluating the recovery error: `norm(A - U * Diagonal(s) * V')`.

Problem 4.14 Use the fact that $\mathcal{R}(A) = \mathcal{R}(U_r)$ to modify the compact SVD code in Section 4.5 to write a function `matrixrange(A)` that returns an orthonormal basis for the range of a given matrix, akin to how `nullspace` returns a null space basis.

Problem 4.15 Spherical manifold optimization problems.

- (a) Suppose $A \in \mathbb{F}^{M \times N}$ has rank $0 < r \leq \min(M, N)$ and SVD $A = U \Sigma V' = \sum_{k=1}^r \sigma_k u_k v_k'$.
- $x_{\text{opt}} = \arg \max_{\|x\|_2=1} \|Ax\|_2 = ?$
 - $y_{\text{opt}} = \arg \max_{\|y\|_2=1} \|A'y\|_2 = ?$
 - When are x_{opt} and y_{opt} unique (to within a sign ambiguity)?
 - Do the answers change if you replace A with $-A$ in the optimization problems? Explain why or why not.
 - What constraints could you add to the above manifold optimization problem (with the same objective function) so you get u_r and v_r ?
- (b) Suppose $A \in \mathbb{F}^{N \times N}$ is Hermitian, and has eigendecomposition $A = V \Lambda V' = \sum_{k=1}^N \lambda_k v_k v_k'$, with descending eigenvalue ordering $\lambda_1 \geq \dots \geq \lambda_N \in \mathbb{R}$. Define $x_{\text{opt}} = \arg \max_{\|x\|_2=1} x' A x$.
- Prove that $x_{\text{opt}} = z v_1$ where $|z| = 1$.
 - What is $x_{\text{opt}}' A x_{\text{opt}}$?
 - When is x_{opt} unique (aside from the sign ambiguity)?
- (c) Using the same A as in (b), for some $1 < K \leq r$, define:

$$x_{\text{opt}} = \arg \max_{\|x\|_2=1} x' A x \text{ subject to } x \perp v_1, x \perp v_2, \dots, x \perp v_{K-1}.$$

- Prove that $x_{\text{opt}} = z v_K$ where $|z| = 1$, via an equivalent formulation involving projections.
- What is $x_{\text{opt}}' A x_{\text{opt}}$?
- When is x_{opt} unique (aside from the sign ambiguity)?

Problem 4.16 For this problem you will show that projecting onto a subspace $\mathcal{S} \subseteq \mathbb{F}^N$ is a linear operation; that is, there exists an $N \times N$ matrix P such that $\mathcal{P}_{\mathcal{S}}(\mathbf{v}) = P\mathbf{v} \forall \mathbf{v} \in \mathbb{F}^N$.

- Using the subspace decomposition theorem (4.14) with (4.17), any $\mathbf{v} \in \mathcal{V}$ has a unique decomposition $\mathbf{v} = \mathbf{x} + \mathbf{z}$, where $\mathbf{x} \in \mathcal{S}$ and $\mathbf{z} \in \mathcal{S}^\perp$. Show that $\mathcal{P}_{\mathcal{S}}(\mathbf{v}) = \mathcal{P}_{\mathcal{S}}(\mathbf{x} + \mathbf{z}) = \mathbf{x}$.
- Show that $\mathcal{P}_{\mathcal{S}}(\mathbf{v} + \mathbf{u}) = \mathcal{P}_{\mathcal{S}}(\mathbf{v}) + \mathcal{P}_{\mathcal{S}}(\mathbf{u})$.
- Show that $\mathcal{P}_{\mathcal{S}}(a\mathbf{v}) = a\mathcal{P}_{\mathcal{S}}(\mathbf{v})$.
- Describe how to construct the matrix P from the projection operator $\mathcal{P}_{\mathcal{S}}(\cdot)$.
- Find P when $\mathcal{V} = \mathbb{F}^3$ and $\mathcal{S} = \text{span}\left(\left\{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}\right\}\right)$.

Problem 4.17 When \mathcal{S} is subspace with orthonormal basis U , verify that $P_{\mathcal{S}}$ and $P_{\mathcal{S}}^\perp$ are orthogonal projection matrices, as discussed in Section 5.9, that satisfy the following properties:

- $P_{\mathcal{S}}^2 = P_{\mathcal{S}}$.
- $(P_{\mathcal{S}}^\perp)^2 = P_{\mathcal{S}}^\perp$.
- $P_{\mathcal{S}}' = P_{\mathcal{S}}$ and $(P_{\mathcal{S}}^\perp)' = P_{\mathcal{S}}^\perp$.
- $P_{\mathcal{S}}P_{\mathcal{S}}^\perp = \mathbf{0}$.

Problem 4.18 Referring to (4.60), determine whether the allowable sets X in (4.47) and (4.48) are convex sets as defined in Section 4.9.1.

Problem 4.19

- Show that $f(\mathbf{x}) = \|A\mathbf{x} - \mathbf{b}\|_2$ is a convex function on \mathbb{R}^N when matrix A has N columns.
Hint: Use the triangle inequality (6.2d): $\|\mathbf{x} + \mathbf{y}\|_2 \leq \|\mathbf{x}\|_2 + \|\mathbf{y}\|_2$.
- Show that the largest singular value of a matrix, that is, the function $\sigma_1(X) : \mathbb{F}^{M \times N} \mapsto \mathbb{R}$, is a convex function of the elements of the $M \times N$ matrix X .
Hint: Use (3.16).
- (Optional) Prove or disprove that the second singular value of a matrix, that is, the function $\sigma_2(X) : \mathbb{F}^{M \times N} \mapsto \mathbb{R}$, is a convex function of the elements of the $M \times N$ matrix X when $\min(M, N) \geq 2$.

5 Linear Least-Squares Regression and Binary Classification

5.1 Introduction

Many applications require solving a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{y}$ for \mathbf{x} given \mathbf{A} and \mathbf{y} . In DS–ML–SP applications, there is often no exact solution for \mathbf{x} , so one seeks an approximate solution. This chapter focuses on least-squares formulations of this type of problem. Section 5.2 briefly reviews the $\mathbf{A}\mathbf{x} = \mathbf{y}$ case and then motivates the more general $\mathbf{A}\mathbf{x} \approx \mathbf{y}$ cases. Section 5.3 then focuses on the over-determined case where \mathbf{A} is tall, emphasizing the insights offered by the SVD of \mathbf{A} . Section 5.4 introduces the pseudoinverse that is especially important in Section 5.5 for the under-determined case where \mathbf{A} is wide. Section 5.6 describes alternative approaches for the under-determined case such as Tikhonov regularization, and Section 5.7 summarizes all the linear least-squares (LLS) methods. Section 5.8 introduces frames, a generalization of unitary matrices. Section 5.9 uses the SVD analysis of this chapter to describe projection onto a subspace, completing the subspace-based classification ideas introduced in Section 4.8, and also introduces a least-squares approach to binary classifier design. Section 5.10 introduces recursive least-squares methods that are important for streaming data.

5.2 Introduction to Linear Equations

Solving a system of linear equations arises in numerous applications. Mathematically, a system of M equations in N unknowns is usually written

$$\mathbf{A}\mathbf{x} = \mathbf{y}, \quad \mathbf{A} \in \mathbb{F}^{M \times N}, \quad \mathbf{x} \in \mathbb{F}^N, \quad \mathbf{y} \in \mathbb{F}^M. \quad (5.1)$$

Typically \mathbf{A} is known from some type of (linear) modeling, \mathbf{y} corresponds to some type of measurements, and the goal is to solve for \mathbf{x} . However, despite the equals sign in the classic expression (5.1), in practice often there does not exist any \mathbf{x} that yields equality, particularly when \mathbf{A} is tall, so the notation $\mathbf{A}\mathbf{x} \approx \mathbf{y}$ would be more realistic. Sometimes there is not a *unique* \mathbf{x} that satisfies (5.1), in particular whenever \mathbf{A} is wide. Linear algebra texts focus on examining when a solution \mathbf{x} exists and is unique for (5.1).

5.2.1 Solving $Ax = y$

Fact 5.1 Key existence and uniqueness results for (5.1) are the following [4, Theorem 6.1].

- There exists a solution x iff $y \in \mathcal{R}(A)$.
- There exists a solution x for all $y \in \mathbb{F}^M$ iff $\mathcal{R}(A) = \mathbb{F}^M$.
- A solution x is unique iff $N(A) = \{\mathbf{0}\}$.
- There exists a unique solution for all $y \in \mathbb{F}^M$ iff A is $M \times M$ and nonsingular (invertible), that is, none of the eigenvalues or singular values of A are zero.
- There is at most one solution for all $y \in \mathbb{F}^M$ iff A has linearly independent columns, that is, $N(A) = \{\mathbf{0}\}$, and this is possible only if $M \geq N$.
- The homogeneous system $Ax = \mathbf{0}$ has a nontrivial (i.e., nonzero) solution iff $\text{rank}(A) < N$.

To understand the third result, suppose $x_0 \in N(A)$ and $x_0 \neq \mathbf{0}$, and suppose we have a solution $Ax_1 = y$. Then $x_2 = x_1 + x_0$ is also a solution because $Ax_2 = Ax_1 + Ax_0 = y + \mathbf{0}$.

When A is wide ($M < N$), $N(A)$ is nontrivial and there are (infinitely) many solutions to (5.1). One must design some way to choose among all those solutions.

When A is square ($M = N$) and full rank (and hence invertible), then there is a unique solution $x = A^{-1}y$. We say that “the number of equations” equals “the number of unknowns.”

5.2.2 Linear Regression and Machine Learning

In linear regression, we are given a set of training data consisting of M input (feature) vectors $a_1, \dots, a_M \in \mathbb{F}^N$ and corresponding responses $y_1, \dots, y_M \in \mathbb{F}$, and we want to find coefficients/weights $x \in \mathbb{F}^N$ such that $a_m^T x \approx y_m$. Stacking up the input vectors into a matrix A and the response variables into a vector y yields the following matrix-vector form:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} \approx \begin{bmatrix} a_1^T \\ \vdots \\ a_M^T \end{bmatrix} x, \quad \text{that is, } y = Ax + \epsilon, \quad (5.2)$$

where $\epsilon \in \mathbb{F}^M$ denotes noise or other errors.

We want to determine (“learn”) the weight vector, or coefficients, x so that given some future feature vector a , we can predict the corresponding (unknown) response by simply computing $a^T x$. This is a classic problem in statistics and it is a key “machine learning” method that one usually should consider (for estimation/regression problems) before considering more complicated methods.

The notation for linear regression varies by discipline. In statistics: $y \approx X\beta$; in machine learning: $y \approx Xw$; linear algebra: $y \approx Ax$.

The variables here have many different names:

- y : response, labels, regressand, endogenous variable, measured variable, criterion variable, dependent variable, predicted variable, ...
- a (rows of A): features, regressors, exogenous variables, explanatory variables, covariates, input variables, predictor variables, independent variables, ...
- x : parameter vector, regression coefficients, unknown, ...
- ϵ : noise, disturbance, error, ...

Example 5.1 Predict a child's height from their parent's height [68].

Example 5.2 Handwritten digit classification using two "hand-crafted" features (plus a bias or offset). The first feature is the image mean (tends to be larger for 0s than 1s); the second feature is the mean of a middle image column (tends to be larger for 1s than 0s). In math, if $f_m[i,j]$ denotes the m th training image of size $K \times K$, with pixel coordinates i,j , then $a'_m = [\sum_{i,j} f_m[i,j], \sum_j f_m[K/2,j], 1]$. Here, A is $M \times 3$, where M is the number of training examples, and $y_m = \pm 1$ for the two class labels. See Section 5.9.4 and Fig. 5.1.

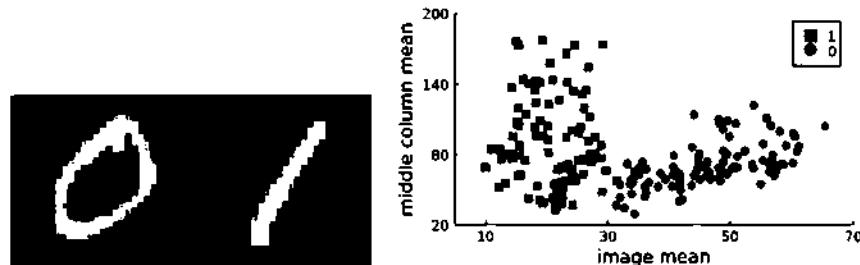


Figure 5.1 Example digits from MNIST [14], and scatter plot of hand-crafted features.

5.2.3 Lifting for Nonlinear Regression



Demo 5.1 illustrates the value of nonlinear "lifting" in regression problems. For this data, a linear model of the form $y_i = \alpha t_i + \epsilon_i$ fits poorly. Adding the nonlinear feature t_i^2 to the model, $y_i = \alpha_1 t_i + \alpha_2 t_i^2 + \epsilon_i$, leads to a much better fit. Viewed another way, after "lifting" from 1D to the 2D model with features (t_i, t_i^2) , the data approximately lies on a plane. Such nonlinearities are central to neural network models. See Section 11.2 and Fig. 5.2.

5.3 Linear Least-Squares Estimation

In a typical situation where $M > N$, called "tall" or "over-determined," one can show that $\mathcal{R}(A) \neq \mathbb{F}^M$. So there will be (infinitely) many y (those not in $\mathcal{R}(A)$) for which no

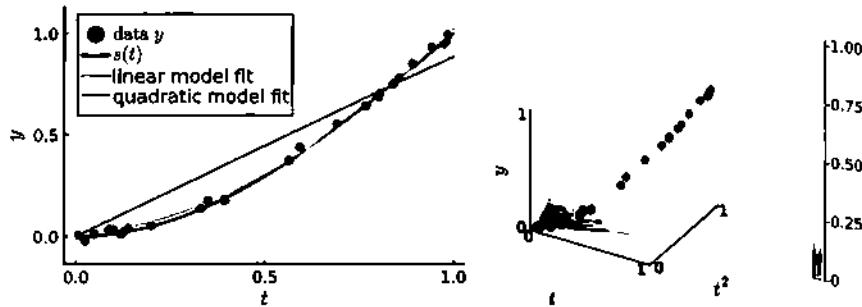


Figure 5.2 Lifting this nonlinear one-dimensional data into a higher dimension allows it to be fitted linearly.

solution \mathbf{x} exists. Thus, when $M > N$, instead of insisting on *exactly* solving $A\mathbf{x} = \mathbf{y}$, usually we look for *approximate* solutions where $A\mathbf{x} \approx \mathbf{y}$. To do this, we must quantify “approximately equal.”

The most important and common approximate solution is to use linear least-squares (LLS) estimation or fitting, where we find an estimate $\hat{\mathbf{x}}$ that “best fits” the data \mathbf{y} using a Euclidean norm distance as follows:

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathbb{F}^N} \frac{1}{2} \|A\mathbf{x} - \mathbf{y}\|_2^2. \quad (5.3)$$

- $\hat{\mathbf{x}}$ is called the linear least-squares estimate (or solution). (The “hat” or caret above \mathbf{x} is often used to denote an estimate.)
- $A\hat{\mathbf{x}} - \mathbf{y}$ is called the residual.
- $\|A\mathbf{x} - \mathbf{y}\|_2$ is the Euclidean norm of the residuals for a candidate solution \mathbf{x} and is a measure of the “error” of the fit. It is sometimes called the “goodness of fit,” even though a larger value means a worse fit!
- $\arg \min_{\mathbf{x}}$ means that we seek the argument $\hat{\mathbf{x}}$ that minimizes the fitting error; the minimum value of the fitting error, $\min_{\mathbf{x} \in \mathbb{F}^N} \|A\mathbf{x} - \mathbf{y}\|_2^2$, is usually of less interest. In words, we often say “we minimized the squared error” but we really mean “we found a solution $\hat{\mathbf{x}}$ that minimized the squared error.”

Although this technique is ancient [69], it remains one of the most important linear algebra methods for DS–ML–SP data analysis.

Example 5.3 Suppose we observe noisy samples of signal:

$$y_m = s(t_m) + \epsilon_m, \quad m = 1, \dots, M,$$

and we believe that the signal is a cubic polynomial,

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \quad (5.4)$$

with unknown coefficients. In matrix–vector form:

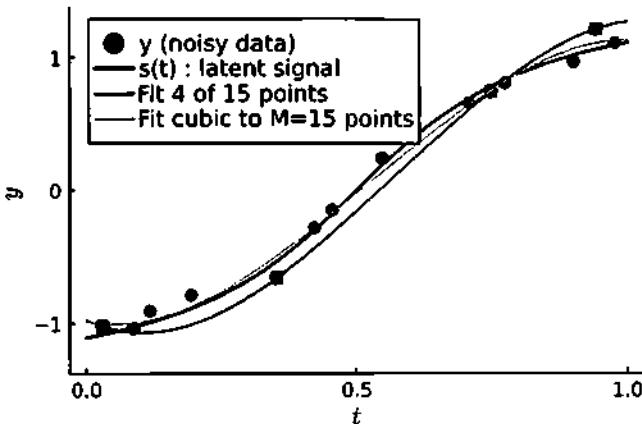


Figure 5.3 Fitting a cubic polynomial to four data points (red) using (5.1), compared to fitting all $M = 15$ points (magenta) by LS using (5.3).

$$y \approx Ax, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}, \quad A = \begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_M & t_M^2 & t_M^3 \end{bmatrix}, \quad x = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}, \quad Ax = \begin{bmatrix} \alpha_0 + \alpha_1 t_1 + \alpha_2 t_1^2 + \alpha_3 t_1^3 \\ \alpha_0 + \alpha_1 t_2 + \alpha_2 t_2^2 + \alpha_3 t_2^3 \\ \vdots \\ \alpha_0 + \alpha_1 t_M + \alpha_2 t_M^2 + \alpha_3 t_M^3 \end{bmatrix}.$$

Figure 5.3 illustrates this problem. Here, the “features” are the time points t_m raised to different powers, and we call it polynomial regression.

To find the coefficient vector x , one option would be to take just $M = 4$ samples. As long as we pick four distinct t_m points then one can show (using the fact that monomials are linearly independent functions) that the 4×4 matrix A_4 has linearly independent columns. Thus, A_4 is invertible and we could use $\hat{x} = A_4^{-1}y$ as an estimate of the coefficients.

However, in the presence of noise in the data, this approach would give very noisy and unreliable estimates of the coefficients. Instead, it is preferable to use all $M \gg 4$ samples and estimate \hat{x} using linear least-squares.

For higher polynomial orders, it is more stable to use orthogonal polynomials as the basis, instead of monomials. We use monomials here for simplicity in this example.

- ➊ Demo 5.2 illustrates polynomial regression and generates Fig. 5.3.

Figure 5.3 illustrates that fitting a cubic polynomial using all $M \gg 4$ points via (5.3) is better than using just four points with A_4^{-1} . (One can prove that statement statistically.)

A key line in the demo code is $xh = A \setminus y$, and next we explain in detail the very important mathematical foundation behind that computation.

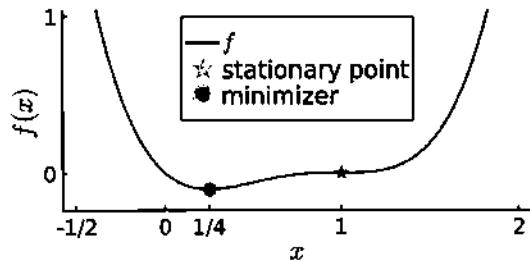


Figure 5.4 Stationary points of the one-dimensional function $f(x) = x(x - 1)^3$.

5.3.1 Minimization and Gradients

The cost function in (5.3) is differentiable, so we can use derivatives to find a minimizer, as reviewed next.

5.3.1.1 Review of 1D Minimization by Example

To find the minimizer of a function $f: \mathbb{R} \mapsto \mathbb{R}$ such as $f(x) = x(x - 1)^3$, first take the derivative and equate it to zero:

$$0 = \dot{f}(x) = (x - 1)^2(4x - 1).$$

This case has two roots, at $x = 1$ and $x = 1/4$, called stationary points. Because $f(1) = 0$ and $f(1/4) < 0$ it looks like $x = 1/4$ is the minimizer; see Fig. 5.4. To verify that $x = 1/4$ is a minimizer, we also take the second derivative:

$$\ddot{f}(x) = 6(2x^2 - 3x + 1) \implies \ddot{f}(1/4) = 9/4 > 0,$$

so $x = 1/4$ is at least a local minimizer. Checking $\pm\infty$ verifies. This example required extra work because f is nonconvex. For convex differentiable functions, simply finding a point x where $\dot{f}(x) = 0$ is sufficient.

5.3.1.2 Calculating Gradients

Similarly, to solve the minimization problem (5.3), we will find a zero of the gradient of its cost function. Here we review gradient operations.

Consider the affine function $f_1: \mathbb{R}^N \mapsto \mathbb{R}$ defined for a vector $v \in \mathbb{R}^N$ and $b \in \mathbb{R}$ by

$$f_1(x) \triangleq b + v'x = b + \sum_{n=1}^N v_n x_n.$$

The (column) gradient of this function is the vector of its partial derivatives:

$$\nabla f_1(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1 \\ \vdots \\ \frac{\partial}{\partial x_N} f_1 \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} = v \implies \boxed{\nabla_x (b + v'x) = v}. \quad (5.5)$$



(There are other layout conventions in matrix calculus.)

Consider the quadratic function $f_2: \mathbb{R}^N \mapsto \mathbb{R}$ defined for a matrix $M \in \mathbb{R}^{N \times N}$ by

$$f_2(\mathbf{x}) \triangleq \frac{1}{2} \mathbf{x}' M \mathbf{x} = \frac{1}{2} \sum_{k=1}^N \sum_{l=1}^N M_{kl} x_k x_l.$$

The partial derivative of this function with respect to x_n is

$$\frac{\partial}{\partial x_n} f_2(\mathbf{x}) = \frac{\partial}{\partial x_n} \frac{1}{2} \sum_{k=1}^N \sum_{l=1}^N M_{kl} x_k x_l.$$

Deliberately choosing different letters k, l for the summations, instead of n , avoids mishaps.

Using the linearity of differentiation and defining $\mathbb{I}_{\{\text{expr}\}} \triangleq \begin{cases} 1 & \text{if expr true} \\ 0 & \text{if expr false} \end{cases}$ we have:

$$\begin{aligned} \frac{\partial}{\partial x_n} f_2(\mathbf{x}) &= \frac{1}{2} \sum_{k=1}^N \sum_{l=1}^N M_{kl} \left(\frac{\partial}{\partial x_n} x_k x_l \right) \\ &= \frac{1}{2} \sum_{k=1}^N \sum_{l=1}^N M_{kl} (x_l \mathbb{I}_{\{n=k \neq l\}} + x_k \mathbb{I}_{\{n=l \neq k\}} + 2x_n \mathbb{I}_{\{n=k=l\}}) \\ &= \frac{1}{2} \left(\sum_{k=1}^N \sum_{l=1}^N M_{kl} x_l \mathbb{I}_{\{n=k \neq l\}} + \sum_{k=1}^N \sum_{l=1}^N M_{kl} x_k \mathbb{I}_{\{n=l \neq k\}} + 2 \sum_{k=1}^N \sum_{l=1}^N M_{kl} x_n \mathbb{I}_{\{n=k=l\}} \right) \\ &= \frac{1}{2} \left(\sum_{l=1}^N M_{nl} x_l \mathbb{I}_{\{n \neq l\}} + \sum_{k=1}^N M_{kn} x_k \mathbb{I}_{\{n \neq k\}} + 2M_{nn} x_n \right) \\ &= \frac{1}{2} \sum_{l=1}^N M_{nl} x_l + \frac{1}{2} \sum_{k=1}^N M_{kn} x_k \\ &= \frac{1}{2} [\mathbf{M}\mathbf{x}]_n + \frac{1}{2} [\mathbf{M}'\mathbf{x}]_n, \end{aligned}$$

where $[\mathbf{A}\mathbf{x}]_i$ denotes the i th element of the vector $\mathbf{A}\mathbf{x}$. (In JULIA: $(\mathbf{A}^*\mathbf{x})[i]$.) Now write the gradient of this function in column form by stacking up all the partial derivatives into a vector:

$$\nabla f_2(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f_2 \\ \vdots \\ \frac{\partial}{\partial x_N} f_2 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} [\mathbf{M}\mathbf{x}]_1 + [\mathbf{M}'\mathbf{x}]_1 \\ \vdots \\ [\mathbf{M}\mathbf{x}]_N + [\mathbf{M}'\mathbf{x}]_N \end{bmatrix} = \frac{1}{2} (\mathbf{M}\mathbf{x} + \mathbf{M}'\mathbf{x}). \quad (5.6)$$

5.3.1.3 Summary of Key Gradients Needed for LS Problems

- Constant: $\nabla c = \mathbf{0}$.
- Linear: $\nabla v' \mathbf{x} = v$.
- Quadratic: $\nabla \frac{1}{2} \mathbf{x}' M \mathbf{x} = \frac{1}{2} (M + M') \mathbf{x}$; if M is symmetric, $\nabla \frac{1}{2} \mathbf{x}' M \mathbf{x} = M \mathbf{x}$.

These apply to gradients with respect to a vector \mathbf{x} . For differentials with respect to a matrix, see [70].

5.3.1.4 Gradient for the LLS Problem

Now consider the LLS cost function corresponding to (5.3) for the real-valued case with $f: \mathbb{R}^N \mapsto \mathbb{R}$:

$$f(\mathbf{x}) = \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2 = \frac{1}{2} (\mathbf{x}' \mathbf{A}' \mathbf{A} \mathbf{x} - 2\mathbf{y}' \mathbf{A} \mathbf{x} + \mathbf{y}' \mathbf{y}). \quad (5.7)$$

Applying the linearity of differentiation and the three gradients derived above, the (column) gradient of the cost function f with respect to \mathbf{x} is:

$$\nabla f(\mathbf{x}) = \mathbf{A}' \mathbf{A} \mathbf{x} - \mathbf{A}' \mathbf{y} = \mathbf{A}' (\mathbf{Ax} - \mathbf{y}). \quad (5.8)$$

5.3.1.5 Complex Case

The above derivation is applicable only when \mathbf{x}, \mathbf{y} , and \mathbf{A} are all real-valued. When they are complex-valued, then strictly speaking the LLS cost function f is not differentiable! Instead of working with the gradient, in the complex case we work with the conjugate cogradient, using methods from Wirtinger calculus [71, 72, 73]. One intermediate step above is different, using (2.32):

$$f(\mathbf{x}) = \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2 = \frac{1}{2} (\mathbf{x}' \mathbf{A}' \mathbf{A} \mathbf{x} - 2 \operatorname{real}\{\mathbf{y}' \mathbf{A} \mathbf{x}\} + \mathbf{y}' \mathbf{y}), \quad (5.9)$$

but conveniently the final expression for the conjugate cogradient is identical to (5.8):

$$\nabla f(\mathbf{x}) = \mathbf{A}' (\mathbf{Ax} - \mathbf{y}). \quad (5.10)$$

The only difference is that here \mathbf{A}' truly denotes a Hermitian transpose, whereas in (5.8) it is simply an ordinary matrix transpose.

We use (5.8) hereafter, knowing that it applies to both real and complex-valued problems. And we will call it the “gradient” even though technically it is the “conjugate cogradient” in the complex case.

5.3.2 Solving LLS Using the Normal Equations

One can show using (4.62) that the LLS cost function corresponding to (5.3),

$$f(\mathbf{x}) = \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2, \quad (5.11)$$

where $f: \mathbb{F}^N \mapsto \mathbb{R}$, is a convex function. Because f is convex and differentiable, a vector $\hat{\mathbf{x}}$ is a minimizer of f if and only if $\hat{\mathbf{x}}$ is a point where the gradient of f is zero. From (5.8), the gradient of f with respect to \mathbf{x} is

$$\nabla f(\mathbf{x}) = \mathbf{A}' (\mathbf{Ax} - \mathbf{y}).$$

Setting this gradient to zero, $\nabla f(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}} = \mathbf{0}$, and rearranging yields the normal equations:

$$\underbrace{\mathbf{A}' \mathbf{A}}_{N \times N \text{ Gram matrix}} \hat{\mathbf{x}} = \underbrace{\mathbf{A}' \mathbf{y}}_{\in \mathbb{F}^N}. \quad (5.12)$$

The set of minimizers of f in (5.3) is the set of points $\hat{\mathbf{x}}$ satisfying (5.12).

Note that we started with an $M \times N$ matrix A in (5.1) and (5.3), but the normal equations always have a square $N \times N$ Gram matrix.

If A is a wide matrix ($M < N$), then the rank of the $N \times N$ Gram matrix $A'A$ is at most M , so $A'A$ is singular. Section 5.5 uses the SVD of A to analyze this under-determined case.

If A is tall or square and has linearly independent columns, that is, has rank N , then $A'A$ is invertible.

5.3.2.1 Practical LLS Solution Using Backslash

When A has full column rank (linearly independent columns), the Gram matrix $A'A$ is invertible and the LLS cost function has a unique minimizer given by the solution to the normal equations (5.12). Specifically:

$$A \in \mathbb{F}^{M \times N} \text{ and } \underbrace{\text{rank}(A) = N}_{\Rightarrow M \geq N} \implies \hat{x} = \arg \min_{x \in \mathbb{F}^N} \|Ax - y\|_2^2 = (A'A)^{-1}A'y. \quad (5.13)$$

This expression is useful for analysis, but it is not the best computational approach!

Q5.1 It might be tempting to implement (5.13) in JULIA using one of the two following code snippets; which is likely to require less computation?

- A: `xh = inv(A'*A) * (A' * y)`
- B: `xh = inv(A'*A) * A' * y`
- C: Both always use the same computation.

Using `inv(A'*A)` is computationally inefficient. Do not use it (at least not for large problem sizes)! An exception is applications where we must solve LLS problems for many y with the same A .

A more efficient approach uses the QR decomposition of the matrix A , without ever forming the Gram matrix $A'A$. In JULIA code this is done simply by `xh = A \ y` or `xh = \(A, y)`, spoken as " A backslash y ." It is also more efficient than applying Gaussian elimination to (5.12). See [74, Section 2.3.2].

Next, we turn to the SVD for more insight into LLS problems.

5.3.3 Solving LLS Problems Using the Compact SVD

The compact SVD provides an insightful way to analyze the LLS problem (5.3). It is also a reasonable way to solve LLS problems where M and N are not too large. (Large problems require iterative methods, discussed later.) Using a compact SVD $A = U_r \Sigma_r V_r'$ from (4.40), the LLS cost function (5.7) becomes:

$$\begin{aligned} \|Ax - y\|_2^2 &= \|U_r \Sigma_r V_r' x - y\|_2^2 \\ &= \|U_r \Sigma_r V_r' x\|_2^2 - 2\text{real}\{(U_r \Sigma_r V_r' x)' y\} + \|y\|_2^2 \\ &= \|\Sigma_r V_r' x\|_2^2 - 2\text{real}\{(\Sigma_r V_r' x)' (U_r' y)\} + \|U_r' y\|_2^2 + \|y\|_2^2 - \|U_r' y\|_2^2 \end{aligned}$$

$$= \underbrace{\|\Sigma_r V_r' x - U_r' y\|_2^2}_{\text{term 1}} + \underbrace{\|y\|_2^2 - \|U_r' y\|_2^2}, \quad (5.14)$$

where we completed the square and used the fact that $\|U_r z_r\|_2 = \|z_r\|_2$ because U is unitary.

Term 2 is independent of x , so to minimize the LLS cost function we must minimize term 1:

$$\hat{x} = \arg \min_x \|\Sigma_r V_r' x - U_r' y\|_2^2. \quad (5.15)$$

If $r = 0$ (i.e., if $A = 0$) then term 1 vanishes, so we focus on the usual case where $1 \leq r \leq N$ hereafter.

Although (5.15) initially might look more complicated than the original LLS problem, it is actually simpler because Σ_r is invertible and V_r has orthonormal columns. By inspection (without taking any gradients!), one possible solution to (5.15) is

$$\hat{x} = \arg \min_x \|\Sigma_r V_r' x - U_r' y\|_2^2 = V_r \Sigma_r^{-1} U_r' y, \quad (5.16)$$

because this solution makes term 1 identically zero. To verify this,

$$\Sigma_r V_r' \hat{x} = \Sigma_r V_r' \underbrace{(V_r \Sigma_r^{-1} U_r' y)}_T = \underbrace{\Sigma_r \Sigma_r^{-1}}_T U_r' y = U_r' y.$$

Recall that Σ_r is $r \times r$ and contains the r nonzero singular values of A along its diagonal, so it is invertible. If $r = N$, that is, if A has full column rank, then $V_r = V$ and the solution (5.16) is the *unique* minimizer.

However, if $r < N$, then there are multiple minimizers. All minimizers are given by

$$\hat{x} = V_r \Sigma_r^{-1} U_r' y + V_0 z_0, \quad (5.17)$$

where $V = [V_r \ V_0]$ and z_0 is *any* vector of the appropriate length! Any such solution makes term 1 identically zero because

$$\Sigma_r V_r' \hat{x} = \Sigma_r V_r' \left(V_r \Sigma_r^{-1} U_r' y + V_0 z_0 \right) = U_r' y + \underbrace{\Sigma_r V_r' V_0 z_0}_0 = U_r' y. \quad (5.18)$$

Q5.2 When A is $M \times N$, what is the length of the vector z_0 in (5.17)?

- A: r B: $N - r$ C: $M - r$ D: N E: M

Letting $x = Vz = [V_r \ V_0] \begin{bmatrix} z_r \\ z_0 \end{bmatrix}$, so z denotes the coordinates of x in the V coordinate system, another way of writing the general solution in (5.17) is

$$\hat{x} = [V_r \ V_0] \begin{bmatrix} \Sigma_r^{-1} U_r' y \\ z_0 \end{bmatrix} = V \hat{z}, \quad \hat{z} = \begin{bmatrix} \Sigma_r^{-1} U_r' y \\ z_0 \end{bmatrix} = \begin{bmatrix} \frac{[U_r' y]_1 / \sigma_1}{z_0} \\ \vdots \\ \frac{[U_r' y]_r / \sigma_r}{z_0} \end{bmatrix}. \quad (5.19)$$

This is a general expression for “the” LLS solution in terms of a compact SVD of A and the data y . The arbitrary choice of z_0 may seem unsettling; we will address that concern shortly.

There are two ways that A can have rank $r < N$, leading to a nonunique solution:

- if A is square or tall but has linearly dependent columns;
- if A is wide, because then $r \leq \min(M, N) = M < N$.

When A is tall with full rank $r = N$, the (unique) LLS solution is

$$\hat{x} = V_r \Sigma_r^{-1} U_r' y = V \Sigma_N^{-1} U_N' y.$$

Example 5.4 We return to the example of fitting a cubic polynomial (5.4). In this case $M \gg N = 4$. As long as at least four of the (t_m) values are distinct, the fact that monomials are linearly independent functions implies that A has full rank, that is, $r = N = 4$. In this (typical) case, the SVD simplifies to

$$\begin{aligned} \underbrace{\mathbf{A}}_{M \times 4} &= \underbrace{\mathbf{U}}_{M \times M} \underbrace{\mathbf{\Sigma}}_{M \times 4} \underbrace{\mathbf{V}'}_{4 \times 4} = \underbrace{\mathbf{U}_4}_{M \times 4} \underbrace{\mathbf{\Sigma}_4}_{4 \times 4} \underbrace{\mathbf{V}'}_{4 \times 4}, \\ \mathbf{\Sigma} &= \left[\begin{array}{c} \mathbf{\Sigma}_4 \\ \hline \mathbf{0}_{(M-4) \times 4} \end{array} \right] = \left[\begin{array}{cccc} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \\ \hline \mathbf{0}_{(M-4) \times 4} \end{array} \right]. \end{aligned}$$

The solution \hat{x} in (5.17) simplifies to

$$\hat{x} = V \Sigma_4^{-1} U_4' y. \quad (5.20)$$

Because $r = N = 4$ here, this is a final and unique LLS solution; there is no arbitrary vector z_0 to select.

5.3.3.1 Practical Implementation

Returning to the JULIA demo for this example, where A is tall and $r = N$, we can implement this SVD-based solution using the following code.

Start with an SVD: $U, s, V = \text{svd}(A)$. Recall that, for this economy SVD version,

- U is $M \times N$ instead of the usual $M \times M$, that is, it is $U_r = U_N$ in (4.40);
- s is a vector of the $(\sigma_1, \dots, \sigma_N)$ values, so $\Sigma_r = \Sigma_N = \text{Diagonal}(s)$ because $r = N$ here.

Using this economy SVD, two mathematically equivalent forms of the LS solution are:

- $xh = V * \text{Diagonal}(1 ./ s) * (U' * y)$, which looks like $\hat{x} = V \Sigma_4^{-1} U_4' y$ in (5.20);
- $xh = V * ((1 ./ s) .* (U' * y))$, which looks like (5.19).

Explore 5.1 Why the parentheses $(U' * y)$?

When A is tall with full rank ($r = N$), the compact SVD (used in our mathematical analysis) and the economy SVD (returned by `svd`) are identical; that is, $U_r = U_N$, $\Sigma_r = \Sigma_N$, and $V_r = V_N = V$.

5.3.4 Uniqueness of LLS Solution

If A is $M \times N$, then by the definition of rank, $\text{rank}(A) = N$ iff A has linearly independent columns. Having $M \geq N$ is a necessary condition for linear independence of the columns of A . Conversely, if $M < N$ then $\text{rank}(A) \neq N$ because $\text{rank}(A) \leq \min(M, N) = M < N$.

5.3.4.1 Over-Determined (Tall) Case

In the frequent case where $M > N$ we have (graphically):

$$\begin{array}{c} \boxed{A} \\ \underbrace{\phantom{\boxed{A}}_{M \times N}} \end{array} \quad \boxed{x} \quad \boxed{y} = \boxed{y} \quad \underbrace{\phantom{\boxed{y}}_M}_{N}$$

In words, there are more equations than unknowns, called an over-determined system. Rarely is there an exact solution in this case due to noise in the data y , so “best-fit” solutions like LLS (5.3) are used instead.

When $M \geq N$ and A has rank $r = N$, the solution (5.17) or (5.19) has no arbitrary terms and simplifies to

$$\hat{z}_n = \underbrace{\frac{[U'y]_n}{\sigma_n}}_{n=1,\dots,N} \implies \hat{z} = \Sigma_N^{-1} U'_N y, \quad \hat{x} = V \hat{z} \implies \hat{x} = V \Sigma_N^{-1} U'_N y. \quad (5.21)$$

This is the *unique* solution to (5.3) when $\text{rank}(A) = N$.

5.3.4.2 Over-Determined Full-Rank Case Using SVD

The expression in (5.21) uses the compact SVD and it is useful to also write it in terms of the full SVD:

$$\hat{x} = V \Sigma_N^{-1} U'_N y = V \left[\begin{array}{c|c} \Sigma_N^{-1} & 0_{N \times (M-N)} \end{array} \right] \left[\begin{array}{c} U'_N \\ U'_0 \end{array} \right] y = V \left[\begin{array}{c|c} \Sigma_N^{-1} & 0_{N \times (M-N)} \end{array} \right] U' y.$$

This form is suboptimal for *computation* because the term $U'_0 y$ is computed but then multiplied by 0 . An even more concise expression is

$$M \geq N \text{ and } \text{rank}(A) = N \implies \hat{x} = V \Sigma^+ U' y = A^+ y, \quad (5.22)$$

where $\Sigma^+ \triangleq \left[\begin{array}{c|c} \Sigma_N^{-1} & 0_{N \times (M-N)} \end{array} \right]$ denotes the Moore–Penrose pseudoinverse of $\Sigma = \left[\begin{array}{c} \Sigma_N \\ \hline 0_{(M-N) \times N} \end{array} \right]$. The next section discusses this new term in detail and explains A^+ .

5.4

Moore–Penrose Pseudoinverse

Definition For any matrix $A \in \mathbb{F}^{M \times N}$, the Moore–Penrose pseudoinverse of A , denoted $A^+ \in \mathbb{F}^{N \times M}$, is a generalization of the usual notion of matrix inverse that satisfies the following four properties:

- $AA^+A = A$ (weaker than $AA^{-1} = I$)
- $A^+AA^+ = A^+$
- $(A^+A)' = A^+A$ (symmetry)
- $(AA^+) = AA^+$

Another notation used in the literature is A^\dagger .

Properties of the Moore–Penrose pseudoinverse (for proofs, see [75]):

- A^+ is unique and $(A^+)^+ = A$.
- If A is invertible, then $A^+ = A^{-1}$.
- $A^+ = (A'A)^+A'$.
- If A has full column rank (linearly independent columns), then $A^+ = (A'A)^{-1}A'$. In this case, A^+ is a left inverse because $A^+A = I_N$.
- $A^+ = A'(AA')^+$.
- If A has full row rank, (linearly independent rows), then $A^+ = A'(AA')^{-1}$. In this case, A^+ is a right inverse because $AA^+ = I_M$.
- $\mathbf{0}_{M \times N}^+ = \mathbf{0}_{N \times M}$, so in particular $0^+ = 0$.
- $(A')^+ = (A^+)'.$

5.4.1 Pseudoinverse and Matrix Products



Caution: in general, $(AB)^+ \neq B^+A^+$, so be careful with matrix products. However, for $B \in \mathbb{F}^{K \times L}$:

(P1) If Q is an $M \times K$ matrix with orthonormal columns, that is, $Q'Q = I_{K \times K}$, then $(QB)^+ = B^+Q'$.

Proof (partial):

$$(QB)(QB)^+(QB) = QB(B^+Q')QB = QBB^+B = QB,$$

$$(QB)^+QB(QB)^+ = (B^+Q')QB(B^+Q') = B^+BB^+Q' = B^+Q' = (QB)^+, \text{ etc. } \square$$

(P2) If Q is an $L \times N$ matrix with orthonormal rows, that is, $QQ' = I_{L \times L}$, then $(BQ)^+ = Q'B^+$.

$$(BQ)(BQ)^+(BQ) = BQ(Q'B^+)BQ = BB^+BQ = BQ,$$

$$(BQ)^+(BQ)(BQ)^+ = (Q'B^+)BQ(Q'B^+) = Q'B^+BB^+ = Q'B^+ = (BQ)^+, \text{ etc. } \square$$

(P3) If $A \in \mathbb{F}^{M \times N}$ has full column rank (linearly independent columns) and $B \in \mathbb{F}^{N \times K}$ has full row rank (linearly independent rows), then $(AB)^+ = B^+A^+$.

Fortunately, these product properties serve our needs.

A special case of (P1) and (P2) is when $B = I$, leading to:

- (P4) If Q is a matrix with orthonormal columns, then $Q^+ = Q'$.
 (P5) If Q is a matrix with orthonormal rows, then $Q^+ = Q$.

In the context of the compact SVD for $r > 0$, (P4) implies that $U_r^+ = U_r'$ and $V_r^+ = V_r'$. Because U_r has orthonormal columns, V_r' has orthonormal rows, and Σ_r is invertible, it follows that

$$\begin{aligned} A = U_r \Sigma_r V_r' \implies A^+ &= (U_r \Sigma_r V_r')^+ = (U_r (\Sigma_r V_r'))^+ \stackrel{(P1)}{=} (\Sigma_r V_r')^+ U_r' \\ &\stackrel{(P2)}{=} V_r \Sigma_r^+ U_r' = V_r \Sigma_r^{-1} U_r'. \end{aligned} \quad (5.23)$$

Explore 5.2 Prove or disprove: $(AB)^+ = (AB)^+ P_{\mathcal{R}(A)}$.

Example 5.5 A pseudoinverse example of particular interest when working with the SVD is that of a rectangular diagonal matrix (Problem 5.10):

$$\Sigma = \underbrace{\left[\begin{array}{c|c} \Sigma_r & \mathbf{0}_{r \times (N-r)} \\ \hline \mathbf{0}_{(M-r) \times r} & \mathbf{0}_{(M-r) \times (N-r)} \end{array} \right]}_{M \times N} \implies \Sigma^+ = \underbrace{\left[\begin{array}{c|c} \Sigma_r^{-1} & \mathbf{0}_{r \times (M-r)} \\ \hline \mathbf{0}_{(N-r) \times r} & \mathbf{0}_{(N-r) \times (M-r)} \end{array} \right]}_{N \times M}. \quad (5.24)$$

An important special case is when $r = N$; that is, A has full column rank, in which case

$$\Sigma = \underbrace{\left[\begin{array}{c} \Sigma_N \\ \hline \mathbf{0}_{(M-N) \times N} \end{array} \right]}_{M \times N} \implies \Sigma^+ = \underbrace{\left[\begin{array}{c|c} \Sigma_N^{-1} & \mathbf{0}_{N \times (M-N)} \\ \hline \hline \end{array} \right]}_{N \times M}. \quad (5.25)$$

Noting that $\Sigma^+ \Sigma = I_N$ in the tall full-rank case, it is trivial to verify the four defining properties for this special case.



Caution: In general, $A^+ A \neq I_N$ and $AA^+ \neq I_M$.

5.4.2 Pseudoinverse and SVD

Using the orthogonal matrix product properties of the pseudoinverse yields the following expression for A^+ in terms of the full SVD of A :

$$\underbrace{A}_{M \times N} = \underbrace{U}_{M \times M} \underbrace{\Sigma}_{M \times N} \underbrace{V'}_{N \times N} \implies \underbrace{A^+}_{N \times M} = \underbrace{V}_{N \times N} \underbrace{\Sigma^+}_{N \times M} \underbrace{U'}_{M \times M}. \quad (5.26)$$

The final expression is not quite an SVD of A^+ , but we can easily define an SVD of A^+ by introducing permutation matrices P_N, P_M that order the singular values in Σ^+ appropriately:

$$A^+ = V \Sigma^+ U' = (V P'_N) (P_N \Sigma^+ P'_M) (P_M U').$$

Here is one sanity check for the pseudoinverse expression (5.26):

$$AA^+A = U \Sigma \underbrace{V' V}_{\Sigma^+} \Sigma^+ \underbrace{U' U}_{\Sigma V'} \Sigma V' = U \underbrace{\Sigma \Sigma^+ \Sigma}_{\Sigma} V' = U \Sigma V' = A.$$

The key expression (5.26) simplifies to the usual matrix inverse in the rare case where A is square and invertible: $A^{-1} = V \Sigma^{-1} U'$. (However, for invertible matrices often we do not need to use the SVD.)

Expressing a pseudoinverse A^+ in terms of the compact SVD of A for $r > 0$ is also very useful:

$$A^+ = V \Sigma^+ U' = [V_r \mid V_0] \Sigma^+ \begin{bmatrix} U'_r \\ \hline U'_0 \end{bmatrix} \implies A^+ = \underbrace{V_r}_{N \times r} \underbrace{\Sigma_r^{-1}}_{r \times r} \underbrace{U'_r}_{r \times M} . \quad (5.27)$$

Q5.3 The LLS solution $\hat{x} = A^+y$ lies in which of the four fundamental spaces of A ?

- A: $N(A)$ B: $N^\perp(A)$ C: $\mathcal{R}(A)$ D: $\mathcal{R}^\perp(A)$ E: None of these

Sir Roger Penrose [76], who developed the pseudoinverse, shared the Nobel Prize in physics in 2020 for work in understanding black holes.

Q5.4 If A has linearly independent columns, then $\|A^+Ax\| = \|x\|$.

- A: True B: False

Q5.5 Let A have full SVD $A = U\Sigma V'$ and compact SVD $A = U_r \Sigma_r V'_r$, where we partition the unitary matrix V as usual as $V = [V_r \mid V_0]$. Which of the following is the tuple $(N(V'), N(V'_r))$?

- A: $(\mathbf{0}, \mathbf{0})$ B: $(\mathbf{0}, \mathcal{R}(V_0))$ C: $(\mathcal{R}(V_0), \mathbf{0})$ D: $(\mathcal{R}(V_0), \mathcal{R}(V_0))$ E: None of these

Q5.6 If matrix A has SVD $A = U\Sigma V' = U_r \Sigma_r V'_r$, then two expressions for its pseudoinverse are $A^+ = V \Sigma^+ U' = V_r \Sigma_r^{-1} U'_r$. Which of the following is the null space of A^+ , $N(A^+)$?

- A: $N(A)$ B: $N^\perp(A)$ C: $\mathcal{R}(A)$ D: $\mathcal{R}^\perp(A)$ E: None of these

Q5.7 If $A = U_r \Sigma_r V'_r$ for $r > 0$, then $A^+ = V_r \Sigma_r^{-1} U'_r$ is a compact SVD of A^+ .

- A: Always B: Sometimes C: Never

5.4.2.1 Projector/Idempotent Preview

The following matrix is called an orthogonal projector (see Section 5.9.2):

$$P_{\mathcal{R}(A')} = P_{N^\perp(A)} \triangleq A^+ A = V \Sigma^+ U' U \Sigma V' = V \Sigma^+ \Sigma V' = V_r V'_r. \quad (5.28)$$

Because $P_{\mathcal{R}(A')}P_{\mathcal{R}(A')} = P_{\mathcal{R}(A')}$, it is called idempotent. Likewise, the following matrix is also a projector (and also idempotent):

$$P_{\mathcal{R}(A)} = P_{N^\perp(A')} \triangleq AA^+ = U\Sigma V'V\Sigma^+U' = U\Sigma\Sigma^+U' = U_rU_r'. \quad (5.29)$$

5.4.2.2 Relating Pseudoinverse and Normal Equations

The following equalities (properties of the pseudoinverse) follow:

$$\begin{aligned} A'AA^+ &= A' \text{ because } A'AA^+ = V_r\Sigma_rU_r'(U_rU_r') = V_r\Sigma_rU_r' = A', \\ A^+AA' &= A' \text{ because } A^+AA' = (V_rV_r')V_r\Sigma_rU_r' = V_r\Sigma_rU_r' = A'. \end{aligned}$$

Multiplying the first of these two equalities by y yields

$$A'AA^+y = A'y \implies A'A\hat{x} = A'y, \quad (5.30)$$

so the pseudoinverse solution $\hat{x} = A^+y$ always satisfies the normal equations, regardless of the rank of A .

5.4.2.3 LLS Solution Using Pseudoinverse

Using (5.22), we write the LLS estimate (5.22) particularly concisely in the full-rank case as follows:

$$A \in \mathbb{F}^{M \times N} \text{ and } \underbrace{\text{rank}(A) = N}_{\implies M \geq N} \implies \hat{x} = \arg \min_{x \in \mathbb{F}^N} \|Ax - y\|_2^2 = A^+y. \quad (5.31)$$

This is a wonderfully elegant form on paper, but it is not the best computational approach! We could implement (5.31) in JULIA using the pseudoinverse function with `xh = pinv(A) * y`. Alternatively, we could use the `inv` form discussed on p. 151: `xh = inv(A'*A) * (A'*y)`. However, both these approaches are computationally inefficient! We rarely use `pinv` or `inv` in practice (at least for large problem sizes) unless we must solve LLS problems for many different y with the same A . Otherwise, the backslash approach `xh = A \ y` discussed on p. 151 with its more efficient QR decomposition is preferable. Nevertheless, the SVD is very helpful conceptually, especially in the under-determined case.

If A is square and does not have full column rank, then $A \backslash y$ may produce an error message like `SingularException`. Then one should first think about why A does not have linearly independent columns, and then perhaps consider using `pinv`, as discussed in Section 5.5, or one of the regularized solutions discussed in Section 5.6.

 If A is wide, then JULIA's `\` silently reverts to providing a pseudoinverse solution.

5.5 LLS: Under-Determined Case

So far we have focused on cases where $M \geq N$ and A has full rank, and derived the concise LLS solution based on the pseudoinverse (5.31). Now we examine cases where $M < N$, called under-determined, as well as cases where A is tall but rank deficient that is, $\text{rank}(A) < N$.

Example 5.6 An example application where $M < N$ is sparse-view X-ray computed tomography (CT) [77]. An X-ray CT system collects line integrals through a 3D object, and then sophisticated algorithms reconstruct images from those measurements. One way to reduce X-ray dose in CT scans is to collect fewer measurements, called sparse-view CT. Figure 5.5 shows an extremely simplified example of (2D) CT imaging where the object is described by a grid of 4×4 pixels and we collect just four parallel line integrals at three projection angles. Here, the number of unknowns is $N = 4^2 = 16$, and the number of measurements is $M = 4 \cdot 3 = 12$. We use a linear model for the measurement process, described by a sensing matrix A .

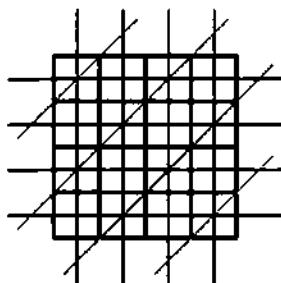


Figure 5.5 Twelve line integrals.

The sensing matrix A that corresponds to this geometry is the following 12×16 matrix, with elements color coded to correspond to Fig. 5.5:

$$A = \begin{bmatrix} I_4 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sqrt{2}I_4 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The $\sqrt{2}$ scaling is because the lines that traverse pixels diagonally have longer intersections. In real CT systems, the matrix A is too large to precompute and store, so elements of it are computed on the fly [78]. Numerous publications describe advanced regularization methods and optimization algorithms for addressing the nonuniqueness of such problems.

In cases where $\text{rank}(A) < N$, there is not a unique minimizer to the LLS cost function $\|Ax - y\|_2^2$. Using the compact SVD of A , we showed in (5.17) that any minimizer has the form

$$\hat{x} = \underbrace{V_r \Sigma_r^{-1} U_r' y}_{\hat{x}_R \in \mathcal{R}(V_r) = N^\perp(A)} + \underbrace{V_0 \hat{z}_0}_{\hat{x}_N \in \mathcal{N}(A)} = A^+ y + \hat{x}_N. \quad (5.32)$$

The choice of \hat{z}_0 is arbitrary because the residual is invariant to the null space component, cf. (5.18):

$$\begin{aligned} A\hat{x} - y &= A(V_r \hat{z}_r + V_0 \hat{z}_0) - y = A(\hat{x}_R + \hat{x}_N) - y = A\hat{x}_R + 0 - y \\ &= A\hat{x}_R - y. \end{aligned} \quad (5.33)$$

Thus, the LLS criterion by itself is insufficient to identify a unique best \hat{x} in under-determined (or rank-deficient) problems.

Explore 5.3 If A is wide, that is, $M < N$, can we have $\text{rank}(A) = N$?

To summarize:

- When $M \geq N$ and A has full rank N , the LLS solution is unique and is $\hat{x} = A^+ y$.
- Otherwise (i.e., if $M < N$ or if A has rank less than N), the LLS solution is not unique and any \hat{x} of the form $\hat{x} = A^+ y + \hat{x}_N$ where $\hat{x}_N \in \mathcal{N}(A)$ is “equally good” from the point of view of the LLS cost function $\|Ax - y\|_2^2$.

To pin down a unique solution in the under-determined case, we must introduce some additional criterion to select on \hat{x} from the (infinitely) many candidates of the form $\hat{x} = A^+ y + \hat{x}_N$. A modern approach uses a sparsity model and seeks a sparse solution; for example, find the solution \hat{x} with the fewest nonzero entries in the set $\{x \in \mathbb{F}^N : \|Ax - y\|_2^2 \leq \epsilon\}$.

We focus for now on the “classic” approach of choosing the minimum norm solution, that is, the LLS minimizer where $\|\hat{x}\|_2^2$ is the smallest. This is a “double minimization” problem, because (conceptually) we first find all the minimizers of $\|Ax - y\|_2^2$, and then among those minimizers we pick the one where $\|\hat{x}\|_2^2$ is the smallest. But first we give a geometric interpretation of the LLS solution.

5.5.1 Orthogonality Principle

To examine LLS geometrically, recall from the normal equations (5.12) that any LLS solution must satisfy

$$\begin{aligned} A'A\hat{x} = A'y &\implies A'(y - A\hat{x}) = 0 \implies A'r = 0 \\ &\implies z'A'r = 0 \implies Az \perp r \forall z \in \mathbb{F}^N \end{aligned}$$

for $A \in \mathbb{F}^{M \times N}$, where the residual after fitting is denoted $r = y - A\hat{x}$. (The normal equations are a necessary condition for \hat{x} to be a minimizer, regardless of the rank of A .)

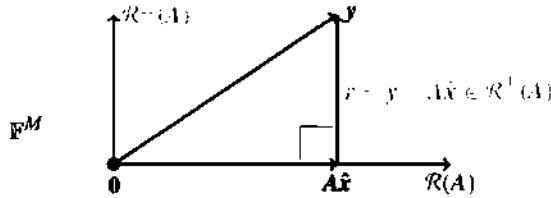


Figure 5.6 Orthogonality principle for LLS estimation.

The following orthogonality principle of LLS estimation is a direct consequence of the above:

$$\langle Az, y - Ax \rangle = (y - Ax)^T Az = 0, \text{ i.e., } Az \perp (y - Ax), \forall z \in \mathbb{F}^N. \quad (5.34)$$

In words, the residual is perpendicular to $\mathcal{R}(A)$. Figure 5.6 gives the geometric interpretation.

The following terminology is (unfortunately) often used interchangeably:

- $y - Ax$ is called the residual (from fitting);
- $y - Ax_{\text{true}}$ is called the error in the data, in situations where we believe $y = Ax_{\text{true}} + \varepsilon$;
- $\hat{x} - x_{\text{true}}$ is called the error in the parameter estimate.

Often one must determine from context which meaning of “error” is meant.

If $y \in \mathcal{R}(A)$ (which rarely happens for noisy data when $M > N$), then there is a solution with zero residual $y - Ax$. However, the parameter error $\hat{x} - x_{\text{true}}$ may still be nonzero!

Here is an alternate derivation that uses the orthogonality principle (5.34) to confirm that \hat{x} in (5.32) is optimal for LLS estimation (for A of any size or rank). Suppose $x? = A^+y + \hat{x}_N + z = \hat{x} + z$ for some arbitrary vector $z \in \mathbb{F}^N$. Could this $x?$ be a better estimator?

Using the norm of a sum in (2.32) and the orthogonality principle (5.34), the squared error criterion for such an $x?$ has the following lower bound:

$$\begin{aligned} \|Ax? - y\|_2^2 &= \|A(\hat{x} + z) - y\|_2^2 = \|(A\hat{x} - y) + Az\|_2^2 \\ &= \|A\hat{x} - y\|_2^2 + \underbrace{2 \operatorname{real}\{(A\hat{x} - y)^T Az\}}_0 + \underbrace{\|Az\|_2^2}_{\geq 0} \geq \|A\hat{x} - y\|_2^2, \end{aligned} \quad (5.35)$$

where the lower bound is achieved by $z = \mathbf{0}$. In words, the LLS fit is best when $\hat{x} = A^+y + \hat{x}_N$ for any vector $\hat{x}_N \in \mathcal{N}(A)$.

Mathematically, the set of LLS solutions is the (set!) sum of a vector and a subspace (the null space of A):

$$\begin{aligned} \{\tilde{x} \in \mathbb{F}^N : \|A\tilde{x} - y\|_2^2 \leq \|Ax - y\|_2^2 \forall x \in \mathbb{F}^N\} &= A^+y + \mathcal{N}(A) \\ &\triangleq \{\tilde{x} = A^+y + \hat{x}_N : \hat{x}_N \in \mathcal{N}(A)\}. \end{aligned} \quad (5.36)$$

The sum of a vector plus a subspace is called a linear variety, or a flat or affine subspace.

5.5.2 Minimum-Norm LS Solution via Pseudoinverse

We have seen that the set of optimal LLS estimates is $\hat{x} = A^+y + N(A)$.

- If A has full column rank, then $N(A) = \mathbf{0}$ and we have a unique solution $\hat{x} = A^+y$.
- If A does not have full column rank, then we want to pick one choice from the set of LLS estimates.

The classic way to pick one of the many possible estimates in the under-determined case is to choose the one with minimum norm.

Fact 5.2 The (unique) minimum-norm LS solution is:

$$\hat{x} \triangleq \arg \min_{x \in \{A^+y + N(A)\}} \|x\|_2^2 = A^+y. \quad (5.37)$$

This is a bilevel optimization problem [79, 80] because first we found a set of candidate solutions by finding minimizers of $\|Ax - y\|_2^2$, and then we solved a different minimization problem involving $\|x\|_2^2$ to select one final solution from that set of candidates.

Proof of (5.37). If $x = A^+y + \hat{x}_N$ where $\hat{x}_N \in N(A)$, then $\hat{x}_N \in \text{span}(V_0)$, where $V = [V_r \mid V_0]$. Using (5.27), $A^+y = V_r \Sigma_r^{-1} U'_r y \in \mathcal{R}(V_r)$. Because V is unitary, the columns of V_r and V_0 are orthogonal. Thus, $A^+y \perp \hat{x}_N$. Using (2.32):

$$\begin{aligned} \|x\|_2^2 &= \|A^+y + \hat{x}_N\|_2^2 = \|A^+y\|_2^2 + 2 \text{real}\{\hat{x}'_N A^+y\} + \|\hat{x}_N\|_2^2 \\ &= \|A^+y\|_2^2 + \|\hat{x}_N\|_2^2 \geq \|A^+y\|_2^2, \end{aligned}$$

where the minimum is achieved iff $\hat{x}_N = \mathbf{0}$. Thus the minimum norm solution is $\hat{x} = A^+y$. \square

The set over which we minimize in (5.37) is $\{A^+y + N(A)\}$.

Q5.8 What is the cardinality of this set when $\text{rank}(A) = N$ for $A \in \mathbb{F}^{M \times N}$?

A: 0 B: 1 C: r D: N E: ∞

Q5.9 What is the cardinality of this set when $M < N$?

A: 0 B: 1 C: r D: N E: ∞

In summary, we have the following fortuitous situation:

- If A has full column rank, then $N(A) = \mathbf{0}$ and we have a unique solution $\hat{x} = A^+y$.
- If A does not have full column rank, then there are multiple LLS estimates and the one with smallest Euclidean norm is $\hat{x} = A^+y$.

Hence the pseudoinverse solution has been very popular historically, and remains important today, except in many highly under-determined problems of the kind known as compressed sensing. There are quantum-computing methods for the pseudoinverse solution [81].

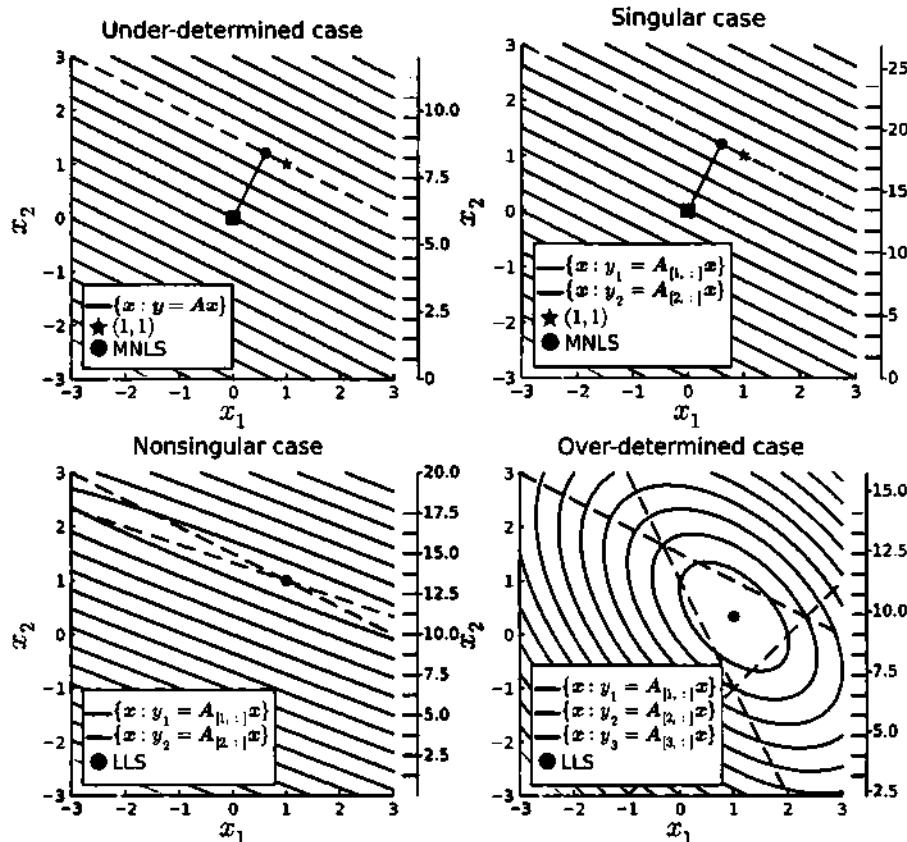


Figure 5.7 Least squares cost functions for $N = 2$ with $M \in \{1, 2, 3\}$.

- Demo 5.3 visualizes the cost function (5.7) and solution \hat{x} for $N = 2$ with $M \in \{1, 2, 3\}$; see Fig. 5.7. For the bottom two plots the minimizer is unique. For the two right plots $y \notin \mathcal{R}(A)$ and $A\hat{x} \neq y$.

Using (5.26), we can interpret the pseudoinverse solution $\hat{x} = A^+y = V\Sigma^+U'y$ as a cascade of three separate transformations:

$$y \xrightarrow{U'} \tilde{y} \xrightarrow{\Sigma^+} z \xrightarrow{V} \hat{x}.$$

The geometric interpretation is similar to what we did for the SVD in Section 3.3.3 except “in reverse” and with Σ^+ . To interpret the residual, note that

$$\begin{aligned} r &= y - A\hat{x} = y - AA^+y = (I - AA^+)y \\ &= (I - U\Sigma\Sigma^+U')y = U(I - \Sigma\Sigma^+)U'y \\ &= [U_r \mid U_0] \left(I - \left[\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array} \right] \right) [U_r \mid U_0]' y \end{aligned}$$

$$\begin{aligned}
 &= [\mathbf{U}_r \mid \mathbf{U}_0] \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} [\mathbf{U}_r \mid \mathbf{U}_0]' \mathbf{y} = \mathbf{U}_0 \mathbf{U}'_0 \mathbf{y} \\
 \implies \|\mathbf{r}\|_2^2 &= \|\mathbf{U}_0 \mathbf{U}'_0 \mathbf{y}\|_2^2 = \mathbf{y}' \mathbf{U}_0 \mathbf{U}'_0 (\mathbf{U}_0 \mathbf{U}'_0 \mathbf{y}) = \mathbf{y}' \mathbf{U}_0 \mathbf{U}'_0 \mathbf{y} = \|\mathbf{U}'_0 \mathbf{y}\|_2^2,
 \end{aligned} \tag{5.38}$$

where $\mathcal{R}^\perp(\mathbf{A}) = \mathcal{R}(\mathbf{U}_0)$ from Section 4.5. Thus, the residual norm squared comes from the portion of \mathbf{y} in $\mathcal{R}^\perp(\mathbf{A})$, consistent with our earlier picture of the orthogonality principle.

5.6 Truncated SVD Solution

We have seen that the minimum-norm LLS minimizer of $\|\mathbf{Ax} - \mathbf{y}\|_2^2$ is the pseudoinverse solution

$$\hat{\mathbf{x}} = \mathbf{A}^+ \mathbf{y} = \mathbf{V} \Sigma^+ \mathbf{U}' \mathbf{y} = \mathbf{V}_r \Sigma_r^{-1} \mathbf{U}'_r \mathbf{y} = \sum_{k=1}^r \frac{1}{\sigma_k} v_k(u'_k \mathbf{y}). \tag{5.39}$$

This solution is mathematically elegant, but in practice it sometimes works very poorly. This section provides one remedy based on the truncated SVD.

Example 5.7 Consider an application where $r = \text{rank}(\mathbf{A}) = 2$ with singular values $\sigma_1 = 1, \sigma_2 = 10^{-8}$. Then the pseudoinverse solution here is

$$\begin{aligned}
 \hat{\mathbf{x}} = \mathbf{A}^+ \mathbf{y} &= \sum_{k=1}^2 \frac{1}{\sigma_k} v_k(u'_k \mathbf{y}) = \frac{1}{\sigma_1} v_1(u'_1 \mathbf{y}) + \frac{1}{\sigma_2} v_2(u'_2 \mathbf{y}) \\
 &= (1) v_1(u'_1 \mathbf{y}) + (10^8) v_2(u'_2 \mathbf{y}).
 \end{aligned}$$

The pseudoinverse solution “blows up” when any of the σ_k values are “too small” relative to the others, that is, if σ_k/σ_1 is near the floating-point precision limits.

5.6.1 Condition Number

Such problems are called poorly conditioned because the matrix $\mathbf{A} \in \mathbb{F}^{M \times N}$ has an undesirably large condition number, defined here as

$$\begin{aligned}
 \kappa(\mathbf{A}) &\triangleq \begin{cases} \infty, & M < N \\ \frac{\sigma_{\max}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})}, & M \geq N \end{cases} = \begin{cases} \infty, & M < N \\ \frac{\sigma_1}{\sigma_N}, & M \geq N \end{cases} \\
 &= \begin{cases} \infty, & M < N \\ \sqrt{\|\mathbf{A}'\mathbf{A}\|_2 \|\mathbf{A}'\mathbf{A}^{-1}\|_2}, & M \geq N \end{cases} = \sqrt{\frac{\sigma_{\max}(\mathbf{A}'\mathbf{A})}{\sigma_{\min}(\mathbf{A}'\mathbf{A})}}.
 \end{aligned} \tag{5.40}$$



Caution: An alternate definition of condition number is σ_1/σ_r (e.g., [5, p. 69]), but we will use (5.40). Caution: Many sources consider only the case where \mathbf{A} is square; hence the expression $\mathbf{A}'\mathbf{A}$ above.

In JULIA, to compute the condition number of a matrix A use `cond(A)`. For an $M \times N$ matrix, JULIA uses a different definition:

$$\text{cond}(A) \triangleq \begin{cases} \infty, & \sigma_1 = 0, \\ \sigma_1/\sigma_{\min(M,N)}, & \text{otherwise.} \end{cases}$$

This definition gives a finite value for a wide matrix with full row rank, which differs from our definition (5.40) and also differs from the ratio σ_1/σ_r used in some texts. So be aware that condition number does not have a standardized definition in the literature for rank-deficient matrices.

The problem of poor conditioning motivates the truncated SVD solution where we discard any singular values that are “too small” and write

$$\hat{x}_K = \sum_{k=1}^K \frac{1}{\sigma_k} v_k(u'_k y), \quad (5.41)$$

where we choose fewer terms, $K < r$, such that $\sigma_K > \delta > 0$ for some tolerance δ .

5.6.2 Practical Implementation of Truncated SVD Solution

The tolerance δ may depend on factors such as the noise level in the data, the size of A , and whether one is using half-, single-, double-, or extended-precision variables. In JULIA, these variable types are `Float16`, `Float32`, `Float64`, and `BigFloat`. JULIA’s `pinv` has optional arguments for specifying the (absolute or relative) tolerance. The default value (Problem 4.6) is

```
(min(size(A)...)*eps(real(float(one(eltype(A)))))).
```

The backslash function in JULIA, `xh = A \ y`, does not have any tolerance parameter δ , so one must use `pinv` instead of backslash to control the tolerance for poorly conditioned problems. Alternatively, one can use some other method to improve the condition number, such as Tikhonov regularization, also called ridge regression, as described in Section 5.6.5.

5.6.3 Low-Rank Approximation Interpretation of Truncated SVD

One way to interpret the truncated SVD solution (5.41) is as follows.

First we form a low-rank approximation A_K of A , defined as

$$A_K = U_K \Sigma_K V'_K = \sum_{k=1}^K \sigma_k u_k v'_k,$$

with $K < r \leq \min(M,N)$. (See Section 7.2 for more details about such approximations.)

Then we express the (truncated) pseudoinverse LLS solution using that approximation:

$$\hat{x}_K = A_K^+ y = \sum_{k=1}^K \frac{1}{\sigma_k} v_k(u_k' y), \quad (5.42)$$

which is the same expression as (5.41). This approach is also called principal component regression [82].

We visualize a low-rank approximation as follows:

$$\begin{array}{c} \boxed{A} \\ \underbrace{}_{M \times N} \end{array} \approx \begin{array}{c} \boxed{A_K} \\ \underbrace{}_{M \times N} \end{array} = \begin{array}{c} \boxed{U_K} \\ \underbrace{}_{M \times K} \end{array} \begin{array}{c} \boxed{\Sigma_K} \\ \underbrace{}_{K \times K} \end{array} \begin{array}{c} \boxed{V'_K} \\ \underbrace{}_{K \times N} \end{array}$$

5.6.4 Noise Effects and Perturbations

There are two sources of perturbations in LLS problems that can degrade the estimate $\hat{x} = A^+ y$:

- Additive noise: $y = Ax + \varepsilon$.
- Errors in the model A . The A we use for estimation might differ from the “true model” A_{true} .

One can bound how these errors affect \hat{x} . Let $A \in \mathbb{R}^{M \times N}$ with $M \geq N$ and $\text{rank}(A) = N$, and define the model perturbation $\Delta A \triangleq A_{\text{true}} - A$. Let $\kappa \triangleq \sigma_1/\sigma_N = \|A\|_2/\sigma_N$ denote the condition number of A , using the matrix spectral norm defined in (3.16). Assume that the model perturbations are not “too large” as follows:

$$\eta \triangleq \frac{\|\Delta A\|_2}{\sigma_N} = \kappa \epsilon_A < 1, \quad \epsilon_A \triangleq \frac{\|\Delta A\|_2}{\|A\|_2}.$$

If the perturbed matrix $A + \Delta A$ has full rank, then the solution perturbation $\delta \hat{x} = \hat{x} - x_{\text{true}}$ has the following bound [5, Theorem 6.12, p. 69] in terms of the residual $r = A\hat{x} - y$:

$$\|\delta \hat{x}\|_2 \leq \frac{\kappa}{1 - \eta} \left(\epsilon_A \|x\|_2 + \frac{\|\varepsilon\|_2}{\|A\|_2} + \epsilon_A \kappa \frac{\|r\|_2}{\|A\|_2} \right). \quad (5.43)$$

(See also [83, 84, 85, 86, 87], as well as the componentwise bounds on δx_n in [88].)

- In the (full-rank) square case where $M = N$, the residual is $r = \mathbf{0}$, so the solution error $\|\delta \hat{x}\|_2$ depends on the condition number κ .
- In the usual over-determined case where $M \geq N$ and $y \notin \mathcal{R}(A)$, then the solution error is proportional to κ^2 so it is particularly important to try to keep κ small.

This bound is a motivation for using the truncated SVD where σ_1/σ_K is better (lower) than σ_1/σ_N .

5.6.5

Tikhonov Regularization, or Ridge Regression

A drawback of the truncated SVD solution to LLS problems is that it requires one to compute an SVD of \mathbf{A} , which can be impractical for large problems. An alternate approach to address ill-conditioned problems is to use Tikhonov regularization, also known as ridge regression.

We saw that the pseudoinverse solution to LLS problems involves $1/\sigma_k$ terms that can “blow up” for small singular values, leading to very large values of $\hat{\mathbf{x}}$. Instead of directly modifying the singular values, Tikhonov regularization modifies the LS cost function to include a term that discourages the estimate from having excessively high values:

$$\hat{\mathbf{x}}_\beta = \arg \min_{\mathbf{x} \in \mathbb{R}^N} \underbrace{\|\mathbf{Ax} - \mathbf{y}\|_2^2}_{\text{data misfit}} + \underbrace{\beta \|\mathbf{x}\|_2^2}_{\rightarrow \text{regularization (energy)}}, \quad (5.44)$$

where $\beta > 0$ is a regularization parameter that one must tune to trade off between how well $\hat{\mathbf{x}}_\beta$ fits the data and how high the energy of $\hat{\mathbf{x}}_\beta$ is.

Combining terms using the “stacking property” of norms (2.33), we can rewrite the Tikhonov estimate (5.44) as

$$\hat{\mathbf{x}}_\beta = \arg \min_{\mathbf{x} \in \mathbb{R}^N} \left\| \begin{bmatrix} \mathbf{A} \\ \sqrt{\beta} \mathbf{I} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} \right\|^2 = \arg \min_{\mathbf{x} \in \mathbb{R}^N} \|\tilde{\mathbf{A}}\mathbf{x} - \tilde{\mathbf{y}}\|_2^2, \quad \tilde{\mathbf{A}} \triangleq \begin{bmatrix} \mathbf{A} \\ \sqrt{\beta} \mathbf{I} \end{bmatrix}, \quad \tilde{\mathbf{y}} \triangleq \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}. \quad (5.45)$$

Q5.10 If \mathbf{A} is $M \times N$, how many rows does $\tilde{\mathbf{A}}$ have?

- A: M B: N C: $M + N$ D: $2M$ E: $2N$

Q5.11 What is the rank of $\tilde{\mathbf{A}}$?

- A: r B: M C: N D: $M + N$ E: None of these

In this simplified form, we know that the (unique!) LLS solution to (5.44) is

$$\hat{\mathbf{x}}_\beta = \tilde{\mathbf{A}}^+ \tilde{\mathbf{y}} = (\tilde{\mathbf{A}}' \tilde{\mathbf{A}})^{-1} \tilde{\mathbf{A}}' \tilde{\mathbf{y}} = (\mathbf{A}' \mathbf{A} + \beta \mathbf{I})^{-1} \mathbf{A}' \mathbf{y}. \quad (5.46)$$

If N is large, this solution requires inverting an $N \times N$ matrix (actually, solving an $N \times N$ system of equations), which is impractical. Instead, we usually apply an optimization approach (such as a conjugate gradient method) directly to (5.44). Nevertheless, the closed-form expression for the solution is useful for analysis. In particular, it is insightful to examine the solution in terms of an SVD $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}'$. Here,

$$\mathbf{A}' \mathbf{A} + \beta \mathbf{I} = \mathbf{V} \Sigma' \mathbf{U}' \mathbf{U} \Sigma \mathbf{V}' + \beta \mathbf{I} = \mathbf{V} (\Sigma' \Sigma + \beta \mathbf{I}) \mathbf{V}', \quad (5.47)$$

so

$$\begin{aligned} \hat{\mathbf{x}}_\beta &= (\mathbf{A}' \mathbf{A} + \beta \mathbf{I})^{-1} \mathbf{A}' \mathbf{y} = \mathbf{V} (\Sigma' \Sigma + \beta \mathbf{I})^{-1} \mathbf{V}' \mathbf{V} \Sigma' \mathbf{U}' \mathbf{y} \\ &= \mathbf{V} (\Sigma' \Sigma + \beta \mathbf{I})^{-1} \Sigma' \mathbf{U}' \mathbf{y} \\ &= \mathbf{V}_r (\Sigma_r' \Sigma_r + \beta \mathbf{I})^{-1} \Sigma_r' \mathbf{U}' \mathbf{y} \end{aligned}$$

$$= \sum_{k=1}^r v_k \underbrace{\left(\frac{\sigma_k}{\sigma_k^2 + \beta} \right)}_{\hookrightarrow \text{"shrinkage" or "Wiener filter" form}} (u'_k y). \quad (5.48)$$

If $\beta \rightarrow 0$ then the ratio $\frac{\sigma_k}{\sigma_k^2 + \beta} \rightarrow \frac{1}{\sigma_k}$, unless $\sigma_k = 0$ (which never happens for $k = 1, \dots, r$). So $\hat{x}_\beta \rightarrow A^+y$ as $\beta \rightarrow 0$.

5.6.5.1 Regularization Parameter Selection

- ◆ An important practical question is how one chooses the regularization parameter β . Unsupervised methods include cross validation and Stein's unbiased risk estimate (SURE; see Section 7.6.1). One can also pursue supervised learning approaches when training data is available [89, 90]. The details are beyond the scope here, but the foundations here are essential for understanding such methods. Tikhonov regularization is just the tip of the iceberg in modern regularization methods [91, 92].

5.7 Summary of LLS Solution Methods in Terms of SVD

This chapter has discussed three different ways of solving the linear least-squares (LLS) problem (5.3), each of which have SVD expressions as follows (see Fig. 5.8):

- “Optimal” solution (minimum-norm LLS):

$$\hat{x} = A^+y = \sum_{k=1}^r \frac{1}{\sigma_k} v_k (u'_k y).$$

- Truncated SVD with K terms:

$$\hat{x} = A_K^+y = \sum_{k=1}^{K < r} \frac{1}{\sigma_k} v_k (u'_k y) = \sum_{k=1}^r \mathbb{1}_{\{k \leq K\}} \frac{1}{\sigma_k} v_k (u'_k y).$$

- Tikhonov regularized:

$$\hat{x} = \sum_{k=1}^r \left(\frac{\sigma_k}{\sigma_k^2 + \beta} \right) v_k (u'_k y).$$

The SVD is a key tool for understanding LLS problems.

5.8 Frames and Tight Frames

A least-squares minimization problem $\arg \min_x \|Ax - y\|_2$ is easiest to solve when A is unitary because in that case $\hat{x} = A'y$. We now discuss a generalization of unitary matrices that leads to equally easy solutions.

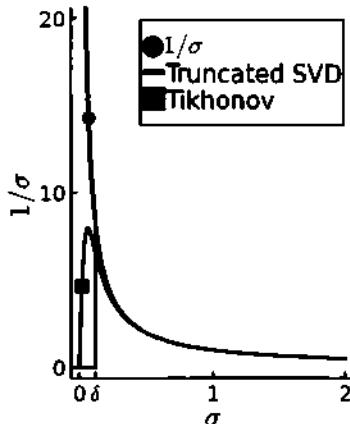


Figure 5.8 Linear least squares via SVD.

For simplicity we focus on the case of a finite number of vectors in a finite-dimensional vector space [93]. The concepts generalize to a countable collection of vectors in a general Hilbert space.

Definition A collection of M vectors $\Phi = \{\phi_1, \dots, \phi_M\}$ in \mathbb{F}^N is called a frame in \mathbb{F}^N iff there exist real numbers $0 < \alpha \leq \beta < \infty$, called the frame bounds [94], such that

$$\alpha \|x\|_2^2 \leq \sum_{m=1}^M |\langle x, \phi_m \rangle|^2 \leq \beta \|x\|_2^2 \quad \forall x \in \mathbb{F}^N. \quad (5.49)$$

In other words, if we arrange those vectors into an $M \times N$ matrix

$$T_\Phi \triangleq \begin{bmatrix} \phi'_1 \\ \vdots \\ \phi'_M \end{bmatrix},$$

then the collection of vectors is a frame iff there exist real numbers $0 < \alpha \leq \beta < \infty$ such that

$$\alpha \|x\|_2^2 \leq \|T_\Phi x\|_2^2 \leq \beta \|x\|_2^2 \quad \forall x \in \mathbb{F}^N. \quad (5.50)$$

The matrix T_Φ is called the analysis operator of the frame, and its Hermitian transpose T'_Φ is called the synthesis operator. For brevity, we call such a Φ a frame. The upper bound is important in infinite-dimensional inner product spaces, but is not very informative in \mathbb{F}^N .

Q5.12 If $\alpha > 0$, what is the minimum possible β here in terms of the singular values of T_Φ ?

- A: σ_1 B: σ_1^2 C: σ_r^2 D: σ_M E: None of these

Explore 5.4 When is the upper frame bound β positive?

So, in \mathbb{F}^N , whether Φ is a frame or not depends on the existence of $\alpha > 0$.

Example 5.8 Consider $\Phi = \{\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}\}$, which is an orthonormal basis for \mathbb{R}^2 augmented with one additional “redundant” vector. Here, $T_\Phi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$ with $\alpha = \sigma_2^2 = 1$, and $T_\Phi^+ = \frac{1}{3} \begin{bmatrix} 2 & -1 & 1 \end{bmatrix}$. Note that $T_\Phi^+ T_\Phi = I_2$. Later examples will illustrate the benefit of such redundancy.

5.8.1 Properties of a Frame

- Fact: $\exists \alpha > 0$ in (5.50) iff T_Φ has full column rank (i.e., $\text{rank}(T_\Phi) = N$), in which case any $0 < \alpha \leq \sigma_N^2$ is a valid lower frame bound. See (3.23).
- Thus, T_Φ must be tall (usually) or square; that is, $M \geq N$.
- Thus, $\alpha > 0 \implies T_\Phi = U_N \Sigma_N V'$ (compact SVD is the same as economy SVD) where $\sigma_N > 0$, because $U_r = U_N$ and $V_r = V_N = V$.
- So $T_\Phi^+ = V \Sigma_N^{-1} U_N' = (T_\Phi' T_\Phi)^{-1} T_\Phi'$ and $T_\Phi^+ T_\Phi = I_N$, so T_Φ^+ is a left inverse of T_Φ .

Now suppose we have a vector $x \in \mathbb{F}^N$ that we would like to express as a linear combination of the frame vectors $\{\phi_1, \dots, \phi_M\}$; that is, we want to write

$$x = \sum_{m=1}^M c_m \phi_m = T_\Phi c \quad (5.51)$$

for some coefficient vector $c \in \mathbb{F}^M$. In the usual case where T_Φ' is wide, there will be numerous possible coefficient vectors c for which $\|x - T_\Phi' c\|_2 = 0$. The unique such c having minimum norm is given by the Moore–Penrose pseudoinverse:

$$c = (T_\Phi')^+ x = (T_\Phi^+)' x = T_\Phi (T_\Phi' T_\Phi)^{-1} x. \quad (5.52)$$

This is not too exciting yet because of the expensive matrix inverse. So we impose more conditions on Φ .

5.8.2 Tight Frame

Definition A frame Φ is a tight frame [33, 34, 95] iff Φ is a frame with $\alpha = \beta$; that is,

$$\alpha \|x\|_2^2 = \|T_\Phi x\|_2^2 = \sum_{m=1}^M |\langle x, \phi_m \rangle|^2 \quad \forall x \in \mathbb{F}^N. \quad (5.53)$$

Fact 5.3 If Φ is a tight frame, then:

- $T_\Phi' T_\Phi = \alpha I_N$, so $\alpha = \sigma_1^2 = \dots = \sigma_N^2$, where $\{\sigma_k\}$ denotes the singular values of T_Φ ;
- the pseudoinverse of the analysis operator is simply $T_\Phi^+ = (1/\alpha) T_\Phi' = (1/\sigma_1^2) T_\Phi'$.

Proof. Using (5.53) and Problem 3.21,

$$\alpha \|x\|_2^2 = \|T_\Phi x\|_2^2 \implies x'(\alpha I - T'_\Phi T_\Phi)x = 0 \quad \forall x \in \mathbb{F}^N \implies \alpha I = T'_\Phi T_\Phi. \quad \square$$

Normally, finding σ_1 requires an SVD, which is expensive for large problems. But, for a tight frame,

$$\|T_\Phi e_1\|_2 = \sqrt{\alpha} = \sigma_1, \quad (5.54)$$

so here we can find σ_1 by a simple matrix–vector multiplication and a norm. One can show (Problem 5.22) that $T_\Phi T'_\Phi \leq \alpha I_M$, using the full SVD

$$T_\Phi = U \begin{bmatrix} \sqrt{\alpha} I \\ \mathbf{0} \end{bmatrix} V.$$

The much simpler pseudoinverse expression in Fact 5.3 makes (5.52) much more practical. For a tight frame, we can write (5.51) and (5.52) as the following expansion:

$$x = \frac{1}{\alpha} T'_\Phi T_\Phi x = \sum_{m=1}^M \frac{1}{\alpha} \langle x, \phi_m \rangle \phi_m. \quad (5.55)$$

Example 5.9 Consider vectors $\phi_m = \begin{bmatrix} \cos(2\pi m/M) \\ \sin(2\pi m/M) \end{bmatrix}$, $m = 1, \dots, M$, that are equally spaced around the unit circle in \mathbb{R}^2 . One can verify that $T'_\Phi T_\Phi = (M/2)I_2$, so Φ is a tight frame with $\alpha = M/2$, using the trigonometric identities $\sum_{m=1}^M \cos^2(2\pi m/M) = \sum_{m=1}^M \sin^2(2\pi m/M) = M/2$.

5.8.3 Parseval Tight Frame

Definition A frame Φ is a Parseval tight frame [33, 34] iff Φ is a tight frame with $\alpha = \beta = 1$; that is,

$$\|x\|_2^2 = \|T_\Phi x\|_2^2 = \sum_{m=1}^M |\langle x, \phi_m \rangle|^2 \quad \forall x \in \mathbb{F}^N. \quad (5.56)$$

In this case the key expansion (5.55) simplifies further because $\alpha = 1$.

Explore 5.5 What are the singular values σ_1 and σ_N of T_Φ in this case?

5.8.4 Properties of Parseval Tight Frames

Fact 5.4 If Φ is a Parseval tight frame, then:

- $T'_\Phi T_\Phi = I_N$;
- the relevant pseudoinverses are simply $T_\Phi^+ = T'_\Phi$ and $(T'_\Phi)^+ = T_\Phi$.

Proof. Using (5.56),

$$\|x\|_2^2 = \|T_\Phi x\|_2^2 \implies x'(I - T'_\Phi T_\Phi)x = 0 \quad \forall x \in \mathbb{F}^N \implies I = T'_\Phi T_\Phi. \quad \square$$

Here we can think of T_Φ as a "transform" and T'_Φ as a (left) "inverse transform," because $x = T'_\Phi T_\Phi x$.

Q5.13 The converse also holds; that is, if $T_\Phi^* T_\Phi = I_N$, then Φ is a Parseval tight frame.

A: True

B: False

Q5.14 The columns of every unitary matrix form a tight frame.

A. True

B: False

Example 5.10 A simple Parseval tight frame is the “Mercedes Benz” frame:

$$T_\Phi = \sqrt{\frac{2}{3}} \begin{bmatrix} 0 & -\sqrt{3}/2 & \sqrt{3}/2 \\ 1 & -1/2 & -1/2 \end{bmatrix},$$

One can verify that $T'_\Phi T_\Phi = I_2$.



Analysis of deep networks shows that the class means of features in the last layer collapse to the vertices of a simplex equiangular tight frame (ETF) [96].

Q5.15 Every Parseval tight frame corresponds to a unitary matrix.

A: True

B: False

Another way to construct a tight frame is to combine multiple unitary matrices:

$$T'_\Phi = [U_1 \quad \cdots \quad U_K]. \quad (5.57)$$

This construction is fairly common in signal processing, for example, combining orthogonal wavelet transforms with other transforms.

Q5.16 If U_1 and U_2 are $N \times N$ unitary matrices, what is the frame bound of $T'_\Phi \triangleq [U_1 \quad U_2]$?

A: 1 B: 2 C: N D: $2N$ E: None

1. [Ex. 7]

If U_1 and U_2 are $N \times N$ unitary matrices, then $T_\Phi \triangleq \frac{1}{\sqrt{2}} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$ is the analysis operator of a Parseval tight frame because $T_\Phi^* T_\Phi = I_N$. For completeness, one can verify (Problem 5.19) the following SVD of T_Φ' :

$$T_{\Phi} = I_N \underbrace{\begin{bmatrix} I_N & \mathbf{0}_N \end{bmatrix}}_{\Sigma} \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} U_1 & U_2 \\ Q & -QU_1'U_2 \end{bmatrix}}_{V'}, \quad (5.58)$$

where Q denotes any $N \times N$ unitary matrix. In particular, choosing $Q = U_1$ leads to

$$\mathbf{T}'_\Phi = \mathbf{I}_N \underbrace{\begin{bmatrix} \mathbf{I}_N & \mathbf{0}_N \\ \Sigma & \end{bmatrix}}_{\mathbf{V}'} \underbrace{\frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \\ \mathbf{U}_1 & -\mathbf{U}_2 \end{bmatrix}}_{\mathbf{U}'}. \quad (5.59)$$

Example 5.11 Consider the $2N \times N$ matrix $\mathbf{T}_\Phi = \frac{1}{2} \begin{bmatrix} \mathbf{A}_1' \\ \mathbf{A}_2' \end{bmatrix}$ where

$$\mathbf{A}_1 \triangleq \begin{bmatrix} -1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 1 & -1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -1 \end{bmatrix}, \quad \mathbf{A}_2 \triangleq \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 1 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & 1 \end{bmatrix}. \quad (5.60)$$

One can verify that $\mathbf{T}'_\Phi \mathbf{T}_\Phi = \mathbf{I}_N$, so \mathbf{T}_Φ is the analysis operator for a Parseval tight frame. The first $N - 1$ columns of \mathbf{A}_1 are the transpose of the finite difference matrix in (4.42). Both \mathbf{A}_1 and \mathbf{A}_2 are circulant matrices (see Section 8.3). This example is related to the over-complete (i.e., undecimated) Haar wavelet transform and to the cycle spinning process used in wavelet denoising [97]. Rearranging the columns of \mathbf{T}'_Φ leads to the form (5.57).

- ➊ Demo 5.4 and Fig. 5.9 illustrate benefits of using a Parseval tight frame (PTF) versus an orthogonal discrete wavelet transform (ODWT) for image denoising. The denoising method is wavelet coefficient shrinkage by the soft thresholding operation in (7.24). See the demo for details.

5.8.5 Frame Summary

The hierarchy of frames is shown in Fig. 5.10.

Frames have numerous uses in signal processing [33, 34, 95, 98]. In the usual case where \mathbf{T}_Φ is tall, they are considered “robust redundant signal representations” [99].

Frame theory underpins recent advances in image denoising and efforts to provide insights into convolutional neural networks by relating them to perfect reconstruction filter banks [100, 101, 102].

Solving LLS problems for a Parseval tight frame Φ is as easy as for a unitary matrix, because $\mathbf{T}_\Phi^+ = \mathbf{T}'_\Phi$. So, given a vector \mathbf{x} in \mathbb{F}^N , the representation of \mathbf{x} using the (typically linearly dependent!) set of “atoms” $\{\phi_1, \dots, \phi_M\}$ for which the coefficient vector has minimum Euclidean norm is

$$\mathbf{x} = \sum_{m=1}^M \langle \mathbf{x}, \phi_m \rangle \phi_m = \mathbf{T}'_\Phi \mathbf{c}, \quad \mathbf{c} = \mathbf{T}_\Phi \mathbf{x}. \quad (5.61)$$

In this sense, Parseval tight frames are generalizations of orthonormal bases and unitary matrices.

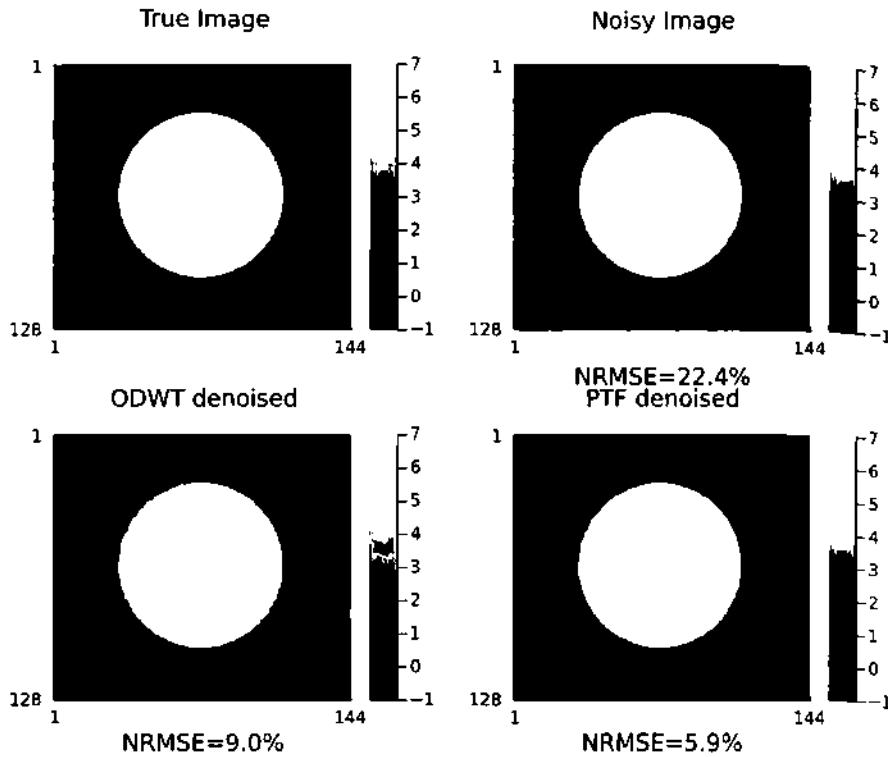


Figure 5.9 Illustration of image denoising using a Parseval tight frame.

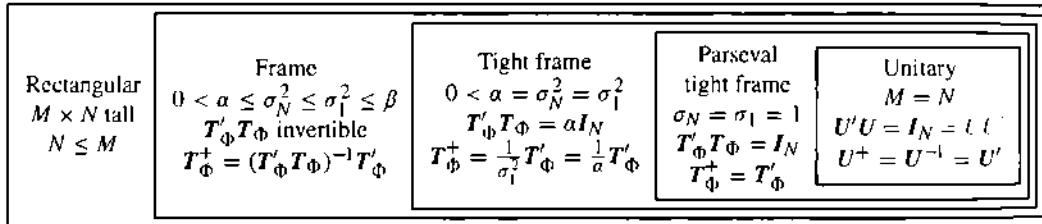


Figure 5.10 Venn diagram for frames.

Other minimum-norm representations are also of interest, such as [103]

$$\arg \min_{\mathbf{c}: \mathbf{x} = \mathbf{T}'_{\Phi} \mathbf{c}} \|\mathbf{c}\|_p.$$

5.9 Projection and Orthogonal Projection

5.9.1 Idempotent Matrix

Definition A (square) matrix \mathbf{P} is called a projection matrix iff $\mathbf{P}^2 = \mathbf{P}\mathbf{P} = \mathbf{P}$. Such a (square) matrix is also called an **idempotent matrix**.

Definition When P is a non-Hermitian idempotent matrix, we call Px the oblique projection of x onto $\mathcal{R}(P)$.

Example 5.12 For any a ,

$$P = \begin{bmatrix} 1 & a \\ 0 & 0 \end{bmatrix} = V\Lambda V^{-1}, \quad V = \begin{bmatrix} 1 & -a/\sqrt{1+a^2} \\ 0 & 1/\sqrt{1+a^2} \end{bmatrix}, \quad \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

is idempotent. For this example, if $x \in \mathbb{R}^2$ then

$$Px = \begin{bmatrix} 1 & a \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 + ax_2 \\ 0 \end{bmatrix} \in \text{span}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \mathcal{R}(P).$$

Figure 5.11 illustrates oblique projection for this example with $a = 1$.

5.9.1.1 Basic Properties of Idempotent Matrices

Fact 5.5 All eigenvalues of an idempotent matrix are either 0 or 1.

Proof. Suppose x is an eigenvector of an idempotent matrix P , that is, $Px = \lambda x$. Multiplying both sides by P yields $P(Px) = \lambda(Px) \implies Px = \lambda^2 x$. Combining, we have $\lambda^2 = \lambda$, so $\lambda = 0$ or $\lambda = 1$. \square

Fact 5.6 Every projection matrix is diagonalizable.

Q5.17 The converse of that property also holds, that is, if A is a (square) diagonalizable matrix with eigenvalues that are all either 0 or 1, then A is always idempotent.

A: True

B: False

In other words, $P^2 = P \iff P$ is diagonalizable and all its eigenvalues are 0 or 1.

Example 5.13 For any matrix A , the matrix $P = AA^+$ is idempotent, because $P^2 = (AA^+)(AA^+) = A(A^+AA^+) = AA^+ = P$.

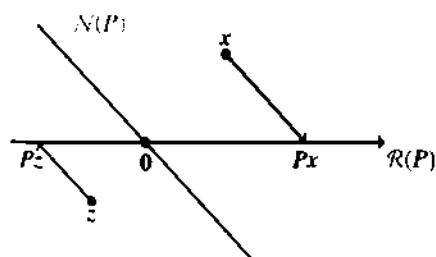


Figure 5.11 Oblique projection.

5.9.2 Orthogonal Projection Matrix

Our primary interest is the subset of projection matrices that are (Hermitian) symmetric matrices.

Definition A (square) matrix P is called an orthogonal projector or orthogonal projection matrix iff P is idempotent and P is Hermitian.

 Caution: An orthogonal projection matrix typically is not an orthogonal matrix! If $P = P' = P^2$ is an orthogonal projection matrix and if P is also an orthogonal matrix then $I = P'P = P^2 = P$. So the only matrix that is both is the identity matrix I .

Example 5.14 An example of an orthogonal projection matrix is

$$P = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \right) \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \right).$$

Q5.18 Every orthogonal projection matrix has a unitary eigendecomposition.

Q5.19 If x has unit norm, then xx' is an orthogonal projection matrix.

Q5.20 Every orthogonal projection matrix P is positive semidefinite.

Alternative explanation: If P is an orthogonal projection matrix, that is, both idempotent and Hermitian, then $P = P^2 = PP = P'P \succeq 0$. More generally, if Q is any (possibly nonsquare) matrix with orthonormal columns, then $P = QQ'$ is an orthogonal projection matrix, because

- $P = QQ'$ is Hermitian;
 - $P^2 = (QQ')(QQ') = QQ' = P$, because $Q'Q = I$.

Q5.21 Is the converse true? Can an $N \times N$ orthogonal projection matrix P be written as QQ' for some matrix Q with orthonormal columns?

- A: Yes, always
 B: Yes, if $\text{rank}(P) \geq 1$
 C: Yes, if $\text{rank}(P) \leq N - 1$
 D: Only if P is nonsingular
 E: No

O5.22 When writing a nonzero orthogonal projection matrix P as OO' , the O is unique.

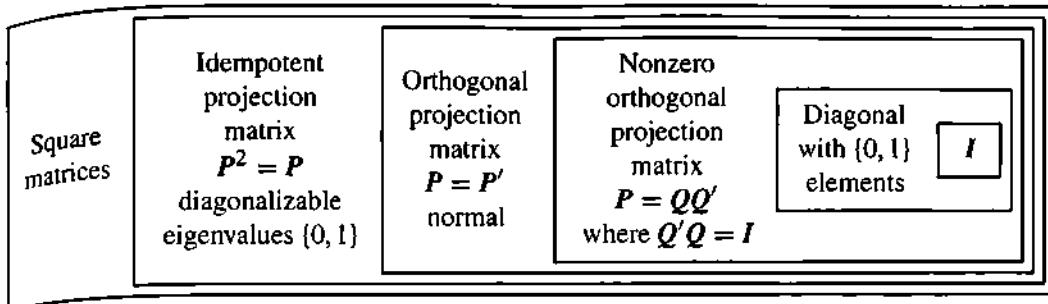


Figure 5.12 Venn diagram for projection matrix relations.

Q5.23 The $M \times M$ Gram matrix $G = T_\Phi T'_\Phi$ of a Parseval tight frame with $T_\Phi \in \mathbb{F}^{M \times N}$ is an orthogonal projection matrix.

A: True

B: False

The Venn diagram in Fig. 5.12 summarizes projection matrix relationships.

To recap, we have used the term projection in two ways here.

- One use is in defining an $N \times N$ projection matrix P that has the property $P^2 = P$. In other words, for any $x \in \mathbb{F}^N$, $P(Px) = P^2x = Px$.
- The other use is in (4.47) when defining the projection onto a (typically convex) set $\mathcal{P}_C(\cdot)$.

One connection between the two uses is the following. If we project a point v onto a set C , we get $\hat{v} = \mathcal{P}_C(v)$. If we then project \hat{v} onto C , we get the same point: $\hat{v} = \mathcal{P}_C(\hat{v})$. In other words, for any $v \in \mathcal{V}$, $\mathcal{P}_C(\mathcal{P}_C(v)) = \mathcal{P}_C(v)$, as in (4.49). Put concisely, we have the expression $\mathcal{P}_C \circ \mathcal{P}_C = \mathcal{P}_C$, which is very similar to $P^2 = P$, hence the similarity in terminology.

5.9.3 Projection onto a Subspace

Now we move towards using projection as a tool for solving DS–ML–SP problems. Let a_1, \dots, a_N denote any collection of N vectors, each in \mathbb{F}^M . The span of this set defines a subspace: $S \triangleq \text{span}(\{a_1, \dots, a_N\})$. Equivalently (because we are working in \mathbb{F}^M here), we can group the vectors into an $M \times N$ matrix A and then describe the subspace as its range:

$$S = \mathcal{R}(A) \triangleq \{Ax : x \in \mathbb{F}^N\}, \quad A \triangleq [a_1 \quad \dots \quad a_N].$$

As introduced in Section 4.8, in many applications (including handwritten digit recognition via nearest subspace) we are given some test vector y and we want to find the vector in a subspace S that is closest to y :

$$\hat{y} = \arg \min_{s \in S = \mathcal{R}(A)} \|y - s\|_2. \quad (5.62)$$

The resulting vector \hat{y} is called the projection of the point y onto the subspace S , and the process (operation) of finding that closest point is called projecting the point y onto the subspace S .

The key to solving this problem is to use the fact that every point in the subspace S has the form Ax for some $x \in \mathbb{F}^N$. Thus, $\hat{y} = A\hat{x}$ for some $\hat{x} \in \mathbb{F}^N$, and we just have to find that best \hat{x} to get \hat{y} . In other words, the closest point or projection problem is equivalent to:

$$\hat{y} = A\hat{x}, \quad \hat{x} = \arg \min_{x \in \mathbb{R}^N} \|y - Ax\|_2, \quad (5.63)$$

because every $s \in S$ is $s = Ax$ for some $x \in \mathbb{F}^N$. This form involves solving a least-squares problem, and a solution to that part is $\hat{x} = A^+y$, so

$$\hat{y} = \underset{s \in S = \mathcal{R}(A)}{\arg \min} \|y - s\|_2 = AA^+y. \quad (5.64)$$

In other words,

$$\mathcal{P}_{R(A)}(y) = AA^+y . \quad (5.65)$$

5.9.3.1 Practical Implementation

Reiterating, for $A \in \mathbb{F}^{M \times N}$, the point in the subspace $\mathcal{R}(A)$ that is closest to the point $y \in \mathbb{F}^M$ is $\hat{y} = AA^+y$. If we need to compute this just once, for one A and one y , then using code like $A^*(A \setminus y)$ is fine if A is small enough for backslash to work. If A is large, then we use an iterative method to compute the LS coefficients \hat{x} first, then compute the projection $\hat{y} = A\hat{x}$. The parentheses are essential in $A^*(A \setminus y)$ because $*$ and \setminus have the same operator precedence in JULIA.



In most applications A is fixed (after some training or modeling process) and we will need to perform projection for many different “test” y vectors. In such cases, it is more efficient to use an SVD at the beginning (as a final step of the training process) to save computation at the test stage. Specifically, when $r = \text{rank}(A) > 0$,

$$\mathbf{P}_{R(A)} \triangleq AA^+ = U_r \Sigma_r V_r' V_r \Sigma_r^{-1} U_r' = U_r U_r'. \quad (5.66)$$

So a more practical approach in such cases is to perform one SVD to find U_r ; then after that we need only use simple matrix–vector multiplies to perform projection. The following JULIA code illustrates this approach. It handles both the usual case where $r > 0$, and the trivial case where $r = 0$. This code also illustrates an efficient way to count the number of elements of a vector (or any JULIA iterator) that exceed a threshold.

```

<>5.1 using LinearAlgebra: svd
(U,s,V) = svd(A)
# r = sum(s .> threshold) # "matlab way" (allocating)
r = count(>(threshold), s) # Julia way!
if r > 0
    Ur = U[:,1:r]
    projector = (y) -> Ur * (Ur' * y)
else
    projector = (y) -> zeros(eltype(y), size(y))
end

```

Note that U_r is an orthonormal basis for $\mathcal{R}(A)$, and $P_{\mathcal{R}(A)} = U_r U_r'$. This is not a coincidence!

In a classification application, we often have many test vectors in an $M \times K$ matrix Y , instead of a single test vector y .

- $U_r * (U_r' * Y)$ requires $2rKM$ scalar multiplications;
- $(U_r * U_r') * Y$ requires $rM^2 + KM^2 = (r + K)M^2$ scalar multiplications.

Typically $2rK \ll (r+K)M$ because $r \ll M$, so usually the first approach is preferable.

5.9.3.2 Orthonormal versus Nonorthonormal Bases for a Subspace

More generally, if Q is a matrix whose columns are orthonormal, then those columns form an orthonormal basis for a subspace, namely $\mathcal{R}(Q)$, and the projection of a vector y onto that subspace is simply

$$P_{\mathcal{R}(Q)}(y) = P_{\mathcal{R}(Q)}y = Q(Q'y). \quad (5.67)$$

So an orthonormal basis for a subspace is extremely convenient for computation, because subspace projection using such bases requires mere matrix–vector multiplication.

If a basis matrix B is not orthonormal, then to compute the projection we would need

$$P_{\mathcal{R}(B)}y = B(B^+y) = B(B'B)^{-1}B'y, \quad (5.68)$$

as derived in (5.65), which is much more expensive in general because B^+ usually requires an SVD. Recall that a basis consists of linearly independent vectors, so B must have full column rank.

We usually use (5.67) or (5.68) rather than the general form in Problem 4.16.

5.9.3.3 Orthogonality Principle Revisited

Another version of the orthogonality principle is the following. Let S denote any subspace of a vector space \mathcal{V} , and $v \in \mathcal{V}$ be any point in that vector space. If the closest point in S to v is

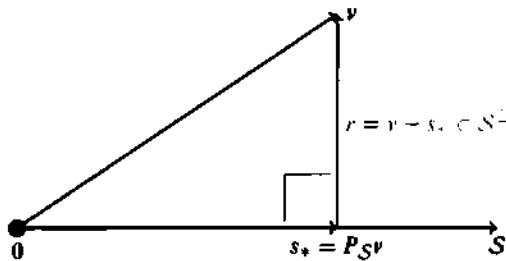


Figure 5.13 Orthogonality principle for subspace projection.

$$s_* = \arg \min_{s \in S} \|v - s\|_2 = P_S v,$$

then the orthogonality principle (see Fig. 5.13) says that the residual vector $r = v - s_*$ is perpendicular to the entire subspace S ; that is,

$$\langle v - s_*, s \rangle = 0 \quad \forall s \in S. \quad (5.69)$$

Note that $r = v - s_* = v - P_S v = (I - P_S)v = P_S^\perp v$, $P_S^\perp \triangleq I - P_S$.

Proof. For $s \in S$ and any $\alpha \in \mathbb{F}$,

$$\begin{aligned} 0 &\leq \| (s_* + \alpha s) - v \|_2^2 - \| s_* - v \|_2^2 = 2 \operatorname{real}\{\alpha \langle s_* - v, s \rangle\} + |\alpha|^2 \|s\|_2^2 \\ &\implies \langle s_* - v, s \rangle = 0. \end{aligned}$$

□

5.9.3.4 Projection onto a Subspace's Orthogonal Complement

Recall from (4.16) that in a finite-dimensional vector space \mathcal{V} , if S is a subspace of \mathcal{V} , then

$$\mathcal{V} = S \oplus S^\perp.$$

Clearly $P_{\mathcal{V}} = I$, so it follows that

$$I = P_S + P_{S^\perp}.$$

Thus, the projection onto the subspace's orthogonal complement S^\perp is simply

$$P_{S^\perp} = I - P_S.$$

If P is any projection matrix, then as a notation convention we define

$$P^\perp \triangleq I - P. \quad (5.70)$$

It then follows that

$$P_{S^\perp} = P_S^\perp = I - P_S.$$

In words, P_S^\perp is the orthogonal projector onto S^\perp , the orthogonal complement of the subspace S .

Example 5.15 If $S = \text{span}(\{\mathbf{1}_n\})$ then $P_S^\perp = I - \mathbf{1}_n \mathbf{1}_n^T = I - (1/n)\mathbf{1}_n \mathbf{1}_n^T$. The matrix–vector product $\mathbf{y} = P_S^\perp \mathbf{x}$ returns a vector \mathbf{y} having zero mean, so \mathbf{y} is orthogonal to $\mathbf{1}_n$.

5.9.3.5 Projectors and the Four Fundamental Subspaces

Recall the SVD anatomy of an $M \times N$ matrix with rank $0 < r < \min(M, N)$:

$$A = U\Sigma V' = \sum_{k=1}^{\min(M,N)} \sigma_k u_k v_k' = \sum_{k=1}^r \sigma_k u_k v_k' \\ = \left[\begin{array}{c|c} U_r & U_0 \end{array} \right] \left[\begin{array}{cc} \Sigma_r & 0 \\ \hline 0 & 0 \end{array} \right] \left[\begin{array}{c|c} V_r & V_0 \end{array} \right]' \quad (5.71)$$

The four fundamental subspaces have the following orthonormal bases and orthogonal projectors.

| Space | Subspace in A | Equivalent subspace in A' | Basis matrix | Projector | Alternate projector |
|----------------|------------------------|-----------------------------------|-----------------|------------|------------------------|
| \mathbb{F}^N | $N(A)$ | $\mathcal{R}^\perp(A')$ | V_0 | $V_0 V'_0$ | $I - V_r V'_r$ |
| \mathbb{F}^N | $N^\perp(A)$ | $\mathcal{R}(A')$ | V_r | $V_r V'_r$ | $I - V_0 V'_0$ |
| \mathbb{F}^M | $\mathcal{R}(A)$ | $N^\perp(A')$ | U_r | $U_r U'_r$ | $I - U_0 U'_0$ |
| \mathbb{F}^M | $\mathcal{R}^\perp(A)$ | $N(A')$ | U_0 | $U_0 U'_0$ | $I - U_r U'_r$ |

For each subspace we have two choices, but the blue forms are usually preferable in practice, because if we have stored “only” the compact SVD, then we will need to use U_r and V_r .

$$\text{Q5.25 } N^\perp(A') \oplus R^\perp(A) = \mathbb{F}^M.$$

A. Type

B. False

Q5.26 If Q is a matrix with orthonormal columns and D is a diagonal matrix whose elements are all 0 or 1, then $P = Q D Q'$ is an orthogonal projection matrix.

A. True

B: False

Q5.27 Let matrix A , rank $r > 0$, have SVD $A = U\Sigma V'$ and compact SVD $A = U_r\Sigma_r V'_r$, where we partition the unitary matrix V as usual as $V = [V_r \quad V_0]$. Which of the following matrices is $P_{N(A'A)}$?

A: $V_0 V'_0$ B: $V_r V'_r$ C: $V_0 V'_r$ D: $V_r V'_0$ E: None of these

Q5.28 Continuing the previous problem, and assuming the rank $r > 0$ is much smaller than the matrix dimensions, which of the following JULIA snippets is the most efficient way to implement $P_{\mathcal{R}^\perp(A^+)}x$ after the line $U, s, V = \text{svd}(A)$?

- A: $V[:, 1:r] * (V[:, 1:r]')' * x$
- B: $V[:, 1:r] * V[:, 1:r]' * x$
- C: $x - V[:, 1:r] * V[:, 1:r]' * x$
- D: $x - V[:, 1:r] * (V[:, 1:r])' * x$
- E: $V[:, (r+1):end] * (V[:, (r+1):end])' * x$

This last question is a “mini review” that uses elements from multiple chapters: Section 2.4 (multiplication order), Section 3.3 (SVD), Section 4.2 (subspaces), Section 5.4 (pseudoinverse), Section 5.9 (orthogonal projection), and JULIA!

5.9.4 Binary Classifier Design Using Least Squares

Linear LS can be used as a simple tool for designing a binary classifier via supervised learning. Given M feature vectors $v_i \in \mathbb{R}^N$ and corresponding binary class labels $y_i = \pm 1$, we want to find a weight vector x such that $\langle x, v_i \rangle$ has the same sign as y_i . Equivalently, we want $\text{sign}(\langle x, y_i v_i \rangle) = 1$.

Form an $M \times N$ data matrix from the training feature vectors and a corresponding label vector:

$$\tilde{A} = \begin{bmatrix} v'_1 \\ \vdots \\ v'_M \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}. \quad (5.72)$$

Then solve a least-squares problem (or regularized variant thereof):

$$\hat{x} = \arg \min_{x \in \mathbb{R}^N} \|\tilde{A}x - y\|_2^2. \quad (5.73)$$

Because each element of y is ± 1 , the diagonal matrix $D \triangleq \text{Diag}(y)$ is unitary and $D^{-1} = D' = D$. Using the unitary invariance of the Euclidean norm, and the fact that $y = D\mathbf{1}$, an equivalent formulation is

$$\hat{x} = \arg \min_{x \in \mathbb{R}^N} \|D\tilde{A}x - \mathbf{1}_M\|_2^2 = \arg \min_{x \in \mathbb{R}^N} \|Ax - \mathbf{1}_M\|_2^2, \quad A \triangleq \begin{bmatrix} y_1 v'_1 \\ \vdots \\ y_M v'_M \end{bmatrix} = \text{Diag}(y) \tilde{A}. \quad (5.74)$$

After “learning” the regression weights \hat{x} , the classifier for a subsequent test data point $v \in \mathbb{R}^N$ is simply

$$\text{sign}(v'\hat{x}). \quad (5.75)$$

It is somewhat unconventional to use LS for a data vector y that is binary (± 1 values), as logistic regression (see Section 9.5) is truly designed for such data. Nevertheless, the LS approach is simple and fast, and can work adequately in some cases.

5.9.5

Empirical Risk Minimization

The cost function (5.73) is for the case where we aim to solve for the classifier weight vector \hat{x} using a set of M training pairs $\{(v_m, y_m) : m = 1, \dots, M\}$.

◆ ◆ A more general framework from statistical learning theory views (5.73) as an example of empirical risk minimization (ERM) [104, §2.2]. Assume that there exists a joint distribution $p(v, y)$ of feature vectors v and labels y , and that our training data $\{(v_m, y_m) : m = 1, \dots, M\}$ consists of independently and identically distributed (IID) samples from that distribution. Let $\hat{y} = h(v; x)$ denote a model that predicts the label y corresponding to the feature vector v , for a given model weight vector x . We define a loss function $L(\hat{y}, y)$ that quantifies how poorly the prediction $\hat{y} = h(v; x)$ predicts the true label y . For example, linear LS estimation is based on the squared-error loss $L(\hat{y}, y) = \|\hat{y} - y\|_2^2$. Ideally we would like to choose the weights x to minimize the risk of the predictor, which is its expected loss:

$$R(x) = E[L(h(v; x), y)] = \int L(h(v; x), y) d p(v, y). \quad (5.76)$$

Typically, the joint distribution $p(v, y)$ is unknown, so one cannot compute the risk (5.76) exactly. However, one can replace the expectation with an average over the training sample, which is equivalent to replacing $p(v, y)$ with the empirical measure of the sample, leading to

$$R(x) \approx R_{\text{emp}}(x) \triangleq \frac{1}{M} \sum_{m=1}^M L(h(v_m; x), y_m). \quad (5.77)$$

We then learn the model parameters x by minimizing this empirical risk:

$$\hat{x} = \arg \min_{x \in \mathbb{F}^N} R_{\text{emp}}(x). \quad (5.78)$$

In particular, for the squared-error loss and the linear predictor

$$\hat{y}_m = h(v_m; x) = v'_m x,$$

ERM becomes

$$\hat{x} = \arg \min_{x \in \mathbb{F}^N} R_{\text{emp}}(x), \quad R_{\text{emp}}(x) = \frac{1}{M} \sum_{m=1}^M (v'_m x - y_m)^2 = \frac{1}{M} \|\tilde{A}x - y\|_2^2, \quad (5.79)$$

which is equivalent to (5.73). The ERM perspective is useful for deriving performance bounds [105]. It is a central idea in statistical machine learning, but not essential for the matrix methods in this book.

5.10 Recursive Least Squares

There are many DS–ML–SP applications where data arrives sequentially, instead of in a single batch, requiring algorithms that update parameter estimates continuously as new data arrives. Such methods have names like adaptive, online, recursive, or streaming. This section introduces the basics of recursive least-squares (RLS) estimation.

Consider the Tikhonov regularized problem (5.44) for $A_1 \in \mathbb{F}^{M \times N}$ and $y_1 \in \mathbb{F}^M$:

$$\hat{x}_1 = \arg \min_{x \in \mathbb{F}^N} \|A_1 x - y_1\|_2^2 + \beta \|x\|_2^2 = H_1^{-1} A_1' y_1, \quad H_1 \triangleq A_1' A_1 + \beta I. \quad (5.80)$$

Now suppose we obtain more data corresponding to $K \geq 1$ additional rows appended to A and y :

$$A_2 = \begin{bmatrix} A_1 \\ B_2 \end{bmatrix}, \quad y_2 = \begin{bmatrix} y_1 \\ z_2 \end{bmatrix}, \quad B_2 \in \mathbb{F}^{K \times N}, z_2 \in \mathbb{F}^K.$$

Naturally we want to update our estimate of the parameters x using *all* available data:

$$\begin{aligned} \hat{x}_2 &= \arg \min_{x \in \mathbb{F}^N} \|A_2 x - y_2\|_2^2 + \beta \|x\|_2^2 = H_2^{-1} A_2' y_2, \\ H_2 &\triangleq A_2' A_2 + \beta I_N = A_1' A_1 + B_2' B_2 + \beta I_N = H_1 + B_2' B_2. \end{aligned} \quad (5.81)$$

To reduce computation for this solution, apply the matrix inversion lemma (2.28) as follows:

$$H_2^{-1} = (H_1 + B_2' B_2)^{-1} = H_1^{-1} - H_1^{-1} B_2' \left(I_K + B_2 H_1^{-1} B_2' \right)^{-1} B_2 H_1^{-1}. \quad (5.82)$$

In this recursive form, computing H_2^{-1} , given that we previously computed H_1^{-1} , requires inverting only a $K \times K$ matrix, where typically $K \ll N$ and often $K = 1$.

The corresponding update of the estimate \hat{x} has the following recursive form (after simplifying):

$$\begin{aligned} \hat{x}_2 &= H_2^{-1} A_2' y_2 = H_2^{-1} \left(H_1 H_1^{-1} \right) (A_1' y_1 + B_2' z_2) = H_2^{-1} H_1 (\hat{x}_1 + H_1^{-1} B_2' z_2) \\ &= \left(I - H_1^{-1} B_2' \left(I_K + B_2 H_1^{-1} B_2' \right)^{-1} B_2 \right) (\hat{x}_1 + H_1^{-1} B_2' z_2) \\ &= \hat{x}_1 + H_1^{-1} B_2' z_2 - H_1^{-1} B_2' \left(I_K + B_2 H_1^{-1} B_2' \right)^{-1} B_2 (\hat{x}_1 + H_1^{-1} B_2' z_2) \\ &= \hat{x}_1 + H_1^{-1} B_2' \left(z_2 - \left(I_K + B_2 H_1^{-1} B_2' \right)^{-1} B_2 (\hat{x}_1 + H_1^{-1} B_2' z_2) \right) \\ &= \hat{x}_1 + H_1^{-1} B_2' \delta_1, \quad \delta_1 \triangleq \underbrace{\left(I_K + B_2 H_1^{-1} B_2' \right)^{-1} (z_2 - B_2 \hat{x}_1)}_{\text{residual}}. \end{aligned} \quad (5.83)$$

Intuitively, if the new data z_2 is consistent with the estimate \hat{x}_1 , then the residual is zero and the parameter estimate is unchanged. Otherwise, the update of \hat{x} involves finding δ_1 by solving just a $K \times K$ system of equations, given that we computed H_1^{-1} previously.

When $K = 1$, then $\mathbf{B}_2 = \mathbf{b}'_2$ for $\mathbf{b}_2 \in \mathbb{F}^N$ and the recursion (5.82) simplifies to the rank-1 update in (2.27) that requires $O(N^2)$ FLOPs:

$$\mathbf{H}_2^{-1} = \mathbf{H}_1^{-1} - \frac{1}{1 + \mathbf{b}'_2 \mathbf{H}_1^{-1} \mathbf{b}_2} \mathbf{H}_1^{-1} \mathbf{b}_2 \mathbf{b}'_2 \mathbf{H}_1^{-1}. \quad (5.84)$$

Similarly, (5.83) simplifies to

$$\hat{\mathbf{x}}_2 = \hat{\mathbf{x}}_1 + \frac{z_2 - \mathbf{b}'_2 \hat{\mathbf{x}}_1}{1 + \mathbf{b}'_2 \mathbf{H}_1^{-1} \mathbf{b}_2} \mathbf{H}_1^{-1} \mathbf{b}_2. \quad (5.85)$$

5.10.1 RLS with a Forgetting Factor

In streaming applications where new data arrives sequentially, it may be appropriate to use a cost function that includes a “forgetting factor” $0 < \alpha \leq 1$ that gives less weight to less recent data. After collecting $k \geq N$ data points, a typical formulation uses weighted LS with a diagonal weighting matrix per (6.8):

$$\hat{\mathbf{x}}_k = \arg \min_{\mathbf{x} \in \mathbb{F}^N} \|\mathbf{A}_k \mathbf{x} - \mathbf{y}_k\|_{W_k}^2 = \mathbf{H}_k^{-1} \mathbf{b}_k, \quad (5.86)$$

$$\mathbf{A}'_k = [\mathbf{a}_1 \ \cdots \ \mathbf{a}_k], \quad W_k = \text{Diag}\{\alpha^{k-1}, \dots, \alpha, 1\},$$

$$\begin{aligned} \mathbf{H}_k &= \mathbf{A}'_k W_k \mathbf{A}_k = \sum_{i=1}^k \alpha^{k-i} \mathbf{a}_i \mathbf{a}'_i, \\ \mathbf{b}_k &= \mathbf{A}'_k W_k \mathbf{y}_k = \sum_{i=1}^k \alpha^{k-i} \mathbf{a}_i y_i. \end{aligned} \quad (5.87)$$

After the next data point arrives, we update the estimate $\hat{\mathbf{x}}$ efficiently using all the data, while continuing to down-weight older data:

$$\hat{\mathbf{x}}_{k+1} = \arg \min_{\mathbf{x} \in \mathbb{F}^N} \|\mathbf{A}_{k+1} \mathbf{x} - \mathbf{y}_{k+1}\|_{W_{k+1}}^2 = \mathbf{H}_{k+1}^{-1} \mathbf{b}_{k+1}. \quad (5.88)$$

The inverse of \mathbf{H}_k has the following recursion akin to (5.84) based on (2.27):

$$\begin{aligned} \mathbf{H}_{k+1} &= \sum_{i=1}^{k+1} \alpha^{k+1-i} \mathbf{a}_i \mathbf{a}'_i = \mathbf{a}_{k+1} \mathbf{a}'_{k+1} + \alpha \mathbf{H}_k \\ \implies \mathbf{H}_{k+1}^{-1} &= \frac{1}{\alpha} \left(\mathbf{H}_k^{-1} - \frac{1}{\alpha + \mathbf{a}'_{k+1} \mathbf{H}_k^{-1} \mathbf{a}_{k+1}} \mathbf{H}_k^{-1} \mathbf{a}_{k+1} \mathbf{a}'_{k+1} \mathbf{H}_k^{-1} \right). \end{aligned} \quad (5.89)$$

Similarly for \mathbf{b} :

$$\mathbf{b}_{k+1} = \mathbf{A}'_{k+1} W_{k+1} \mathbf{y}_{k+1} = \sum_{i=1}^{k+1} \alpha^{k+1-i} \mathbf{a}_i y_i = \alpha \mathbf{b}_k + \mathbf{a}_{k+1} y_{k+1}. \quad (5.90)$$

Similar to (5.85), the $\hat{\mathbf{x}}$ update has the following recursion:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{H}_{k+1}^{-1} \mathbf{b}_{k+1} = \hat{\mathbf{x}}_k + \frac{y_{k+1} - \mathbf{a}'_{k+1} \hat{\mathbf{x}}_k}{\alpha + \mathbf{a}'_{k+1} \mathbf{H}_k^{-1} \mathbf{a}_{k+1}} \mathbf{H}_k^{-1} \mathbf{a}_{k+1}. \quad (5.91)$$

There is extensive literature on RLS methods [106] with many extensions, including kernel methods [107], sparsity [108, 109], and missing data [110].

5.11 Summary

Key points about solving LS problems $\hat{x} = \arg \min_x \|Ax - y\|_2^2$:

- Every minimizer \hat{x} satisfies the normal equations $A'A\hat{x} = A'y$.
- If A has full column rank, then $A'A$ is invertible and the unique solution is $\hat{x} = (A'A)^{-1}A'y$.
- Otherwise there is a family of solutions of the form $\hat{x} = A^+y + N(A)$.
- Of that family, the minimum-norm LS solution is always $\hat{x} = A^+y$.
- If A is poorly conditioned, then use regularization or a truncated SVD instead of the pseudoinverse.
- If A is unitary or a Parseval tight frame, then $A^+ = A'$.
- The projection of a point y onto $\mathcal{R}(A)$ is $AA^+y = UU'y$ when $r > 0$ and is $\mathbf{0}$ when $r = 0$.

Solutions to Explorations

Explore 5.1 They ensure that we do matrix–vector products rather than matrix–matrix products.

Explore 5.2 Let $X \triangleq AB$ and $Y \triangleq (AB)^+P_{\mathcal{R}(A)}$. We want to show that $Y = X^+$ by checking the four defining properties of pseudoinverse.

$$\begin{aligned} XYX &= AB((AB)^+P_{\mathcal{R}(A)})AB = AB(AB)^+AB = XX^+X = X. \\ YXY &= ((AB)^+P_{\mathcal{R}(A)})AB((AB)^+P_{\mathcal{R}(A)}) = (AB)^+AB(AB)^+P_{\mathcal{R}(A)} \\ &= X^+XX^+P_{\mathcal{R}(A)} = X^+P_{\mathcal{R}(A)} = Y. \\ YX &= ((AB)^+P_{\mathcal{R}(A)})AB = (AB)^+AB \text{ is symmetric.} \\ XY &= AB((AB)^+P_{\mathcal{R}(A)}) = P_{\mathcal{R}(A)}AB(AB)^+P_{\mathcal{R}(A)} \text{ is symmetric.} \end{aligned}$$

Explore 5.3 No, because $\text{rank}(A) \leq \min(M, N) = M < N$ per (4.25). So wide cases are always under-determined.

Explore 5.4 Always, unless $T_\Phi = \mathbf{0}$.

Explore 5.5 $\sigma_1 = \dots = \sigma_N = 1$.

Problems

Problem 5.1 Suppose $A \in \mathbb{F}^{19 \times 48}$ has rank 8. Determine how many linearly independent solutions can be found to the homogeneous linear system $Ax = \mathbf{0}$.

Hint: Use (4.37).

Problem 5.2 Consider the following set of three measurements (x_i, y_i) : $(1, 2)$, $(2, 1)$, $(3, 3)$.

- (a) Find the line of the form $y = \alpha x + \beta$ that best fits (in the 2-norm sense) this data.
 - (b) Find the line of the form $x = \gamma y + \delta$ that best fits (in the 2-norm sense) this data.
- Hint:* Reuse your answer from part (a).

Problem 5.3 Polynomial regression application. Let $f(t) = 0.5e^{0.8t}$, $t \in [0, 2]$.

- (a) Suppose you are given 16 exact measurements of $f(t)$, taken at the times t in the following 1D array: $T = \text{range}(0, 2, 16)$.

Use JULIA to generate 16 exact measurements, $b_i = f(t_i)$, $i = 1, \dots, 16$, for $t_i \in T$. Now determine the coefficients of the least-squares polynomial approximation of the data b for

- (i) a polynomial of degree 15: $p_{15}(t)$;
- (ii) a polynomial of degree 2: $p_2(t)$.

Compare the quality of the two approximations graphically. Use `scatter` to first show b_i versus t_i for $i = 1, \dots, 16$, then use `plot!` to add plots of $p_{15}(t)$, $p_2(t)$, and $f(t)$ to see how well they approximate the function on the interval $[0, 2]$. Pick a very fine grid for the interval, for example, $t = (0:1000)/500$. Make the y -axis range equal $[-1, 4]$ by using the `ylim=(-1, 4)` option. As always, include axis labels and a clear legend. Submit your plot and also summarize the results qualitatively in one or two sentences.

For this problem, write your own JULIA code rather than using `fit` from the JULIA Polynomials package. (You may check your solutions using that function.)

- (b) Now suppose the measurements are affected by some noise. Generate the measurements using $y_i = f(t_i) + e_i$, $i = 1, \dots, 16$, as follows:

`using Random; seed!`

`seed!(3); e = randn(length(T))`

Determine the coefficients of the least-squares polynomial approximation of the (noisy) measurements y for

- (i) a polynomial of degree 15: $p_{15}(t)$;
- (ii) a polynomial of degree 2: $p_2(t)$.

Compare the two approximations as in part (a). Again, make the y -axis range equal $[-1, 4]$ by using the `ylim=(-1, 4)` option. Submit your plot and also summarize the results qualitatively in a couple of sentences, including comparing the behavior in (a) and (b).

- (c) Let $\hat{x}_n(b)$ and $\hat{x}_n(y)$ denote the LLS polynomial coefficients from noiseless b and noisy y , respectively, for a polynomial of degree n . Report the values of the residual norms $\|A\hat{x}_n(b) - b\|_2$ and $\|A\hat{x}_n(y) - y\|_2$ for the polynomial fits of degree 2 and degree 15. These residual norms describe how closely the fitted curve fits the data. Also report the fitting errors $\|A\hat{x}_n(y) - b\|_2$ for $n = 2, 15$. Arrange your results in a table as follows:

| Polynomial degree: | $d = 2$ | $d = 15$ |
|---|---------|----------|
| Residual norm $\ A\hat{x}(b) - b\ _2$ noiseless (a) | ? | ? |
| Residual norm $\ A\hat{x}(y) - y\ _2$ noisy (b) | ? | ? |
| Fitting error $\ A\hat{x}(y) - b\ _2$ | ? | ? |

- (d) Explain why the residual norm for degree 2 is smaller or larger than that for degree 15.

Problem 5.4 In the ordinary least-squares problem, we found the (minimum norm) \hat{x} that minimized $\|r\|_2$, where $r = b - Ax$ is the residual. We saw that the optimal x can be expressed entirely in terms of b and an SVD of A . Let A be an $M \times N$ matrix so that $x \in \mathbb{F}^N$, $b \in \mathbb{F}^M$, and $r \in \mathbb{F}^M$. In applications with heteroscedastic measurement errors, we prefer to minimize $\|Wr\|_2$, that is, a weighted squared error, where W is a diagonal matrix having diagonal entries $w_1, \dots, w_M \geq 0$. Determine the optimal x for this weighted least-squares problem. (Again, you should express the answer in terms of b , W , and an SVD of an appropriate matrix.) Your answer must not have any pseudoinverse in it! (It may have an inverse as long as you are sure that the matrix is invertible and trivial to invert.) You may assume that $W^*A == \text{zeros}(\text{size}(W^*A))$ is false in JULIA.

Problem 5.5 Recall that the least-squares problem (5.3) has the solution $\hat{x} = A^+b$, where A^+ denotes the pseudoinverse. When A is large, it can be expensive to compute \hat{x} using the pseudoinverse of A . In such settings, the gradient descent (GD) iteration given by

$$x_{k+1} = x_k - \alpha A' (Ax_k - b) \quad (5.92)$$

will converge (see Section 9.2.3) to a minimizer of $\|Ax - b\|_2$ as iteration $k \rightarrow \infty$ when $0 < \alpha < 2/\sigma_1^2(A)$. Note that the iteration has a fixed point $x_{k+1} = x_k$ if

$$A'(Ax_k - b) = 0,$$

which are exactly the normal equations. So any fixed point minimizes the least-squares cost function.

- (a) Write a function that implements the LS GD algorithm. The required inputs should be A and b . The optional arguments are the step size α (default $1/\sigma_1^2(A)$), an initial guess x_0 (default 0), and the number of iterations (default 200).
- (b) Use your function to generate a plot of $\log_{10}(\|x_k - \hat{x}\|)$ versus $k = 0, 1, \dots, 200$ using $\alpha = 1/\sigma_1^2(A)$ for A and b generated as follows:

```
</> 5.2
using Random; seed!
m, n = 100, 50; sigma = 0.1
seed!(0) # seed random number generator
A = randn(m, n); xtrue = rand(n); noise = randn(m)
b = A * xtrue + sigma * noise # b and xhat change with σ
```

Repeat for $\sigma = 0.5, 1, 2$ and submit one plot with all four curves on it, using a logarithmic scale for the vertical axis. Does $\|x_k - \hat{x}\|$ decrease monotonically with k in the plots? Note that $\sigma = \text{sigma}$ here is a noise standard deviation unrelated to singular values.

Problem 5.6 The GD iteration for the LS problem (5.3) for an $M \times N$ matrix A and vector $y \in \mathbb{R}^M$ is $x_{k+1} = x_k - \alpha A'(Ax_k - y)$. Instead of using a fixed step size α , an alternative is to do a line search to find the best step size *at each iteration*. This variation is called steepest descent (or GD with a line search). See (9.28). Find an expression for the step size α_k that is easily implemented, in terms of (some of) x_k, g_k, d_k, A, P , and/or y . Strive to find an expression that requires as little computation as possible by using quantities already computed.

Problem 5.7 Linear regression has a myriad of uses, including investigation of social justice issues.

⑥ Demo 5.5 has data collected by the College Board – the organization that runs the SAT exam for high-school students, from [111]. This data includes average SAT math scores for ten different family annual income brackets.

This problem uses this data to explore the relationship between income and SAT scores. Make a scatter plot of the midpoint of each income bracket versus the SAT math score. The last income bracket says “100+,” which does not have a midpoint, so just set it to be 120 (K\$). Use the template code in the demo. The scatter plot suggests there is a strong correlation between income and SAT score. You are going to fit four different models to the data (one at a time) and examine the fits and the coefficients:

$$\text{SAT_Math} \approx \begin{cases} \beta_0 & (\text{constant}), \\ \beta_0 + \beta_1 \cdot \text{income} & (\text{linear}), \\ \beta_0 + \beta_1 \cdot \text{income} + \beta_2 \cdot (\text{income})^2 & (\text{affine}), \\ \beta_0 + \beta_1 \cdot \text{income} + \beta_2 \cdot (\text{income})^2 & (\text{quadratic}). \end{cases}$$

For each of the four models, use an LLS fit to determine the β coefficient(s).

- (a) Make a single plot showing the data and the four fits for incomes between 0 and 130 (K\$), that is, $\text{income} = 0:130$. (We cannot predict anything for even higher incomes with this data.) Be sure to label your axes and use a legend to explain which points/lines are which.
- (b) Make a table to report your β coefficients that looks like this:

| Constant fit | Linear fit | Affine fit | Quadratic fit |
|--------------|------------|------------|---------------|
| β_0 | 0 | β_0 | β_0 |
| 0 | β_1 | β_1 | β_1 |
| 0 | 0 | 0 | β_2 |

- (c) In a statistics class you would analyze the coefficients β_1 and β_2 to assess whether they are significantly different from 0 and establish evidence of a relationship.

Even without formal statistics, you should be able to look at your plot and your table of coefficients and draw some conclusions about how equitable the SAT exam is.

Problem 5.8 Suppose $A \in \mathbb{F}^{M \times N}$ with $M \geq N$ has full column rank, that is, $\text{rank}(A) = N$. Show, using an SVD of A , that $A^+ = (A'A)^{-1}A'$.

Problem 5.9 Let $t_k = k\Delta$ for some integer k and $\Delta \in \mathbb{R}$. Consider the sum-of-sinusoids signal $y(t_k) = \sum_{i=1}^r b_i e^{i w_i t_k}$, where $y(t_k) \in \mathbb{C}^N$, $b_1, \dots, b_r \in \mathbb{C}^N$ are linearly independent vectors, $w_i \in [0, 1]$ are distinct, and $i = \sqrt{-1}$. Express the recursion in the simple matrix form $y(t_{k+1}) = Ay(t_k)$.

- Determine the matrix A .
- Determine the eigenvalues and eigenvectors of A .

Hint: For a matrix B of full column rank r we have $B^+B = I_r$, where I_r is an $r \times r$ identity matrix.

Problem 5.10 Verify that Σ^+ in (5.24) satisfies the four conditions for a pseudoinverse on p. 155.

Problem 5.11 For $A \in \mathbb{F}^{M \times N}$, $b \in \mathbb{F}^M$, and $x \in \mathbb{F}^N$, show that $(I - A^+A)x$ and A^+b are orthogonal vectors.

Hint: Use a compact SVD.

Problem 5.12 Let q_1 and q_2 denote two orthonormal vectors and b some fixed vector, all in \mathbb{F}^N .

- Find the optimal linear combination $\alpha q_1 + \beta q_2$ that is closest to b (in the Euclidean norm sense).
- Let $r = b - \alpha q_1 - \beta q_2$ denote the “residual error vector.” Show that r is orthogonal to both q_1 and q_2 .

Problem 5.13 Consider the problem of finding the minimum 2-norm solution of the linear least-squares problem

$$\hat{x} = \arg \min_x \|Ax - b\|_2 \text{ when } A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The solution is $\hat{x} = A^+b = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

- Consider a perturbation $E_1 = \begin{bmatrix} 0 & \delta \\ 0 & 0 \end{bmatrix}$ of A , where δ is a small positive number. Solve the perturbed version of the above problem: $z^* = \arg \min_z \|A_1 z - b\|_2$, where $A_1 = A + E_1$. What happens to $\|\hat{x} - z^*\|_2$ as δ approaches 0?
- Now consider the perturbation $E_2 = \begin{bmatrix} 0 & 0 \\ 0 & \delta \end{bmatrix}$, where again δ is a small positive number. Solve the perturbed problem $z^* = \arg \min_z \|A_2 z - b\|_2$ where $A_2 = A + E_2$. What happens to $\|\hat{x} - z^*\|_2$ here as $\delta \rightarrow 0$?

Problem 5.14

- (a) Find (by hand) all solutions of the linear least-squares problem

$$\arg \min_{x \in \mathbb{R}^2} \|Ax - b\|_2 \text{ when } A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \text{ and } b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

- (b) Describe briefly how you would modify your solution for this variation of the problem: $\arg \min_{x \in \mathbb{C}^2} \|Ax - b\|_2$.

Problem 5.15 In many artificial neural network models used in machine learning, the final layer is often “dense” or “fully connected” and often is affine or linear. This problem focuses on the linear case.

In a supervised learning setting, we are given training data (x_n, y_n) , $n = 1, \dots, N$, consisting of pairs of features $x_n \in \mathbb{R}^M$ and responses $y_n \in \mathbb{R}$. A linear artificial neuron makes a prediction simply by computing the inner product of an input feature $x \in \mathbb{R}^M$ with a (learned) weight vector $w \in \mathbb{R}^M$, that is, $\hat{y}_n = w' x_n$. We want to train the weight vector w to minimize the average loss over the training data by solving the following optimization problem:

$$\hat{w} = \arg \min_w L(w), \quad L(w) = \frac{1}{N} \sum_{n=1}^N \ell(y_n, \hat{y}_n), \quad \hat{y}_n = w' x_n.$$

Determine analytically the optimal weight vector \hat{w} in the case where the loss function is the squared error $\ell(y_n, \hat{y}_n) = \frac{1}{2}(y_n - \hat{y}_n)^2$.

You may assume that the data matrix $X = [x_1 \ \dots \ x_N]$ has full row rank. State any conditions needed on N for this assumption (and hence your answer) to be valid.

Hint: You should be able to set this up as a linear least-squares problem and express your answer in terms of the training data feature correlation matrix $K_x = (1/N) \sum_{n=1}^N x_n x_n'$ and the cross-correlation between the training data features and responses, $K_{yx} = (1/N) \sum_{n=1}^N y_n x_n'$.

Problem 5.16

- (a) For $\delta > 0$, determine the solution of the regularized LS problem

$$\hat{x}(\delta) = \arg \min_x \frac{1}{2} \|Ax - b\|_2^2 + \delta^2 \frac{1}{2} \|x\|_2^2.$$

Express the answer in terms of the SVD of an appropriate matrix.

Use the fact that $B^+ = (B'B)^{-1}B'$ when $B'B$ is invertible to simplify the expression.

Hint: Rewrite the cost function so it looks like a usual least-squares problem.

- (b) What does $\hat{x}(\delta)$ tend to as $\delta \rightarrow \infty$? Does this answer make sense?
- (c) Write (by hand, not code) an iteration based on the gradient descent (GD) method such that the iterates converge to the minimizer $\hat{x}(\delta)$.
- (d) Determine a condition on the step size μ that guarantees convergence of your GD method. Express the condition in terms of the original problem quantities A , b , and δ .

- Problem 5.17** (a) Consider the LLS problem (5.3). When A has full column rank, the solution is $\hat{x} = (A'A)^{-1}A'y$, which involves inverting $A'A$. Express the condition number of $A'A$ in terms of the singular values of A .
- (b) The Tikhonov regularized solution (5.44) involves inverting a different matrix. Express the condition number of that matrix in terms of the singular values of A and $\beta > 0$. Verify that the regularized solution has a “better” condition number.

Problem 5.18 Prove that $\sigma_1^2(A)I \succeq A'A$. This inequality is important for establishing the convergence of certain iterative algorithms for LS and related problems.

Problem 5.19 Generalize the SVD construction in (5.59) to the case

$$T_\Phi = [Q_1 \quad \cdots \quad Q_K] / \sqrt{K},$$

where each Q_k is a unitary matrix.

Problem 5.20 Suppose $b = T_\Phi z$ for some vector $z \in \mathbb{F}^M$, where $T_\Phi \in \mathbb{F}^{M \times N}$ is the analysis operator for a Parseval tight frame.

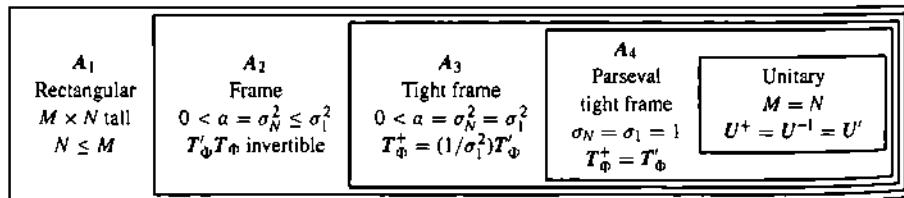
- (a) For $\delta \geq 0$, express a solution of the regularized LS problem

$$\hat{x} = \arg \min_{x \in \mathbb{F}^M} \|T_\Phi x - b\|_2^2 + \delta^2 \|x\|_2^2$$

in terms of z as simply as possible.

- (b) When is your solution unique?
 (c) What value of δ would be the most appropriate in this special case where T_Φ corresponds to a Parseval tight frame?
 (d) What is the condition number of T_Φ here?

Problem 5.21 The following Venn diagram (cf. Fig. 5.10) describes the relationship between a frame analysis operator T_Φ and other matrices.



Each category is a *strict superset* of the categories nested within it.

- (a) Provide example matrices A_1, \dots, A_4 that belong to each of the categories above but *not* the next category nested within it. In each case, choose the simplest possible example of T_Φ (!) from these options:

$$B_0 = [0], \quad B_1 = [1], \quad B_2 = [2], \quad B_3 = [1], \quad B_4 = [1 \ 1], \quad B_5 = [1 \ 1]/\sqrt{2},$$

$$B_6 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad B_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}/\sqrt{2}, \quad B_8 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad B_9 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \quad B_{10} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

$$\mathbf{B}_{11} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B}_{12} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

- (b) A subset of unitary matrices is certain diagonal matrices. Describe the necessary and sufficient conditions for a diagonal matrix to be unitary.

Problem 5.22

- (a) Show that if \mathbf{T}_Φ is the analysis operator for a tight frame with frame bound α , then $\mathbf{T}_\Phi \mathbf{T}'_\Phi \preceq \alpha I$, that is, $\alpha I - \mathbf{T}_\Phi \mathbf{T}'_\Phi$ is positive semidefinite.
 (b) What can you say about $\alpha I - \mathbf{T}'_\Phi \mathbf{T}_\Phi$?

Problem 5.23 Let matrix \mathbf{A} have compact SVD $\mathbf{A} = \mathbf{U}_r \Sigma_r \mathbf{V}'_r$ where $\mathbf{U} = [\mathbf{U}_r \ \mathbf{U}_0]$ and $\mathbf{V} = [\mathbf{V}_r \ \mathbf{V}_0]$. Express the following subspaces (ranges and null spaces) in terms of \mathbf{U}_r , \mathbf{U}_0 , \mathbf{V}_r , and/or \mathbf{V}_0 .

- (a) $N^\perp(\mathbf{A}')$
- (b) $R(\mathbf{A}')$
- (c) $R^\perp(\mathbf{A}\mathbf{A}^+)$
- (d) $N(\mathbf{A}^+)$
- (e) $R(\mathbf{A}^+\mathbf{A})$

Problem 5.24 Given constants a , b , and c , consider $\{(x, y, z) \in \mathbb{R}^3 : ax + by + cz = 0\}$, a plane that intersects the origin.

- (a) Describe how you would use an SVD to find basis vectors for the plane. How many basis vectors are required to express a point on the plane?
 (b) Find the point on the plane that is closest to an arbitrary point $(\alpha, \beta, \gamma) \in \mathbb{R}^3$. Your expression should be general and reasonably simple.
 (c) Using your preceding answer (and probably JULIA or a calculator), find the point on the plane $x + 2y + 3z = 0$ that is closest to the point $(4, 5, 6)$.

Problem 5.25 Prove that if \mathbf{P} is a projection matrix, and, for a square matrix \mathbf{A} , we define

$$e^{\mathbf{A}} = \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{A}^k,$$

where $k!$ denotes the factorial of k , then $e^{\mathbf{P}} = \mathbf{I} + (\mathbf{e} - 1)\mathbf{P}$.

Hint: What is \mathbf{P}^k when \mathbf{P} is a projection matrix?

Problem 5.26 Throughout the problems below you may reuse properties you derived in earlier parts as long as you refer back to the properties you use.

- (a) Let $\mathbf{A} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}'_i$ denote a compact SVD of \mathbf{A} , where r denotes the rank of \mathbf{A} . Express the following in terms of the singular vectors $\{\mathbf{u}_i\}$ and $\{\mathbf{v}_i\}$:
- (i) $\mathbf{P}_{R(\mathbf{A})}$
 - (ii) $\mathbf{P}_{R^\perp(\mathbf{A})}$
 - (iii) $\mathbf{P}_{R(\mathbf{A}')}$
 - (iv) $\mathbf{P}_{R^\perp(\mathbf{A}')}$

- (v) $P_{N(A)}$
 (vi) $P_{N^\perp(A)}$
 (vii) $P_{N(A')}$
 (viii) $P_{N^\perp(A')}$
- (b) Let $A = U\Sigma V'$ denote a (full) SVD of A , and r the rank of A . Express the following in terms of the singular vectors $\{u_i\}$ and $\{v_i\}$:
- $P_{\mathcal{R}(A)}^2$
 - $P_{\mathcal{R}(A)}^3$
 - $P_{\mathcal{R}(A)}^k$ for a positive integer k
 - $(I - P_{\mathcal{R}(A)})P_{\mathcal{R}(A)}$
 - $(I - P_{\mathcal{R}(A)})^k$ for a positive integer k
 - $(I - P_{\mathcal{R}(A)})^k P_{\mathcal{R}(A)}^j$ for positive integers j and k
 - $P_{\mathcal{R}(A)} - P_{\mathcal{R}(A)}'$
 - $\text{rank}(P_{\mathcal{R}(A)})$
 - $\text{rank}(P_{\mathcal{R}^\perp(A)})$
 - $\text{rank}(I - P_{\mathcal{R}(A)})$
- (c) Let $A = U\Sigma V'$ be an SVD of A , and r the rank of A . Express the following in terms of the singular vectors $\{u_i\}$ and $\{v_i\}$:
- $\|P_{\mathcal{R}(A)}\|_F$
 - $\|P_{\mathcal{R}(A)}^k\|_F$ for a positive integer k
 - $\|I - P_{\mathcal{R}(A)}\|_F$
 - $\|(I - P_{\mathcal{R}(A)})^k\|_F$ for a positive integer k
 - $\|P_{\mathcal{R}(A)}(I - P_{\mathcal{R}(A)})\|_F$
 - $\prod_{i=1}^k (I - P_{\mathcal{R}(u_i)}) - P_{\mathcal{R}^\perp(u_1, \dots, u_k)}$ for any positive integer k
- (d) Let $A = U\Sigma V'$ be an SVD of A , and r the rank of A . Find expressions for both a compact SVD and a full SVD for each of the following matrices.
- $P_{\mathcal{R}(A)}$
 - $P_{\mathcal{R}(A)}^\perp$
 - $P_{\mathcal{R}(u_1)} A$
 - $P_{\mathcal{R}(u_1, \dots, u_k)} A$ for $k = 1, \dots, r$
 - $P_{\mathcal{R}(u_1, \dots, u_k)} A$ for $r + 1 \leq k \leq \min(m, n)$
 - $A P_{\mathcal{R}(v_1)}$
 - $A P_{\mathcal{R}(v_1, \dots, v_k)}$ for $k = 1, \dots, r$
 - $A P_{\mathcal{R}(v_1, \dots, v_k)}$ for $1 \leq k \leq \min(m, n)$
Hint: Think carefully about r .
 - $P_{\mathcal{R}(u_1, \dots, u_k)} A P_{\mathcal{R}(v_1, \dots, v_k)}$ for $k = 1, \dots, r$
 - $P_{\mathcal{R}(u_1, \dots, u_j)} A P_{\mathcal{R}(v_1, \dots, v_k)}$ for $1 \leq j, k \leq \min(m, n)$

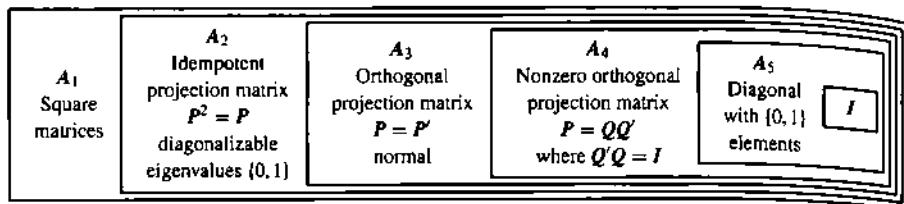
Problem 5.27 Let A be an $M \times N$ matrix with SVD $A = U\Sigma V'$. Let u_k denote the k th column of U , and v_k denote the k th column of V . Let σ_k be the k th diagonal element of

Σ , and let A have rank r . Let I_d denote the $d \times d$ identity matrix. Simplify the following pseudoinverse expressions as much as possible.

- (a) (i) $(u_i)^+$
 (ii) $(u_i u_i')^+$
 (iii) $(u_i v_i')^+$
 (iv) $(\sigma_i u_i v_i')^+$
- (b) (i) U^+
 (ii) V^+
 (iii) UU^+
 (iv) VV^+
 (v) Σ^+
 (vi) $(U\Sigma)^+$
 (vii) $(\Sigma V')^+$
 (viii) $A^+ = (U\Sigma V')^+$
 (ix) $(UV')^+$
- (c) (i) $U[:, 1:k]^+$ for $1 \leq k \leq M$
 (ii) $V[:, 1:k]^+$ for $1 \leq k \leq N$
 (iii) $\Sigma[1:k, 1:k]^+$ for $1 \leq k \leq \min(M, N)$
 (iv) $U[:, 1:k]U[:, 1:k]^+$ for $1 \leq k \leq M$
 (v) $V[:, 1:k]V[:, 1:k]^+$ for $1 \leq k \leq N$
 (vi) $U[:, 1:k]^+U[:, 1:k]$ for $1 \leq k \leq M$
 (vii) $V[:, 1:k]^+V[:, 1:k]$ for $1 \leq k \leq N$
- (d) (i) $I_M - U[:, 1:k]U[:, 1:k]^+$ for $1 \leq k \leq M$
 (ii) $I_N - V[:, 1:k]V[:, 1:k]^+$ for $1 \leq k \leq N$
 (iii) $I_k - U[:, 1:k]^+U[:, 1:k]$ for $1 \leq k \leq M$
 (iv) $I_k - V[:, 1:k]^+V[:, 1:k]$ for $1 \leq k \leq N$
- (e) (i) $(U[:, 1:k]\Sigma[1:k, 1:k]V[:, 1:k]')^+$ for $1 \leq k \leq r$
 (ii) $(A')^+$
 (iii) $(A^+)^t$
 (iv) $(\alpha A)^+$ for $\alpha \neq 0$
 (v) $(AA^+)^t$
 (vi) $(A^+A)^t$
- (f) (i) AA^+
 (ii) A^+A
 (iii) AA^+A
 (iv) A^+AA^+
 (v) $(P_{R(A)})^+$
- (g) (i) $(A'A)^+A'$
 (ii) $A'(AA')^+$
 (iii) A^+A if $r = N$

- (iv) $\mathbf{A}\mathbf{A}^+$ if $r = M$
- (v) $(\mathbf{A}'\mathbf{A})^{-1}\mathbf{A}'$ if $r = N$
- (vi) $\mathbf{A}'(\mathbf{A}\mathbf{A}')^{-1}$ if $r = M$

Problem 5.28 The following Venn diagram (cf. Fig. 5.12) summarizes relationships related to projection operations.



Each category is a *strict superset* of the categories nested within it. Provide example matrices A_1, \dots, A_5 that belong to each of the categories above but *not* the next category nested within it. Try to provide the simplest possible example in each case.

Problem 5.29

- (a) Let A be a real $N \times N$ idempotent matrix that satisfies $x'Ax = 0 \forall x \in \mathbb{R}^N$. Prove that $A = \mathbf{0}$ or provide a counterexample.
- (b) Let A be a real $N \times N$ positive semidefinite matrix that satisfies $x'Ax = 0 \forall x \in \mathbb{R}^N$. Prove that $A = \mathbf{0}$ or provide a counterexample.

6 Norms and Procrustes Problems

6.1 Introduction

So far, the norms we have considered are the Euclidean norm $\|x\|_2$, the spectral norm $\|A\|_2$, and the Frobenius norm $\|A\|_F$ in Problem 3.9. These three norms are particularly important, but there are many other important norms for DS–ML–SP applications.

This chapter discusses vector norms in Section 6.2 and matrix norms and operator norms in Section 6.4. Section 6.5 uses these norms to analyze convergence of sequences. Section 6.6 revisits the Moore–Penrose pseudoinverse from a norm-minimizing perspective. Section 6.7 uses norms in the Procrustes problem.

The Tikhonov regularized least-squares problem in Section 5.6.5 illustrates the two primary uses of norms, namely, quantifying *distance* and *size*:

$$\hat{x} = \arg \min_x \underbrace{\|Ax - y\|_2^2}_{\text{distance: how far}} + \underbrace{\beta \|x\|_2^2}_{\text{size: how big}} \quad (6.1)$$

(how different)

The DS–ML–SP literature has many such optimization problems involving different norms.

6.2 Vector Norms

So far, the only vector norm discussed here has been the common Euclidean norm. Many other vector norms are also important in DS–ML–SP.

Definition [4, p. 57] A norm on a vector space \mathcal{V} defined over a field \mathbb{F} is a function $\|\cdot\|$ from \mathcal{V} into $[0, \infty)$ that satisfies the following properties $\forall x, y \in \mathcal{V}$:

$$\|x\| \geq 0 \quad (\text{nonnegative}), \quad (6.2a)$$

$$\|x\| = 0 \text{ iff } x = \mathbf{0} \quad (\text{positive}), \quad (6.2b)$$

$$\|\alpha x\| = |\alpha| \|x\| \forall \alpha \in \mathbb{F} \quad (\text{absolute homogeneity}), \quad (6.2c)$$

$$\|x + y\| \leq \|x\| + \|y\| \quad (\text{triangle inequality or subadditivity}). \quad (6.2d)$$

Explore 6.1 Show that the nonnegative condition is redundant, that is, that it follows from the other three.

Definition A function from \mathcal{V} to $[0, \infty)$ that satisfies all the properties above except positivity is called a seminorm.

6.2.1 Examples of Vector Norms

For $1 \leq p < \infty$, the ℓ_p norm is

$$\|x\|_p \triangleq \left(\sum_i |x_i|^p \right)^{1/p}. \quad (6.3)$$

- The vector 2-norm or Euclidean norm is the case $p = 2$: $\|x\|_2 \triangleq \sqrt{\sum_i |x_i|^2}$.
- The 1-norm or Manhattan norm is the case $p = 1$: $\|x\|_1 \triangleq \sum_i |x_i|$.

The triangle inequality for this norm is the Minkowski inequality, that in turn uses Hölder's inequality.

The infinity norm or ℓ_∞ norm or uniform norm or sup norm is

$$\|x\|_\infty \triangleq \sup \{|x_1|, |x_2|, \dots\}, \quad (6.4)$$

where sup denotes the supremum (least upper bound) of a set. One can show [112, Problem 2.12] that

$$\|x\|_\infty = \lim_{p \rightarrow \infty} \|x\|_p. \quad (6.5)$$

For the finite-dimensional vector space \mathbb{F}^N , the supremum simplifies to a maximum also called the max norm:

$$\|x\|_\infty \triangleq \max \{|x_1|, \dots, |x_N|\}. \quad (6.6)$$

For quantifying the (non)sparsity of a vector, it is useful to note that

$$\lim_{p \rightarrow 0} \|x\|_p^p = \sum_i \mathbb{I}_{[x_i \neq 0]} \triangleq \|x\|_0, \quad (6.7)$$

where $\mathbb{I}_{\{\cdot\}}$ denotes the indicator function that is unity if the argument is true and zero if false. However, the “0-norm” $\|x\|_0$ is *not* a vector norm because it does not satisfy at least one of the conditions of the norm definition above. The proper name for $\|x\|_0$ is counting measure.

Sometimes we want a weighted norm; for example, a weighted Euclidean norm is

$$\|x\|_W = \sqrt{(x' W x)}. \quad (6.8)$$

Example 6.1 In particular, if W is an $N \times N$ diagonal matrix with positive diagonal elements w_i , then

$$\|x\|_W = \left(\sum_{i=1}^N w_i |x_i|^2 \right)^{1/2} \quad (6.9)$$

Fact 6.1 The functional $\|x\|_W$ in (6.8) is a norm iff W is a positive definite matrix (Problem 6.1).

Q6.1 Which of the four properties of a vector norm in (6.2) does the counting measure $\|\cdot\|_0$ satisfy?

- A: (6.2a), (6.2b)
- B: (6.2a), (6.2c)
- C: (6.2a), (6.2b), (6.2c)
- D: (6.2a), (6.2b), (6.2d)
- E: (6.2a), (6.2c), (6.2e)

Explore 6.2 For a vector $x \in \mathbb{F}^N$ the (discrete) total variation (TV) is defined as $\text{TV}(x) \triangleq \sum_{n=2}^N |x_n - x_{n-1}|$. Determine whether this functional is a norm, a seminorm, or neither of these.

6.2.2 Practical Implementation

For the preceding examples, in JULIA, first invoke `using LinearAlgebra` then use:

```

 $\|v\|_p$  norm(v, p)
 $\|v\|_2$  norm(v, 2) or just norm(v)
 $\|v\|_1$  norm(v, 1)
 $\|v\|_\infty$  norm(v, Inf)
 $\|v\|_0$  norm(v, 0) or count(!=(0), v)

```

 Caution. For $p < 1$, $\|\cdot\|_p$ is not a proper vector norm, though it is sometimes used in practical problems and `norm(v, p)` will evaluate (6.3) for any $-\infty \leq p \leq \infty$.

Q6.2 If W is `Diagonal(w)` for $w > 0$, then which command computes the weighted norm $\|x\|_W$?

- A: `norm(x .* w)`
- B: `norm(x .* w, 2)`
- C: `norm(x .* w.^2)`
- D: `norm(x .* sqrt.(w))`
- E: None of these

6.2.3 Properties of Vector Norms

- Let $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ be any two vector norms on a *finite-dimensional* space. Then there exist finite positive constants C_m and C_M (that depend on α , β , and the vector space dimension) such that:

$$C_m \|\cdot\|_\alpha \leq \|\cdot\|_\beta \leq C_M \|\cdot\|_\alpha. \quad (6.10)$$

In a sense, then, “all norms are equivalent” to within constant factors. This equivalence is especially relevant when examining the convergence of a sequence (see Section 6.5); if a sequence converges with respect to one norm, then it converges with respect to any norm.

- For any vector norm, the reverse triangle inequality is

$$\| \|x\| - \|y\| \| \leq \|x - y\| \quad \forall x, y \in \mathcal{V}. \quad (6.11)$$

Proof. $\|x\| = \|x - y + y\| \leq \|x - y\| + \|y\| \implies \|x\| - \|y\| \leq \|x - y\|$. Similarly, $\|y\| - \|x\| \leq \|x - y\|$. Now combine these two inequalities. \square

- Any vector norm $\|\cdot\|$ on a vector space \mathcal{V} is a convex function:

$$\|\alpha x + (1 - \alpha)z\| \leq \alpha \|x\| + (1 - \alpha)\|z\|, \quad \forall \alpha \in [0, 1], \quad \forall x, z \in \mathcal{V}. \quad (6.12)$$

This fact is easy to prove using the triangle inequality and the homogeneity property (Problem 4.19).

- For $1 < p < \infty$, the function $f(x) \triangleq \|x\|_p^p$ is strictly convex.
- For any norm, the ball of radius $r > 0$, $\{x : \|x\| \leq r\}$ is a convex set (Problem 6.2).

Example 6.2 Figure 6.1 illustrates some norm balls in \mathbb{R}^2 .

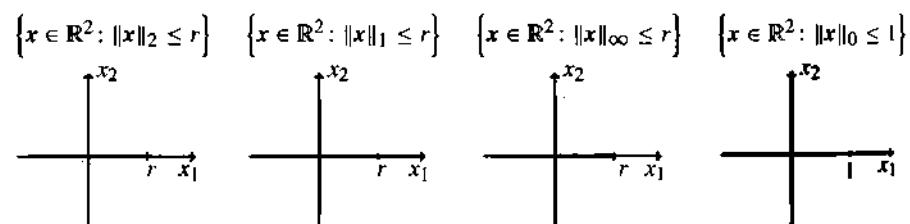


Figure 6.1 Norm balls in \mathbb{R}^2 .

Explore 6.3 For $1 < p \leq \infty$, determine whether the norm function $f(x) \triangleq \|x\|_p$ is strictly convex on \mathbb{R}^N .

Q6.3 For $C = \{x \in \mathbb{R}^N : \|x\|_\infty \leq 5\}$, which of these is the projection of a point $z \in \mathbb{R}^N$ onto C ? (See Fig. 6.2.)

- A: $\min.(z, 5)$
 B: $\min.(\text{abs.}(z), 5)$

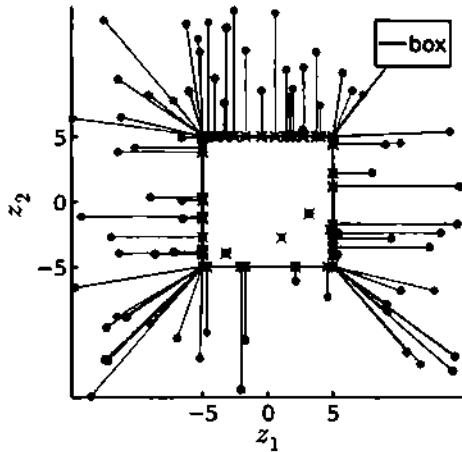


Figure 6.2 Illustration of projection onto ball C for Q6.3.

C: `min.(abs.(z), 5).*sign.(z)`

D: None

Explore 6.4 What if we use \mathbb{C}^N instead of \mathbb{R}^N here?

6.2.4 Norm Notation

Some math literature uses $|x|$ instead of $\|x\|$ to denote a vector norm. That notation should be avoided for matrices where $|A|$ often denotes the determinant of A . Sometimes one must determine from context what $|\cdot|$ means in such literature.

6.2.5 Robust Regression Application

This section describes one of many applications of the ℓ_p norm (6.3). By defining the residual $r \triangleq Ax - y \in \mathbb{R}^M$, we can rewrite the LLS cost function (5.7) as

$$f(x) = \frac{1}{2} \|Ax - y\|_2^2 = \frac{1}{2} \|r\|_2^2 = \frac{1}{2} \sum_{m=1}^M |r_m|^2. \quad (6.13)$$

The squared residual term $|r_m|^2$ associated with the squared Euclidean norm $\|\cdot\|_2^2$ is convenient for deriving the gradient (5.8) and the normal equations (5.12), but it is *not* robust to outliers in the measurements y . The parabola defined by $|r_m|^2$ rises very rapidly for residuals that become far from zero, so the LLS estimate ends up overfitting outlier data. To develop a regression method that is more robust to outliers, we want to use a function of the residuals that rises less rapidly than the parabola $|r_m|^2$. A natural choice is to use $|r_m|^p$ for $1 \leq p \ll 2$. That choice leads to the following optimization problem:

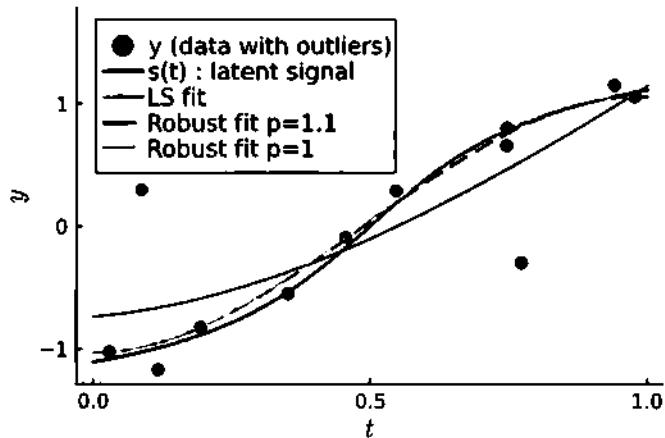


Figure 6.3 Illustration of robust polynomial regression using $\|Ax - y\|_p$ for $1 \leq p \ll 2$.

$$\hat{x} = \arg \min_{x \in \mathbb{F}^N} f_p(x), \quad f_p(x) \triangleq \frac{1}{p} \|Ax - y\|_p^p. \quad (6.14)$$

The choice $p = 1$, corresponding to the cost function $\|Ax - y\|_1$, is a classic one for robust regression, but it has the drawback of not being differentiable, complicating the optimization problem. Using $1 < p \ll 2$ provides robustness while still being differentiable.

- Demo 6.1 and Fig. 6.3 illustrate the benefits of replacing the LS cost function $\|Ax - y\|_2$ with the ℓ_p norm $\|Ax - y\|_p$ for $1 \leq p \ll 2$. The results for $p = 1$ and $p = 1.1$ are very similar, and both fit the latent signal much better by avoiding over-fitting the two outlier points.

6.2.6 Unitarily Invariant Vector Norms

Some vector norms have the following useful invariance property.

Definition A vector norm $\|\cdot\|$ on \mathbb{F}^N is unitarily invariant iff for every unitary matrix $U \in \mathbb{F}^{N \times N}$,

$$\|Ux\| = \|x\| \quad \forall x \in \mathbb{F}^N. \quad (6.15)$$

Example 6.3 The Euclidean norm $\|\cdot\|_2$ on \mathbb{F}^N is unitarily invariant, because for any unitary U , per (2.38),

$$\|Ux\|_2 = \sqrt{(Ux)'(Ux)} = \sqrt{x'U'Ux} = \sqrt{x'x} = \|x\|_2 \quad \forall x \in \mathbb{F}^N. \quad (6.16)$$

As noted after (2.38), this property is related to Parseval's identity and the Plancherel theorem.

Explore 6.5 Determine whether $\|\cdot\|_1$ is unitarily invariant.

◆ ◆ **Example 6.4** Another unitary invariant norm on \mathbb{F}^N is $\|\cdot\|_{(\alpha)} \triangleq \alpha \|\cdot\|_2$ for any $\alpha > 0$.

6.3 Inner Products

Many of the vector spaces used in DS–ML–SP are inner product spaces, meaning a vector space with an associated inner product operation.

Definition For a vector space \mathcal{V} over the field \mathbb{F} , an inner product is a function $\langle \cdot, \cdot \rangle : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{F}$ that must satisfy the following axioms $\forall x, y \in \mathcal{V}, \alpha \in \mathbb{F}$:

$$\langle x, y \rangle = \langle y, x \rangle^* \quad (\text{Hermitian symmetry}), \quad (6.17a)$$

$$\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle \quad (\text{additivity}), \quad (6.17b)$$

$$\langle \alpha x, y \rangle = \alpha \langle x, y \rangle \quad (\text{scaling}), \quad (6.17c)$$

$$\langle x, x \rangle \geq 0 \text{ and } \langle x, x \rangle = 0 \text{ iff } x = 0 \quad (\text{positive definite}). \quad (6.17d)$$

6.3.1 Examples of Inner Products

Example 6.5 For vectors in \mathbb{F}^N , the usual inner product is

$$\langle x, y \rangle = \sum_{n=1}^N x_n y_n^*. \quad (6.18)$$

Example 6.6 The weighted bilinear form $\langle x, y \rangle = y' A x$ is an inner product for \mathbb{F}^N iff $A \succ 0$ (Problem 6.4 and Problem 6.5).

Example 6.7 For the (infinite-dimensional) vector space of square integrable functions on the interval $[a, b]$, the following integral is a valid inner product:

$$\langle f, g \rangle = \int_a^b f(t)g^*(t) dt. \quad (6.19)$$

◆ ◆ An integral like this is used to calculate Fourier series coefficients.

Example 6.8 For two real, zero-mean random variables X, Y defined on a joint probability space, a natural inner product is $E[XY]$. (Keep in mind that random variables are functions.) With this definition, the corresponding norm is

$\|X\| \triangleq \sqrt{\langle X, X \rangle} = \sqrt{E[X^2]} = \sigma_X$, the standard deviation of X . Here, the

◆ ◆ Cauchy–Schwarz inequality is equivalent to the usual bound on the correlation coefficient:

$$\rho_{X,Y} \triangleq \frac{E[XY]}{\sigma_X \sigma_Y} \implies |\rho_{X,Y}| \leq 1.$$

Explore 6.6 With this definition of inner product, what types of random variables are “orthogonal”?

Example 6.9 For two matrices $A, B \in \mathbb{F}^{M \times N}$ (a vector space!), the Frobenius inner product, also called the Hilbert–Schmidt inner product, is defined as (Problem 6.6):

$$\langle A, B \rangle \triangleq \text{trace}\{AB'\} = \sum_{m=1}^M \sum_{n=1}^N a_{mn} b_{mn}^* = \langle \text{vec}(A), \text{vec}(B) \rangle, \quad (6.20)$$

where the latter inner product is the ordinary Euclidean inner product on \mathbb{F}^{MN} .

6.3.2 Properties of Inner Products

- Any valid vector inner product induces a valid (Problem 6.7) vector norm satisfying (6.2):

$$\|x\| = \sqrt{\langle x, x \rangle}. \quad (6.21)$$

Example 6.10 The induced norm corresponding to the Frobenius inner product is the Frobenius norm:

$$\|A\|_F = \sqrt{\langle A, A \rangle}. \quad (6.22)$$

- Bilinearity. $\forall \{x_i\}, \{y_j\} \in \mathcal{V}, \forall \{\alpha_i\}, \{\beta_j\} \in \mathbb{F}$:

$$\left\langle \sum_i \alpha_i x_i, \sum_j \beta_j y_j \right\rangle = \sum_i \sum_j \alpha_i \beta_j^* \langle x_i, y_j \rangle. \quad (6.23)$$

- ◆ ◆ ◆ A vector norm satisfies the parallelogram law,

$$\frac{1}{2} (\|x+y\|^2 + \|x-y\|^2) = \|x\|^2 + \|y\|^2 \quad \forall x, y \in \mathcal{V}, \quad (6.24)$$

iff it is induced by an inner product via (6.21). The required inner product is

$$\begin{aligned} \langle x, y \rangle &\triangleq \frac{1}{4} (\|x+y\|^2 - \|x-y\|^2 + i\|x+iy\|^2 - i\|x-iy\|^2) \\ &= \frac{\|x+y\|^2 - \|x\|^2 - \|y\|^2}{2} + i \frac{\|x+iy\|^2 - \|x\|^2 - \|y\|^2}{2}. \end{aligned} \quad (6.25)$$

- The Cauchy–Schwarz inequality (or Schwarz or Cauchy–Bunyakovsky–Schwarz inequality) states that

$$|\langle x, y \rangle| \leq \|x\| \|y\| = \sqrt{\langle x, x \rangle} \sqrt{\langle y, y \rangle} \quad \forall x, y \in \mathcal{V} \quad (6.26)$$

for a norm $\|\cdot\|$ induced by an inner product $\langle \cdot, \cdot \rangle$ via (6.21), with equality iff x and y are linearly dependent.

Example 6.11 Applying the inequality (6.26) to the Frobenius inner product for matrices yields

$$|\text{trace}\{AB'\}| = |\langle A, B \rangle| \leq \|A\|_F \|B\|_F. \quad (6.27)$$

Q6.4 In an inner product space on \mathbb{R}^N , is $\langle x, y \rangle \leq \|x\| \|y\|$, where $\|\cdot\|$ is the induced norm for $\langle \cdot, \cdot \rangle$?

A: Yes, always

B: Not always

C: Never

Proof of Cauchy-Schwarz inequality for \mathbb{R}^N . For any $x, y \in \mathbb{R}^N$ let $A = [x \ y]$, so $A'A = \begin{bmatrix} x'x & x'y \\ y'x & y'y \end{bmatrix}$. $A'A$ is Hermitian symmetric \Rightarrow its eigenvalues are all real and nonnegative $\Rightarrow \det\{A'A\} \geq 0 \Rightarrow (x'x)(y'y) - (y'x)(x'y) \geq 0 \Rightarrow |x'y|^2 \leq (x'x)(y'y) = \|x\|_2^2 \|y\|_2^2$. Taking the square root of both sides yields the inequality. (We used the fact that $(y'x)(x'y) = \langle x, y \rangle \langle y, x \rangle = \langle x, y \rangle (\langle x, y \rangle)^* = |\langle x, y \rangle|^2$.) \square

6.3.3 More Inner Product Inequalities

For the usual inner product on \mathbb{R}^N :

$$|\langle x, y \rangle| \leq \sum_{n=1}^N |x_n y_n| \leq \|x\|_1 \|y\|_\infty. \quad (6.28)$$

Proof. $|\langle x, y \rangle| = \left| \sum_{n=1}^N x_n y_n^* \right| \leq \sum_n |x_n| |y_n| \leq \sum_n |x_n| \|y\|_\infty = \|x\|_1 \|y\|_\infty$. \square

If $1/p + 1/q = 1$ and $1 < p, q < \infty$, then Hölder's inequality states that

$$|\langle x, y \rangle| \leq \sum_{n=1}^N |x_n y_n| \leq \|x\|_p \|y\|_q, \quad (6.29)$$

again for the usual inner product on \mathbb{R}^N .

6.3.4 Angle Between Vectors

Definition The angle θ between two nonzero vectors $x, y \in \mathcal{V}$ with respect to the inner product $\langle \cdot, \cdot \rangle$ having induced norm $\|\cdot\|$ is defined by

$$\cos \theta = \frac{|\langle x, y \rangle|}{\|x\| \|y\|} \implies \theta \in [0, \pi/2]. \quad (6.30)$$

For real vectors, we can omit the absolute value and obtain $\theta \in [0, \pi]$. The Cauchy-Schwarz inequality is equivalent to the statement $|\cos \theta| \leq 1$. Section 2.5.3 introduced this inequality in Euclidean space, but in fact it holds in any inner product space.

Explore 6.7 Determine the angle in $\mathbb{F}^{N \times N}$ between I_N and xx' for $x \in \mathbb{F}^N \setminus \{0\}$.

6.3.5 Angle Between Subspaces

Definition The angle between subspaces S and T of a vector space V is the minimum angle between nonzero vectors in those subspaces [113] and has the following cosine:

$$\cos \theta_1 = \max_{s \in S - \{0\}, t \in T - \{0\}} \frac{|\langle s, t \rangle|}{\|s\|_2 \|t\|_2} = \max_{s \in S, t \in T} |\langle s, t \rangle| \text{ such that } \|s\|_2 = \|t\|_2 = 1. \quad (6.31)$$

If either subspace is just the trivial subspace $\{0\}$ (a single point), then the angle is undefined. In other words, this definition applies to subspaces having nonzero dimensions.

If S and T denote orthonormal bases for S and T , then one can show [113] that $\cos \theta_1 = \|S' T\|_2$ and $\sin \theta_1 = \|S'_\perp T\|_2$, where S'_\perp denotes an orthonormal basis for S^\perp .

If $S \cap T = \{0\}$, then there is a stronger Cauchy–Schwarz inequality [114]:

$$|\langle s, t \rangle| \leq \gamma \|s\|_2 \|t\|_2 \quad \forall s \in S, t \in T, \text{ where } 0 \leq \gamma < 1 \text{ depends on } S \text{ and } T.$$

The angle θ_1 in (6.31) is the first of the sequence of principal angles between two ♦♦ subspaces. One can generalize to examine the angle between flats.

6.4 Matrix Norms and Operator Norms

Also important are matrix norms and operator norms; these functions quantify the “size” of the elements of a matrix in different ways.

Definition [4, p. 59] A matrix norm on the vector space of matrices $\mathbb{F}^{M \times N}$ is a function $\|\cdot\|$ from $\mathbb{F}^{M \times N}$ to $[0, \infty)$ that satisfies the following properties $\forall A, B \in \mathbb{F}^{M \times N}$:

$$\|A\| \geq 0 \quad (\text{nonnegative}), \quad (6.32a)$$

$$\|A\| = 0 \text{ iff } A = \mathbf{0}_{M \times N} \quad (\text{positive}), \quad (6.32b)$$

$$\|\alpha A\| = |\alpha| \|A\| \quad \forall \alpha \in \mathbb{F} \quad (\text{absolute homogeneity}), \quad (6.32c)$$

$$\|A + B\| \leq \|A\| + \|B\| \quad (\text{triangle inequality}). \quad (6.32d)$$

The nonnegativity property again follows from the others.

Because the set of all $M \times N$ matrices $\mathbb{F}^{M \times N}$ is itself a vector space, matrix norms are simply vector norms for that space. So at first, having a new definition might seem to have modest utility. However, many, but not all, matrix norms are submultiplicative, also called consistent [4, p. 61], meaning that they satisfy the following inequality:

$$\|AB\| \leq \|A\| \|B\| \quad \forall A, B \text{ conformable}. \quad (6.33)$$

This book uses the notation $\|\cdot\|$ to distinguish such matrix norms from the ordinary matrix norms $\|\cdot\|$ on the vector space $\mathbb{F}^{M \times N}$ that need not satisfy this extra condition. Some books include (6.33) as part of the definition, for example, [115, p. 290].

6.4.1 Examples of Matrix Norms

The max norm on $\mathbb{F}^{M \times N}$ is the elementwise maximum:

$$\|A\|_{\max} \triangleq \max_{i,j} |a_{ij}| = \|\text{vec}(A)\|_\infty. \quad (6.34)$$

This norm is akin to the infinity norm for vectors of length MN . One can compute it in JULIA using `norm(A, Inf)` after invoking `using LinearAlgebra`. Equivalently, one may use `norm(vec(A), Inf)` because the matrix shape is unimportant for this norm. (However, this differs from `opnorm(A, Inf)`, which computes $\|A\|_\infty$ described below.)

 The max norm is a matrix norm on the vector space $\mathbb{F}^{M \times N}$ but it does not satisfy the submultiplicative condition (6.33) so it is of limited use, though Section 12.2.1 uses it to analyze roundoff error. Most of the norms of interest in DS–ML–SP are submultiplicative, so such matrix norms are our primary focus hereafter.

Fact 6.2 Any matrix norm, including the max norm, can be rescaled to be submultiplicative (Problem 6.18).

The Frobenius norm (otherwise known as the Hilbert–Schmidt norm or matrix Euclidian norm) is defined on $\mathbb{F}^{M \times N}$ by

$$\|A\|_F \triangleq \sqrt{\sum_{m=1}^M \sum_{n=1}^N |a_{mn}|^2} = \sqrt{\text{trace}\{A'A\}} = \sqrt{\text{trace}\{AA'\}} = \|\text{vec}(A)\|_2, \quad (6.35)$$

and is also called the Schur norm and Schatten 2-norm. It is a very easy norm to compute. The equalities related to trace are an exercise (Problem 2.27). Practical implementation: `norm(A, 2)`, `norm(A)`, `norm(vec(A), 2)`, or `norm(vec(A))`. Again, the shape of A is unimportant for this norm.

To relate the Frobenius norm of a matrix to its singular values, use a compact SVD:

$$\begin{aligned} \|A\|_F &= \sqrt{\text{trace}\{AA'\}} = \sqrt{\text{trace}\{U_r \Sigma_r V_r' V_r \Sigma_r U_r'\}} = \sqrt{\text{trace}\{\Sigma_r \Sigma_r U_r' U_r\}} \\ &= \sqrt{\text{trace}\{\Sigma_r^2\}} = \sqrt{\sum_{k=1}^r \sigma_k^2} = \|\Sigma_r\|_F = \sqrt{\text{trace}\{\Sigma \Sigma'\}}. \end{aligned} \quad (6.36)$$

 This norm is invariant to unitary transformations because of the trace property (2.63); see (6.75). From a vector perspective, this norm is induced by the Frobenius inner product on $\mathbb{F}^{M \times N}$. As a matrix norm, however, it is not induced by any vector norm on \mathbb{F}^N (see Section 6.4.2) [116], but nevertheless it is compatible with the Euclidean vector norm because

$$\|Ax\|_2 \leq \|A\|_2 \|x\|_2 \leq \|A\|_{\text{F}} \|x\|_2. \quad (6.37)$$

However, this upper bound is not tight in general. (It is tight only for matrices with rank ≤ 1 .) By combining (6.37) with the definition of matrix multiplication, one can easily show that the Frobenius norm is submultiplicative [115, p. 291].

Q6.5 What is the Frobenius norm of the outer product $\|uv'\|_{\text{F}}$ for $u \in \mathbb{F}^M$, $v \in \mathbb{F}^N$?
 A: $\|u\|_2 \|v\|_2$ B: $\sqrt{\|u\|_2 \|v\|_2}$ C: $|u'v|^2$ D: $|u'v|$ E: None of these

6.4.1.1 $\ell_{p,q}$ Norms

For certain DS–ML–SP problems involving group sparsity [117, 118], the following ♦♦ family of $\ell_{p,q}$ matrix norms is useful:

$$\|A\|_{p,q} \triangleq \left(\sum_{n=1}^N (\|A_{:,n}\|_p)^q \right)^{1/q} = \left(\sum_{n=1}^N \left(\sum_{m=1}^M |a_{m,n}|^p \right)^{q/p} \right)^{1/q} \quad (6.38)$$

This family considers an $M \times N$ matrix A as a collection of N columns of length M . A particularly popular special case for group sparsity problems [119] is

$$\|A\|_{2,1} = \sum_{n=1}^N \|A_{:,n}\|_2. \quad (6.39)$$

Note that in general $\|A\|_{p,p} = \|\text{vec}(A)\|_p$, and specifically $\|A\|_{2,2} = \|A\|_{\text{F}}$.

6.4.2 Induced Matrix Norms

The matrix norms introduced previously are perhaps most appropriate for matrices that contain data. Recall from Chapter 2 that we often work with $M \times N$ matrices that represent linear mappings from \mathbb{F}^N to \mathbb{F}^M . For such matrices (otherwise known as operators), induced matrix norms are often especially relevant because they bound the effect of multiplying the matrix with a vector.

Definition Let $\|\cdot\|_\alpha$ denote a vector norm for \mathbb{F}^N and let $\|\cdot\|_\beta$ denote a vector norm for \mathbb{F}^M . The induced matrix norm with respect to those two vector norms is defined for any matrix $A \in \mathbb{F}^{M \times N}$ as

$$\|A\|_{\alpha,\beta} \triangleq \sup_{x: \|x\|_\alpha=1} \|Ax\|_\beta = \sup_{x \neq 0} \frac{\|Ax\|_\beta}{\|x\|_\alpha}. \quad (6.40)$$

We also call this an operator norm because now A acts as a linear operator. We say such a matrix norm $\|\cdot\|$ is induced by the vector norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$.

Fact 6.3 By construction, if $\|\cdot\|_{\alpha, \beta}$ is a matrix norm induced by vector norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$, then this definition ensures the following inequality:

$$\|Ax\|_\beta \leq \|A\|_{\alpha, \beta} \|x\|_\alpha \quad \forall x \in \mathbb{F}^N. \quad (6.41)$$

Due to the definition in (6.40), this is a tight inequality (no smaller constant than $\|A\|_{\alpha, \beta}$ suffices for all x).

Often we use the same vector norm (e.g., the Euclidean norm) for both \mathbb{F}^M and \mathbb{F}^N and then simplify the notation to write $\|Ax\| \leq \|A\| \|x\|$. Here we say the matrix norm $\|\cdot\|$ is induced by the vector norm $\|\cdot\|$.

Importantly, when we use the same vector norm throughout, any induced matrix norm will satisfy the submultiplicative property (6.33) provided the number of columns of A matches the number of rows of B . This fact follows readily from the definition (6.40) and the property (6.41) because

$$\|AB\| = \sup_{x: \|x\|=1} \|ABx\| \leq \sup_{x: \|x\|=1} \|A\| \|Bx\| = \|A\| \|B\|. \quad (6.42)$$

Generalizing this inequality to the case of different vector norms (e.g., $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$) is left as an exercise.

Example 6.12 The most important matrix norms (operator norms) are induced by the vector norm $\|\cdot\|_p$ for $1 \leq p \leq \infty$, that is,

$$\|A\|_p \triangleq \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}. \quad (6.43)$$

The spectral norm $\|\cdot\|_2$, often denoted simply $\|\cdot\|$, is defined on $\mathbb{F}^{M \times N}$ by (6.43) with $p = 2$. This is the matrix norm induced by the Euclidean vector norm in (3.16):

$$\|A\|_2 = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \max \{ \sqrt{\lambda} : \lambda \in \text{eig}\{A'A\} \} = \sigma_1(A). \quad (6.44)$$

The maximum row sum matrix norm is defined on $\mathbb{F}^{M \times N}$ by

$$\|A\|_\infty \triangleq \sup_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \max_{1 \leq i \leq M} \sum_{j=1}^N |a_{ij}|. \quad (6.45)$$

It is induced by the ℓ_∞ vector norm. It differs from the max norm in (6.34). Here, shape matters!

Proof.

$$\begin{aligned} \|A\|_\infty &\triangleq \sup_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty} = \sup_{x \neq 0} \frac{\max_{m=1,\dots,M} |(Ax)_m|}{\|x\|_\infty} = \sup_{x \neq 0} \frac{\max_{m=1,\dots,M} \left| \sum_{n=1}^N a_{mn} x_n \right|}{\|x\|_\infty} \\ &\leq \sup_{x \neq 0} \frac{\max_{m=1,\dots,M} \sum_{n=1}^N |a_{mn}| |x_n|}{\|x\|_\infty} \leq \sup_{x \neq 0} \frac{\max_{m=1,\dots,M} \sum_{n=1}^N |a_{mn}| \|x\|_\infty}{\|x\|_\infty} \end{aligned}$$

$$= \max_{1 \leq m \leq M} \sum_{n=1}^N |a_{mn}|.$$

□

The maximum column sum matrix norm is defined on $\mathbb{F}^{M \times N}$ by

$$\|A\|_1 \triangleq \sup_{x \neq 0} \frac{\|Ax\|_1}{\|x\|_1} = \max_{1 \leq j \leq N} \sum_{i=1}^M |a_{ij}|. \quad (6.46)$$

It is induced by the ℓ_1 vector norm. Note that $\|A\|_1 = \|A'\|_\infty$.

6.4.3 Norms Defined in Terms of Singular Values

Here are three important norms used in modern DS–ML–SP problems.

Definition The nuclear norm, also called the trace norm [4, p. 60], is the sum of the singular values:

$$\|A\|_* \triangleq \sum_{k=1}^{\min(M,N)} \sigma_k = \text{trace}\{(A'A)^{1/2}\} = \text{trace}\{(AA')^{1/2}\}, \quad (6.47)$$

where matrix square root is discussed in Section 9.2.1.

Definition For $1 \leq p \leq \infty$, the Schatten p -norm of an $M \times N$ matrix is defined using the ℓ_p norm of its singular values:

$$\|A\|_{S,p} \triangleq \left(\sum_{k=1}^{\min(M,N)} \sigma_k^p \right)^{1/p} = \left(\sum_{k=1}^{\text{rank}(A)} \sigma_k^p \right)^{1/p}. \quad (6.48)$$

Definition The Ky Fan K -norm is the sum of the first $1 \leq K \leq \min(M,N)$ singular values of a matrix:

$$\|A\|_{\text{Ky Fan},K} \triangleq \sum_{k=1}^K \sigma_k(A). \quad (6.49)$$

For a PCA generalization that uses this latter norm, see [120]. For a Schatten 4-norm used in a data science application, see [121]. For an imaging application that uses $p = 0.9$ (not a norm, but useful still in practice), see [122].

Q6.6 If A is a positive semidefinite matrix, then $\|A\|_* = \text{trace}(A)$.

A: True B: False

6.4.3.1 Relationships Between These Norms

Nuclear norm:

$$\|A\|_* = \|A\|_{S,1} = \|A\|_{\text{Ky Fan}, \min(M,N)}. \quad (6.50)$$

Spectral norm:

$$\|A\|_2 = \sigma_1(A) = \|A\|_{S,\infty} = \|A\|_{\text{Ky Fan}, 1}. \quad (6.51)$$

Frobenius norm:

$$\|A\|_F = \|A\|_{S,2}. \quad (6.52)$$

Explore 6.8 Relate $\|A\|_F$ to a Ky Fan norm and to a nuclear norm involving A .

Explore 6.9 Show that the Schatten p -norm $\|\cdot\|_{S,p}$ of an $N \times N$ matrix is *not* an induced matrix norm for $1 \leq p < \infty$ when $N \geq 2$.

Hint: Consider $A = I_N$.

Explore 6.10 Define a matrix norm that unifies the spectral, nuclear, Frobenius, Schatten, and Ky Fan norms.

6.4.3.2 Matrix Inner Product Inequalities

If A and B are $M \times N$ (possibly complex) matrices with singular values $\{\sigma_k\}$ and $\{\gamma_k\}$ respectively, then von Neumann's trace inequality [123, 124, 125, 126] for the Frobenius inner product (6.20) is

$$\text{real}\{\langle A, B \rangle\} = \text{real}\{\text{trace}\{AB'\}\} \leq \sum_{k=1}^{\min(M,N)} \sigma_k \gamma_k. \quad (6.53)$$

The left term is a matrix inner product, and the sum is a Euclidean inner product between singular values. This inequality is tight and equality holds iff there exists a simultaneous SVD of A and B of the form $A = U\Sigma V'$ and $B = U\Omega V'$ [126]. For a lower bound, see [127].

Multiplying a matrix by $e^{i\phi}$ for any ϕ does not change its singular values, so by taking $\phi = \langle A, B \rangle$ one can slightly strengthen (6.53) to the following:

$$|\langle A, B \rangle| = |\text{trace}\{AB'\}| \leq \sum_{k=1}^{\min(M,N)} \sigma_k \gamma_k. \quad (6.54)$$

Applying the Cauchy–Schwarz inequality (6.26) to the singular value inner product in (6.54) leads to the following Cauchy–Schwarz inequality for the Frobenius inner product already shown in (6.27):

$$|\langle A, B \rangle| \leq \sqrt{\left(\sum_k \sigma_k^2\right) \left(\sum_k \gamma_k^2\right)} = \|A\|_F \|B\|_F. \quad (6.55)$$

Explore 6.11 Is the inequality (6.55) tight?

One can bound von Neumann's trace inequality (6.54) in terms of the nuclear norm and spectral norm. If A and B are both in $\mathbb{F}^{M \times N}$, then, from (6.54),

$$|\text{trace}\{AB'\}| \leq \sum_{k=1}^{\min(M,N)} \sigma_k(A) \sigma_k(B) \leq \sum_{k=1}^{\min(M,N)} \sigma_k(A) \sigma_1(B)$$

$$= \|A\|_* \|B\|_2 = \|A\|_{S,1} \|B\|_{S,\infty}, \quad (6.56)$$

which is a matrix extension of the version of Hölder's inequality in (6.57). We explore this further next.

Using (6.28), the Frobenius inner product (6.20) for matrices in $\mathbb{F}^{M \times N}$ satisfies

$$\text{real}(\langle A, B \rangle) \leq |\langle A, B \rangle| = |\langle \text{vec}(A), \text{vec}(B) \rangle| \leq \|\text{vec}(A)\|_1 \|\text{vec}(B)\|_\infty. \quad (6.57)$$



Caution: In general, $\|\text{vec}(A)\|_1 \neq \|A\|_1$ and $\|\text{vec}(B)\|_\infty \neq \|B\|_\infty$.

Explore 6.12 Prove or disprove $|\langle A, B \rangle| \stackrel{?}{\leq} \|A\|_1 \|B\|_\infty$, a matrix analogue of (6.28).

Applying Hölder's inequality (6.29) to the sum in (6.54) yields the following Hölder inequality for matrices for $1/p + 1/q = 1$:

$$\begin{aligned} |\langle A, B \rangle| &= |\text{trace}\{AB'\}| \leq \sum_{k=1}^{\min(M,N)} \sigma_k(A) \sigma_k(B) \\ &\leq \left(\sum_{k=1}^{\min(M,N)} \sigma_k^p(A) \right)^{1/p} \left(\sum_{k=1}^{\min(M,N)} \sigma_k^q(B) \right)^{1/q} = \|A\|_{S,p} \|B\|_{S,q}. \end{aligned} \quad (6.58)$$

Combining with (6.57), this inequality holds for $p, q \in [1, \infty]$ [128, p. 96].

Another Hölder inequality for matrix products involving the Schatten norm for $1/r = 1/p + 1/q$ [129] is

$$\|AB\|_{S,r} \leq \|A\|_{S,p} \|B\|_{S,q}. \quad (6.59)$$

Explore 6.13 Examine the special case $r = 1$ and $p = q = 2$.

6.4.3.3 Triangle Inequalities

- ◆◆ To prove that a function is indeed a norm, often the only nontrivial part is establishing the triangle inequality. For a (complicated!) proof that the Schatten p -norms are in fact norms, see [128, p. 91]. For the Ky Fan norm (and hence nuclear norm), we have easier proofs based on writing them in variational forms, that is, as the solution to an optimization problem.

The Ky Fan norm has a variational formulation. For $A \in \mathbb{F}^{M \times N}$, pick any $X \in \mathcal{V}_K(\mathbb{F}^M)$ and $Y \in \mathcal{V}_K(\mathbb{F}^N)$, where the Stiefel manifold was defined in (4.45). Using von Neumann's trace inequality (6.54),

$$\begin{aligned} |\text{trace}\{X'AY\}| &= |\text{trace}\{AYX'\}| \leq \sum_{k=1}^{\min(M,N)} \sigma_k(A) \sigma_k(XY') = \sum_{k=1}^K \sigma_k(A) \\ &= \|A\|_{\text{Ky Fan}, K}, \end{aligned}$$

because $XY' = XI_KY'$ is its own compact SVD. Thus, we have the following lower bound for the Ky Fan K -norm:

$$\sup_{X \in \mathcal{V}_K(\mathbb{F}^M), Y \in \mathcal{V}_K(\mathbb{F}^N)} |\text{trace}\{X'AY\}| \leq \|A\|_{\text{Ky Fan}, K}.$$

On the other hand, using the SVD $A = U\Sigma V'$ and choosing $X = U_K \in \mathcal{V}_K(\mathbb{F}^M)$ and $Y = V_K \in \mathcal{V}_K(\mathbb{F}^N)$, we have

$$|\text{trace}\{X'AY\}| = |\text{trace}\{U'_K U \Sigma V' V_K\}| = |\text{trace}\{\Sigma_K\}| = \|A\|_{\text{Ky Fan}, K},$$

so $\sup_{X \in \mathcal{V}_K(\mathbb{F}^M), Y \in \mathcal{V}_K(\mathbb{F}^N)} |\text{trace}\{X'AY\}| \geq \|A\|_{\text{Ky Fan}, K}$. Combining yields the variational expression

$$\|A\|_{\text{Ky Fan}, K} = \sup_{X \in \mathcal{V}_K(\mathbb{F}^M), Y \in \mathcal{V}_K(\mathbb{F}^N)} |\text{trace}\{X'AY\}|. \quad (6.61)$$

Explore 6.14 Use (6.61) to establish the triangle inequality for the Ky Fan K -norm.

Taking $K = \min(M, N)$ as a special case yields the following variational formulation of the nuclear norm:

$$\|A\|_* = \sup_{B \in \mathbb{F}^{M \times N}: \|B\|_2 \leq 1} |\text{trace}\{AB'\}|. \quad (6.63)$$

This form readily helps establish the triangle inequality for the nuclear norm, akin to (6.109).

Another variational formulation for the nuclear norm that is useful for analyzing matrix factorization approaches is [130]:

$$\|X\|_* = \min_{AB'=X} \|A\|_F \|B\|_F = \min_{AB'=X} \frac{1}{2} (\|A\|_F^2 + \|B\|_F^2). \quad (6.64)$$

Explore 6.15 Find A, B pairs that satisfy the two equalities in (6.64).

6.4.3.4 Submultiplicativity

The nuclear, Schatten, and Ky Fan norms are all submultiplicative [128, p. 94]. In particular, this means that $\|BA\|_* \leq \|B\|_* \|A\|_*$. However, there is a tighter inequality:

$$\|BA\|_* \leq \|B\|_2 \|A\|_*. \quad (6.65)$$

Proof. For $A \in \mathbb{F}^{M \times N}$ and $B \in \mathbb{F}^{K \times M}$ with $B \neq \mathbf{0}$, define $X = B/\|B\|_2$ so that $\|X\|_2 = 1$ and apply (6.63) twice:

$$\begin{aligned} \|BA\|_* &= \|B\|_2 \|XA\|_* = \|B\|_2 \sup_{C \in \mathbb{F}^{K \times N}: \|C\|_2 \leq 1} |\text{trace}\{XAC'\}| \\ &= \|B\|_2 \sup_{C \in \mathbb{F}^{K \times N}: \|C\|_2 \leq 1} |\text{trace}\{A(X'C)\}| \\ &\leq \|B\|_2 \sup_{Y \in \mathbb{F}^{M \times N}: \|Y\|_2 \leq 1} |\text{trace}\{AY'\}| = \|B\|_2 \|A\|_*. \end{aligned}$$

because $\|X'C\|_2 \leq 1$ implies that $\{X'C : \|C\|_2 \leq 1\} \subset \{Y \in \mathbb{F}^{M \times N} : \|Y\|_2 \leq 1\}$. The case $B = \mathbf{0}$ is trivial. \square

6.4.4 Practical Implementation

JULIA commands (after invoking `using LinearAlgebra`) for some of these norms are as follows:

| | |
|----------------|--|
| $\ A\ _1$ | <code>opnorm(A, 1)</code> |
| $\ A\ _2$ | <code>opnorm(A, 2)</code> or just <code>opnorm(A)</code> |
| $\ A\ _\infty$ | <code>opnorm(A, Inf)</code> |
| $\ A\ _{\max}$ | <code>norm(A, Inf)</code> (be careful!) |
| $\ A\ _*$ | <code>sum(svdvals(A))</code> or <code>sum(svd(A).S)</code> |

A preview of practical use of such norms: $\|A\|_2$ is relatively expensive to compute because it requires an SVD of A , whereas $\|A\|_1$ and $\|A\|_\infty$ are easy to compute; see Section 6.4.7. $\|A\|_*$ is useful as a convex relaxation of $\text{rank}(A)$; see Section 7.5.

Q6.7 For $A = [1 \ -3]'[1 \ 1 \ 1]$, what is `norm(A, Inf)` ?

- A: 2 B: 3 C: 6 D: 9 E: None of these

Q6.8 For $A = [1 \ -3]'[1 \ 1 \ 1]$, what is `opnorm(A, Inf)` ?

- A: 2 B: 3 C: 6 D: 9 E: None of these

Q6.9 For $A = [1 \ -3]'[1 \ 1 \ 1]$, what is `opnorm(A, 2)` ?

- A: 2 B: 3 C: 6 D: 9 E: None of these

6.4.5 Properties of Matrix Norms

All matrix norms are also equivalent (to within constants that depend on the matrix dimensions). See [4, p. 61] for inequalities relating various matrix norms.

Example 6.13 (Problem 6.12.) $A \in \mathbb{F}^{M \times N} \implies \|A\|_1 \leq \sqrt{M}\|A\|_2$.

Example 6.14 To relate the spectral norm and nuclear norm for a matrix A having rank r :

$$\begin{aligned}\|A\|_* &= \sum_{k=1}^r \sigma_k \leq \sum_{k=1}^r \sigma_1 = r\sigma_1 = r\|A\|_2 \\ \|A\|_2 &= \sigma_1 \leq \sum_{k=1}^r \sigma_k = \|A\|_*.\end{aligned}$$

Combining yields the following inequalities:

$$\frac{1}{r}\|A\|_* \leq \|A\|_2 \leq \|A\|_* \leq r\|A\|_2. \quad (6.66)$$

To express it in a way that depends on the norm only (not r , which is a property of a specific matrix):

$$\frac{1}{\min(M, N)} \|A\|_* \leq \|A\|_2 \leq \|A\|_* \leq \min(M, N) \|A\|_2. \quad (6.67)$$

A related concept for the Schatten norm is monotonicity; for $1 \leq p \leq q \leq \infty$,

$$\|A\|_{S,\infty} \leq \|A\|_{S,q} \leq \|A\|_{S,p} \leq \|A\|_{S,1}. \quad (6.68)$$

Taking $p = q = 2$ yields

$$\|A\|_2 \leq \|A\|_F \leq \|A\|_*. \quad (6.69)$$

6.4.5.1 Singular Value Inequalities

If A and B are both $M \times N$ matrices, then the triangle inequality for the spectral norm is

$$\|A + B\|_2 \leq \|A\|_2 + \|B\|_2,$$

or equivalently, in terms of singular values,

$$\sigma_1(A + B) \leq \sigma_1(A) + \sigma_1(B). \quad (6.70)$$

- ◆ The following Weyl's inequality provides a generalization to other singular values (see [131] and [132, Theorem 2]):

$$\begin{aligned} \sigma_{m+n+1}(A + B) &\leq \sigma_{m+1}(A) + \sigma_{n+1}(B), & 0 \leq m, n \leq \min(M, N) - 1, \\ && m + n + 1 \leq \min(M, N). \end{aligned} \quad (6.71)$$

Example 6.15 Considering $m + n + 1 = 3$ we have

$$\begin{aligned} \sigma_3(A + B) &\leq \sigma_3(A) + \sigma_1(B) \quad (m = 2, n = 0), \\ \sigma_3(A + B) &\leq \sigma_2(A) + \sigma_2(B) \quad (m = 1, n = 1), \\ \sigma_3(A + B) &\leq \sigma_1(A) + \sigma_3(B) \quad (m = 0, n = 2), \\ \implies \sigma_3(A + B) &\leq \min(\sigma_3(A) + \sigma_1(B), \sigma_2(A) + \sigma_2(B), \sigma_1(A) + \sigma_3(B)). \end{aligned}$$

Such inequalities are used less often than the triangle inequality, but we do use them in the proofs in Section 7.2.

The Hoffman–Wielandt inequality [133] provides lower and upper bounds relating the Frobenius norm of the difference of two normal matrices to the differences of their eigenvalues. Let A and B denote $N \times N$ normal matrices, and let \mathcal{P}_N denote the permutation group of $\{1, \dots, N\}$. Then

$$\min_{p \in \mathcal{P}_N} \sum_{n=1}^N |\lambda_n(A) - \lambda_{p(n)}(B)|^2 \leq \|A - B\|_F^2 \leq \max_{p \in \mathcal{P}_N} \sum_{n=1}^N |\lambda_n(A) - \lambda_{p(n)}(B)|^2. \quad (6.72)$$

One can generalize this inequality to any matrices $A, B \in \mathbb{F}^{M \times N}$ using the Jordan-Wielandt matrix $\begin{bmatrix} 0 & A \\ A' & 0 \end{bmatrix}$, leading to the singular value inequalities

$$\min_{p \in \mathcal{P}_K} \sum_{k=1}^K |\sigma_k(A) - \sigma_{k(p)}(B)|^2 \leq \|A - B\|_F^2 \leq \max_{p \in \mathcal{P}_n} \sum_{k=1}^K |\sigma_k(A) - \sigma_{p(k)}(B)|^2, \quad (6.73)$$

where $K \triangleq \min(M, N)$.

For matrix products, we know that $\sigma_1(AB) \leq \sigma_1(A)\sigma_1(B)$ because the spectral norm is submultiplicative. More generally, for $1 \leq K \leq \min(L, M, N)$ with $A \in \mathbb{F}^{L \times M}$ and $B \in \mathbb{F}^{M \times N}$ we have [134]:

$$\sum_{k=1}^K \sigma_k(AB) \leq \sum_{k=1}^K \sigma_k(A)\sigma_k(B). \quad (6.74)$$

6.4.5.2 Unitarily Invariant Matrix Norms

Definition A matrix norm $\|\cdot\|$ on $\mathbb{F}^{M \times N}$ is called unitarily invariant iff, for all unitary matrices $U \in \mathbb{F}^{M \times M}$ and $V \in \mathbb{F}^{N \times N}$,

$$\|UAV\| = \|A\| \quad \forall A \in \mathbb{F}^{M \times N}. \quad (6.75)$$

Fact 6.4

The spectral norm $\|A\|_2$ is unitarily invariant.

Any Schatten p -norm $\|A\|_{S,p}$ is unitarily invariant.

Proof (sketch). Unitary matrix rotations do not change singular values. □

The Frobenius norm $\|A\|_F$ is unitarily invariant.

Proof for the Frobenius case:

$$\begin{aligned} \|UAV\|_F^2 &= \text{trace}\{V'A'U'UAV\} = \text{trace}\{V'A'AV\} \\ &= \text{trace}\{VV'A'\} = \text{trace}\{AA'\} = \|A\|_F^2. \end{aligned} \quad \square$$

(We could also prove it using singular values.)

The Frobenius norm has an even more general invariance. If U has M orthonormal columns and Q has N orthonormal rows, then by the same proof $\|UAQ\|_F = \|A\|_F$ for any $A \in \mathbb{F}^{M \times N}$.

Fact 6.5 Every unitarily invariant matrix norm is submultiplicative. (See [128, p. 94] for a complicated proof.)

Fact 6.6 If A and B are positive semidefinite matrices, then, for every unitarily invariant norm [135],

$$2\|AB\| \leq \|A^2 + B^2\| \quad \text{and} \quad 4\|AB\| \leq \|(A + B)^2\|. \quad (6.76)$$

These properties generalize the arithmetic–geometric mean inequality to PSD matrices.

6.4.6 Spectral Radius

Definition For any square matrix, the spectral radius is the maximum absolute eigenvalue:

$$A \in \mathbb{F}^{N \times N} \implies \rho(A) \triangleq \max_i |\lambda_i(A)|. \quad (6.77)$$

- By construction, $|\lambda_i(A)| \leq \rho(A)$ so all eigenvalues lie within a disk in the complex plane of radius $\rho(A)$, hence the name.
- In general, $\rho(A)$ is *not* a matrix norm and $\|Ax\| \not\leq \rho(A)\|x\|$. Consider $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$.
- However, if A is normal, then recall from (3.26) that we can order its eigenvalues in decreasing order of their absolute values, then (3.26) relates its unitary eigendecomposition to an SVD as follows:

$$A = V\Lambda V' = \sum_{n=1}^N \lambda_n v_n v_n' = \sum_{n=1}^N \underbrace{|\lambda_n|}_{\sigma_n} \underbrace{\text{sign}(\lambda_n)}_{u_n} v_n v_n'.$$

Thus, if A is normal (e.g., if $A = A'$) then $\rho(A) = \sigma_1(A) = \|A\|_2$, so $\|Ax\|_2 \leq \rho(A)\|x\|_2$.

Furthermore, A normal $\implies |x'Ax| \leq \rho(A)\|x\|_2^2$ because, applying the Cauchy–Schwarz inequality,

$$|x'Ax| = |\langle Ax, x \rangle| \leq \|Ax\|_2 \|x\|_2 \leq \|A\|_2 \|x\|_2^2 = \rho(A)\|x\|_2^2. \quad (6.78)$$

If $A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ and $x = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ then $\rho(A) = 1$ but $x'Ax = 6 > \|x\|_2^2 = 5$. So A being diagonalizable is insufficient to conclude that $|x'Ax| \leq \rho(A)\|x\|_2^2$.

- If $\|\cdot\|$ is any induced matrix norm on $\mathbb{F}^{N \times N}$ and if $A \in \mathbb{F}^{N \times N}$, then

$$\rho(A) \leq \|A\| \quad \text{and} \quad \rho(A) \leq \|A^k\|^{1/k}, k \in \mathbb{N}. \quad (6.79)$$

Proof. If $Av = \lambda v$ for $v \neq 0$, then $|\lambda|^k \|v\| = \|\lambda^k v\| = \|A^k v\| \leq \|A^k\| \|v\|$. Dividing by $\|v\|$ yields $|\lambda|^k \leq \|A^k\|$. This inequality holds for all eigenvalues, including the one with maximum magnitude. \square

The same proof shows that $\rho(A) \leq \|A\|_F$ because of the compatible property (6.37).

- If $A \in \mathbb{F}^{N \times N}$, then $\lim_{k \rightarrow \infty} A^k = 0$ if and only if $\rho(A) < 1$. This property is particularly important for analyzing the convergence of iterative algorithms, including training recurrent neural networks [136].

♦♦ • For any $A \in \mathbb{F}^{N \times N}$, the spectral radius is an infimum of all induced matrix norms:

$$\rho(A) = \inf\{\|A\| : \|\cdot\| \text{ is an induced matrix norm}\}. \quad (6.80)$$

◆◆ • Gelfand's formula for any induced matrix norm $\|\cdot\|$ for a square matrix A is

$$\rho(A) = \lim_{k \rightarrow \infty} \|A^k\|^{1/k}. \quad (6.81)$$

Q6.10 Which equality (if any) correctly relates a singular value and a spectral radius for any general matrix $A \in \mathbb{F}^{M \times N}$?

- A: $\sigma_1(A) = |\rho(A)|$
- B: $\sigma_1(A) = \rho^2(A)$
- C: $\sigma_1(A) = \rho(A'A)$
- D: $\sigma_1(A) = \sqrt{\rho(A'A)}$
- E: None of these

We have seen that A normal $\implies \rho(A) = \sigma_1(A)$, but the converse is not true.

Example 6.16 Consider $A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sqrt{2} \end{bmatrix}$ for which $\rho(A) = \sigma_1(A) = \sqrt{2}$, but A is not normal.

6.4.7 Practical Step Size for Gradient Descent

For solving an LLS problem of the form $\arg \min_x \frac{1}{2} \|Ax - y\|^2$, the gradient descent (GD) method (Problem 5.5 and (9.2)) $x_{k+1} = x_k - \alpha A'(Ax_k - y)$ converges (for any x_0 ; see Section 9.2.3) iff $\rho(I - \alpha A'A) < 1$, that is, iff the step size α satisfies

$$0 < \alpha < 2/\sigma_1^2(A). \quad (6.82)$$

However, computing the spectral norm $\sigma_1(A) = \|A\|_2$ is expensive for large problems.

One option for avoiding the SVD is to use the bound from (6.69) and Problem 6.12:

$$\sigma_1(A) = \|A\|_2 \leq \|A\|_F.$$

So, choosing $0 < \alpha < 2/\|A\|_F^2$ would always provide a valid step size. This is convenient because it is much easier to compute $\|\cdot\|_F$ than an SVD. However, the upper bound above is often quite loose. If $A = I_N$, then $\|A\|_2 = 1$ but $\|A\|_F = \sqrt{N}$. Using this bound, the step size α would be much smaller than necessary, slowing GD convergence.

A more practical alternative is to use the inequality (6.79) to upper bound on the spectral norm:

$$\sigma_1^2(A) = \sigma_1(A'A) = \|A'A\|_2 = \rho(A'A) \leq \|A'A\|. \quad (6.83)$$

for any induced matrix norm. Because $\sigma_1(A'A) = \sigma_1(AA')$, a similar argument shows that $\sigma_1^2(A) \leq \|AA'\|$. Combining for the choice $\|\cdot\|_1$ yields the upper bound

$$\sigma_1^2(A) \leq \min(\|A'A\|_1, \|AA'\|_1). \quad (6.84)$$

Thus, choosing

$$0 < \alpha < \frac{2}{\min(\|A'A\|_1, \|AA'\|_1)} \quad (6.85)$$

would provide a step size that lies in the proper interval in (6.82).

Q6.11 We chose $\|A'A\|_1$ instead of $\|A'A\|_\infty$ in (6.84). Which norm is bigger?

- A: $\|A'A\|_1$ usually
- B: $\|A'A\|_1$ always
- C: $\|A'A\|_\infty$ usually
- D: $\|A'A\|_\infty$ always
- E: They are the same

If A is an $M \times N$ matrix, then computing $A'A$ would require N^2M multiplies whereas AA' would require M^2N multiplies; both are expensive if M and N are large. In fact, they have the same order $O(MN^2)$ (for $M \geq N$) as computing an SVD of A . To avoid matrix multiplication, we can use the upper bound

$$\sigma_1^2(A) \leq \|A'A\|_1 \leq \|A'\|_1 \|A\|_1 = \|A\|_\infty \|A\|_1,$$

because the matrix 1-norm is submultiplicative. Thus, choosing

$$0 < \alpha < \frac{2}{\|A\|_\infty \|A\|_1} \quad (6.86)$$

is a valid step size that lies in the proper interval in (6.82), and hence ensures GD convergence.

It is much easier to compute $\|A\|_\infty$ and $\|A\|_1$ than $\|A\|_2$.

Q6.12 Approximately how many additions are needed to compute $\|A\|_1$?

- A: M
- B: N
- C: $\min(M, N)$
- D: $\max(M, N)$
- E: MN

A drawback of (6.86) is that it may be a loose upper bound. If A is the $N \times N$ unitary DFT matrix discussed in Section 8.3, then $\|A\|_2 = 1$ but $\|A\|_\infty = \|A\|_1 = \sqrt{N}$. Problem 6.12 shows that $\|A\|_\infty \leq \sqrt{N}\|A\|_2$ for an $M \times N$ matrix A , so this unitary DFT example achieves that worst-case upper bound.

6.5 Convergence of Sequences of Vectors and Matrices

Later chapters discuss iterative optimization algorithms and analyze when sequences produced by such algorithms converge. This is another topic involving vector norms and matrix norms. Recall the following definition for convergence of a sequence of numbers.

Definition We say a sequence of (possibly complex) numbers $\{x_k\}$ converges to a limit x_* iff $|x_k - x_*| \rightarrow 0$ as $k \rightarrow \infty$, where $|\cdot|$ denotes absolute value (or complex magnitude more generally). Specifically,

$$\forall \epsilon > 0, \exists N_\epsilon \in \mathbb{N} \text{ such that } |x_k - x_*| < \epsilon \quad \forall k \geq N_\epsilon. \quad (6.87)$$

We now define convergence of a sequence of vectors or matrices by using a norm to quantify distance, then relating convergence of vectors to that of a sequence of scalars.

Definition We say a sequence of vectors $\{x_k\}$ in a vector space \mathcal{V} converges to a limit $x_* \in \mathcal{V}$ iff $\|x_k - x_*\| \rightarrow 0$ for some norm $\|\cdot\|$ as $k \rightarrow \infty$. Specifically,

$$\forall \epsilon > 0, \exists N_\epsilon \in \mathbb{N} \text{ such that } \|x_k - x_*\| < \epsilon \quad \forall k \geq N_\epsilon. \quad (6.88)$$

Often we write $x_k \rightarrow x_*$ as a shorthand for $\|x_k - x_*\| \rightarrow 0$.

A matrix is simply a point in a vector space of matrices so we use essentially the same definition for convergence of a sequence of matrices.

Definition We say a sequence of matrices $\{X_k\}$ (in a vector space \mathcal{V} of matrices) converges to a limit $X_* \in \mathcal{V}$ iff $\|X_k - X_*\| \rightarrow 0$ for some (matrix) norm $\|\cdot\|$ as $k \rightarrow \infty$. Specifically,

$$\forall \epsilon > 0, \exists N_\epsilon \in \mathbb{N} \text{ such that } \|X_k - X_*\| < \epsilon \quad \forall k \geq N_\epsilon. \quad (6.89)$$

Example 6.17 Consider (for simplicity) the sequence of diagonal matrices $\{D_k\}$ defined by

$$D_k = \begin{bmatrix} 3 + 2^{-k} & 0 \\ 0 & (-1)^k/k^2 \end{bmatrix}.$$

This sequence converges to the limit $D_* = \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}$ because

$$\|D_k - D_*\|_F = \left\| \begin{bmatrix} 2^{-k} & 0 \\ 0 & (-1)^k/k^2 \end{bmatrix} \right\|_F = \sqrt{4^{-k} + 1/k^4} \rightarrow 0.$$

Example 6.18 For a square matrix A , define the partial sum of powers $S_k \triangleq \sum_{j=0}^k A^j$. If $\|A\|_2 < 1$, then one can show that $I - A$ is invertible and the matrix sequence $\{S_k\}$ converges to the Neumann series: $\sum_{j=0}^{\infty} A^j = (I - A)^{-1}$. See [137] for a recent use in neural networks.

6.6 Generalized Inverse of a Matrix

The Moore–Penrose pseudoinverse defined in Section 5.4 is just one (particularly important) type of generalized inverse of a matrix. This section uses the Frobenius norm to characterize the Moore–Penrose pseudoinverse.

Definition A matrix $G \in \mathbb{F}^{N \times M}$ is a generalized inverse of a matrix $A \in \mathbb{F}^{M \times N}$ iff $AGA = A$.

If A has full column rank, then $A'A$ is invertible, so multiplying both sides of $AGA = A$ on the left by $A^+ = (A'A)^{-1}A'$ yields that G is a generalized inverse of such an A iff $GA = I_N$, that is, iff G is a left inverse of A .

Conversely, if A has full row rank, then AA' is invertible and G is a generalized inverse of such an A iff $AG = I_M$, that is, iff G is a right inverse of A .

Considering an SVD $A = U\Sigma V'$, one can verify from the definition that every generalized inverse of A has the form

$$G = V \begin{bmatrix} \Sigma_r^{-1} & S_2 \\ S_3 & S_4 \end{bmatrix} U', \quad (6.90)$$

where the matrices S_2, S_3, S_4 have certain sizes but otherwise have completely arbitrary values. In other words, the (very general!) set of generalized inverses \mathcal{G}_A of an $M \times N$ matrix A is a linear variety in the vector space of $N \times M$ matrices.

Explore 6.16 Determine the sizes of matrices S_2, S_3, S_4 .

There are many ways to choose a specific generalized inverse from the set \mathcal{G}_A .

6.6.1 Minimum Frobenius Norm Generalized Inverse

A simple way is to choose the generalized inverse having the smallest Frobenius norm. This solution turns out to be simply the (Moore–Penrose) pseudoinverse of A :

$$\arg \min_{G \in \mathcal{G}_A} \|G\|_F = A^+. \quad (6.91)$$

Proof. $G \in \mathcal{G}_A \implies G = V \begin{bmatrix} \Sigma_r^{-1} & S_2 \\ S_3 & S_4 \end{bmatrix} U' \implies \|G\|_F = \left\| \begin{bmatrix} \Sigma_r^{-1} & S_2 \\ S_3 & S_4 \end{bmatrix} \right\|_F$ because

the Frobenius norm is unitarily invariant. Because $\left\| \begin{bmatrix} \Sigma_r^{-1} & S_2 \\ S_3 & S_4 \end{bmatrix} \right\|_F^2 = \|\Sigma_r^{-1}\|_F^2 + \|S_2\|_F^2 + \|S_3\|_F^2 + \|S_4\|_F^2$, the minimum Frobenius norm solution arises when each S_i is all zeros. Thus that solution has the form $G = V \begin{bmatrix} \Sigma_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} U' = V\Sigma^+U' = A^+$. \square

In words, the Moore–Penrose pseudoinverse of A is the unique generalized inverse of A with minimal Frobenius norm. See [138] for other choices.

6.7 Procrustes Analysis

One use of matrix norms is quantifying the dissimilarity of two matrices by using a norm of their difference. This section illustrates that use by solving the orthogonal Procrustes problem [139, 140]. This problem provides another practical application of the SVD and the Frobenius norm for matrices.

The goal of the Procrustes problem is to find an orthogonal matrix Q in $\mathbb{R}^{M \times M}$ that makes two other matrices B and A in $\mathbb{R}^{M \times N}$ as similar as possible by “rotating” each of the columns of A :

$$\hat{Q} = \arg \min_{Q: Q'Q=I_M} f(Q), \quad \underbrace{f(Q)}_{\hookrightarrow \text{cost function}} \triangleq \|B - QA\|_F^2. \quad (6.92)$$

- One could use some other norm but the Frobenius is simple and natural here. (Think about why!)
- We put “rotating” in quotes because the condition $Q'Q = I$ ensures that Q has orthonormal columns, but the class of matrices for which $Q'Q = I$ also includes examples like $Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ that are not rotations.
- ◆◆ • There are extensions that require $\det(Q) = 1$ to ensure Q corresponds to a rotation [141].
- This problem is also known as Wahba’s problem [142], after the pioneering statistician Grace Wahba, and is used in satellite attitude determination with GPS receivers.
- See Section 6.7.2 for several generalizations (nonsquare, complex, translation).

One of many motivating applications is performing image registration of two pictures of the same scene acquired with different sensor orientations, using a technique called landmark registration.

Example 6.19 In Fig. 6.4 the goal is to match (by rotation) two sets of landmark coordinates:

$$A = \begin{bmatrix} -59 & -25 & 49 \\ 6 & -33 & 20 \end{bmatrix}, \quad B = \begin{bmatrix} -54.1 & -5.15 & 32.44 \\ -24.3 & -41.08 & 41.82 \end{bmatrix} \approx QA = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} A.$$

Here, $M = 2$ and $N = 3$ (typically $M < N$ in such problems). Here we found the landmarks manually, but there are also automatic methods [143].

To solve this problem, first analyze the cost function:

$$\begin{aligned} f(Q) &= \|B - QA\|_F^2 = \text{trace}\{(B - QA)'(B - QA)\} \\ &= \text{trace}\{B'B - B'QA - A'Q'B + A'Q'QA\} \quad \text{expand via FOIL} \\ &= \text{trace}\{B'B - B'QA - A'Q'B + A'A\} \quad Q'Q = I \\ &= \text{trace}\{B'B\} + \text{trace}\{A'A\} \\ &\quad - \text{trace}\{A'Q'B\} - \text{trace}\{B'QA\} \quad \text{linearity} \end{aligned}$$

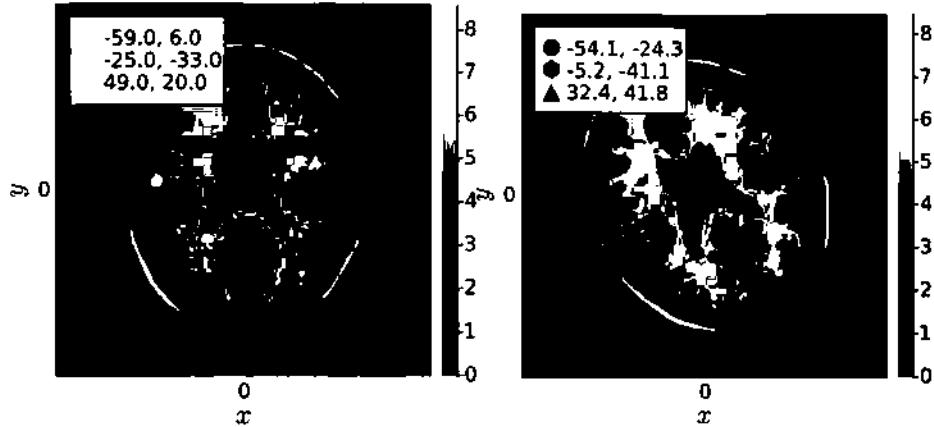


Figure 6.4 Landmarks for image registration.

$$\begin{aligned}
 &= \text{trace}\{B'B\} + \text{trace}\{A'A\} \\
 &\quad - \text{trace}\{A'Q'B\} - \text{trace}\{(A'Q'B)'\} && \text{transpose} \\
 &= \text{trace}\{B'B\} + \text{trace}\{A'A\} - 2\text{trace}\{A'Q'B\} && \text{transpose invar.} \\
 &= \text{trace}\{B'B\} + \text{trace}\{A'A\} - 2\text{trace}\{Q'BA'\} && \text{by (2.63).}
 \end{aligned}$$

So the goal of *minimizing* $f(Q)$ is equivalent to the following *maximization problem*:

$$\hat{Q} = \arg \max_{Q: Q^T Q = I} g(Q), \quad g(Q) \triangleq \text{trace}\{Q'BA'\}. \quad (6.93)$$

Use an SVD (of course!) of the $M \times M$ matrix $C \triangleq BA' = U\Sigma V'$, so

$$\begin{aligned}
 g(Q) &= \text{trace}\{Q'BA'\} = \text{trace}\{Q'U\Sigma V'\} = \text{trace}\{V'Q'U\Sigma\} \\
 &= \text{trace}\{W\Sigma\}, \quad W = W(Q) \triangleq V'Q'U.
 \end{aligned}$$

Using the orthogonality of U , V , and Q , it is clear that the $M \times M$ matrix W is orthogonal (Problem 2.16):

$$W'W = U'QV'V'Q'U = U'QI_MQ'U = U'U = I_M.$$

We must maximize $\text{trace}\{W\Sigma\}$ over W orthogonal, where W depends on Q but Σ does not. Observe:

$$[W\Sigma]_{mm} = w_{mm}\sigma_m \implies \text{trace}\{W\Sigma\} = \sum_{m=1}^M w_{mm}\sigma_m.$$

To proceed, we look for an upper bound for this sum. Because W is an orthogonal matrix, each of its columns have unit norm, that is, $\sum_{m=1}^N |w_{mn}|^2 = 1$ for all m , so $w_{mn} \leq 1$ for all m, n . This inequality yields the following upper bound:

$$\text{trace}\{W\Sigma\} \leq \sum_{m=1}^M \sigma_m = \text{trace}(I\Sigma). \quad (6.94)$$

This upper bound is achieved when $W = I$. Now solve for Q :

$$W = V'Q'U = I \Rightarrow VV'Q'UU' = VU' \Rightarrow Q' = VU' \Rightarrow \hat{Q} = UV'.$$

Fact 6.7 In summary, the solution to the orthogonal Procrustes problem is

$$\hat{Q} = \arg \min_{Q: Q^T Q = I} \|B - QA\|_F^2 = UV^T, \text{ where } C = BA' = U\Sigma V'. \quad (6.95)$$

Problem 3.18 expresses $C = QP$ where P is positive semidefinite using a polar decomposition or polar factorization of the square matrix BA' [4, p. 41].

Explore 6.17 The choice $W = I$ achieves the upper bound in (6.94); does any other orthogonal matrix W also achieve that upper bound?

Q6.13 The solution to the orthogonal Procrustes problem is unique.

One can show that the product UV' is the same for any SVD of BA' , that is, the solution \tilde{Q} is unique, iff BA' has full rank, (i.e., is invertible, or has no zero singular values); see [144]. If in fact $B = \tilde{Q}A$ for some unitary matrix \tilde{Q} , then one can verify, cf. (6.96), that $\tilde{Q} = \tilde{Q}$ and that $\|B - \tilde{Q}A\|_F = 0$.

6.7.1 Sanity Check and Scale Invariance

Suppose \mathbf{B} is exactly a rotated version of the columns of \mathbf{A} , along with an additional spatial scale factor, that is, $\mathbf{B} = \alpha \tilde{\mathbf{Q}}\mathbf{A}$ for some orthogonal matrix $\tilde{\mathbf{Q}}$; equivalently, $\mathbf{A} = \frac{1}{\alpha} \tilde{\mathbf{Q}}' \mathbf{B}$. We now verify that the Procrustes method finds the correct rotation, that is, $\hat{\mathbf{Q}} = \tilde{\mathbf{Q}}$. Let $\mathbf{B} = \tilde{\mathbf{U}}\tilde{\Sigma}\tilde{\mathbf{V}}'$ denote an SVD of \mathbf{B} . Then an SVD of \mathbf{C} is evident by inspection:

$$C = BA' = \frac{1}{\alpha} BB' \tilde{Q} = \frac{1}{\alpha} \underbrace{\tilde{U} \tilde{\Sigma} \tilde{V}'}_{B} \underbrace{\tilde{V} \tilde{\Sigma}' \tilde{U}'}_{B'} \tilde{Q} = \underbrace{\tilde{U}}_U \underbrace{\frac{1}{\alpha} \tilde{\Sigma} \tilde{\Sigma}'}_{\tilde{\Sigma}} \underbrace{\tilde{U}'}_{V'} \tilde{Q}.$$

The Procrustes solution is indeed correct, and invariant to the scale parameter α :

$$\hat{Q} = UV' = (\hat{U})(\hat{V}'\hat{Q}) = \tilde{Q}. \quad (6.96)$$

After finding \hat{Q} , if we also want to estimate the scale then we can solve a linear least-squares problem:

$$\begin{aligned} \arg \min_{\alpha} \|B - \alpha \hat{Q}A\|_F &= \arg \min_{\alpha} \text{trace}\{(B - \alpha \hat{Q}A)(B - \alpha \hat{Q}A)'\} \\ &= \arg \min_{\alpha} \alpha^2 \text{trace}\{AA'\} - 2\alpha \text{trace}\{BA'\hat{Q}\} \\ &= \frac{\text{trace}\{BA'\hat{Q}'\}}{\text{trace}\{AA'\}} = \frac{\text{trace}\{U\Sigma V'VU'\}}{\text{trace}\{AA'\}} = \frac{\sum_{k=1}^r \sigma_k}{\|A\|_F^2}, \quad (6.97) \end{aligned}$$

where $\{\sigma_k\}$ are the singular values of $C = BA'$.

Example 6.20 For determining 2D image rotation, even a single nonzero point in each image suffices! For example, suppose the first point is at $(1, 0)$ and the second point is at (x, y) where $x = 5 \cos \phi$ and $y = 5 \sin \phi$. (This example includes scaling by a factor of five just to illustrate the generality.) Then $A = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $B = \begin{bmatrix} x \\ y \end{bmatrix}$ so

$$C = BA' = \begin{bmatrix} 5 \cos \phi \\ 5 \sin \phi \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos \phi & -q_1 \sin \phi \\ \sin \phi & q_1 \cos \phi \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & q_2 \end{bmatrix}}_{V'} \quad (6.98)$$

for $q_1, q_2 \in \{\pm 1\}$. Here, C is a simple outer product so finding a (full!) SVD by hand was easy. In fact, (6.98) provides four distinct SVDs, corresponding to different signs for u_2 and v_2 . For each of these SVDs, the optimal rotation matrix per (6.95) is

$$\hat{Q} = UV' = \begin{bmatrix} \cos \phi & -q_1 \sin \phi \\ \sin \phi & q_1 \cos \phi \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & q_2 \end{bmatrix} = \begin{bmatrix} \cos \phi & -q \sin \phi \\ \sin \phi & q \cos \phi \end{bmatrix}, \quad (6.99)$$

where $q \triangleq q_1 q_2 \in \{\pm 1\}$. The two Procrustes solutions here (for $q = \pm 1$) both have the correct $\cos \phi$ in the upper left and both exactly satisfy $B = 5Q_A$. So there are two Procrustes solutions that fit the data exactly, one of which (for $q = 1$) corresponds to a rotation matrix, and the other of which (for $q = -1$) has sign flip for the second coordinate. In 2D, any rotation matrix is a unitary matrix, but the converse is not true!

Explore 6.18 Determine what happens to the solution (6.99) when the second point (x, y) is along the ray between the origin and the first point, or antipodal with the first point.

6.7.2 Procrustes Generalizations

This section generalizes the Procrustes problem (6.92) in three ways: we consider complex data, we account for a possible translation, and we allow Q to be nonsquare, meaning that B and A can have different numbers of rows [145]. (They still must have the same number of columns so that B and QA have matched dimensions.) Procrustes was a figure in Greek mythology who “stretched” people to fit an iron bed. An even more general version of the Procrustes problem considers a scaling factor (stretching), like α in Section 6.7.1 (Problem 6.24). A practical application of such generalizations is in landmark-based image registration where one can encounter rotation, translation, and scale factors (due to different pixel sizes); see Fig. 6.5.

Here we assume that $B \in \mathbb{F}^{M \times N}$ but $A \in \mathbb{F}^{K \times N}$, so $Q \in \mathbb{F}^{M \times K}$. We still want Q to have orthonormal columns, so we must have $1 \leq K \leq M$, that is, we want Q to be in the Stiefel manifold $\mathcal{V}_K(\mathbb{F}^M)$ defined in (4.45).

In many practical applications of the Procrustes problem, there can be both rotation and an unknown translation between the two sets of coordinates. Instead of the model $B_{:,n} \approx QA_{:,n}$, a more realistic model is $B_{:,n} \approx QA_{:,n} + d$ where $d \in \mathbb{F}^M$ is an unknown displacement vector. In matrix form,

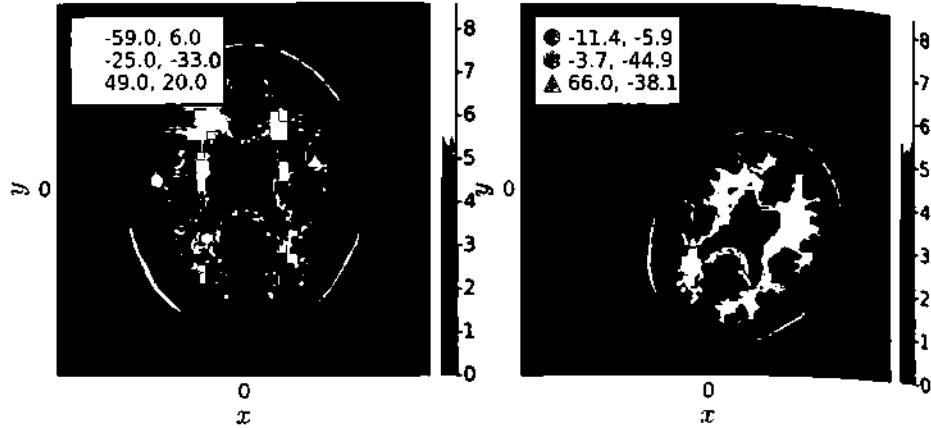


Figure 6.5 Landmark-based image registration where translation, rotation, and scaling are needed.

$$\mathbf{B} \approx \mathbf{Q}\mathbf{A} + \mathbf{d}\mathbf{1}_N' \quad (6.100)$$

Now we must determine both a matrix $\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)$ in the Stiefel manifold and the vector $\mathbf{d} \in \mathbb{F}^M$ by a double minimization using a Frobenius norm:

$$(\hat{\mathbf{Q}}, \hat{\mathbf{d}}) \triangleq \arg \min_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \arg \min_{\mathbf{d} \in \mathbb{F}^M} g(\mathbf{d}, \mathbf{Q}), \quad g(\mathbf{d}, \mathbf{Q}) \triangleq \|\|\mathbf{B} - (\mathbf{Q}\mathbf{A} + \mathbf{d}\mathbf{1}_N')\|_F^2. \quad (6.101)$$

We first focus on the inner minimization over the displacement \mathbf{d} for any given \mathbf{Q} , defining $\mathbf{Z} \triangleq \mathbf{B} - \mathbf{Q}\mathbf{A}$:

$$\begin{aligned} g(\mathbf{d}, \mathbf{Q}) &= \|\|\mathbf{B} - (\mathbf{Q}\mathbf{A} + \mathbf{d}\mathbf{1}_N')\|_F^2 = \text{trace}\{(\mathbf{Z} - \mathbf{d}\mathbf{1}_N')'(\mathbf{Z} - \mathbf{d}\mathbf{1}_N')\} \\ &= \text{trace}\{\mathbf{Z}'\mathbf{Z}\} - \text{trace}\{\mathbf{Z}'\mathbf{d}\mathbf{1}_N'\} - \text{trace}\{\mathbf{1}_N\mathbf{d}'\mathbf{Z}\} + \text{trace}\{\mathbf{1}_N\mathbf{d}'\mathbf{d}\mathbf{1}_N'\} \\ &= \text{trace}\{\mathbf{Z}'\mathbf{Z}\} - \text{trace}\{\mathbf{1}_N'\mathbf{Z}'\mathbf{d}\} - \text{trace}\{\mathbf{d}'\mathbf{Z}\mathbf{1}_N\} + \text{trace}\{\mathbf{d}'\mathbf{d}\mathbf{1}_N'\mathbf{1}_N\} \\ &= \text{trace}\{\mathbf{Z}'\mathbf{Z}\} - \mathbf{1}_N'\mathbf{Z}'\mathbf{d} - \mathbf{d}'\mathbf{Z}\mathbf{1}_N + N\mathbf{d}'\mathbf{d} \\ &= \text{trace}\{\mathbf{Z}'\mathbf{Z}\} - \frac{1}{N}\|\mathbf{Z}\mathbf{1}_N\|_2^2 + N\|\mathbf{d} - \frac{1}{N}\mathbf{Z}\mathbf{1}_N\|_2^2. \end{aligned}$$

It is clear from this expression that the optimal estimate of the displacement \mathbf{d} for any \mathbf{Q} is $\hat{\mathbf{d}}(\mathbf{Q}) = (1/N)\mathbf{Z}\mathbf{1}_N = (1/N)(\mathbf{B} - \mathbf{Q}\mathbf{A})\mathbf{1}_N$. Now, to find the optimal matrix \mathbf{Q} we must solve the outer minimization:

$$\hat{\mathbf{Q}} \triangleq \arg \min_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} f(\mathbf{Q}), \quad f(\mathbf{Q}) \triangleq g(\hat{\mathbf{d}}(\mathbf{Q}), \mathbf{Q}); \quad (6.102)$$

$$\begin{aligned} f(\mathbf{Q}) &= \text{trace}\{(\mathbf{B} - \mathbf{Q}\mathbf{A})'(\mathbf{B} - \mathbf{Q}\mathbf{A})\} - \frac{1}{N}\|(\mathbf{B} - \mathbf{Q}\mathbf{A})\mathbf{1}_N\|_2^2 \\ &\triangleq -2 \text{real}\{\text{trace}\{\mathbf{Q}'\mathbf{B}\mathbf{A}'\}\} + \frac{2}{N} \text{real}\{\mathbf{1}_N'\mathbf{A}'\mathbf{Q}'\mathbf{B}\mathbf{1}_N\} \\ &= -2 \text{real}\{\text{trace}\{\mathbf{Q}'\mathbf{B}\mathbf{A}'\}\} + 2 \text{real}\left\{\text{trace}\left\{\mathbf{Q}'\mathbf{B}\frac{1}{N}\mathbf{1}_N\mathbf{1}_N'\mathbf{A}'\right\}\right\} \end{aligned}$$

$$= -2 \operatorname{real} \left\{ \operatorname{trace} \left[Q' \tilde{C} \right] \right\}, \quad \tilde{C} \triangleq \underbrace{BMA'}_{M \times K}, \quad M \triangleq I - \frac{1}{N} \mathbf{1}_N \mathbf{1}'_N,$$

where $\stackrel{c}{=}$ means “equal to within constant terms that are irrelevant for minimization.”

The matrix M is a “de-meaning” (or “centering”) operator that projects onto the orthogonal complement of $\mathcal{R}(\mathbf{1}_N)$, that is,

$$M = P_{\mathbf{1}_N}^\perp = I - \frac{1}{N} \mathbf{1}_N \mathbf{1}'_N. \quad (6.103)$$

Multiplying M by any vector in \mathbb{F}^N produces a new vector whose mean value is zero, so we say it “de-means” the input vector. For example, $y = Mx$ subtracts the mean of x from each element of x . In code: $y = x . - \operatorname{mean}(x)$.

After finding a (full) SVD $\tilde{C} = \underbrace{U}_{M \times M} \underbrace{\Sigma}_{M \times K} \underbrace{V'}_{K \times K}$, we want:

$$\hat{Q} = \underset{Q \in \mathcal{V}_K(\mathbb{F}^M)}{\arg \max} \operatorname{real} \left\{ \operatorname{trace} \left[Q' \tilde{C} \right] \right\}, \quad (6.104)$$

where, using the Frobenius inner product inequality (6.57) and defining $W \triangleq U' Q V \in \mathcal{V}_K(\mathbb{F}^M)$, we have:

$$\begin{aligned} \operatorname{real} \left\{ \operatorname{trace} \left[Q' \tilde{C} \right] \right\} &= \operatorname{real} \left\{ \operatorname{trace} \left[Q' U \Sigma V' \right] \right\} = \operatorname{real} \left\{ \operatorname{trace} \left[W' \Sigma \right] \right\}, \\ &= \operatorname{real} \left\{ \langle \Sigma, W \rangle \right\} \leq | \langle \Sigma, W \rangle | \leq \| \operatorname{vec}(\Sigma) \|_1 \| \operatorname{vec}(W) \|_\infty \\ &= \| \Sigma \|_* \| \operatorname{vec}(W) \|_\infty. \end{aligned}$$

Since $\Sigma = \begin{bmatrix} \Sigma_K \\ \mathbf{0}_{(M-K) \times K} \end{bmatrix}$ is rectangular diagonal, the matrix $W = \begin{bmatrix} I_K \\ \mathbf{0}_{(M-K) \times K} \end{bmatrix} \in \mathcal{V}_K(\mathbb{F}^M)$ achieves the upper bound. Solving for Q yields

$$\hat{Q} = UWV' = U \begin{bmatrix} I_K \\ \mathbf{0}_{(M-K) \times K} \end{bmatrix} V' = U_K V',$$

where U_K denotes the first K columns of the $M \times M$ matrix U . In summary, the optimal Q is

$$\hat{Q} = U_K V', \text{ where } \tilde{C} \triangleq BMA' = U \Sigma V'. \quad (6.105)$$

The de-meaning matrix M is a symmetric idempotent matrix so $M = MM'$ and we can rewrite \tilde{C} above as $\tilde{C} = (BM)(AM)' = \tilde{B}\tilde{A}'$ where $\tilde{A} \triangleq AM$, $\tilde{B} \triangleq BM$ are versions of A and B where each column has its mean subtracted out.

In words, to find the optimal rotation matrix when there is possible translation, we first de-mean each column of A and B , and then compute the usual SVD of $\tilde{B}\tilde{A}'$ and use the left and right bases via $\hat{Q} = U_K V'$.

6.7.3 Subspace/Span Comparisons

Another application of the orthogonal Procrustes problem is quantifying the “alignment” between two subspace bases. Suppose \mathbf{B}_1 and \mathbf{B}_2 are $M \times N$ matrices that we think span the same (or similar) subspace in \mathbb{F}^M . In general it does not make sense to use $d(\mathbf{B}_1, \mathbf{B}_2) = \|\mathbf{B}_1 - \mathbf{B}_2\|_F$ as a measure of dissimilarity because we could have $\mathcal{R}(\mathbf{B}_1) = \mathcal{R}(\mathbf{B}_2)$ even if \mathbf{B}_1 and \mathbf{B}_2 are themselves different matrices, for example if $\mathbf{B}_1 = -\mathbf{B}_2$.

A more useful measure of dissimilarity involves first rotating the basis for one subspace to be as similar to the other as possible, and then examining the difference. When \mathbf{B}_1 and \mathbf{B}_2 are both in the same Stiefel manifold, a reasonable measure is

$$d(\mathbf{B}_1, \mathbf{B}_2) \triangleq \min_{Q \in \mathcal{V}_N(\mathbb{F}^N)} \|\mathbf{B}_1 - \mathbf{B}_2 Q'\|_F = \min_{Q \in \mathcal{V}_N(\mathbb{F}^N)} \|\mathbf{B}'_1 - Q\mathbf{B}'_2\|_F. \quad (6.106)$$

The best Q is $\hat{Q} = \mathbf{U}\mathbf{V}'$ where $\mathbf{B}'_1\mathbf{B}_2 = \mathbf{U}\Sigma\mathbf{V}'$, so the simplified dissimilarity measure is

$$d(\mathbf{B}_1, \mathbf{B}_2) = \|\mathbf{B}_1 - \mathbf{B}_2\mathbf{V}\mathbf{U}'\|_F.$$

If the \mathbf{B} matrices are not in the Stiefel manifold, then one should include scale factors akin to (6.97).

Example 6.21 If $\mathbf{B}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$ and $\mathbf{B}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 0 & 0 \end{bmatrix}$ then $\mathbf{B}'_1\mathbf{B}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \mathbf{U}\mathbf{I}_2\mathbf{I}_2 = \hat{Q}$ and $d(\mathbf{B}_1, \mathbf{B}_2) = 0$, as expected because both bases span the xy -plane in \mathbb{R}^3 .

6.7.4 Weighted Procrustes Problems

Some alternatives to (6.92) involve weighted norms (distances) with positive semidefinite weighting matrices (see [146, 147, 148, 149] and [140, Chapter 8]), for example, using cost functions of the form

$$\begin{aligned} \hat{Q} &= \arg \min_{Q: Q'Q=I} f_i(Q), & (6.107) \\ f_1(Q) &\triangleq \|(B - QA)W\|_F \quad \text{for } W \in \mathbb{R}^{N \times N}, \\ f_2(Q) &\triangleq \|W(B - QA)\|_F \quad \text{for } W \in \mathbb{R}^{M \times M}, \\ f_3(Q) &\triangleq \|W \odot (B - QA)\|_F \quad \text{for } W \in \mathbb{R}^{M \times N}. \end{aligned}$$

◆◆ Solving f_1 is easy (Problem 6.27), but f_2 and f_3 require iterative methods in general.

6.7.5 Practical Implementation

-  Demo 6.2 illustrates that the solution to the Procrustes problem requires just a couple of JULIA statements. The key ingredient is simply the `svd` command.

6.8 Summary

- We use vector norms and matrix norms to measure sizes and distances.
- Some matrix norms are essentially just vector norms in terms of $\text{vec}(\mathbf{A})$, some matrix norms satisfy the important submultiplicative property, and operator norms are induced by vector norms.
- Many of the matrix norms can be expressed in terms of singular values, and those are unitarily invariant.
- Classical methods (like linear LS) use 2-norms, but many modern methods use other norms. One vector norm of recent interest is the ordered weighted ℓ_1 (OWL) norm [150].
- We assess the convergence of a sequence of vectors or matrices using norms.
- The spectral radius is a related quantity for square matrices, where

$$\sigma_1(\mathbf{A}) = \sqrt{\sigma_1(\mathbf{A}'\mathbf{A})} = \sqrt{\rho(\mathbf{A}'\mathbf{A})}.$$

- The Moore–Penrose pseudoinverse is the generalized inverse having minimum Frobenius norm.
- As one application, the orthogonal Procrustes problem has an SVD-based solution:

$$\hat{\mathbf{Q}} = \underset{\mathbf{Q}: \mathbf{Q}'\mathbf{Q}=\mathbf{I}_M}{\arg \min} \|\mathbf{B} - \mathbf{Q}\mathbf{A}\|_F^2 = \mathbf{U}\mathbf{V}', \quad \mathbf{C} = \mathbf{B}\mathbf{A}' = \mathbf{U}\Sigma\mathbf{V}'.$$

- The solution is invariant to scaling factors, that is, multiplying $\mathbf{Q}\mathbf{A}$ by α .
- Unknown displacement (translation) simply requires de-meaning \mathbf{A} and \mathbf{B} before doing an SVD.
- A displacement estimate (if needed) is $(1/N)(\mathbf{B} - \hat{\mathbf{Q}}\mathbf{A})\mathbf{1}_N$ (Problem 6.24).
- Deriving the solution to this problem used *many* of the tools discussed so far: Frobenius norm, matrix trace and its properties, SVD, matrix/vector algebra.

Explore 6.19 Suppose \mathbf{A} and \mathbf{B} are both real $1 \times N$ vectors (each with mean 0 for simplicity). How can we interpret the orthogonal Procrustes solution in this case geometrically?

Hint: What is the SVD of $\mathbf{B}\mathbf{A}'$ here?

Explore 6.20 What if $\mathbf{B} = e^{i\varphi} \mathbf{A}$?

Solutions to Explorations

Explore 6.1 For any x , take $y = -x$ and then apply the triangle inequality and the homogeneity property:

$$0 = \|0\| = \|x - x\| = \|x + (-x)\| \leq \|x\| + \|(-x)\| = \|x\| + |-1|\|x\| = 2\|x\|.$$

Explore 6.2

It is not a norm, because $\text{TV}(\mathbf{1}_N) = 0$, so positivity does not hold. However, homogeneity and nonnegativity are easily seen. Also, we can write $\text{TV}(x) = \|Cx\|_1$, where C is the $(N - 1) \times N$ matrix shown in (4.42). Thus, $\text{TV}(x + y) = \|C(x + y)\|_1 \leq \|Cx\|_1 + \|Cy\|_1 = \text{TV}(x) + \text{TV}(y)$, verifying the triangle inequality. Thus, TV is a seminorm.

Explore 6.3 No. Simply take $N = 1$ and then $f(x) \triangleq \|x\|_p = |x_1|$.

Explore 6.4 Same answer as Q6.3.

Explore 6.5 No. If $U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$ and $x = e_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ then $\|x\|_1 = 1$, but $\|Ux\|_1 = \sqrt{2}$.

Explore 6.6 Pairs of random variables that are uncorrelated, that is, where $E[XY] = 0$.

Explore 6.7 $\cos \theta = |\langle J, xx' \rangle| / (\|J\|_{\text{F}} \|xx'\|_{\text{F}}) = \text{trace}\{xx'\} / (\sqrt{N} \|x\|_2^2) = 1/\sqrt{N}$.

Explore 6.8 $\|A\|_{\text{F}}^2 = \|A'A\|_{\text{Ky Fan, min}(M,N)} = \|A'A\|_{\infty}$.

Explore 6.9 Consider $\|I_N\|_{S,p} = N^{1/p} \neq 1 = \sup_{x \in \mathbb{R}^N : x \neq 0} (\|I_N x\| / \|x\|)$, no matter what vector norm $\|\cdot\|$ is used.

Explore 6.10 For $1 \leq p \leq \infty$ and $1 \leq K \leq \min(M, N)$, use the ℓ_p norm of its first K singular values:

$$\|A\|_{K,p} = \left(\sum_{k=1}^K \sigma_k^p \right)^{1/p}.$$

Of course, some work is needed to prove the triangle inequality for this norm.

Explore 6.11 Yes, it holds if $B = \alpha A$, as usual for Cauchy-Schwarz inequalities.

Explore 6.12 For $A = B = I_2$, we have $\langle A, B \rangle = \|A\|_{\text{F}}^2 = 2 > \|A\|_1 \|A\|_{\infty} = 1$.

Explore 6.13 Applying (6.59) leads to the inequality

$$\|AB\|_* = \|AB\|_{S,1} \leq \|A\|_{S,2} \|B\|_{S,2} = \|A\|_F \|B\|_F. \quad (6.108)$$

This is a different inequality than submultiplicativity and (6.65).

Explore 6.14

$$\begin{aligned} \|A + B\|_{Ky\ Fan, K} &= \sup_{X \in \mathcal{V}_K(\mathbb{F}^M), Y \in \mathcal{V}_K(\mathbb{F}^N)} |\text{trace}\{X'(A + B)Y\}| \\ &\leq \sup_{X \in \mathcal{V}_K(\mathbb{F}^M), Y \in \mathcal{V}_K(\mathbb{F}^N)} |\text{trace}\{X'AY\}| \\ &\quad + \sup_{X \in \mathcal{V}_K(\mathbb{F}^M), Y \in \mathcal{V}_K(\mathbb{F}^N)} |\text{trace}\{X'BY\}| \\ &= \|A\|_{Ky\ Fan, K} + \|B\|_{Ky\ Fan, K}. \end{aligned} \quad (6.109)$$

Explore 6.15 Let $X = U_r \Sigma_r V_r'$. Then $A = U_r \Sigma_r^{1/2}$ and $B = V_r \Sigma_r^{1/2}$ satisfy $AB' = X$ and $\|A\|_F = \|B\|_F = \sqrt{\sum_k \sigma_k}$, so both equalities in (6.64) hold.

Explore 6.16 $S_2 \in \mathbb{F}^{r \times (N-r)}$, $S_3 \in \mathbb{F}^{(M-r) \times r}$, $S_4 \in \mathbb{F}^{(M-r) \times (N-r)}$.

Explore 6.17 No.

Explore 6.18 Along the ray means $(x,y) = (x,0)$ for $x \geq 0$, so $\phi = 0$ and $\hat{Q} = \begin{bmatrix} 1 & 0 \\ 0 & \pm 1 \end{bmatrix}$. Antipodal means $(x,y) = (-1,0)$, so $\phi = \pi$ and $\hat{Q} = \begin{bmatrix} -1 & 0 \\ 0 & \pm 1 \end{bmatrix}$.

Explore 6.19 If $A = x'$ and $B = y'$ where $x, y \in \mathbb{R}^N$, then

$$BA' = y'x = \underbrace{\text{sgn}(y'x)}_U \underbrace{|y'x|}_{\sigma_1} \underbrace{1}_V$$

so $Q = UV' = \text{sgn}(y'x) = \pm 1$. Here, the “rotation” is just possibly negating the sign to match in 1D.

Explore 6.20 $BA' = B(e^{-i\phi} B)' = e^{i\phi} V_{BB'} \Lambda_{BB'} V_{BB}' \implies Q = UV' = e^{i\phi} V_{BB'} V_{BB}' = e^{i\phi} I$.

Problems

Problem 6.1 Show that the weighted Euclidean norm $\|\mathbf{x}\|_W$ is a valid vector norm iff W is a positive definite matrix. Assume W is (Hermitian) symmetric.

Problem 6.2 Show that for any vector norm on \mathbb{F}^N , the ball of radius $r \geq 0$ with respect to that norm, $\mathcal{B}_r \triangleq \{\mathbf{x} \in \mathbb{F}^N : \|\mathbf{x}\| \leq r\}$, is a convex set.

Problem 6.3

- (a) Show that if $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_n(\mathbf{x})$ are all convex functions of \mathbf{x} , then the following are also convex functions:
 - (i) $\sum_{i=1}^n f_i(\mathbf{x})$
 - (ii) $\sum_{i=1}^n w_i f_i(\mathbf{x})$ for scalars $w_i \geq 0$. Provide an example wherein w_i being negative breaks the convexity.
 - (iii) $\max \{f_1, f_2, \dots, f_n\}$
- (b) Which of the following cost functions are convex functions of \mathbf{x} ? Assume that \mathbf{A} and \mathbf{D} are matrices and \mathbf{b} is a vector with compatible dimensions. Justify your answer; using the results from a previous part may help.
 - (i) $\|\mathbf{Ax} - \mathbf{b}\|_2^2 + \|\mathbf{Dx}\|_2^2$
 - (ii) $\|\mathbf{Ax} - \mathbf{b}\|_2^2 + \|\mathbf{Dx}\|_1$
 - (iii) $\|\mathbf{Ax} - \mathbf{b}\|_2^2 - \|\mathbf{Dx}\|_2^2$
 - (iv) $\|\mathbf{Ax} - \mathbf{b}\|_2^2 - \|\mathbf{Dx}\|_1$

Problem 6.4 Define $f: \mathbb{F}^N \times \mathbb{F}^N \mapsto \mathbb{F}$ by $f(\mathbf{x}, \mathbf{y}) = \mathbf{y}'\mathbf{A}\mathbf{x}$. Show that f is an inner product iff $\mathbf{A} \succ 0$, that is, \mathbf{A} is a positive definite matrix.

Problem 6.5 Prove or disprove: All inner products on \mathbb{F}^N have the weighted bilinear form $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{y}'\mathbf{A}\mathbf{x}$ for some positive definite matrix \mathbf{A} .

Problem 6.6 Verify the four defining inner product properties on p. 203 for the Frobenius inner product (6.20).

Problem 6.7 Verify that an induced norm of the form (6.21) satisfies the conditions for a norm in (6.2).

Problem 6.8 Consider two vector spaces \mathcal{V}_α and \mathcal{V}_β with corresponding norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$, respectively. A function $f: \mathcal{V}_\alpha \mapsto \mathcal{V}_\beta$ is called an \mathcal{L} -Lipschitz function with respect to those norms if, for every \mathbf{x}, \mathbf{y} in the domain of f , there exists a real constant $\mathcal{L} > 0$ such that $\|f(\mathbf{x}) - f(\mathbf{y})\|_\beta \leq \mathcal{L}\|\mathbf{x} - \mathbf{y}\|_\alpha$. The constant \mathcal{L} is called a Lipschitz constant of f and the smallest such constant is called the best Lipschitz constant. See (9.29).

We say that f is Lipschitz continuous with Lipschitz constant \mathcal{L} . In general, the norm on the left-hand side of the inequality can differ from the norm on the right-hand side. When the norm $\|\cdot\|$ is not specified, it is assumed that f is a Lipschitz function with respect to the Euclidean norm.

- (a) Show that $f(\mathbf{x}) = \nabla g(\mathbf{x}) = \mathbf{A}^\top(\mathbf{Ax} - \mathbf{b})$ is a Lipschitz function of \mathbf{x} when \mathbf{A} is an $M \times N$ matrix. Express its best Lipschitz constant concisely.

Note that $f(\mathbf{x})$ is the gradient of the least-squares cost function (5.3).

- (b) Show that the largest singular value of a matrix is a 1-Lipschitz function (of its MN entries) with respect to the spectral norm for \mathcal{V}_α .
Hint: Use the reverse triangle inequality.
- (c) Optional. Repeat (b) but show it with respect to the Frobenius norm.
Hint: Use Problem 3.13.

Functions with Lipschitz continuous gradients are important because if the function has a Lipschitz constant \mathcal{L} , then, for any $x, z \in \mathbb{R}^N$,

$$f(x) \leq f(z) + \langle \nabla f(z), x - z \rangle + \frac{\mathcal{L}}{2} \|x - z\|_2^2.$$

This inequality is very helpful in analyzing iterative algorithms, such as gradient descent, because substituting $x = x_{k+1}$ and $z = x_k$ yields a bound on the value of the cost function after k iterations that can be used to select the step size to ensure that the value of the cost function will decrease after every iteration (unless we are already at a minimizer where the gradient is zero).

Problem 6.9 Consider the regularized LS cost function (5.44).

- (a) Determine the gradient of this cost function.
- (b) Determine an easily computed upper bound on the Lipschitz constant (see Problem 6.8) of the gradient of this cost function.
 Your final expression must not involve any singular values, because the SVD is too expensive to compute for large-scale problems. Your bound should be one that is reasonably tight (see below); if your bound is too loose (too large), then using its reciprocal as the step size in gradient descent (or related algorithms) would lead to undesirably slow convergence.
- (c) Determine (analytically) the numerical value of your upper bound for the case where $A = \begin{bmatrix} I_5 \\ 1_5 1'_5 \end{bmatrix}$ and $\beta = 2$.
- (d) Determine (analytically) the exact value of the Lipschitz constant of the gradient of the cost function for the A in the previous part. If your bound is more than 10% larger than the exact Lipschitz constant for this case then, you have not found a good enough bound and you should return to part (b) and improve your answer.

Problem 6.10 Let $A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 0 \\ 2 & 3 \\ 0 & 0 \end{bmatrix}$. Determine the cosine of the angle between subspaces $\mathcal{R}(A)$ and $\mathcal{R}(B)$.

Problem 6.11 Consider the vector space of $M \times N$ matrices $\mathbb{C}^{M \times N}$, with the *real* field of scalars, and define the function $f(X, Y) = \text{real}\{\text{trace}[XY']\}$. Show that f is an inner product on this vector space, or provide a counterexample.

Problem 6.12 Show that the following matrix norm inequalities hold when A has rank r . For any part you may use previous parts if helpful.

- (a) $\|A\|_2 \leq \|A\|_F$
Hint: Work with singular values for (a)–(d).

(b) $\|A\|_F \leq \sqrt{r}\|A\|_2$

(c) $\|A\|_F \leq \|A\|_*$

Hint: Start with $\|A\|_F^2$.

(d) Optional. $\|A\|_* \leq \sqrt{r}\|A\|_F$

Hint: Consider using the convexity of $f(x) = x^2$, as in Jensen's inequality.

(e) Optional. $\|A\|_\infty \leq \sqrt{n}\|A\|_2$

Hint: First show that $x \in \mathbb{F}^n \implies (1/\sqrt{n})\|x\|_2 \leq \|x\|_\infty \leq \|x\|_2$.

(f) $\|A\|_2 \leq \sqrt{m}\|A\|_\infty$

Hint: Use the previous hint.

(g) Optional. $(1/\sqrt{m})\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1$

Hint: Use the preceding inequalities.

These inequalities can provide upper and lower bounds for the largest singular value via simple operations on the elements of the matrix. One application where this is useful is in determining the step size for gradient descent algorithms; there, the largest singular value (which equals $\|A\|_2$) has a pivotal role.

Optional challenge. For each of the above inequalities, find a small and simple (but nonzero) matrix A that shows that the inequality is tight, that is, achieved for some A . All of the inequalities hold as equalities when $A = [1]$, so to make this more interesting, your matrix must be bigger than a 1×1 matrix.

Problem 6.13

- (a) Let A_1, A_2, \dots, A_K and B denote $M \times N$ matrices. Determine a solution to the “matrix fitting” least-squares optimization problem:

$$\hat{x} = \arg \min_{x=(x_1, \dots, x_K) \in \mathbb{F}^K} \|B - \sum_{k=1}^K x_k A_k\|_F.$$

- (b) Specify conditions on the given matrices that ensure \hat{x} is unique.
 (c) Write (by hand) a simple JULIA expression for computing \hat{x} given B and the A_k matrices when $K = 3$.
 (d) Optional challenge. How would you handle the general K case in JULIA?

Problem 6.14 Let A be an $M \times N$ matrix with full column rank, and let B be an $M \times K$ matrix.

- (a) Find a concise expression for the solution to $\hat{X} = \arg \min_{X \in \mathbb{F}^{N \times K}} \|AX - B\|_F$.
 (b) Write (by hand) a short JULIA expression for computing \hat{X} efficiently given A and B in the case that B is very wide, that is, $M \approx N \ll K$.

Problem 6.15 Express $\text{trace}((AA')^2)$ concisely in terms of a Schatten p -norm.

Problem 6.16

- (a) Show that $\|[A \quad B]\|_2^2 \leq \|A\|_2^2 + \|B\|_2^2$.
 (b) We would like to verify that this is a “tight” upper bound. Simply showing equality for arbitrary A and B is uninteresting because $A = B = \mathbf{0}$ is a trivial example.

To define tightness more meaningfully, consider $\mathcal{B}_b \triangleq \{B \in \mathbb{F}^{M \times K} : \|B\|_2 = b\}$. Show that for any $A \in \mathbb{F}^{M \times N}$, and any $b \geq 0$, and any $M, N, K \in \mathbb{N}$, there exists $B \in \mathcal{B}_b$ such that $\|[A \quad B]\|_2^2 = \|A\|_2^2 + \|B\|_2^2$. Show existence constructively, that is, show how to define B in terms of A .

- (c) Find a tight upper bound on the spectral norm of the matrix $\begin{bmatrix} A \\ B \end{bmatrix}$ in terms of the spectral norms of A and B .

Problem 6.17 Prove that the squared Frobenius norm $f(X) = \|X\|_F^2$ is a strictly convex function.

Problem 6.18 Not every matrix norm is submultiplicative, but every matrix norm can be *rescaled* to be submultiplicative.

- (a) Show that the max norm $\|A\|_{\max} = \|\text{vec}(A)\|_\infty$ is not submultiplicative.
 (b) Define a matrix norm as a scaled version of the max norm that is submultiplicative.

Problem 6.19 This problem explores extensions of unitary invariance of norms. In this problem, let $A \in \mathbb{F}^{M \times N}$ and assume that X is a $K \times M$ matrix with orthonormal columns and Y is an $N \times L$ matrix with orthonormal rows. (Do not assume X or Y are square.)

- (a) Prove that $\|XAY\|_F = \|A\|_F$.
 This is a kind of generalization of unitary invariance. (Maybe it should be called “orthonormal invariance” or “semi-unitary invariance?”)
 (b) For any matrix B with $J \in \mathbb{N}$ rows, define the function $f(B) = J\|B\|_*$. Show that $f(B)$ is a unitarily invariant norm.
 (c) Prove or disprove: $f(XAY) = f(A)$.
 (d) Consider the following “unified” matrix norm defined in Section 6.4.3: $\|A\|_{K,p} \triangleq \left(\sum_{k=1}^K \sigma_k^p\right)^{1/p}$, where $K \in \mathbb{N}$ and $1 \leq p < \infty$. It is easy to verify that this “unified” norm is unitarily invariant. Prove or disprove: $\|A\|_{K,p} = \|XAY\|_{K,p}$.

Problem 6.20 A previous problem showed that if A is an $M \times N$ matrix, then $\|A\|_2 \leq \sqrt{N}\|A\|_1$.

- (a) Find a simple (but nonzero) 1×2 or 2×1 matrix where this inequality is equality. (When an inequality is achievable, we call it a tight bound.)
 (b) This bound is not tight for normal matrices. If A is normal and $N \times N$, then $\|A\|_2 \leq c\|A\|_1$ for a constant c that is much smaller than N in general. Determine (and prove) the best value for c .
Hint: Think about spectral radius.
 (c) Give a nonzero 2×2 matrix where the inequality in (b) is an equality, so it is also tight.

Problem 6.21 Prove or disprove. If A and B are $N \times N$ orthogonal projection matrices, then the spectral radius of their average, $\frac{1}{2}A + \frac{1}{2}B$, is at most 1.

Hint: Do not “invent” an incorrect triangle inequality.

Problem 6.22 Let A and B be two matrices of the same size. Because the nuclear norm is a matrix norm, we know (by the triangle inequality) that $\|A + B\|_* \leq \|A\|_* + \|B\|_*$. This problem provides a sufficient condition for equality.

Show that if $AB' = 0$ and $A'B = 0$ then $\|A + B\|_* = \|A\|_* + \|B\|_*$.

Hint: Construct a valid compact SVD for $A + B$ from compact SVDs $A = U_r \Sigma_r V_r'$ and $B = X_s \Omega_s Y_s'$ where $r = \text{rank}(A)$ and $s = \text{rank}(B)$, using the given properties. Also think about an upper bound for $\text{rank}(A) + \text{rank}(B)$.

Problem 6.23 Prove or disprove (with the simplest and smallest possible concrete counterexamples) the following statements about the generalized inverse of a matrix. (See Section 6.6.)

- (a) Every matrix $A \in \mathbb{F}^{M \times N}$ with rank $r > 0$ and singular values $\sigma_1, \sigma_2, \dots$ has a generalized inverse matrix G whose spectral norm is $1/\sigma_r$.
- (b) Every matrix $A \in \mathbb{F}^{M \times N}$ with rank $0 < r < \min(M, N)$ and singular values $\sigma_1, \sigma_2, \dots$ for which $\sigma_r = 1$ has a generalized inverse matrix G whose spectral norm is 7.

Problem 6.24 This problem provides the analytical background for aligning shapes using the Procrustes method. Suppose $A, B \in \mathbb{R}^{d \times n}$ and that we would like to rotate, translate, and scale A to best align with B . One way to express this mathematically is by computing the matrix $\hat{A} \in \mathbb{R}^{d \times n}$ defined as

$$\hat{A} = \alpha Q (A - \lambda \mathbf{1}_n') + \mu \mathbf{1}_n', \quad (6.110)$$

where Q is a $d \times d$ orthogonal matrix (rotation or generalization thereof), $\lambda, \mu \in \mathbb{R}^d$ are translation vectors, and $\alpha \in \mathbb{R}$ is a scalar. Geometrically, (6.110) corresponds to translating each column of A by $-\lambda$, “rotating” by Q , scaling by α , and then translating again by μ . However, optimizing both λ and μ would be redundant, because we can replace $\lambda \rightarrow \lambda + \Delta$ and $\mu \rightarrow \mu + \alpha Q \Delta$ for any $\Delta \in \mathbb{R}^d$ without changing \hat{A} . Thus, without loss of generality, we choose

$$\lambda \triangleq \mu_A \triangleq \frac{1}{n} A \mathbf{1}_n = \frac{1}{n} \sum_{k=1}^n A_{[:, k]},$$

the centroid of (the columns of) A .

Then optimize the remaining parameters by solving

$$(\alpha_*, \mu_*, Q_*) = \underset{\alpha \geq 0, \mu \in \mathbb{R}^d, Q: Q'Q = I_d}{\arg \min} \|B - \alpha Q (A - \mu \mathbf{1}_n') - \mu \mathbf{1}_n'\|_F. \quad (6.111)$$

Show that the solution is

$$\mu_* = \mu_B, \quad Q_* = UV', \quad \alpha_* = \frac{\text{trace}\{B_0 A'_0 Q'_*\}}{\text{trace}\{A'_0 A'_0\}},$$

where $\mu_B \triangleq (1/n)B \mathbf{1}_n$ is the centroid of (the columns of) B , and $U \Sigma V'$ is an SVD of $B_0 A'_0$, where we have defined the “de-meansed” matrices $B_0 \triangleq B - \mu_B \mathbf{1}_n'$ and $A_0 \triangleq A - \mu_A \mathbf{1}_n'$.

Hint: First show that $\mu = \mu_*$ minimizes (6.111) for any fixed α and Q (it is a least-squares problem). Then show that $Q = Q_*$ minimizes (6.111) for any fixed α when $\mu = \mu_*$. Finally, show that $\alpha = \alpha_*$ minimizes (6.111) when $\mu = \mu_*$ and $Q = Q_*$.

With $\mu = \mu_*$ and fixed α , (6.111) is a Procrustes problem. Along the way, it will be useful to remember that Z and αZ have the same sets of singular vectors for any matrix Z when $\alpha > 0$.

Problem 6.25 Let $X \in \mathbb{F}^{M \times N}$ for $1 \leq N \leq M$, and let $\mathcal{V}_N(\mathbb{F}^M)$ denote the Stiefel manifold of $M \times N$ matrices having orthonormal columns. Describe how to compute the projection of X onto $\mathcal{V}_N(\mathbb{F}^M)$.

Problem 6.26 For $1 \leq N \leq M$, define the spectral-norm unit ball as

$$\mathcal{B} = \{A \in \mathbb{F}^{M \times N} : \|A\|_2 \leq 1\}.$$

Now define \mathcal{X} to be the subset of matrices in \mathcal{B} having maximal Frobenius norm:

$$\mathcal{X} = \{X \in \mathcal{B} : \|X\|_F \geq \|Y\|_F \forall Y \in \mathcal{B}\}.$$

Show, or disprove with a counterexample, that $\mathcal{X} = \mathcal{V}^{M \times N}$, where $\mathcal{V}^{M \times N}$ denotes the Stiefel manifold of $M \times N$ matrices.

Problem 6.27 Consider the following weighted version of the Procrustes problem:

$$\hat{Q} = \arg \min_{Q: Q^T Q = I} \sum_{n=1}^N w_n \|y_n - Qx_n\|_2^2.$$

Describe a method for computing \hat{Q} .

7 Low-Rank Approximation and Multidimensional Scaling

7.1 Introduction

In many applications, dimensionality reduction is important. Uses of dimensionality reduction include visualization, removing noise, and decreasing compute and memory requirements, such as in image compression. This chapter focuses on low-rank approximation of a matrix. There are theoretical models for why big matrices should be approximately low rank [151]. Low-rank approximations are also used to compress large neural network models to reduce computation and storage [152].

The chapter begins with the classic approach to approximating a matrix by a low-rank matrix, using a nonconvex formulation that has a remarkably simple SVD solution. Section 7.3 applies this approach to the source localization application via the multidimensional scaling (MDS) method. Section 7.5 turns to convex formulations of low-rank approximation based on proximal operators from Section 7.4 that involve singular value shrinkage. Section 7.6 discusses methods for choosing the rank of the approximation, and describes the optimal shrinkage method called OptShrink. Section 7.7 discusses related dimensionality reduction methods including (linear) autoencoders and principal component analysis (PCA). Section 7.8 applies the methods to learning low-dimensionality subspaces from training data for subspace-based classification problems. Finally, Section 7.9 extends the method to streaming applications with time-varying data.

7.2 Low-Rank Approximation via Frobenius Norm

We are given a matrix $A \in \mathbb{F}^{M \times N}$ (often large), having rank $r \leq \min(M, N)$. To perform dimensionality reduction, we want to approximate A by another matrix \hat{A}_K having rank $K \leq r$. Often we pick $K \ll r$. To find the “best” \hat{A}_K we must define how closely \hat{A}_K approximates A . The simplest metric is the Frobenius norm of the difference. This criterion leads to the following low-rank approximation problem:

$$\begin{aligned}\hat{A}_K &\triangleq \arg \min_{B \in \mathcal{L}_K^{M \times N}} \underbrace{\|B - A\|_F}_{\hookrightarrow \text{Frobenius norm}} = \mathcal{P}_{\mathcal{L}_K^{M \times N}}(A), \\ \mathcal{L}_K^{M \times N} &\triangleq \left\{ B \in \mathbb{F}^{M \times N} : \text{rank}(B) \leq K \right\}. \end{aligned}\tag{7.1}$$

This is a nonconvex optimization problem because the set $\mathcal{L}_K^{M \times N}$ of rank- K matrices is not convex.

Example 7.1 To see why $\text{rank}(\cdot)$ is a nonconvex function, and $\mathcal{L}_K^{M \times N}$ is a nonconvex set, consider $A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$, $\alpha \in (0, 1) \implies \text{rank}(\alpha A + (1 - \alpha)B) = \text{rank}\left(\begin{bmatrix} \alpha & 0 \\ 0 & 1 - \alpha \end{bmatrix}\right) = 2 \not\leq \alpha \text{rank}(A) + (1 - \alpha)\text{rank}(B) = \alpha \cdot 1 + (1 - \alpha) \cdot 1 = 1$.

Remarkably, despite this nonconvexity, there is a simple solution for \hat{A}_K based on any SVD of A [153].

7.2.1 Eckart–Young–Mirsky Theorem

Fact 7.1 (Eckart–Young–Mirsky theorem [153, 154]). Earlier by Schmidt [155] in infinite-dimensional spaces.) The best rank (at most) K approximation uses the first (largest) K singular components of A :

$$\hat{A}_K = \sum_{k=1}^K \sigma_k u_k v'_k = U_K \Sigma_K V'_K, \quad \text{where } A = U \Sigma V' = \sum_{k=1}^{\min(M,N)} \sigma_k u_k v'_k. \quad (7.2)$$

The approximation error depends on the singular values of the remaining (discarded) terms:

$$\|\hat{A}_K - A\|_F = \sqrt{\sum_{k=K+1}^{\text{rank}(A)} \sigma_k^2} \leq \|B - A\|_F \quad \forall B \in \mathcal{L}_K^{M \times N}. \quad (7.3)$$

- Clearly, the approximation error will be small if the remaining singular values are small compared to the first K singular values.
- Once again, we see that the SVD is key tool. Here, the SVD is used both to construct the approximation and to quantify the approximation error.
- The original matrix A has MN values. The approximation \hat{A}_K is formed from $K(M + N + 1)$ values. This saves memory if $K \ll \min(M, N)$.

Q7.1 What is the rank of $\sum_{k=1}^K \sigma_k u_k v'_k$ when $1 \leq K \leq r = \text{rank}(A)$?

A: Always 1 B: Always r C: Usually r D: Always K E: Usually K

Proof. The approximation error expression is simple:

$$\begin{aligned} A - \hat{A}_K &= \sum_{k=1}^r \sigma_k u_k v'_k - \sum_{k=1}^K \sigma_k u_k v'_k = \sum_{k=K+1}^r \sigma_k u_k v'_k \\ \implies \|\hat{A}_K - A\|_F^2 &= \left\| \sum_{k=K+1}^r \sigma_k u_k v'_k \right\|_F^2 = \sum_{k=K+1}^r \sigma_k^2. \end{aligned}$$

For any $B \in \mathcal{L}_K^{M \times N}$, $\sigma_{K+1}(B) = 0$ because B has rank at most K . Now apply Weyl's inequality for singular values (6.71) with $m = i - 1$ and $n = K$ to see that

$$\sigma_{K+i}(A) = \sigma_{K+i}(A - B + B) \leq \sigma_i(A - B) + \sigma_{K+1}(B) = \sigma_i(A - B) \quad (7.4)$$

for $i = 1, \dots, \min(M, N) - K$. Thus, B provides no better approximation than \hat{A}_K :

$$\begin{aligned} \|A - B\|_F^2 &= \sum_{i=1}^{\min(M, N)} \sigma_i^2(A - B) \geq \sum_{i=1}^{\min(M, N) - K} \sigma_i^2(A - B) \\ &\geq \sum_{i=1}^{\min(M, N) - K} \sigma_{K+i}^2(A) = \sum_{k=K+1}^{\min(M, N)} \sigma_k^2(A) = \|A - \hat{A}_K\|_F^2. \quad \square \quad (7.5) \end{aligned}$$

In words: the best rank (at most) K approximation to a matrix is given by its truncated SVD (to K terms).

Explore 7.1 Prove that $\hat{A}_K = P_{U_K} A$.

7.2.2 Subspace Approximation Perspective

Often we arrange a collection of N data points $x_1, \dots, x_N \in \mathbb{F}^M$ into a matrix $X = [x_1 \ \cdots \ x_N] \in \mathbb{F}^{M \times N}$. From (7.2), the best rank- K approximation to this matrix is $X \approx \hat{X}_K = U_K \Sigma_K V'_K$, based on the SVD $X = U \Sigma V'$. Now look at this approximation from the perspective of an individual data point, the n th column of X :

$$\begin{aligned} x_n = X_{:,n} = X e_n &\approx \hat{x}_n \triangleq \hat{X}_K e_n = U_K \Sigma_K V'_K e_n \\ &= \sum_{k=1}^K (\sigma_k v_{nk}^*) u_k \in \mathcal{R}(U_K). \quad (7.6) \end{aligned}$$

In words, data point x_n has an approximation that lies in the K -dimensional subspace $\mathcal{R}(U_K)$. Geometrically, we often think of the data points as being in a cloud, and here we are approximating that cloud by a K -dimensional hyperplane.

7.2.3 Implementation

For implementation, one must choose $K \leq r = \text{rank}(A) \leq \min(M, N)$. The approximation error is given by the 2-norm of the excluded singular values ($\sigma_{K+1}, \dots, \sigma_r$), so it is useful to plot all the singular values versus k and look for a "knee in the curve." Low-rank approximation is related to factor analysis, and a scree plot shows the singular values [156].

Example 7.2 Figure 7.1 shows scree plots for the MRI data in Fig. 7.3 and for a synthetic example for $r = 3$ with noise. Hopefully your scree plot has a roughly linear section corresponding to the noise, shown as red squares in Fig. 7.1.

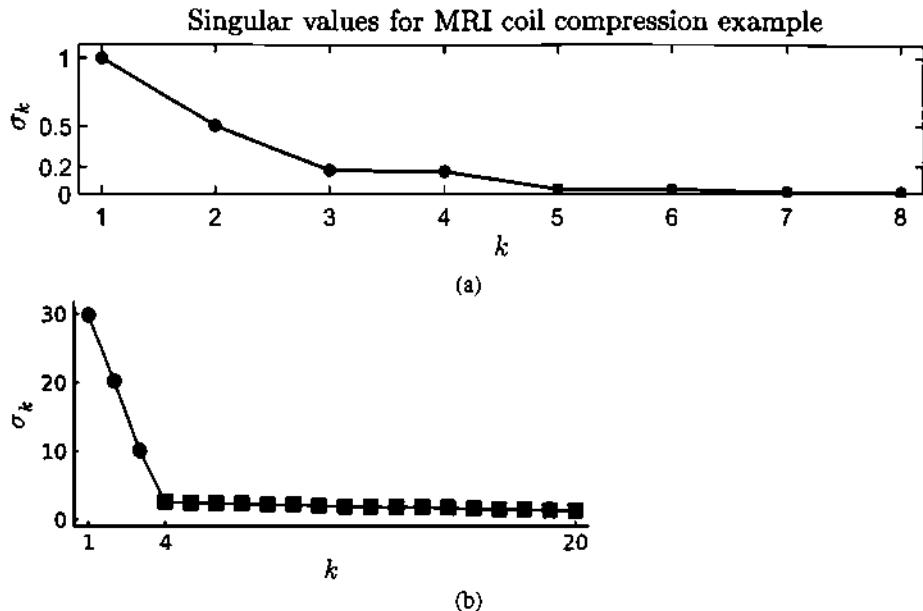


Figure 7.1 Scree plots from MRI data (a) and synthetic data (b).

For moderate-sized matrices, low-rank approximation needs just a few lines of JULIA code. One approach uses the sum of outer products solution (7.2) literally:

```
</> 7.1 using LinearAlgebra: svd
    (U, s, V) = svd(A)
    B = +([U[:,k] * s[k] * V[:,k]' for k in 1:K]...)
```

Alternatively, one can use matrix multiplication corresponding to the compact SVD form in (7.2):

```
</> 7.2 using LinearAlgebra: svd, Diagonal
    (U, s, V) = svd(A) # default full=false saves memory
    B = U[:,1:K] * Diagonal(s[1:K]) * V[:,1:K]'
```

Explore 7.2 Why use small s instead of capital S above?

In both versions, the final B is \hat{A}_K . Quality code would also include a bounds check that $1 \leq K \leq \min(M, N)$.

Yet another option is to compute only the first K SVD components using `svds`:

```
</> 7.3 using LinearAlgebra: Diagonal
using Arpack: svds
tmp = svds(A, nsv=K, ritzvec=true)[1] # special SVD data structure
(U, s, Vt) = (tmp.U, tmp.S, tmp.Vt) # extract needed components
B = U * Diagonal(s) * Vt
```

Explore 7.3 Why use Vt above?

7.2.4 Choosing Rank via Permutation

Looking for a jump in the scree plot seems subjective. A more quantitative approach uses a permutation method [157]. Consider a matrix that is the sum of a low-rank component and IID random noise. If we permute (randomly) each of the columns (or rows? or all?) of that matrix, the noise part will remain the same (statistically) but the low-rank structure will be destroyed. The singular values of this shuffled matrix will basically correspond to noise. The singular values of the original data that are larger than those of the shuffled data might be considered to be the “significant” ones and can help guide rank choice. See [157] for theoretical analysis. One might need to permute multiple times and average to get a reliable baseline. Another approach uses random sign flips [158]. See also [159]. Here is JULIA code for such permutations.

```
</> 7.4  using Random: shuffle
Ys = mapslices(shuffle, Y, dims=1) # shuffle each column of Y
Ys = reshape(shuffle(vec(Y)), size(Y)) # or, shuffle all
```

Example 7.3 Figure 7.2 illustrates a case where the matrix $Y = X + Z$, where X is a matrix with rank $r = 2$ and Z is a Gaussian noise matrix.

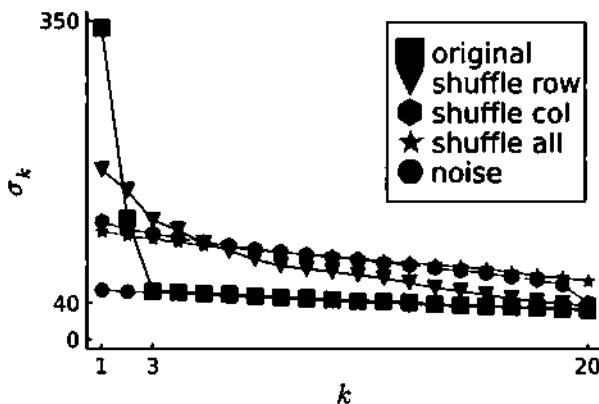


Figure 7.2 This plot uses synthetic data that is rank 2 plus noise. The two largest singular values stand out above the shuffled case.

Example 7.4 Modern MRI scanners use multiple receive coils to collect more image information at the same time. Processing many sets of 3D images can require undesirably large computation times, and typically the data recorded by multiple coils is close to being linearly dependent. A technique called coil compression reduces data size [160, 161], essentially by making a low-rank approximation to the data and then just storing and processing the K singular vectors. Figure 7.3 shows MRI brain images recorded simultaneously by eight coils around the head. The scree plot in Fig. 7.1 suggests that most of the information is in the first four components. We reshape the data into an ($M = 256^2 \times N = 8$) matrix, use an SVD to find the



Figure 7.3 (a) MRI brain images from eight different receive coils. (b) Virtual coil images based on low-rank approximation with $K = 4$.

rank-4 approximation, and reshape the first $K = 4$ vectors $\mathbf{u}_1, \dots, \mathbf{u}_4$ into 256×256 “virtual coil” images for display in Fig. 7.3. (In practice this operation is done on the raw data before making images.)

7.2.5 Nonuniqueness of SVD and Low-Rank Approximation

 As noted previously, “the” SVD of a matrix A is not unique. If two singular values are the same, then one can swap the corresponding columns of U and V (or perform more general rotations) and have two equally valid SVD expressions. When we write $A = \sum_{k=1}^{\min(M,N)} \sigma_k \mathbf{u}_k \mathbf{v}'_k$, the order of the terms in the sum is unique if $\sigma_1 > \sigma_2 > \dots > \sigma_{\min(M,N)}$, but if any of the singular values are identical (including two or more zero singular values) then the ordering is not unique, and one can swap corresponding columns of U and V .

Example 7.5 Here are two different SVDs for a diagonal matrix with

$7 = \sigma_1 > \sigma_2 = \sigma_3 = 5$:

$$\begin{bmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Example 7.6 shows that there are two distinct rank $K = 2$ approximations to this A .

Even if all the singular values are distinct, “the” SVD is still not unique because we can multiply both \mathbf{u}_k and \mathbf{v}_k by $e^{i\phi}$ and get a “different” U and V . However, the low-rank approximation is still the same in this case because $\sigma_k(e^{i\phi} \mathbf{u}_k)(e^{i\phi} \mathbf{v}_k)' = \sigma_k \mathbf{u}_k \mathbf{v}'_k$. We have not focused on this issue too much yet because often uniqueness is relatively unimportant.

Now consider the question of whether the rank- K approximation of A is unique. This is simpler to answer.

- If $\sigma_K > \sigma_{K+1}$ then one can extend the proof in (7.5) to show that the rank- K approximation is unique.
- If $\sigma_K = \sigma_{K+1}$ then the rank- K approximation is *not unique* because we could swap the K th and $(K + 1)$ th terms.

Example 7.6 Here are two distinct rank-1 approximations to a simple diagonal matrix:

$$A \triangleq \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix};$$

$$\begin{aligned} A \approx B_1 &\triangleq \sigma_1 u_1 v_1' = 5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix} \\ &\approx B_2 \triangleq \sigma_1 \bar{u}_1 \bar{v}_1' = 5 \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 5 \end{bmatrix}. \end{aligned}$$

Q7.2 What is the approximation error $\|A - B_1\|_F$ in Example 7.6?

- A: 0 B: $\sqrt{5}$ C: 5 D: 10 E: 5^2

In practice, any rank- K approximation (for suitable K) is often equally useful, so one infrequently worries about nonuniqueness. Furthermore, in practical finite-precision computing, it is dubious to test whether two floating-point values σ_K and σ_{K+1} are “exactly” equal. Avoid code like `if (a == b)` with floating-point numbers! For analysis of the numerical stability of low-rank matrix approximation, including consideration of the “singular value gap” $\sigma_K - \sigma_{K+1}$, see [162].

7.2.6 One-Dimensional Example

Example 7.7 Here we consider a 2×9 matrix where each column is an (x_n, y_n)

coordinate for $n = 1, \dots, N = 9$, that is, $A = \begin{bmatrix} x_1 & \cdots & x_9 \\ y_1 & & y_9 \end{bmatrix}$.

Explore 7.4 Based on the blue dots in Fig. 7.4, what is $\text{rank}(A)$?

Suppose we want to perform dimensionality reduction of the 2D data by reducing it to a 1D line. One option for processing this data is to perform a rank-1 approximation, leading to dashed black line in Fig. 7.4. The black points are the nearest points in the subspace for each of the data points. An alternative is to perform a linear least-squares fit assuming $y_n \approx \alpha x_n$. This leads to the green line with slope $\hat{\alpha}$ in Fig. 7.4.

 Demo 7.1 provides details.

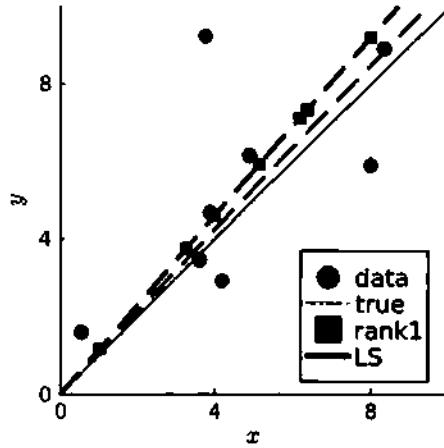


Figure 7.4 Least-squares versus rank-1 approximation.

It might seem surprising that these two methods lead to different fits. The rank-1 approximation is

$$\arg \min_{B \in \mathcal{L}_1^{M \times N}} \left\| \begin{bmatrix} x_1 & \cdots & x_N \\ y_1 & \cdots & y_N \end{bmatrix} - B \right\|_F^2 = \arg \min_{B \in \mathcal{L}_1^{M \times N}} \sum_{n=1}^N \left\| \begin{bmatrix} x_n \\ y_n \end{bmatrix} - B_{:,n} \right\|_2^2.$$

The linear least-squares estimator is

$$\arg \min_{\alpha} \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \alpha \right\|_2^2 = \arg \min_{\alpha} \sum_{n=1}^N |y_n - \alpha x_n|^2.$$

These two methods measure approximation error differently, as illustrated in Fig. 7.5.

7.2.7 Generalization to Other Norms

Fact 7.2 (Eckart–Young–Mirsky theorem; see [154] and [163] for a proof.) For any unitarily invariant matrix norm $\|\cdot\|_{U1}$, the low-rank approximation problem has the same solution using the first (largest) K singular components of $A = U\Sigma V' = \sum_{k=1}^r \sigma_k u_k v'_k$:

$$\hat{A}_K \triangleq \underbrace{\arg \min_{B \in \mathcal{L}_K^{M \times N}} \|B - A\|_{U1}}_{\hookrightarrow \text{unitarily invariant norm}} = \sum_{k=1}^K \sigma_k u_k v'_k. \quad (7.7)$$

Although the proof in (7.5) was for the Frobenius norm, remarkably the same result holds (with a different proof) for other unitarily invariant norms such as the spectral norm.

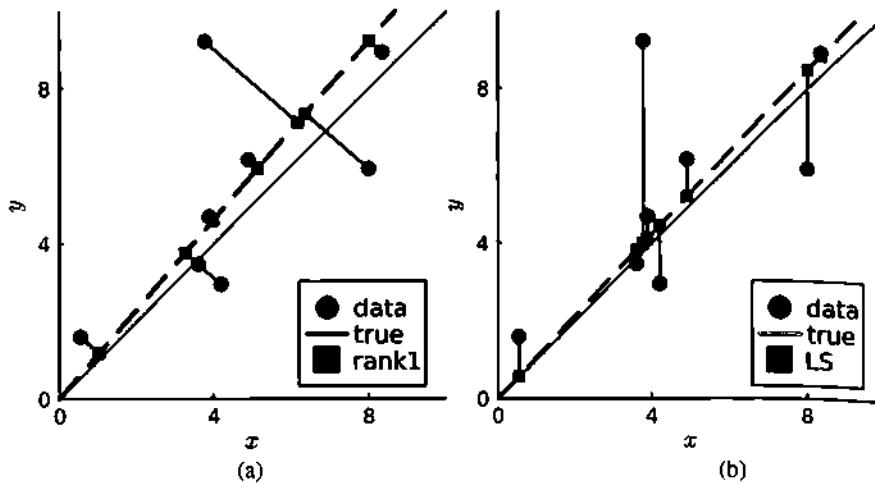


Figure 7.5 Error measures for rank-1 (a) and LS (b) approximations.

- ◆ To be robust to data outliers, Section 6.2.5 showed that using $\|\cdot\|_p$ for $1 \leq p \ll 2$ is preferable to using $\|\cdot\|_2$ for regression problems. Similarly, to perform robust low-rank approximation, we can replace the Frobenius norm, which is equivalent to $\|\text{vec}(B - A)\|_2$, with a cost function like $\|\text{vec}(B - A)\|_p$ [164, 165]. Usually such alternatives are *not* unitarily invariant, so (7.7) is inapplicable. They require different solution methods (typically, iterative algorithms). SVD operations are still used as one part of many such algorithms [165].

Q7.3 What is $\|A_K - A\|_2$ when A has rank r ? Assume $K < r \leq \min(M, N)$.

- A: $\sqrt{\sum_{k=K+1}^r \sigma_k^2}$ B: $\sum_{k=K+1}^r \sigma_k$ C: σ_{K+1} D: 0 E: None of these

Explore 7.5 What is $\|\hat{A}_K - A\|_*$? (Choose from the same answer list.)

7.2.7.1 Proof for Spectral Norm

Claim: $\text{rank}(B) \leq K \implies \|A - \hat{A}_K\|_F^2 \leq \|A - B\|_F^2$, where $\hat{A}_K \triangleq \sum_{k=1}^K \sigma_k u_k v_k'$.

Proof. Let $W = [v_1 \ \dots \ v_{K+1}]$, so $\dim(\mathcal{R}(W)) = K + 1$ because the columns of W are orthonormal. Now,

$$\begin{aligned} \text{rank}(B) \leq K &\implies \dim(\mathcal{N}(B)) \geq N - K \\ &\implies \dim(\mathcal{N}(B)) + \dim(\mathcal{R}(W)) \geq (N - K) + (K + 1) = N + 1. \end{aligned}$$

But both $N(B)$ and $R(W)$ are subspaces in \mathbb{F}^N , so they must have a nonzero intersection by (4.12). Thus, there exists some $x \neq 0$ such that $x \in N(B)$ and $x \in R(W)$. Without loss of generality, take $\|x\|_2 = 1$. $x \in R(W) \implies x = WW'x \implies 1 = \|x\|_2 = \|WW'x\|_2 = \|W'x\|_2$ by (2.39). To complete the proof:

$$\begin{aligned}
\|A - \hat{A}_K\|_2^2 &= \left\| \sum_{k=K+1}^r \sigma_k u_k v_k' \right\|^2 = \sigma_{K+1}^2 = \sigma_{K+1}^2 \|W'x\|_2^2 \text{ (because } \|W'x\|_2 = 1) \\
&= \sigma_{K+1}^2 \sum_{k=1}^{K+1} \|v_k'x\|_2^2 \leq \sum_{k=1}^{K+1} \sigma_k^2 \|v_k'x\|_2^2 \leq \sum_k \sigma_k^2 \|v_k'x\|_2^2 = \|Ax\|_2^2 \\
&= \|(A - B)x\|_2^2 \quad (\text{because } x \in N(B)) \\
&\leq \|A - B\|_2^2 \|x\|_2^2 = \|A - B\|_2^2. \quad \square
\end{aligned}$$

Proof (alternative proof based on Weyl's inequality (6.71)). Taking $i = 1$ in (7.4), for $\text{rank}(B) \leq K$ we have $\|A - \hat{A}_K\|_2 = \sigma_{K+1}(A) \leq \sigma_1(A - B) = \|A - B\|_2$. \square

7.2.8 Bases for $\mathbb{F}^{M \times N}$

Recall that a set of linearly independent vectors b_1, b_2, \dots is a basis for a vector space \mathcal{V} iff $\text{span}(\{b_1, b_2, \dots\}) = \mathcal{V}$; that is, we can write every vector in \mathcal{V} as a linear combination of the basis vectors: $v \in \mathcal{V} \implies v = \sum_k b_k \alpha_k$ for some $\alpha_k \in \mathbb{F}$. Now we consider two types of bases for the vector space of matrices $\mathbb{F}^{M \times N}$.

7.2.8.1 Canonical Basis for $\mathbb{F}^{M \times N}$

First, the obvious (canonical) basis for $\mathbb{F}^{M \times N}$ is

$$\mathcal{B} = \{e_m \tilde{e}_n' : m = 1, \dots, M, n = 1, \dots, N\},$$

where e_m denotes the m th unit vector of length M and \tilde{e}_n denotes the n th unit vector of length N . These are clearly linearly independent, and $\text{span}(\mathcal{B}) = \mathbb{F}^{M \times N}$ because

$$A \in \mathbb{F}^{M \times N} \implies A = \sum_{m=1}^M \sum_{n=1}^N a_{mn} e_m \tilde{e}_n'.$$

To attempt “dimensionality reduction” using this basis, we could seek an approximation of the form $B = \sum_{m=1}^M \sum_{n=1}^N b_{mn} e_m \tilde{e}_n'$, where we limit the number of nonzero coefficients b_{mn} . Let $\|\text{vec}(B)\|_0$ denote the number of nonzero elements of matrix B . Then a natural optimization problem is

$$\hat{B} = \underset{B \in \mathbb{F}^{M \times N}, \|\text{vec}(B)\|_0 \leq K}{\arg \min} \|A - B\|_F. \quad (7.8)$$

This problem has a trivial solution: choose the K elements of A having the largest magnitudes $|a_{mn}|$ and set all other elements of \hat{B} to zero. The approximation error is the square root of the sum of the squared magnitudes of all those other elements.

Example 7.8 Consider the following “0-norm” formulation:

$$\hat{B} = \underset{B \in \mathbb{F}^{M \times N}, \|B\|_0 \leq 3}{\arg \min} \left\| \begin{bmatrix} 3 & 5 & 7 \\ 6 & 0 & 2 \end{bmatrix} - B \right\|_F = \begin{bmatrix} 0 & 5 & 7 \\ 6 & 0 & 0 \end{bmatrix}.$$

The approximation error is $\|A - \hat{B}\|_F = \left\| \begin{bmatrix} 3 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix} \right\|_F = \sqrt{2^2 + 3^2}$.

Explore 7.6 How large must K be here for this approximation to have zero error?

Explore 7.7 How large must K be here for the rank- K approximation to have zero error?

The low-rank approximation seems more parsimonious than approximation using \mathcal{B} . To state this rigorously, we must first show that our low-rank approximation also uses a basis for $\mathbb{F}^{M \times N}$.

7.2.8.2 Orthonormal Bases for $\mathbb{F}^{M \times N}$

Let U denote any unitary $M \times M$ matrix and V denote any unitary $N \times N$ matrix. Now endow $\mathbb{F}^{M \times N}$ with the (Frobenius) inner product $\langle A, B \rangle = \text{trace}(B'A)$ and corresponding (Frobenius) norm $\|A\|_F = \sqrt{\langle A, A \rangle}$.

Claim: The following set of MN rank-1 $M \times N$ outer-product matrices is an orthonormal basis for $\mathbb{F}^{M \times N}$:

$$\{u_m v'_n : m = 1, \dots, M, n = 1, \dots, N\}. \quad (7.9)$$

Proof (of orthonormality for the Frobenius inner product):

$$\begin{aligned} \langle u_m v'_n, u_k v'_l \rangle &= \text{trace}\left\{(u_k v'_l)' (u_m v'_n)\right\} = \text{trace}\{v_l u'_k (u_m v'_n)\} \\ &= \text{trace}\{u'_k u_m v'_n v_l\} = (u'_k u_m)(v'_n v_l) = \begin{cases} 1, & k = m, l = n, \\ 0, & \text{otherwise.} \end{cases} \quad \square \end{aligned}$$

Being orthonormal, the set $\{u_m v'_n\}$ spans an MN -dimensional space and is a basis for $\mathbb{F}^{M \times N}$. One can write any $X \in \mathbb{F}^{M \times N}$ in terms of that basis, that is, $X = UCV' = \sum_{m=1}^M \sum_{n=1}^N c_{mn} u_m v'_n$, where $C \triangleq U'XV$.

When working with a matrix A , using the basis for $\mathbb{F}^{M \times N}$ that is formed from its own left and right singular vectors, that is, U and V , respectively, turns out to provide the most parsimonious representation. Note that the original low-rank problem formulation (7.1) did not involve any SVD, but an SVD arose in the solution.

Q7.4 Does the canonical basis $\{e_m \tilde{e}'_n : m = 1, \dots, M, n = 1, \dots, N\}$ form an orthonormal basis for $\mathbb{F}^{M \times N}$ when using the Frobenius inner product?

A: Yes

B: No

C: Insufficient information

7.2.9 Low-Rank Approximation Summary

The low-rank approximation problem (7.1) and its SVD-based solution (7.2) provide the best low-rank *representation* (or approximation) of a given matrix (in the Frobenius norm sense). Specifically, if B is any rank- K matrix having the same size as a matrix $A = \sum_{k=1}^r \sigma_k u_k v'_k$, then for $\hat{A}_K = \sum_{k=1}^K \sigma_k u_k v'_k$ we have

$$\begin{aligned}
 \sqrt{\sum_{k=K+1}^r \sigma_k^2} &= \|A - \hat{A}_K\|_F \\
 &\leq \|A - B\|_F \quad \forall B \in \mathcal{L}_K^{M \times N} \\
 &< \|A - B\|_F \quad \forall B \in \mathcal{L}_K^{M \times N} \setminus \{\hat{A}_K\}, \quad \sigma_K > \sigma_{K+1}.
 \end{aligned} \tag{7.10}$$

Explore 7.8 The definition in (7.1) describes low-rank matrix approximation as a projection onto the (nonconvex) set of rank (at most) K matrices. In contrast, [166] defines the operation $A \mapsto U'_K A V_K$. Discuss whether that operation is a projection operator.

7.2.9.1 Generalizations

What if the data is corrupted by noise? What if $Y = X + Z$, where X is a low-rank matrix (or close to low rank)? Due to the noise matrix Z , typically Y is not low rank. We really want to recover X here, not just represent or approximate Y . How do we do that? An answer called OptShrink is given in [167] and discussed in Section 7.6.2.

Low-rank approximation is closely related to principal component analysis, discussed in Section 7.7.2. There are many PCA generalizations, including robust methods (see Section 10.5.1), nonlinear models, multilinear (tensor) models, nonnegative matrix factorization (NMF; see Section 10.6) [168, 169], methods that use sparsity, and so on.

7.2.10 Rank and Stability

Although rank is a simple mathematical concept, it is not stable numerically. Precisely, it is not a continuous function of the elements of a matrix.

Example 7.9 Consider the following two matrices for $a \neq 0$:

$$A = \begin{bmatrix} a & a \\ a & a \end{bmatrix}, \quad B = \begin{bmatrix} a & a \\ a(1+\varepsilon) & a \end{bmatrix}, \quad 0 < |\varepsilon| \ll 1.$$

These two matrices are nearly indistinguishable numerically, yet $\text{rank}(A) = 1$ and $\text{rank}(B) = 2$.

An alternative matrix property used in some situations is the stable rank [170], defined (for $A \neq \mathbf{0}$) as

$$\frac{\|A\|_F^2}{\|A\|_2^2}, \quad \text{where } 1 \leq \frac{\|A\|_F^2}{\|A\|_2^2} = \frac{1}{\sigma_1^2} \sum_{k=1}^{\min(M,N)} \sigma_k^2 = 1 + \sum_{k=2}^{\min(M,N)} \frac{\sigma_k^2}{\sigma_1^2} \leq r, \tag{7.11}$$

because (Problem 6.12) $\|A\|_2 \leq \|A\|_F \leq \sqrt{r}\|A\|_2$. For Example 7.9,

$$\frac{\|B\|_F^2}{\|B\|_2^2} \approx \frac{4a^2 + 2\epsilon a^2}{4a^2} = 1 + \frac{\epsilon}{2}.$$

Example 7.10 For $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$, the stable rank is $\frac{\|A\|_F^2}{\|A\|_2^2} = \frac{2^2 + 1^2}{2^2} = 1.25 \in [1, 2]$.

Q7.5 What is the stable rank of $A = xy'$, when x and y are nonzero vectors?

- A: 0 B: 1 C: $\|x\|_2\|y\|_2$ D: $\|x\|_2^2\|y\|_2^2$

7.2.11 Example: Photometric Stereo

In computer vision, photometric stereo is a set of methods for estimating the surface normals of 3D objects from a set of (often grayscale) images of that object taken with different lighting conditions. The original method assumed knowledge of the lighting directions [171], whereas later methods relaxed this assumption [172, 173]. Extensions include handling shadows and specular reflections [166, 174, 175, 176]. For a Lambertian surface illuminated by a distance point light source, the reflected light intensity at a point on the object is proportional to the inner product between the surface normal vector and the normalized light direction vector. Given N images (for N light sources), each with M pixels, we create an $M \times N$ data array Y where the n th column of Y is the vec of the n th image. The Lambertian model is $Y = XL$, where X is the (unknown) matrix of $M \times 3$ surface normals, and L is the $3 \times N$ matrix of light directions. Ideally (in the absence of noise and shadows, etc.), the matrix Y has (at most) rank 3, so one can use a rank-3 factorization to estimate the surface normals and the lighting directions.

-  Demo 7.2 illustrates the input and output data for photometric stereo; see Fig. 7.6. That demo also uses the Procrustes method from Section 6.7 to align coordinate systems.

7.3 Sensor Localization Application: Multidimensional Scaling

We now turn to another application of matrix factorization: sensor localization via multidimensional scaling (MDS). Suppose $J \geq d$ sensors are located at unknown locations $c_1, \dots, c_J \in \mathbb{R}^d$, where typically $d = 2$ or $d = 3$. All we are given is the $J \times J$ distance matrix D having elements

$$d_{ij} = d_{ji} = \underbrace{\|c_i - c_j\|_2}_{\hookrightarrow \text{usual Euclidean distance}}, \quad i, j = 1, \dots, J. \quad (7.12)$$

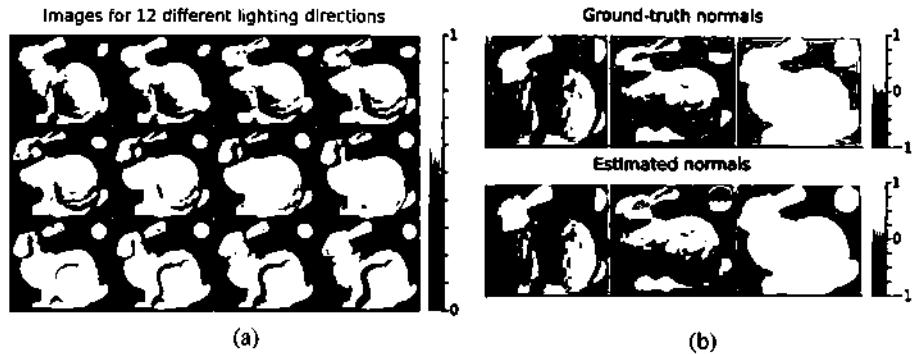


Figure 7.6 Illustration of photometric stereo. (a) Twelve input images of a static 3D object taken with different lighting directions (data from [175]). (b) x , y , and z components of true and estimated surface normals.

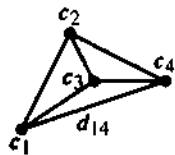


Figure 7.7 Example sensor set.

Clearly, by definition the diagonal elements of D are zero: $d_{jj} = 0$. The goal of MDS is, given D , to determine the locations $\{c_j\}$. An example scenario is shown in Fig. 7.7.

There is a fundamental ambiguity because translating or rotating or reflecting coordinates does not change D . Specifically, if $\tilde{c}_j = Qc_j + d$ where Q is a $d \times d$ orthogonal matrix (such as a rotation matrix) and $d \in \mathbb{R}^d$ is a displacement vector, then

$$\|\tilde{c}_i - \tilde{c}_j\|_2 = \|(Qc_i + d) - (Qc_j + d)\|_2 = \|Q(c_i - c_j)\|_2 = \|c_i - c_j\|_2, \quad (7.13)$$

because the Euclidean norm is unitarily invariant. So D would be the same for $\{\tilde{c}_j\}$ and $\{c_j\}$. Thus, we must be content with determining locations $\{c_j\}$ to within some translation and rotation factor. Note the choice to use a matrix representation of the data! And of course, the locations are vectors.

7.3.1 Derivation (Analysis)

To derive a solution (for noiseless data), define another matrix S by the square of the elements of D :

$$s_{ij} \triangleq d_{ij}^2 = \|c_i - c_j\|_2^2 = \|c_i\|_2^2 + \|c_j\|_2^2 - 2 \langle c_i, c_j \rangle. \quad (7.14)$$

Naturally we write S in matrix form:

$$\begin{aligned}
 S &= \begin{bmatrix} s_{11} & \cdots & s_{1J} \\ \vdots & \cdots & \vdots \\ s_{J1} & \cdots & s_{JJ} \end{bmatrix} \\
 &= \begin{bmatrix} \|c_1\|_2^2 & \cdots & \|c_1\|_2^2 \\ \vdots & \cdots & \vdots \\ \|c_J\|_2^2 & \cdots & \|c_J\|_2^2 \end{bmatrix} + \begin{bmatrix} \|c_1\|_2^2 & \cdots & \|c_J\|_2^2 \\ \vdots & \cdots & \vdots \\ \|c_1\|_2^2 & \cdots & \|c_J\|_2^2 \end{bmatrix} - 2 \begin{bmatrix} \langle c_1, c_1 \rangle & \cdots & \langle c_1, c_J \rangle \\ \vdots & \cdots & \vdots \\ \langle c_J, c_1 \rangle & \cdots & \langle c_J, c_J \rangle \end{bmatrix} \\
 &= \begin{bmatrix} \|c_1\|_2^2 \\ \vdots \\ \|c_J\|_2^2 \end{bmatrix} \mathbf{1}'_J + \mathbf{1}_J \begin{bmatrix} \|c_1\|_2^2 & \cdots & \|c_J\|_2^2 \end{bmatrix} - 2 \begin{bmatrix} c'_1 \\ \vdots \\ c'_J \end{bmatrix} [c_1 \cdots c_J] \\
 &= r \mathbf{1}'_J + \mathbf{1}_J r' - 2 \mathbf{C}' \mathbf{C}, \quad r \triangleq \begin{bmatrix} \|c_1\|_2^2 \\ \vdots \\ \|c_J\|_2^2 \end{bmatrix}, \quad \mathbf{C} \triangleq \underbrace{[c_1 \cdots c_J]}_{d \times J \text{ unknown locations}}. \tag{7.15}
 \end{aligned}$$

Unfortunately, we do not know the values in the vector r .

So we know the $J \times J$ matrix S and we want to solve for the $d \times J$ unknown location matrix \mathbf{C} using

$$S = r \mathbf{1}'_J + \mathbf{1}_J r' - 2 \mathbf{C}' \mathbf{C}. \tag{7.16}$$

Because S on the left is symmetric and zero along its diagonal, it contains $(J^2 - J)/2$ distinct known values. The right-hand side depends on the dJ unknown values in \mathbf{C} .

Because source localization has a translation ambiguity, we may as well use a coordinate system where the centroid is $\mathbf{0}$; that is, we assume $\mathbf{C}\mathbf{1}_J = \sum_{j=1}^J c_j = \mathbf{0}$ without losing any further generality. Following (6.103), define the following “de-meaning” operator that projects onto the orthogonal complement of $\mathcal{R}(\mathbf{1}_J)$:

$$\mathbf{P}^\perp \triangleq I - \frac{1}{J} \mathbf{1}_J \mathbf{1}'_J. \tag{7.17}$$

Clearly, $\mathbf{P}^\perp \mathbf{1}_J = \mathbf{0}$, so $\mathbf{1}'_J \mathbf{P}^\perp = \mathbf{0}'$ and $\mathbf{C}\mathbf{P}^\perp = \mathbf{C}(I - (1/J)\mathbf{1}_J \mathbf{1}'_J) = \mathbf{C}$, so, from (7.16),

$$\mathbf{P}^\perp S \mathbf{P}^\perp = \mathbf{P}^\perp (r \mathbf{1}'_J + \mathbf{1}_J r' - 2 \mathbf{C}' \mathbf{C}) \mathbf{P}^\perp = -2 \mathbf{C}' \mathbf{C}.$$

Rearranging leads to the following simple expression for the $J \times J$ Gram matrix:

$$\mathbf{G} \triangleq \mathbf{C}' \mathbf{C} = -\frac{1}{2} \mathbf{P}^\perp S \mathbf{P}^\perp. \tag{7.18}$$

In words, we remove the row and column means of S and divide by -2 .

Now we have the Gram matrix $\mathbf{G} = \mathbf{C}' \mathbf{C}$, but we really want the coordinates matrix \mathbf{C} itself. Although $\mathbf{C}' \mathbf{C}$ is a $J \times J$ matrix, $d \leq J$ so typically $\text{rank}(\mathbf{C}' \mathbf{C}) = d$ (in the absence of noise). If the locations happen to be linearly dependent, that is, if the sensors

are along one line through the origin in 2D or in the same plane through the origin in 3D, then $\text{rank}(\mathbf{C}'\mathbf{C}) < d$. To elaborate, consider $d = 2$ where

$$\mathbf{C} \triangleq [\mathbf{c}_1 \ \cdots \ \mathbf{c}_J] = \begin{bmatrix} x_1 & \cdots & x_J \\ y_1 & \cdots & y_J \end{bmatrix},$$

where $\mathbf{c}_j = (x_j, y_j)$. Because \mathbf{C} has two rows, $\text{rank}(\mathbf{C}'\mathbf{C}) = \text{rank}(\mathbf{C}) \leq 2$.

Explore 7.9 Determine when $\text{rank}(\mathbf{C}) = 1$.

The Gram matrix $\mathbf{C}'\mathbf{C}$ consists of inner products, and it is invariant to rotations of the source locations:

$$\tilde{\mathbf{C}} = \mathbf{Q}\mathbf{C} \implies \tilde{\mathbf{C}}'\tilde{\mathbf{C}} = (\mathbf{Q}\mathbf{C})'(\mathbf{Q}\mathbf{C}) = \mathbf{C}'\mathbf{Q}'\mathbf{Q}\mathbf{C} = \mathbf{C}'\mathbf{C}.$$

Given the (noiseless) $J \times J$ Gram matrix $\mathbf{G} \triangleq \mathbf{C}'\mathbf{C} = -\frac{1}{2}\mathbf{P}^\perp \mathbf{S} \mathbf{P}^\perp$, we determine its compact SVD:

$$\mathbf{G} = \mathbf{V}\Sigma\mathbf{V}' = \underbrace{\sum_{k=1}^J \sigma_k \mathbf{v}_k \mathbf{v}_k'}_{\text{(noiseless)}} \underset{k=1}{=} \sum_{k=1}^d \sigma_k \mathbf{v}_k \mathbf{v}_k' = \mathbf{V}_d \Sigma_d \mathbf{V}_d', \quad (7.19)$$

$$\Sigma_d \triangleq \underbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_d \end{bmatrix}}_{d \times d}, \quad \mathbf{V}_d \triangleq \underbrace{\begin{bmatrix} 1 & & & \\ \mathbf{v}_1 & \cdots & \mathbf{v}_d \\ 1 & & 1 \end{bmatrix}}_{J \times d}.$$

- Because \mathbf{G} is (symmetric) positive semidefinite (in the absence of noise), we know $\mathbf{U} = \mathbf{V}$.
- Because \mathbf{G} has rank at most d (in the absence of noise), the above low-rank form is *exact*, not an approximation like we used earlier in this chapter.

We define our estimate of the source locations to be the J columns of this $d \times J$ matrix:

$$\hat{\mathbf{C}} = \Sigma_d^{1/2} \mathbf{V}_d' = \underbrace{\begin{bmatrix} \sqrt{\sigma_1} & & \\ & \ddots & \\ & & \sqrt{\sigma_d} \end{bmatrix}}_{d \times d} \underbrace{\begin{bmatrix} - & \mathbf{v}_1' & - \\ \vdots & & \vdots \\ - & \mathbf{v}_d' & - \end{bmatrix}}_{d \times J}, \quad (7.20)$$

because this estimate $\hat{\mathbf{C}}$ is consistent with \mathbf{C} to within an unknown rotation and translation:

$$\hat{\mathbf{C}}'\hat{\mathbf{C}} = (\Sigma_d^{1/2} \mathbf{V}_d')' (\Sigma_d^{1/2} \mathbf{V}_d') = \mathbf{V}_d \Sigma_d \mathbf{V}_d' \underset{\text{(noiseless)}}{=} \underbrace{\mathbf{V}\Sigma\mathbf{V}'}_! = \mathbf{G} = \mathbf{C}'\mathbf{C}. \quad (7.21)$$

Now it seems like we are done because we have an expression for \mathbf{G} that is computable from \mathbf{S} , so we can use its SVD to find the source location estimates $\hat{\mathbf{C}}$. However, we assumed that $\mathbf{C}\mathbf{1}_J = \mathbf{0}$, so we must verify that $\hat{\mathbf{C}}\mathbf{1}_J = \mathbf{0}$ to ensure that our approach is self-consistent. Because $\mathbf{P}^\perp \mathbf{1}_J = \mathbf{0}$, it follows from (7.17) that $\mathbf{G}\mathbf{1}_J = \mathbf{0}$, so

$V\Sigma V' \mathbf{1}_J = \mathbf{0}$. In the noiseless case, from (7.21) we then have $V_d \Sigma_d V_d' \mathbf{1}_J = \mathbf{0}$, which in turn implies that $\hat{C} \mathbf{1}_J = \Sigma_d^{1/2} V_d' \mathbf{1}_J = \mathbf{0}$.

Q7.6 What is the size of the final $\mathbf{0}$ in the immediately preceding line?

- A: $J \times J$ B: J C: $J \times d$ D: $d \times J$ E: d

7.3.2 MDS Method

We summarize the MDS method for finding locations from distances (in the possible presence of noise):

- Given distances arranged in a matrix D .
- Compute S from D by squaring elements.
- Compute $\hat{G} \triangleq -\frac{1}{2} P^\perp S P^\perp \approx C'C$ by de-meaning.
- Optional: force \hat{G} to be symmetric by replacing it with $(\hat{G} + \hat{G}')/2$.
- Compute the compact SVD $\hat{G} \approx U_d \Sigma_d V_d'$. This is a rank- d matrix approximation in the presence of noise (the topic of this chapter). (If \hat{G} is symmetric then $U_d = V_d$.)
- Use the first d terms for source location estimate: $\hat{C} = \Sigma_d^{1/2} V_d'$.

This remarkably simple algorithm for finding \hat{C} is called (classic) multidimensional scaling (MDS) [177, Chapter 12], and using the low-rank compact SVD is a key step. Because $G = C'C$, matrix G is positive semidefinite (in the absence of noise). If we force \hat{G} to be likewise symmetric, then its eigenvalues are all real, so alternatively one could use an eigendecomposition of \hat{G} with suitably ordered eigenvalues.

Example 7.11 To see how symmetrizing a matrix does not ensure nonnegative eigenvalues in general (in the presence of noise), consider the (asymmetric, but still dc-meanned) matrix

$$\hat{G} = \begin{bmatrix} 0 & 2 & -2 \\ 4 & 0 & -4 \\ -4 & -2 & 6 \end{bmatrix} \implies (\hat{G} + \hat{G}')/2 = \begin{bmatrix} 0 & 3 & -3 \\ 3 & 0 & -3 \\ -3 & -3 & 6 \end{bmatrix}.$$

One can verify that the eigenvalues of $(\hat{G} + \hat{G}')/2$ are $\{-3, 0, 9\}$.

This situation is why the SVD is preferable to using an eigendecomposition for MDS.

Q7.7 Consider a given noiseless distance matrix D with $J > d$. The position estimates \hat{C} returned by the SVD-based MDS algorithm are unique (to within numerical precision), that is, are the same for any SVD of the Gram matrix.

- A: True B: False

Q7.8 In the possible presence of noise, the product $\hat{C}' \hat{C}$ is unique (to within numerical precision) for any SVD of the calculated Gram matrix \hat{G} .

- A: Never B: Usually C: Always

7.3.3 Practical Implementation

JULIA's `MultivariateStats.jl` has a `classical_mds` command, but the method is so simple that it is just as easy to code our own.

Example 7.12 Consider $J = 3$ sources that are all equally distant:

$$S = D .^2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Just a few lines of JULIA suffices (see the demo code linked below).

```
</> 7.5  using Plots; using LinearAlgebra: svd, Diagonal
       using Statistics: mean
       D = [0 1 1; 1 0 1; 1 1 0] + 0e-9 * randn(3,3) # given distances
       S = D.^2
       tmp = S .- mean(S, dims=1)
       G = -0.5 * (tmp .- mean(tmp, dims=2))
       (L, s, V) = svd(G)
       C = Diagonal(sqrt.(s[1:2])) * V[:,1:2]'
       scatter(C[1,:], C[2,:], aspect_ratio=1, label="", 
               xaxis=("x", (-0.6,0.6), -.5:.5:.5),
               yaxis=("y", (-0.6,0.6), -.5:.5:.5), title="s 3(s)")
```

Explore 7.10 What “shape” should appear?

The SVD has nontrivial nonuniqueness in this case:

$$\Sigma = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$G = \frac{1}{6} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} = V \Sigma V', \quad V = \begin{bmatrix} -\sqrt{2/3} & 0 & \sqrt{1/3} \\ \sqrt{1/6} & \sqrt{1/2} & \sqrt{1/3} \\ \sqrt{1/6} & -\sqrt{1/2} & \sqrt{1/3} \end{bmatrix},$$

$$= \tilde{V} \Sigma \tilde{V}', \quad \tilde{V} = \begin{bmatrix} \sqrt{1/2} & \sqrt{1/6} & \sqrt{1/3} \\ 0 & -2\sqrt{1/6} & \sqrt{1/3} \\ -\sqrt{1/2} & \sqrt{1/6} & \sqrt{1/3} \end{bmatrix}.$$

These decompositions lead to two (of many possible) different location estimates, as shown in Fig. 7.8:

$$\hat{C} = \Sigma_2^{1/2} V'_2 = \sqrt{1/2} \begin{bmatrix} -\sqrt{2/3} & \sqrt{1/6} & \sqrt{1/6} \\ 0 & \sqrt{1/2} & -\sqrt{1/2} \end{bmatrix},$$

$$\hat{\tilde{C}} = \Sigma_2^{1/2} \tilde{V}'_2 = \sqrt{1/2} \begin{bmatrix} \sqrt{1/2} & 0 & -\sqrt{1/2} \\ \sqrt{1/6} & -2\sqrt{1/6} & \sqrt{1/6} \end{bmatrix}.$$

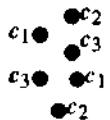


Figure 7.8 Multidimensional scaling results.

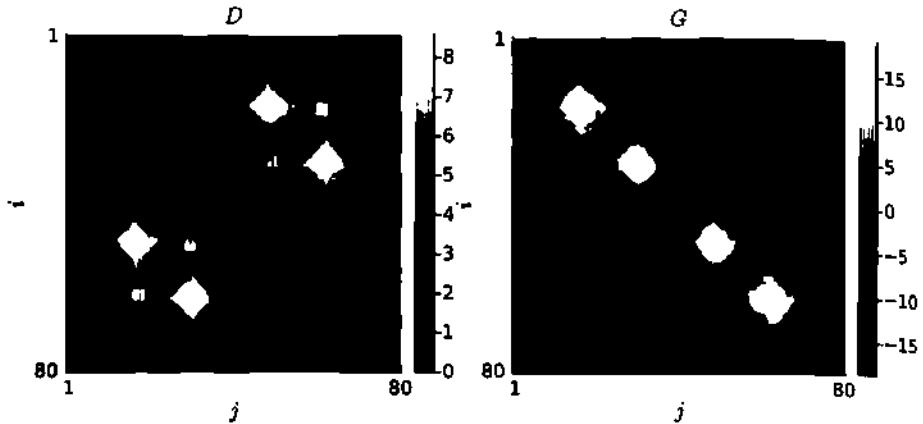


Figure 7.9 Images of D and G for source localization demo.

Demo 7.3 considers an unknown collection of $J = 80$ source points. See Fig. 7.9 for images of D and G . Run the demo code to see \hat{C} and consider noisy cases to examine robustness.

7.3.4 Extensions

- When the distance measurements are noisy, then D may not be perfectly symmetric and $P^\perp S P^\perp$ may have (hopefully small) negative eigenvalues. Use the truncated SVD: truncate the rank to the dimension d of the embedding, or shrink the eigenvalues [178].
- What if some sensors cannot reach some other sensors, that is, if D has some missing elements?
- Is there a way to formulate the problem that considers the possibility of noise (errors in the distance values) at the outset, instead of simply empirically investigating the effects of noise on our solution?
- What if there are outliers in the data, for example some sensors that give incorrect values (either due to errors or because the sensor was hacked or otherwise compromised)? Is MDS robust to such errors, or is a different formulation needed, perhaps based on a robust distance measure like $\| \cdot \|_1$?
- If we want to compare the location estimates \hat{C} to some ground truth, we can use the Procrustes method to align them, compensating for the inherent rotation/translation ambiguity.

Multidimensional scaling is used to track flu and corona virus mutations [179]. More generally, MDS is used as a dimensionality reduction method (cf. Section 8.9.6).

Explore 7.11 What happens if D is geodesic distances or distances along roads between locations?

Explore 7.12 The derivation here was for the usual case where $C \in \mathbb{R}^{d \times J}$. Suppose $C \in \mathbb{C}^{d \times J}$. Does the derivation extend to this case? If not, which step does not generalize?

7.4 Proximal Operators

Our next topic is some alternative low-rank approximation methods, also formulated as optimization problems. To describe those alternatives, we first need to introduce an important operation.

Definition The proximal operator [180, 181] (also called the proximal mapping [182, Chapter 6]) associated with a (typically convex) function $f: \mathbb{F}^N \mapsto \mathbb{R}$ is defined, for any $v \in \mathbb{F}^N$, as the following minimizer:

$$\text{prox}_f(v) \triangleq \arg \min_{x \in \mathbb{F}^N} \frac{1}{2} \|v - x\|_2^2 + f(x). \quad (7.22)$$

- The norm squared is a strictly convex function, so when f is convex the “arg min” is unique $\forall v \in \mathbb{F}^N$.
- Often the function f is additively separable, in which case the “arg min” separates into N individual 1D minimization problems. See Example 7.13.
- The proximal operator transforms one function $f(\cdot)$ into another function $\text{prox}_f(\cdot)$.

Q7.9 In general, the functions f and prox_f have the same domain and/or codomain?

A: True, True B: True, False C: False, True D: False, False

7.4.1 Soft Thresholding

Example 7.13 The case $\mathbb{F} = \mathbb{R}$ and $f(x) = \beta \|x\|_p^p = \beta \sum_{n=1}^N |x_n|^p$, for $\beta \geq 0$, is especially important. The most important case is when $p = 1$, for which

$$\text{prox}_f(v) = \arg \min_{x \in \mathbb{R}^N} \frac{1}{2} \|v - x\|_2^2 + \beta \|x\|_1 = \arg \min_{x \in \mathbb{R}^N} \sum_{n=1}^N \left(\frac{1}{2} (v_n - x_n)^2 + \beta |x_n| \right).$$

This sum is additively separable, meaning each x_n appears in just one term of the sum. So we can solve the minimization problem separately for each x_n term by solving

$$\hat{x}_n \triangleq \arg \min_{x_n \in \mathbb{R}} g_n(x_n), \quad g_n(x_n) \triangleq \frac{1}{2}(v_n - x_n)^2 + \beta |x_n|, \quad n = 1, \dots, N. \quad (7.23)$$

Often for minimization problems we take the derivative and set it to zero. That approach does not quite work here because $|\cdot|$ is not differentiable at 0. So we need a “braces and cases” approach. Suppose the minimizer satisfies $\hat{x}_n > 0$. In this regime the function g is differentiable with derivative $\dot{g}_n(x_n) = x_n - v_n + \beta 1$. Equating to zero yields $\hat{x}_n = v_n - \beta$. However, we must keep in mind that we assumed here that $\hat{x}_n > 0$, so this solution is valid only when $v_n > \beta$.

Now suppose the minimizer satisfies $\hat{x}_n < 0$. In this regime the function g is differentiable with derivative $\dot{g}_n(x_n) = x_n - v_n + \beta(-1)$. Equating to zero yields $\hat{x}_n = v_n + \beta$. Here we assumed that $\hat{x}_n < 0$, so this solution is valid only when $v_n < -\beta$. Because we have considered all cases where the minimizer is nonzero, for any other input value v_n , the minimizer must be zero. Thus we have shown that

$$\begin{aligned} \arg \min_{x_n \in \mathbb{R}} \frac{1}{2}(v_n - x_n)^2 + \beta |x_n| &= \text{soft}(v_n; \beta), \\ \text{soft}(v_n; \beta) &\triangleq \begin{cases} v_n - \beta, & v_n > \beta, \\ v_n + \beta, & v_n < -\beta, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \quad (7.24)$$

The profile of $\text{soft}(v_n; \beta)$ is shown in Fig. 7.10.

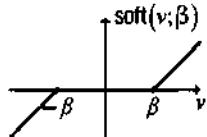


Figure 7.10 Soft thresholding.

This operation is called soft thresholding and it induces sparsity because of the range of values set to zero. Putting it all together, we have shown that the proximal operator for the 1-norm scaled by β is elementwise soft thresholding:

$$\text{prox}_{\beta \|\cdot\|_1}(v) = \text{soft.}(v; \beta). \quad (7.25)$$

(The `.` here in `soft.` denotes Julia-style elementwise broadcast.)

Figure 7.11 illustrates the cases where $v < \beta$ and $v > \beta$ by plotting

$$\frac{1}{2}(v - x)^2 - \beta |x|$$

- When $0 < \beta \leq v$ the minimizer is at $x = v - \beta$.
- When $-\beta \leq v \leq \beta$, the minimizer is at $x = 0$.

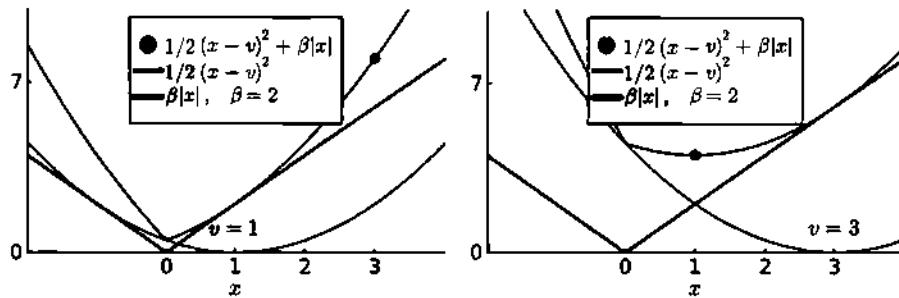


Figure 7.11 Soft thresholding component functions.

Fact 7.3 The generalization of the proximal operator (7.24) to complex numbers is (Problem 7.3):

$$\arg \min_{z \in \mathbb{C}} \frac{1}{2} |s - z|^2 + \beta |z| = [|z| - \beta]_+ e^{i \angle z}. \quad (7.26)$$

$$[x]_+ \triangleq \max(x, 0). \quad (7.27)$$

7.4.2 Hard Thresholding

Example 7.14 Another main case of interest here is the (nonconvex) 0-norm $f(x) = \beta \|x\|_0$ for $\beta \geq 0$, for which the proximal operator is (elementwise) hard thresholding. Here is the derivation:

$$\begin{aligned} \text{prox}_{\beta\|x\|_0}(v) &= \arg \min_{x \in \mathbb{R}^N} \frac{1}{2} \|v - x\|_2^2 + \beta \|x\|_0 \\ &= \arg \min_{x \in \mathbb{R}^N} \sum_{n=1}^N \left(\frac{1}{2} |v_n - x_n|^2 + \beta \mathbb{I}_{\{x_n \neq 0\}} \right) = h_{\text{hard}}(v; \beta), \\ h_{\text{hard}}(v; \beta) &\triangleq \arg \min_{x \in \mathbb{R}} \frac{1}{2} |v - x|^2 + \beta \mathbb{I}_{\{x \neq 0\}} \\ &= \begin{cases} v, & \frac{1}{2} |v|^2 > \beta \\ 0, & \text{otherwise} \end{cases} = v H(|v| - \sqrt{2\beta}), \end{aligned} \quad (7.28)$$

where $H(t) = \mathbb{I}_{\{t>0\}}$ denotes the Heaviside step function. We cannot derive the minimizer by differentiation, because the indicator function is not differentiable at zero. So we must apply “braces and cases.” The profile of $h_{\text{hard}}(v; \beta)$ is shown in Fig. 7.12.

If the minimizer is nonzero, then the second term is β and the minimizer of the first term is v , and the overall cost is $0 + \beta = \beta$. If the minimizer is 0, then the first term is $(v - 0)^2/2$ and the second term is 0, so the cost is $v^2/2$. So 0 is the minimizer when $v^2/2 < \beta$, otherwise v is the minimizer.

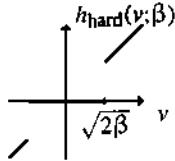


Figure 7.12 Hard thresholding.

The minimizer is not unique if $|v| = \sqrt{2\beta}$; in such cases, both v and 0 are global minimizers. The threshold is $\sqrt{2\beta}$ due to the $1/2$ in front of the norm in the proximal operator definition.

Now we return to low-rank approximation where we will use these results.

7.5 Alternative Low-Rank Approximation Formulations

If $Y \in \mathbb{F}^{M \times N}$ is a given data matrix with SVD $Y = U\Sigma V'$, and $Y = X + Z$ where we believe $\text{rank}(X) \leq K$ and Z denotes an $M \times N$ noise matrix, then the constrained formulation discussed so far is natural [153]:

$$\hat{X} = \arg \min_{X \in \mathcal{L}_K^{M \times N}} \|Y - X\|_F^2 = U_K \Sigma_K V'_K = \sum_{k=1}^K \sigma_k u_k v'_k = \sum_{k=1}^r \sigma_k \mathbb{I}_{\{k \leq K\}} u_k v'_k. \quad (7.29)$$

This problem is nonconvex, but still has a nice solution (7.2), reiterated in (7.29) in three different forms. The final form reminds us that we have “set to 0” all the singular values for $k > K$.

7.5.1 Unconstrained/Regularized Formulation

When the rank is unknown, an alternative approach is to consider an *unconstrained* cost function that discourages (penalizes) high rank (high complexity) rather than constraining it [183]:

$$\hat{X} = \arg \min_{X \in \mathbb{F}^{M \times N}} \frac{1}{2} \|Y - X\|_F^2 + \beta \text{rank}(X) = \sum_{k=1}^r h_{\text{hard}}(\sigma_k; \beta) u_k v'_k, \quad (7.30)$$

where $h_{\text{hard}}(\cdot; \beta)$ is the hard thresholding function defined in (7.28).

7.5.2 General Unitarily Invariant Formulations

There are several ways to form low-rank approximations that balance between data misfit and model complexity, all having the general form

$$\hat{X} = \arg \min_{X \in \mathbb{F}^{M \times N}} \frac{1}{2} \|Y - X\|_{\text{UI}}^2 + \beta R(X) \quad (7.31)$$

for some unitarily invariant matrix norm $\|\cdot\|$ and for some regularizer $R(X)$, where $R: \mathbb{F}^{M \times N} \mapsto \mathbb{R}$, such as $R(X) = \text{rank}(X)$. The regularization parameter (or hyper-parameter) $\beta > 0$ controls the trade-off between fit to the data Y , as measured by $\|Y - X\|$, and model complexity, as quantified by $R(X)$.

We assume hereafter that the regularizer is also a unitarily invariant function, that is, $R(UXV) = R(X)$ for any suitably sized unitary matrices U and V . When both the data misfit norm and the regularizer are unitarily invariant norms (or monotonically increasing functions thereof [163, Remark 3]), using the symmetric gauge principles (Problem 7.7) of [123, 154, 184] or [163, Theorem 4], one can use the compact SVD $Y = U_r \Sigma_r V_r'$ to show that any minimizer of (7.31) has the form [163, Corollary 2]

$$\hat{X} = \sum_{k=1}^r \hat{w}_k u_k v_k' = U_r \hat{\Sigma}_r V_r' \text{ where } \hat{\Sigma}_r = \text{Diag}\{\hat{w}_k\}, \quad \hat{w}_k = h_k(\sigma_1, \dots, \sigma_r; \beta), \quad (7.32)$$

for some shrinkage or thresholding function $h_k(\cdot; \beta)$ that depends on the data misfit norm and the regularizer.

Q7.10 What is the rank of the \hat{X} in (7.32) when $\text{rank}(Y) = r$? (Choose the best answer.)
A: Usually 1 B: Always 1 C: Usually r D: Always r E: None of these

Because (7.32) is the form of the solution, we can rewrite the general optimization problem (7.31) as

$$\hat{X} = U_r \hat{\Sigma}_r V_r', \quad \hat{\Sigma}_r = \underset{S=\text{Diag}\{s_1, \dots, s_r\}}{\arg \min} \frac{1}{2} \| \Sigma_r - S \|_F^2 + \beta R(S) = \text{Diag}\{\hat{w}_1, \dots, \hat{w}_r\}. \quad (7.33)$$

7.5.3 Singular Value Hard Thresholding

Now return to the special case (7.30) with the Frobenius norm and the rank penalty. The equivalent minimization problem in terms of the singular values requires minimizing over $S = \text{Diag}\{s_1, \dots, s_r\}$:

$$\begin{aligned} \frac{1}{2} \| \Sigma_r - S \|_F^2 + \beta \text{rank}(S) &= \sum_{k=1}^r \left(\frac{1}{2} (\sigma_k - s_k)^2 + \beta \mathbb{I}_{\{s_k \neq 0\}} \right) \\ \implies \hat{w}_k &= \arg \min_s \frac{1}{2} (\sigma_k - s)^2 + \beta \mathbb{I}_{\{s \neq 0\}} = h_{\text{hard}}(\sigma_k; \beta) = \sigma_k H(\sigma_k - \sqrt{2\beta}). \end{aligned} \quad (7.34)$$

We started with the cost function (7.30), simplified it to the equivalent problem with diagonal matrices (7.33), and now we have r separate 1D problems that we solved earlier when deriving the proximal operator for the 0-norm that resulted in hard thresholding; see (7.28). This analysis leads to $\hat{w}_k = h_{\text{hard}}(\sigma_k; \beta)$ in the method (7.30) and is called singular value hard thresholding (SVHT).

7.5.4 Singular Value Soft Thresholding

Any formulation involving $\text{rank}(\cdot)$ is nonconvex, and it is somewhat remarkable that a nice solution to (7.30) exists despite that nonconvexity. Often it can be preferable to have a convex formulation. The convex relaxation [185, 186] of rank is the nuclear norm that leads to soft thresholding of singular values:

$$\begin{aligned}\hat{X} &= \arg \min_{X \in \mathbb{R}^{M \times N}} \frac{1}{2} \|Y - X\|_F^2 + \beta \|X\|_* \\ &= \sum_{k=1}^r [\sigma_k - \beta]_+ u_k v_k', \quad \text{where } [t]_+ = \begin{cases} t, & t > 0, \\ 0, & \text{otherwise.} \end{cases}\end{aligned}\quad (7.35)$$

So $\hat{\sigma}_k = h_{\text{soft}}(\sigma_k; \beta)$, $h_{\text{soft}}(\sigma; \beta) \triangleq [\sigma - \beta]_+ = \max(\sigma - \beta, 0)$. This approach is called singular value soft thresholding (SVST). See Fig. 7.13.

Proof (sketch). $\frac{1}{2} \|\Sigma_r - S\|_F^2 + \beta \|S\|_* = \sum_{k=1}^r \frac{1}{2} (\sigma_k - s_k)^2 + \beta s_k \implies \hat{\sigma}_k = \arg \min_s \frac{1}{2} (\sigma_k - s)^2 + \beta |s|$. This is just the proximal operator of $\beta |\cdot|$ that we solved following (7.23), leading to $\hat{s}_k = [\sigma_k - \beta]_+$. \square

Q7.11 If Y has nonzero singular values 3, 5, 6, 7, 9 and $\beta = 6$, what is the rank of \hat{X} here?

- A: 1 B: 2 C: 3 D: 4 E: 5

Q7.12 This \hat{X} equals the best rank- K approximation to Y , where K is answer to previous problem.

- A: True B: False

Q7.13 In general, what is $\|\hat{X}\|_2$ in terms of the notation used throughout here?

- A: β B: σ_1 C: $\sigma_1 - \beta$ D: $\sqrt{\sigma_r - \beta}$ E: $[\sigma_1 - \beta]_+$

Q7.14 In general, if $\|Y\|_2 \geq \beta$, then what is $\|Y - \hat{X}\|_2$ in terms of the notation used throughout here?

- A: β B: σ_1 C: $\sigma_1 - \beta$ D: $\sqrt{\sigma_r - \beta}$ E: $[\sigma_1 - \beta]_+$

One motivation for these unconstrained formulations is solving matrix completion or matrix sensing problems where we must replace $\|Y - X\|_F^2$ with something like $\|y - A \text{vec}(X)\|_2^2$ where A is a known matrix. See Section 10.4.9.1. These problems

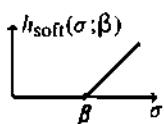


Figure 7.13 Singular value soft thresholding function.

do not have closed-form solutions and require iterative methods, and convergence of iterative methods is better understood for convex problems.

7.5.5 Other Extensions of Low-Rank Approximation

- Sometimes we might want to retain some columns of the data exactly, leading to a different type of constrained low-rank approximation [187].
- Another variation replaces the Frobenius norm by the spectral norm (Problem 7.5):

$$\begin{aligned} \arg \min_X \frac{1}{2} \|Y - X\|_F^2 + \beta \operatorname{rank}(X), \\ \arg \min_X \frac{1}{2} \|Y - X\|_F^2 + \beta \|X\|_{**}. \end{aligned}$$

- Another variation replaces the Frobenius norm by the nuclear norm:

$$\arg \min_X \|Y - X\|_* + \beta \|X\|_*.$$

The solution to this variation is left as an exercise. It seems not to be useful.

- For further generalizations using weighted norms, see [188, 189, 190].

Q7.15 If we use two Frobenius norms, the solution has the following general form:

$$\hat{X} = \arg \min_X \frac{1}{2} \|Y - X\|_F^2 + \beta \frac{1}{2} \|X\|_F^2 = \sum_{k=1}^r \hat{w}_k u_k v'_k, \quad \hat{w}_k = h_{FF}(\sigma_k, \beta). \quad (7.36)$$

What is $h_{FF}(\sigma, \beta)$?

- A: $[\sigma - \beta]_+$ B: $\sigma H(\sigma - \sqrt{2\beta})$ C: $\max(\sigma, \beta)$ D: $\frac{\sigma}{1 + \beta}$ E: $(\sigma - \beta)^2$

Q7.16 If Y is $M \times N$ with rank r , then what is the rank of the solution \hat{X} here?

- A: 0 B: 1 C: r D: $\min(N, M)$ E: None of these

7.6 Choosing the Rank or Regularization Parameter

In all the above low-rank approximation formulations, we must either choose the rank K directly, or choose the regularization parameter β that influences the rank of \hat{X} . As illustrated in Fig. 7.14, this selection process is nontrivial because when $Y = X + Z$ and the latent matrix X has low rank:

- For $\|\hat{X} - Y\|_F$, the difference between the noisy data Y and the low-rank approximation \hat{X} always
 - decreases monotonically as K increases (discretely);
 - increases monotonically as β increases (continuously).

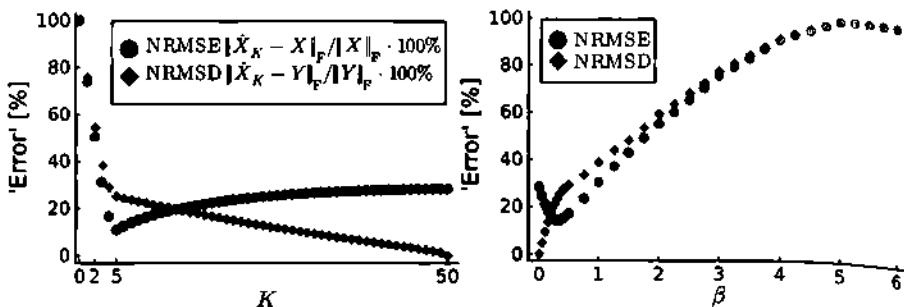


Figure 7.14 Trade-offs between NRMSE and NRMSD as one varies rank K or regularization parameter β .

- For $\|\hat{X} - X\|_F$, the difference between the latent low-rank matrix X and the low-rank approximation \hat{X} :
 - usually decreases initially as K increases, then increases;
 - usually decreases initially as β increases, then increases.

Often we normalize the difference between \hat{X} and Y or that between \hat{X} and X and define the (unitless) normalized root mean-squared difference (NRMSD) and normalized root mean-squared error (NRMSE):

$$\text{NRMSD} = \frac{\|\hat{X} - Y\|_F}{\|Y\|_F} \cdot 100\%, \quad \text{NRMSE} = \frac{\|\hat{X} - X\|_F}{\|X\|_F} \cdot 100\%. \quad (7.37)$$

(There are many variations of these definitions in the literature.)

- ◻ Demo 7.4 considers a case where the 100×50 matrix X has rank 5 and $Y = X + Z$, where Z is an $M \times N$ matrix with IID Gaussian noise. See Fig. 7.14.

Q7.17 What is the y-axis value of the right blue curve in Fig. 7.14 at $\beta = 0$?

- A: $\|Y - X\|_F / \|Y\|_F \cdot 100\%$ B: $\|Y - X\|_F / \|X\|_F \cdot 100\%$ C: Neither

Ideally we would like to find K_* or β_* that minimizes $\|\hat{X} - X\|_F$, the minimizers of the blue curves in Fig. 7.14. Finding either of the choices exactly is impossible because X is unknown in practice! All we have is $Y = X + Z$, and usually we are *hoping* that X is low rank, without knowing for sure.

7.6.1 Stein's Unbiased Risk Estimate

Researchers have developed surrogates for $\|\hat{X} - X\|_F$ that help choose β . One notable method is Stein's unbiased risk estimate (SURE). If $\hat{x}_\beta = \hat{x}(y; \beta)$ is a weakly differentiable estimate of a parameter $x \in \mathbb{R}^d$ from data with Gaussian noise, $y \sim N(x, \sigma_0^2 I)$, then an unbiased estimate of the risk, that is, the mean-squared error (MSE), is:

$$\text{divergence}\{\hat{x}(\cdot; \beta)\} \triangleq \text{trace}\{\nabla_y \hat{x}(y; \beta)\} \sim \underbrace{\text{SURE}(\beta) \triangleq \|\hat{x}(y; \beta) - y\|_2^2 - d\sigma_0^2 + 2\sigma_0^2 \sum_i \frac{\partial}{\partial y_i} \hat{x}_i(y; \beta)}_{\text{Independent of } x!} \quad (7.38)$$

$$E[\text{SURE}(\beta)] = \text{MSE}(\beta) = E[\|\hat{x}(y; \beta) - x\|^2]. \quad (7.39)$$

For a proof, see [191]. Note that σ_0^2 is a noise variance here, not a singular value!

Explore 7.13 Verify (7.39) for the case where $\hat{x}(y; \beta) = U_\beta U'_\beta y$, where U_β is a unitary basis for a β -dimensional subspace of \mathbb{R}^d for $\beta \in \mathbb{N}$.

The SURE approach uses the following approximation to select β :

$$\beta_* \triangleq \arg \min_{\beta} \|\hat{x}_\beta - x\|^2 \approx \arg \min_{\beta} E[\|\hat{x}_\beta - x\|^2] = \arg \min_{\beta} \text{SURE}(\beta). \quad (7.40)$$

The hard thresholding function is *not* weakly differentiable, so we cannot apply the SURE method to rank-constrained or rank-regularized cases. The soft thresholding function *is* weakly differentiable. Remarkably, the divergence expression derived in [183] for any method of the form $\hat{X} = \sum_k u_k h_k(\sigma_k; \beta) v'_k$ turns out to depend only on the singular values, so it is practical to evaluate. When the singular values of Y are distinct:

$$\begin{aligned} \text{SURE}(\beta) &= \|\hat{X} - Y\|_F^2 - MN\sigma_0^2 \\ &+ 2\sigma_0^2 \left(|M-N| \sum_{i=1}^{\min(M,N)} \frac{h(\sigma_i; \beta)}{\sigma_i} + \sum_{i=1}^{\min(M,N)} h_i(\sigma_i; \beta) \right. \\ &\quad \left. + 2 \sum_{i \neq j}^{\min(M,N)} \frac{\sigma_i h_i(\sigma_j; \beta)}{\sigma_i^2 - \sigma_j^2} \right). \end{aligned} \quad (7.41)$$

Example 7.15 See the demo notebook and Fig. 7.15. A practical challenge is that one must know the noise variance σ_0^2 to apply this method. As seen in Fig. 7.15, the minimizer β_* of the SURE function is remarkably close to the minimum MSE choice of β .

7.6.2 OptShrink

Both SVST and SVHT require the user to select a regularization parameter β . The OptShrink method, based on random matrix theory, provides optimal singular value shrinkage for matrix denoising [167], as seen in Fig. 7.16.

The model here is $Y = X + Z$, where Y is the observed matrix, X is the latent matrix, and Z is noise. By random matrix theory, for quite general noise models (biunitarily invariant), the best estimate of X is

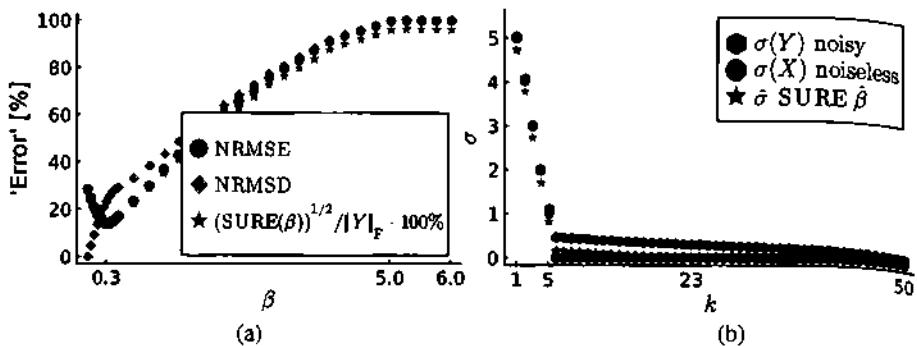


Figure 7.15 (a) Similarity of SURE to NRSME. (b) Singular values of data, latent matrix, and SURE estimate.

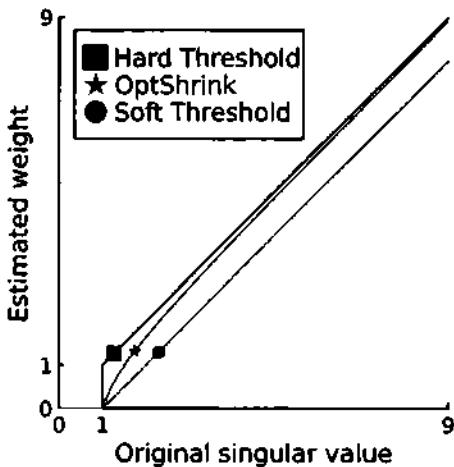


Figure 7.16 Singular value shrinkage.

$$\hat{X} = \sum_{k=1}^r \hat{w}_k u_k v'_k \quad (7.42)$$

where \hat{w}_k is the best “weight” for the k th SVD component [167].

The OptShrink algorithm has the following steps.

- Input: $Y \in \mathbb{R}^{M \times N}$ signal+noise matrix.
- Input: \hat{r} = estimate of rank of latent matrix X .
- Compute SVD: $Y = \sum_{k=1}^{\min(M,N)} \sigma_k u_k v'_k$.
- Form tail singular value matrix:

$$\Sigma_{\hat{r}} = \text{Diag}\{\sigma_{\hat{r}+1}, \dots, \sigma_{\min(M,N)}\} \in \mathbb{R}^{(M-\hat{r}) \times (N-\hat{r})}. \quad (7.43)$$

- For $k = 1, \dots, \hat{r}$, compute $D(\sigma_k, \Sigma_{\hat{r}})$ and $D'(\sigma_k, \Sigma_{\hat{r}})$ using (7.45) and (7.46), then the optimal weight is $\hat{w}_k = -2D(\sigma_k, \Sigma_{\hat{r}})/D'(\sigma_k, \Sigma_{\hat{r}})$.

- Form the final denoised estimate of the latent matrix via

$$\hat{X} = \sum_{k=1}^r \hat{w}_k u_k v'_k. \quad (7.44)$$

For estimates of the MSE of this estimate, see [167, Eq. (17)].

Let S denote a $K \times L$ rectangular diagonal matrix with diagonal entries $s_1, \dots, s_{\min(K,L)}$. The two key expressions in this algorithm are

$$D(z, S) = \frac{1}{K} \text{trace} \left\{ z(z^2 I_K - SS')^{-1} \right\} \frac{1}{L} \text{trace} \left\{ z(z^2 I_L - S'S)^{-1} \right\} \quad (7.45)$$

and

$$\begin{aligned} D'(z, S) &= \frac{1}{K} \text{trace} \left\{ z(z^2 I - SS')^{-1} \right\} \frac{1}{L} \text{trace} \left\{ -2z^2(z^2 I - S'S)^{-2} + (z^2 I - S'S)^{-1} \right\} \\ &\quad + \frac{1}{L} \text{trace} \left\{ z(z^2 I - S'S)^{-1} \right\} \frac{1}{K} \text{trace} \left\{ -2z^2(z^2 I - SS')^{-2} + (z^2 I - SS')^{-1} \right\}. \end{aligned} \quad (7.46)$$

These equations are practical only if both M and N are sufficiently small. Here we adapt the approach to allow one of M or N (but not both) to be large. To evaluate $D(z, S)$, note that $D(z, S) = D(z, S')$ so, without loss of generality, assume $K \geq L$ (tall).

Then $S = \begin{bmatrix} S_L \\ \mathbf{0} \end{bmatrix}$ and, for $z \neq s_k$,

$$\begin{aligned} z^2 I - SS' &= z^2 I - \begin{bmatrix} S_L^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} z^2 I - S_L^2 & \mathbf{0} \\ \mathbf{0} & z^2 I_{K-L} \end{bmatrix} \\ \implies \text{trace} \left\{ z(z^2 I - SS')^{-1} \right\} &= \text{trace} \left\{ z \left[\begin{bmatrix} z^2 I - S_L^2 & \mathbf{0} \\ \mathbf{0} & z^2 I \end{bmatrix}^{-1} \right] \right\} \\ &= \frac{K-L}{z} + \sum_{k=1}^L \frac{z}{z^2 - s_k^2}. \end{aligned}$$

Similarly, as long as $z \neq s_k$,

$$\begin{aligned} \text{trace} \left\{ z(z^2 I - S'S)^{-1} \right\} &= \sum_{k=1}^L \frac{z}{z^2 - s_k^2}, \\ \implies D(z, S) &= \frac{1}{KL} \left(\frac{K-L}{z} + \sum_{k=1}^L \frac{z}{z^2 - s_k^2} \right) \left(\sum_{k=1}^L \frac{z}{z^2 - s_k^2} \right). \end{aligned}$$

A similar simplification is feasible for $D'(z, S)$ (Problem 7.8). Combining, one can implement OptShrink to calculate the optimal SVD component weights $\{\hat{w}_k\}$ when at most one of M or N is large.

7.6.2.1 Illustration of OptShrink's Robustness to Rank Estimate

Example 7.16 Figure 7.17 shows X , a 100×30 logo image (or 2D array or matrix) with $\text{rank}(X) = 4$, the noisy data $Y = X + Z$, and the denoised matrix \hat{X} for two rank estimates. Figure 7.18 shows that the error of \hat{X} for conventional low-rank approximation increases faster (due to over-fitting noise) than that of OptShrink. OptShrink finds the “threshold” for singular value shrinkage *adaptively* from the data, namely from the tail singular values, using random matrix theory.

To extend optimal singular value shrinkage to heteroscedastic noise, see [192].

Example 7.17 Figure 7.19 illustrates why singular value shrinkage is needed when $Y = X + Z$ where $X \in \mathbb{R}^{100 \times 50}$ is rank 5 with singular values 5, 4, 3, 2, 1 and the noise is zero mean with standard deviation $\sigma_0 = 0.1$. The figure shows histograms of $\sigma_k(Y)$ for 2000 realizations of Z . Noise causes a positive bias, $E[\sigma_k(Y)] > \sigma_k(X)$, as discussed in more detail in Chapter 12.

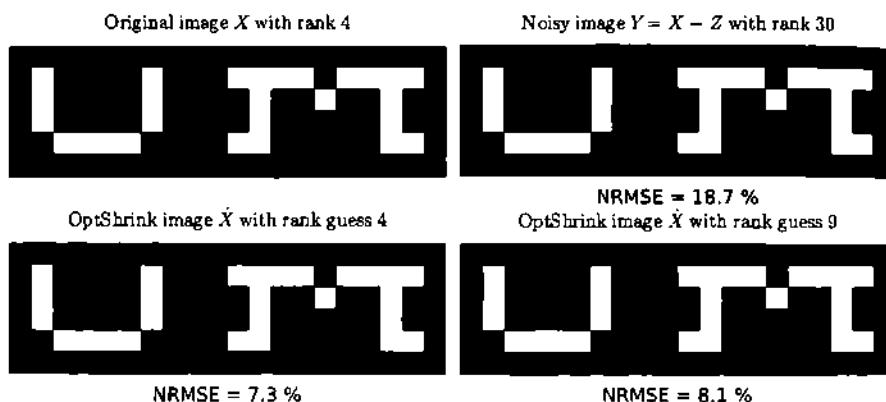


Figure 7.17 OptShrink demo results. The NRMSE increases only slightly when $\hat{r} = 9 \gg r = 4$.

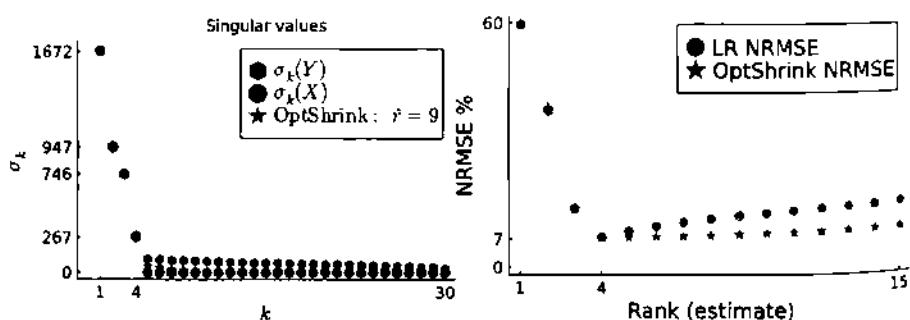


Figure 7.18 OptShrink demo plots.

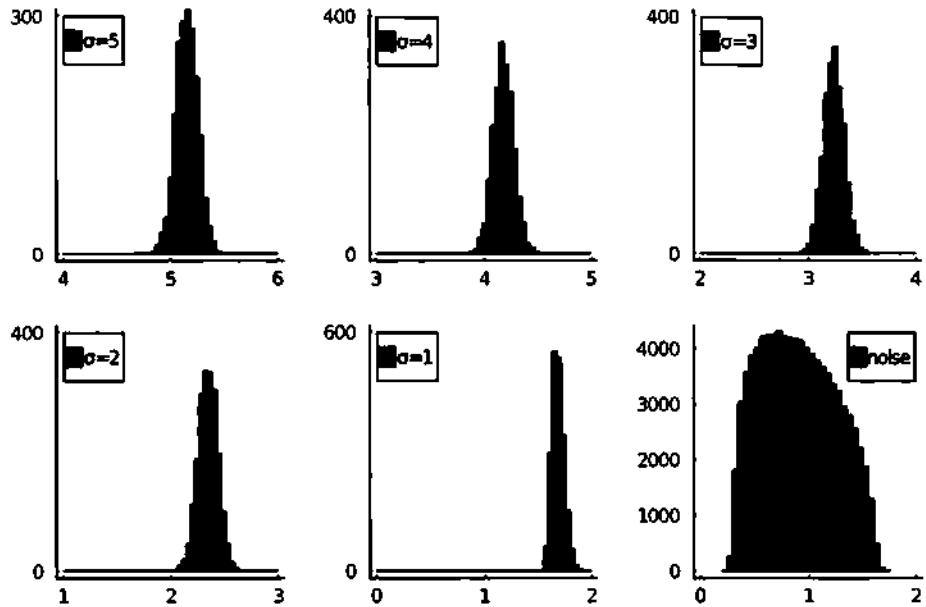


Figure 7.19 Singular value distributions for a signal+noise matrix $Y = X + Z$, where $\text{rank}(X) = 5$. The bottom right plot shows all the singular values of the noise-only matrix Z .

7.7

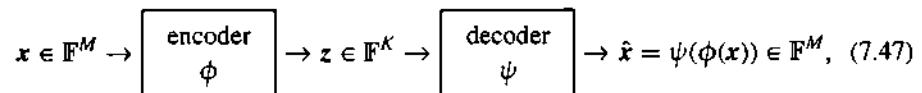
Related Methods: Autoencoders and PCA

This section discusses unsupervised dimensionality reduction methods that are closely related to low-rank matrix approximation [193].

7.7.1

Relation to Autoencoder with Linear Layers

An autoencoder is a type of artificial neural network (ANN) that is designed to perform dimensionality reduction [194]. An autoencoder consists of an encoder and a decoder:



where typically $K \ll M$, that is, the dimension of the code or latent variable z is much less than that of a data point x .

The simplest possible case is where the encoder and decoder are each linear, and hence each can be represented via a (learnable) matrix:

$$\begin{aligned} \phi(x) &= Ex, & E &\in \mathbb{F}^{K \times M} \text{ (wide),} \\ \psi(z) &= Dz, & D &\in \mathbb{F}^{M \times K} \text{ (tall).} \end{aligned} \quad (7.48)$$

In this case, the output is $\hat{x} = DEx$.

The natural approach to training this ANN given training data $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{F}^M$ for $N > K$ is

$$\begin{aligned}\hat{(\mathbf{D}, \mathbf{E})} &= \arg \min_{\mathbf{D} \in \mathbb{F}^{M \times K}, \mathbf{E} \in \mathbb{F}^{K \times M}} \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{D}\mathbf{E}\mathbf{x}_n\|_2^2 \\ &= \arg \min_{\mathbf{D} \in \mathbb{F}^{M \times K}, \mathbf{E} \in \mathbb{F}^{K \times M}} \|\mathbf{X} - \mathbf{D}\mathbf{E}\mathbf{X}\|_F^2, \quad \mathbf{X} \triangleq [\mathbf{x}_1 \ \cdots \ \mathbf{x}_N].\end{aligned}\quad (7.49)$$

The product $\mathbf{D}\mathbf{E}\mathbf{X}$ has at most rank K due to the dimensions of \mathbf{D} and \mathbf{E} . Choices for $\hat{\mathbf{D}}$ and $\hat{\mathbf{E}}$ such that $\hat{\mathbf{D}}\hat{\mathbf{E}}\mathbf{X} = \mathbf{U}_K \Sigma_K \mathbf{V}'_K$ provide an optimal rank- K approximation to \mathbf{X} , using an SVD $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}'$.

Based on matching the dimensions, a natural choice for $\hat{\mathbf{D}}$ looks to be $\hat{\mathbf{D}} = \mathbf{U}_K$, so we start with that choice and see if we can choose $\hat{\mathbf{E}}$ such that $\hat{\mathbf{E}}\mathbf{X} = \Sigma_K \mathbf{V}'_K$, that is, for which $\hat{\mathbf{E}}\mathbf{U}\Sigma\mathbf{V}' = \Sigma_K \mathbf{V}'_K$. Multiplying on the right by \mathbf{V}_K yields the goal

$$\hat{\mathbf{E}}\mathbf{U}\Sigma \begin{bmatrix} \mathbf{I}_K \\ \mathbf{0} \end{bmatrix} = \Sigma_K.$$

So evidently we want $\hat{\mathbf{E}}\mathbf{U} = [\mathbf{I}_K \ \mathbf{0}]$, which is achieved when $\hat{\mathbf{E}} = \mathbf{U}'_K$. Now we verify that a solution to (7.49) is

$$\hat{\mathbf{E}} = \mathbf{U}'_K, \quad \hat{\mathbf{D}} = \mathbf{U}_K, \quad (7.50)$$

where \mathbf{U}_K is the first K left singular vectors of \mathbf{U} . For that choice of \mathbf{E} and \mathbf{D} , and assuming $K \leq \min(M, N)$, the product becomes

$$\begin{aligned}\hat{\mathbf{D}}\hat{\mathbf{E}}\mathbf{X} &= \mathbf{U}_K \mathbf{U}'_K \underbrace{\mathbf{U}\Sigma\mathbf{V}'}_{\mathbf{X}} = \mathbf{U}_K \mathbf{U}'_K [\mathbf{U}_K \ \mathbf{U}_{:, (K+1):M}] \Sigma \mathbf{V}' \\ &= \mathbf{U}_K [\mathbf{I}_K \ \mathbf{0}_{K \times M}] \begin{bmatrix} \Sigma_K & \mathbf{0}_{K \times (N-K)} \\ \mathbf{0}_{(M-K) \times K} & \Sigma_{(M-K) \times (N-K)} \end{bmatrix} [\mathbf{V}_K \ \mathbf{V}_{:, (K+1):N}]' \\ &= \mathbf{U}_K \Sigma_K \mathbf{V}'_K = \hat{\mathbf{X}}_K,\end{aligned}$$

namely the optimal rank at most K approximation to \mathbf{X} .

So, performing low-rank approximation of training data is closely related to learning a (linear) autoencoder with a single hidden layer of reduced dimension. If the data dimension is too large to use an SVD, then one can use backpropagation instead [195, 196, 197].

Nonlinear autoencoders are generalizations using nonlinear operations such as $\phi(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $\sigma: \mathbb{F}^M \rightarrow \mathbb{F}^M$ is some nonlinear function, \mathbf{W} is an $M \times K$ matrix, and $\mathbf{b} \in \mathbb{F}^M$ is a bias vector, and one must learn \mathbf{W} and \mathbf{b} from training data [198]. See Problem 7.10 for the case where the encoder and decoder are affine instead of linear.

7.7.2 Relation to Principal Component Analysis

Up to this point we have focused on low-rank approximation of a matrix. Often, the low-dimensional subspaces $\text{span}(\mathbf{U}_K)$ and $\text{span}(\mathbf{V}_K)$ are more important to us than the

overall approximation $\hat{A}_K = U_K \Sigma_K V'_K$, because those subspaces provide the opportunity for dimensionality reduction. A classic unsupervised method for dimensionality reduction is principal component analysis (PCA). Here we derive PCA using two perspectives: a subspace learning perspective, and a variation maximization perspective. In both cases, we begin with a set of N training data vectors $x_1, \dots, x_N \in \mathbb{F}^M$, organized into an $M \times N$ data matrix $X \triangleq [x_1 \ \dots \ x_N]$.

The subspace learning perspective makes an affine modeling assumption:

$$x_n \approx \mu + Qz_n, z_n \in \mathbb{F}^K,$$

where $\mu \in \mathbb{F}^M$ allows the data to be centered away from $\mathbf{0}$, and where $Q \in \mathcal{V}_K(\mathbb{F}^M)$ is an $M \times K$ semiunitary matrix whose columns provide an orthonormal basis for the subspace $\mathcal{R}(Q)$. The z_n values are coefficients for vectors in that subspace. (In factor analysis, one also makes statistical assumptions about z_n .) The primary goal is to learn the basis Q , but we must also fit μ and $\{z_n\}$. A natural approach is the following least-squares formulation:

$$\hat{Q} = \arg \min_{Q \in \mathcal{V}_K(\mathbb{F}^M)} \min_{z_1, \dots, z_N \in \mathbb{F}^K} \min_{\mu \in \mathbb{F}^M} \sum_{n=1}^N \|\mu + Qz_n - x_n\|_2^2. \quad (7.51)$$

We start by minimizing with respect to z_n , which is simply an LLS problem, so by Section 5.5.2:

$$\hat{z}_n = \arg \min_{z_n \in \mathbb{F}^K} \|Qz_n - (x_n - \mu)\|_2^2 = Q^+(x_n - \mu) = Q'(x_n - \mu)$$

because Q is semiunitary. Substituting this solution into (7.51) yields the simplified minimization problem

$$\begin{aligned} \hat{Q} &= \arg \min_{Q \in \mathcal{V}_K(\mathbb{F}^M)} \min_{\mu \in \mathbb{F}^M} \sum_{n=1}^N \|\mu + QQ'(x_n - \mu) - x_n\|_2^2 \\ &= \arg \min_{Q \in \mathcal{V}_K(\mathbb{F}^M)} \min_{\mu \in \mathbb{F}^M} \sum_{n=1}^N \|P_Q^\perp(\mu - x_n)\|_2^2. \end{aligned} \quad (7.52)$$

Next, we minimize with respect to μ by zeroing the gradient, recalling (5.12):

$$\mathbf{0} = \sum_{n=1}^N (P_Q^\perp)'(P_Q^\perp)(\mu - x_n) \implies NP_Q^\perp \hat{\mu} = P_Q^\perp \sum_{n=1}^N x_n.$$

These normal equations have multiple solutions (when $K < M$) because translating the origin of an affine subspace along the subspace has no effect. The standard solution choice is simply the sample average of the data: $\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n$. Substituting this solution into (7.52) yields a simpler subspace learning problem,

$$\hat{Q} = \arg \min_{Q \in \mathcal{V}_K(\mathbb{F}^M)} \sum_{n=1}^N \|P_Q^\perp y_n\|_2^2, \quad (7.53)$$

where $y_n \triangleq x_n - \mu$ denotes the de-meaned data. To simplify further:

$$\sum_{n=1}^N \|P_Q^\perp y_n\|_2^2 = \sum_{n=1}^N \|(I - QQ')y_n\|_2^2 = \|Y - QQ'Y\|_F^2,$$

where $\mathbf{Y} \triangleq [\mathbf{y}_1 \quad \cdots \quad \mathbf{y}_N] \in \mathbb{F}^{M \times N}$. Thus, (7.53) becomes

$$\hat{\mathbf{Q}} = \arg \min_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \|\mathbf{Y} - \mathbf{Q}\mathbf{Q}'\mathbf{Y}\|_F^2. \quad (7.54)$$

This problem is a special case of (7.49) where we require $\mathbf{D} = \mathbf{E}'$ and $\mathbf{D}, \mathbf{E}' \in \mathcal{V}_K(\mathbb{F}^M)$. There, we found that the solution in (7.50) is simply \mathbf{U}_K , the first K left singular vectors of the data (here \mathbf{Y}), so we conclude that $\hat{\mathbf{Q}} = \mathbf{U}_K$.

An alternative approach is to expand the norm in (7.53):

$$\begin{aligned} \|\mathbf{P}_{\hat{\mathbf{Q}}}^\perp \mathbf{y}_n\|_2^2 &= (\mathbf{P}_{\hat{\mathbf{Q}}}^\perp \mathbf{y}_n)' (\mathbf{P}_{\hat{\mathbf{Q}}}^\perp \mathbf{y}_n) = \mathbf{y}_n' \mathbf{P}_{\hat{\mathbf{Q}}}^\perp \mathbf{y}_n \\ &= \mathbf{y}_n' (\mathbf{I} - \mathbf{P}_{\hat{\mathbf{Q}}}) \mathbf{y}_n = \|\mathbf{y}_n\|_2^2 - \text{trace}\{\mathbf{Q}' \mathbf{y}_n \mathbf{y}_n' \mathbf{Q}\}. \end{aligned}$$

The first term is independent of \mathbf{Q} , so (7.53) simplifies to

$$\hat{\mathbf{Q}} = \arg \min_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \sum_{n=1}^N (-\text{trace}\{\mathbf{Q}' \mathbf{y}_n \mathbf{y}_n' \mathbf{Q}\}) = \arg \max_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \text{trace}\{\mathbf{Q}' \mathbf{Y} \mathbf{Y}' \mathbf{Q}\}, \quad (7.55)$$

where $\mathbf{Y} \mathbf{Y}' = \sum_{n=1}^N \mathbf{y}_n \mathbf{y}_n' \in \mathbb{F}^{M \times M}$ is the sample covariance matrix of the de-meansed data \mathbf{Y} . This trace maximization problem must have the same solution, leading to the useful trace maximization property

$$\arg \max_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \text{trace}\{\mathbf{Q}' \mathbf{Y} \mathbf{Y}' \mathbf{Q}\} = \arg \min_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \|\mathbf{Y} - \mathbf{Q}\mathbf{Q}'\mathbf{Y}\|_F^2. \quad (7.56)$$

In summary, we conclude that, from the SVD $\mathbf{Y} = \mathbf{U}\Sigma\mathbf{V}'$ of the de-meansed data, the subspace estimate is

$$\hat{\mathbf{Q}} = \arg \min_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \|\mathbf{Y} - \mathbf{Q}\mathbf{Q}'\mathbf{Y}\|_F^2 = \arg \max_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \text{trace}\{\mathbf{Q}' \mathbf{Y} \mathbf{Y}' \mathbf{Q}\} = \mathbf{U}_K. \quad (7.57)$$

We have now solved (7.51) for $\{\mathbf{z}_n\}$, μ , and \mathbf{Q} . The fitting error (lowest cost) is similar to (7.3):

$$\begin{aligned} \min_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M)} \min_{\mathbf{z}_1, \dots, \mathbf{z}_N \in \mathbb{F}^K} \min_{\mu \in \mathbb{F}^M} \sum_{n=1}^N \|\mu + \mathbf{Q}\mathbf{z}_n - \mathbf{x}_n\|_2^2 &= \sum_{n=1}^N \|\hat{\mu} + \hat{\mathbf{Q}}\hat{\mathbf{z}}_n - \mathbf{x}_n\|_2^2 \\ &= \|\mathbf{P}_{\hat{\mathbf{Q}}}^\perp \mathbf{Y}\|_F^2 = \sum_{k=K+1}^M \sigma_k^2(\mathbf{Y}). \end{aligned} \quad (7.58)$$

7.7.2.1 Maximum Variance Projections

The variance maximizing perspective on PCA focuses on learning a linear transform $\mathbf{T} \in \mathbb{F}^{N \times K}$ (having orthonormal columns) that maximizes variance when applied to the data via $\mathbf{T}'\mathbf{X}$. The matrix \mathbf{T}' will be wide ($K \ll N$), so this product provides dimensionality reduction. PCA first subtracts from each the mean of all training vectors to form a de-meansed data matrix:

$$\mathbf{X}_1 \triangleq [\mathbf{x}_1 - \hat{\mu} \quad \cdots \quad \mathbf{x}_N - \hat{\mu}] = \mathbf{X} - \mathbf{X} \frac{1}{N} \mathbf{1}_N \mathbf{1}_N', \quad (7.59)$$

where X and $\hat{\mu}$ were defined above. Note that $X_1 \mathbf{1}_N = \mathbf{0}_M$. Now PCA finds the first column of T by seeking the (unit norm) linear combination of rows of X (elements of x) that maximizes the (empirical) variance as follows:

$$\begin{aligned} T_{:,1} &\triangleq \arg \max_{\mathbf{u}: \|\mathbf{u}\|=1} \frac{1}{N} \sum_{n=1}^N |\mathbf{u}' X_1[:, n]|^2 = \arg \max_{\mathbf{u}: \|\mathbf{u}\|=1} \|(\mathbf{u}' X_1)'\|_2 \\ &= \arg \max_{\mathbf{u}: \|\mathbf{u}\|=1} \|X_1' \mathbf{u}\|_2 = \mathbf{u}_1, \end{aligned}$$

where an SVD of X_1 is $X_1 = U \Sigma V'$. Another way of writing this uses a Rayleigh quotient [5, §6.4]:

$$\arg \max_{\mathbf{u} \in \mathbb{F}^M \setminus \{0\}} \frac{\mathbf{u}' X_1 X_1' \mathbf{u}}{\|\mathbf{u}\|_2^2}, \quad (7.60)$$

where we recognize $X_1 X_1' / N \in \mathbb{F}^{M \times M}$ as an empirical covariance matrix.

Having found that \mathbf{u}_1 is the (unit norm) linear combination that maximizes the (empirical) variance, we can now remove the component along that direction and then seek another linear combination that maximizes the variance of what is left. To remove the component along \mathbf{u}_1 we construct a new matrix as follows:

$$X_2 \triangleq P_{R(\mathbf{u}_1)}^\perp X_1 = (\mathbf{I} - P_{R(\mathbf{u}_1)}) X_1 = (\mathbf{I} - \mathbf{u}_1 \mathbf{u}_1') X_1. \quad (7.61)$$

Note that $X_2 \mathbf{1} = \mathbf{0}$ so we still have a zero-mean array. Now PCA finds the second column of T as follows:

$$T_{:,2} \triangleq \arg \max_{\mathbf{u}: \|\mathbf{u}\|=1} \|(\mathbf{u}' X_2)'\|_2 = \arg \max_{\mathbf{u}: \|\mathbf{u}\|=1} \|X_1' (\mathbf{I} - \mathbf{u}_1 \mathbf{u}_1') \mathbf{u}\|_2 = \mathbf{u}_2.$$

To see why \mathbf{u}_2 is the solution here, use an SVD of X_1 to write

$$X_2 = (\mathbf{I} - \mathbf{u}_1 \mathbf{u}_1') X_1 = (\mathbf{I} - \mathbf{u}_1 \mathbf{u}_1') U \Sigma V' = U \text{Diag}(0, \sigma_2, \sigma_3, \dots) V',$$

which is almost an SVD of X_2 except for a permutation. Clearly the nonzero singular values of X_2 are $(\sigma_2, \sigma_3, \dots, \sigma_r)$, the spectral norm of X_2 is σ_2 , and the corresponding left singular vector, which is its principal left singular vector, is \mathbf{u}_2 . One can continue this reasoning to show that $T = U_K$, the first K columns of U , provides a linear transform T having orthonormal columns that maximizes the resulting variances of the scores $T' X_1$. We say that U'_K represents the linear combinations that “explain the most variance” in the data.

Using the same SVD of X_1 , we have $U'_K X_1 = U'_K U \Sigma V' = \Sigma_K V'_K$, where V_K denotes the first K columns of V . Thus, instead of storing all MN elements of X_1 it suffices to store the MK and NK elements of U_K and V_K (and the K diagonal elements of Σ_K) reflecting dimensionality reduction.

7.7.2.2 Units and Normalization

The requirement that T have orthonormal columns means that the elements of T must be unitless. When T is unitless, then the matrix product $T' X$ is valid only if the rows of X have the same units (or are unitless); cf. Section 2.4.5.2. When the features in x_n have different units, it is essential to normalize the rows of X as a preprocessing

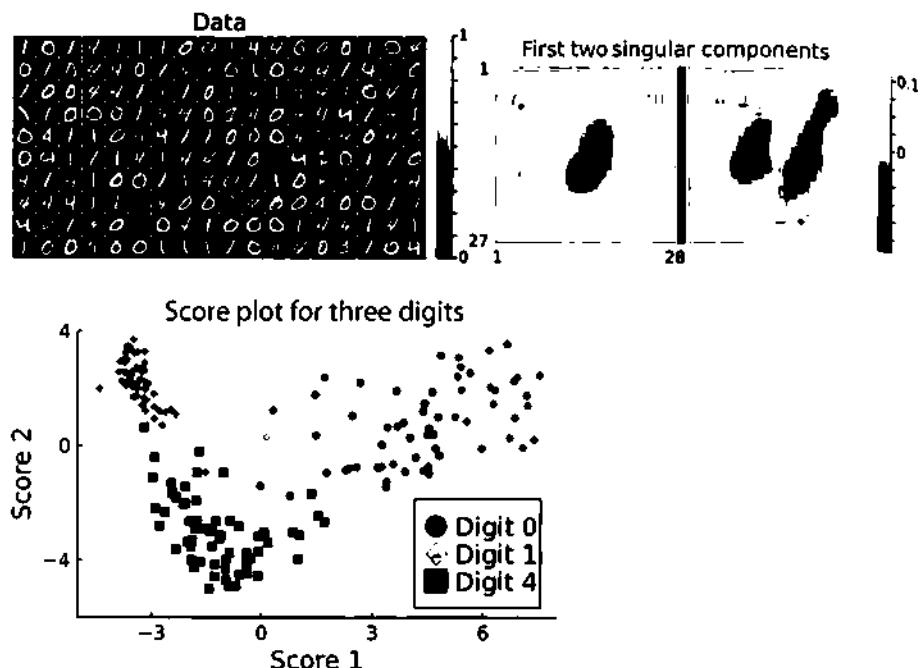


Figure 7.20 Results from PCA demo.

step, typically by dividing each row by its standard deviation, so that the product $T'X$ is valid.

Demo 7.5 illustrates applying PCA to images of hand-written digits. Figure 7.20 shows 60 images from each of three handwritten digits (0, 1, 4) that are reduced to $K = 2$ dimensions. To apply PCA here, we vec each 28×27 image into a vector in \mathbb{R}^{756} , and then arrange those vectors into a 756×180 matrix. After computing the SVD, we reshape each of the first two left singular vectors into a 28×27 matrix for display in Fig. 7.20. This approach of forming a matrix where each column is a vectorized version of one image in a set of images is very common for SVD methods.

7.7.2.3 PCA Extensions

Many nonlinear dimensionality reduction methods exist that generalize PCA, including kernel PCA [199], nonlinear PCA [200], Laplacian eigenmaps [201, 202] (see Section 8.9.6), t -distributed stochastic neighbor embedding (t -SNE) [203], locally linear embedding [204], and PaCMAP [205].

Applications of PCA methods are innumerable, including face recognition using eigenfaces [49] and analysis of macromolecules using cryo-EM [206].

7.8

Subspace Learning for Classification

This section discusses an application of low-rank matrix approximation to subspace learning. In this setting, the low-dimensional subspace $\mathcal{R}(U_K)$ and its basis U_K are more important to us than the overall approximation $\hat{A}_K = U_K \Sigma_K V_K'$.

In supervised classification, we are given N labeled training data samples $x_{j,1}, \dots, x_{j,N}$ for each of J classes of objects, where each feature vector $x_{j,n} \in \mathbb{F}^M$. We would like to learn something about the nature of each class so that later, when we get a new sample vector x_0 , we can assess which class it is most like.

One basic approach to classification is the K -nearest neighbors method. When $K = 1$, this method simply chooses the class of the nearest neighbor to a test point x_0 among the training samples. Figure 7.21 illustrates this process that is expressed mathematically as

$$\hat{j} = \arg \min_{j \in \{1, \dots, J\}} \min_{n=1, \dots, N} \|x_0 - x_{j,n}\|. \quad (7.62)$$

This basic approach requires JN comparisons, so complexity grows with training data size. Also, its performance may not improve as N increases when the marginal distributions overlap.

Instead, we often try to learn a model from the training data, and then use the fit to the model as the basis for classification. Here we focus on learning a subspace model for each class. Then, to classify a test point x_0 we simply compute the distance of x_0 to each of the J subspaces and see which is closest, as illustrated in (4.57) and Fig. 4.12. To simplify notation, we drop the class subscript j and focus on learning a subspace for a set of samples for one class type, x_1, \dots, x_N , where each $x_n \in \mathbb{F}^M$.

Saying that all the x_n vectors lie in a subspace of dimension K means that there is some $M \times K$ matrix Q having orthonormal columns such that $x_n \in \mathcal{R}(Q)$. The columns of Q form an orthonormal basis for the subspace. In practice the data will only lie approximately in a subspace, so $x_n \approx Qz_n$ for some coefficient vector $z_n \in \mathbb{F}^K$. We want to learn a subspace basis matrix Q from the training data $X = [x_1 \ \cdots \ x_N]$, an $M \times N$ matrix. Writing $x_n \approx Qz_n$ in matrix form:

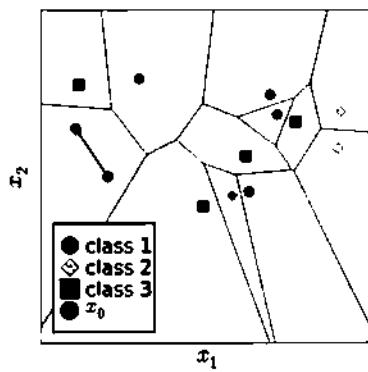


Figure 7.21 Decision boundaries for nearest-neighbor classification.

$$\underbrace{\mathbf{X}}_{M \times N} \approx \underbrace{\mathbf{Q}}_{M \times K} \underbrace{\mathbf{Z}}_{K \times N}, \quad \text{where } \mathbf{Z} \triangleq [\mathbf{z}_1 \ \cdots \ \mathbf{z}_N]. \quad (7.63)$$

To find \mathbf{Q} and \mathbf{Z} we could pursue the following optimization problem on the Stiefel manifold of (4.45):

$$\hat{\mathbf{Q}} = \arg \min_{\mathbf{Q} \in \mathcal{V}_K(\mathbb{F}^M), \mathbf{Z} \in \mathbb{F}^{K \times N}} \|\mathbf{X} - \mathbf{Q}\mathbf{Z}\|_{\text{F}}. \quad (7.64)$$

In this setting, typically $K \ll \min(M, N)$ so the product $\mathbf{Q}\mathbf{Z}$ is a matrix with (at most) rank K .

Thus, this problem is essentially a low-rank approximation problem, except that here we really care only about the subspace basis \mathbf{Q} and not the coefficients \mathbf{Z} (nor their product). The low-rank solution is

$$\hat{\mathbf{X}}_K = \sum_{k=1}^K \sigma_k \mathbf{u}_k \mathbf{v}'_k = \mathbf{U}_K \boldsymbol{\Sigma}_K \mathbf{V}'_K = \underbrace{\mathbf{U}_K}_{\hat{\mathbf{Q}}} \underbrace{\boldsymbol{\Sigma}_K}_{\mathbf{Z}} \mathbf{V}'_K. \quad (7.65)$$

So, to learn a subspace basis $\hat{\mathbf{Q}}$ we simply use the first K left singular vectors of the training data \mathbf{X} .

After learning an orthonormal basis \mathbf{Q}_j for a subspace approximation for each of the J classes, we classify a test point \mathbf{x}_0 by finding the nearest subspace:

$$\begin{aligned} j &= \arg \min_{j \in \{1, \dots, J\}} \min_{\mathbf{z} \in \mathbb{F}^k} \|\mathbf{x}_0 - \mathbf{Q}_j \mathbf{z}\|_2^2 = \arg \min_{j \in \{1, \dots, J\}} f_j(\mathbf{x}_0), \\ f_j(\mathbf{x}_0) &\triangleq \underbrace{\|\mathbf{x}_0 - \mathbf{Q}_j (\mathbf{Q}_j' \mathbf{x}_0)\|_2^2}_{(a)} = \underbrace{\|\mathbf{x}_0 - \mathbf{P}_j \mathbf{x}_0\|_2^2}_{(b)} = \underbrace{\|\mathbf{P}_j^\perp \mathbf{x}_0\|_2^2}_{(c)} = \underbrace{\|\mathbf{x}_0\|_2^2 - \|\mathbf{Q}_j' \mathbf{x}_0\|_2^2}_{(d)}, \end{aligned} \quad (7.66)$$

where $\mathbf{P}_j \triangleq \mathbf{Q}_j \mathbf{Q}_j'$ performs the orthogonal projection onto the subspace $\mathcal{R}(\mathbf{Q}_j)$, and $\mathbf{P}_j^\perp \triangleq \mathbf{I} - \mathbf{P}_j$.

Q7.18 Which form above is the most efficient (per sample) in the usual case where $K \ll \min(M, N)$?

- A: (a) B: (b) C: (c) D: (d)

Q7.19 For that efficient form, how many multiplies are needed per test sample?

- A: $JM(2K + 1)$ B: $JM(M + 1)$ C: $J2MK$ D: JM^2 E: None

Learning subspaces via (7.64) is optimal in the Frobenius norm sense of low-rank approximation, but is *not* necessarily optimal for the purposes of classification. An extension called supervised PCA aims to find subspaces that are good for both approximation and classification [207, 208, 209]. See [210] for a special issue on subspace learning and a generalization called submanifold learning.

7.8.1

Subspace Clustering

The subspace learning approach described above, especially (7.64), assumes that the training data is labeled by class type, so it is supervised learning. There are also unsupervised subspace learning methods that perform subspace clustering as part of the subspace learning process [163, 211, 212].

Given unlabeled training data $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{F}^M$, in a J -subspace clustering approach we want to learn a set of J orthogonal bases $\mathbf{Q}_1, \dots, \mathbf{Q}_J \in \mathcal{V}_K(\mathbb{F}^M) \subset \mathbb{F}^{M \times K}$ such that each training point \mathbf{x}_n is close to one of the J subspaces $\{\mathcal{R}(\mathbf{Q}_j)\}$. In other words, the model here is that all of the data points lie on (or near) a union of subspaces $\cup_{j=1}^J \mathcal{R}(\mathbf{Q}_j)$. One possible optimization formulation for learning the subspace bases is

$$\arg \min_{\mathbf{Q}_1, \dots, \mathbf{Q}_J \in \mathcal{V}_K(\mathbb{F}^M)} \min_{j_1, \dots, j_N \in \{1, \dots, J\}} \sum_{n=1}^N \|\mathbf{x}_n - \mathbf{P}_{\mathbf{Q}_{j_n}} \mathbf{x}_n\|_2^2. \quad (7.67)$$

An alternative way of writing this is to use weights, where w_{nj} indicates whether the n th training point is assigned to the j class:

$$\begin{aligned} & \arg \min_{\mathbf{Q}_1, \dots, \mathbf{Q}_J \in \mathcal{V}_K(\mathbb{F}^M)} \min_{\mathbf{W} \in \mathcal{W}} \sum_{n=1}^N \sum_{j=1}^J w_{nj} \|\mathbf{x}_n - \mathbf{P}_{\mathbf{Q}_j} \mathbf{x}_n\|_2^2, \\ & \mathcal{W} \triangleq \left\{ \mathbf{W} \in \mathbb{R}^{N \times J} : w_{nj} \in \{0, 1\}, \sum_{j=1}^J w_{nj} = 1 \right\}. \end{aligned}$$

These are challenging discrete optimization problems called integer programming problems. One can alternate between updating the basis $\{\mathbf{Q}_j\}$ and the cluster assignments $\{j_n\}$ or \mathbf{W} . An alternative formulation called sparse subspace clustering (SSC) relaxes the discrete problem [211]. A related concept for unlabeled data is called generalized PCA (GPCA) [213].

7.9

Subspace Tracking and Streaming PCA

The methods described thus far in this chapter are suitable when an entire data matrix \mathbf{X} is well modeled as being low rank. This type of “global low rank” model can require a lot of memory when SVD-based methods are used, and is poorly suited to applications with time-varying data. To reduce memory and/or to accommodate time-varying data, one can employ adaptive / incremental / online / recursive / streaming / tracking / updating methods [214, 215]. (The terminology varies in the literature.)

The specific algorithms used depend in part on whether one’s goal is to perform computation incrementally or to track time-varying subspaces, in analogy with the RLS methods in Section 5.10. We summarize here one example of each type. For a thorough survey, see [215].

7.9.1 Incremental SVD

Suppose that data vectors $\mathbf{x}_1, \mathbf{x}_2, \dots$ arrive sequentially. The incremental SVD (ISVD) [216] provides a method to update an SVD as each new data vector arrives. Let $\mathbf{X}_n \triangleq [\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_n] \in \mathbb{F}^{d \times n}$ denote the data matrix at time n , for $n \ll d$ (high-dimensional data). Suppose we previously computed the SVD of the data matrix at time $n-1$, $\mathbf{X}_{n-1} = \mathbf{U}_{n-1} \Sigma_{n-1} \mathbf{V}'_{n-1}$, where $\mathbf{U}_{n-1} \in \mathbb{F}^{d \times d}$, $\Sigma_{n-1} \in \mathbb{F}^{d \times (n-1)}$, and $\mathbf{V}'_{n-1} \in \mathbb{F}^{(n-1) \times (n-1)}$. After the n th data point arrives, we want to “update” the SVD by finding the SVD of the new data matrix that has one additional column in it:

$$\mathbf{X}_n = [\mathbf{X}_{n-1} \ \mathbf{x}_n] = \mathbf{U}_n \Sigma_n \mathbf{V}'_n. \quad (7.68)$$

The data covariance involves a rank-1 update:

$$\mathbf{X}_n \mathbf{X}'_n = [\mathbf{X}_{n-1} \ \mathbf{x}_n] [\mathbf{X}_{n-1} \ \mathbf{x}_n]' = \mathbf{X}_{n-1} \mathbf{X}'_{n-1} + \mathbf{x}_n \mathbf{x}'_n. \quad (7.69)$$

Furthermore, defining $\mathbf{D}_{n-1} \triangleq \Sigma_{n-1} \Sigma'_{n-1} \in \mathbb{F}^{d \times d}$, we have

$$\mathbf{U}'_{n-1} \mathbf{X}_n \mathbf{X}'_n \mathbf{U}_{n-1} = \mathbf{D}_{n-1} + \mathbf{z}_n \mathbf{z}'_n, \quad \mathbf{z}_n \triangleq \mathbf{U}'_{n-1} \mathbf{x}_n \in \mathbb{F}^d, \quad (7.70)$$

so, by Problem 3.1 and [217], the squared singular values of \mathbf{X}_n are the (suitably ordered) roots of the rational function

$$1 + \sum_{i=1}^d \frac{|[\mathbf{z}_n]_i|^2}{d_i - \lambda} \quad (7.71)$$

with respect to λ , where $\mathbf{D}_{n-1} = \text{Diag}\{d_i\}$. To update the left singular vectors, that is, to find $\mathbf{U}_n = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_d]$, use the equality $\mathbf{X}_n \mathbf{X}'_n \mathbf{u}_i = \lambda_i \mathbf{u}_i$, where λ_i is the square of the i th singular value of \mathbf{X}_n . Thus,

$$(\mathbf{U}_{n-1} \mathbf{D}_{n-1} \mathbf{U}'_{n-1} + \mathbf{x}_n \mathbf{x}'_n) \mathbf{u}_i = \lambda_i \mathbf{u}_i,$$

or, equivalently, $(\mathbf{D}_{n-1} + \mathbf{z}_n \mathbf{z}'_n) \mathbf{b}_i = \lambda_i \mathbf{b}_i$, where $\mathbf{b}_i \triangleq \mathbf{U}'_{n-1} \mathbf{u}_i$. Rearranging [214] yields $(\mathbf{D}_{n-1} - \lambda_i \mathbf{I}_d) \mathbf{b}_i = -(\mathbf{z}'_n \mathbf{b}_i) \mathbf{z}_n$, so $\mathbf{b}_i \propto (\mathbf{D}_{n-1} - \lambda_i \mathbf{I}_d)^{-1} \mathbf{z}_n$. Expressing in terms of \mathbf{u}_i and normalizing yields

$$\mathbf{u}_i = \frac{\mathbf{U}_{n-1} (\mathbf{D}_{n-1} - \lambda_i \mathbf{I}_d)^{-1} \mathbf{z}_n}{\|(\mathbf{D}_{n-1} - \lambda_i \mathbf{I}_d)^{-1} \mathbf{z}_n\|_2}, \quad i = 1, \dots, d. \quad (7.72)$$

For further refinements that reduce computation and accommodate missing data, see [215, 218].

Explore 7.14 After computing $\{\mathbf{u}_i\}$ via (7.72), how would you compute the right singular vectors (if needed)?

7.9.2 Streaming PCA

Streaming PCA methods are designed for cases where we believe each \mathbf{x}_n lies in (or near) a K -dimensional subspace, where that subspace varies (somewhat slowly) with time instead of being static. The goal is to estimate an orthonormal subspace basis

$U_n \in \mathcal{V}_K(\mathbb{F}^d)$ for each time point n , from the data acquired up until that point, that is, from X_n , where the U_n should evolve slowly. One must initialize the process with some $U_0 \in \mathcal{V}_K(\mathbb{F}^d)$, for example by applying the Gram–Schmidt process to the first K observations. One popular approach is Oja’s method [219] that updates the subspace basis recursively as follows:

$$U_n = \mathcal{P}_{\mathcal{V}_K(\mathbb{F}^d)}(U_{n-1} + \eta_n x_n x_n' U_{n-1}), \quad (7.73)$$

where $\mathcal{P}_{\mathcal{V}_K(\mathbb{F}^d)}(Y)$ denotes the projection of the matrix Y onto the Stiefel manifold $\mathcal{V}_K(\mathbb{F}^d)$. The step-size parameter(s) $\{\eta_n\}$ control how rapidly the subspace bases can change with time. See [220] for statistical performance analysis and see [215] for numerical examples with code.

Explore 7.15 Relate the units of the step size η_n to the units of the data vectors $\{x_n\}$.

7.10 Summary

This chapter described several methods for computing a low-rank approximation to a matrix, all of which use an SVD. It also discussed methods for choosing the rank or regularization parameter. It applied the methods to sensor localization (multidimensional scaling) and to subspace learning, which is useful for classification problems.

There are two main perspectives in this chapter. First, we started by taking an arbitrary matrix A and finding a low-rank approximation $\hat{A}_K \approx A$. This perspective has applications in data compression and in dimensionality reduction. In these applications, we want K to be small, but we also want the approximation error $\|\hat{A}_K - A\|_F$ to be small, and there is a trade-off between the two desires as we vary K .

Second, we then considered matrix denoising applications where we are given a noisy matrix Y that we model as $Y = X + Z$, where we think X is a low-rank matrix. Then we process Y to make an estimate \hat{X} of the latent matrix X . In this setting, we want the estimation error $\|\hat{X} - X\|_F$ to be small, and we care less about the approximation error $\|\hat{X} - Y\|_F$.

All the methods in this chapter require an SVD of an $M \times N$ matrix, which is impractical if both M and N are very large. Extensions for “big data” are an active research area, for example using sketching [221, 222], streaming/online methods [219, 220, 223], and nonconvex optimization [224] using, for example, the Burer–Monteiro factorization [225, 226, 227]; see Section 10.4.9.

Solutions to Explorations

Explore 7.1 $P_{U_K} A = U_K U_K' A = U_K U_K' U \Sigma V' = U_K [I \ 0] \Sigma V' = U_K \Sigma_K V'_K = \hat{A}_K$.

Explore 7.2 As a reminder that the output is a vector, not a matrix.

Explore 7.3 It avoids an unnecessary transpose operation.

Explore 7.4 The blue dots do not lie on a line, so $\text{rank}(A) = 2$.

Explore 7.5 $\sum_{k=K+1}^r \sigma_k$.

Explore 7.6 $K = 5$, because A has five nonzero elements.

Explore 7.7 $K = 2$, because A has rank 2.

Explore 7.8 This operator maps $M \times N$ matrices into $K \times K$ matrices, so it cannot be a projection operation.

Explore 7.9 Only if the two rows of C are linearly dependent; that is, if $y = \alpha x$ for some $\alpha \in \mathbb{R}$, or, in other words, if $y_j = \alpha x_j$ for all j , which is the equation for points along a line through the origin with slope α .

Explore 7.10 The three vertices of an equilateral triangle.

Explore 7.11 Distortion in location estimates. One should consider other dimensionality reduction methods in such situations.

Explore 7.12 $P^\perp S P^\perp$ would be $\text{real}\{C'C\}$, because $\langle c_i, c_j \rangle$ in (7.14) becomes $\text{real}\{\langle c_i, c_j \rangle\}$ per (2.32).

Explore 7.13 In this linear case where $y = x + \epsilon$ and $\epsilon \sim N(0, \sigma_0^2 I)$,

$$\begin{aligned} \text{SURE}(\beta) &= \|UU'y - y\|_2^2 - d\sigma_0^2 + 2\sigma_0^2 \text{trace}\{UU'\} = \|P_U^\perp y\|_2^2 - (d - 2\beta)\sigma_0^2, \\ E[\text{SURE}(\beta)] &= \text{trace}\{P_U^\perp(x' + \sigma_0^2 I_d)\} - (d - 2\beta)\sigma_0^2 \\ &= x'P_U^\perp x + \sigma_0^2(d - \beta) - (d - 2\beta)\sigma_0^2 = x'P_U^\perp x + \sigma_0^2\beta, \\ \text{MSE}(\beta) &= E[\|UU'y - x\|_2^2] = E[\|P_U\epsilon - P_U^\perp x\|_2^2] = x'P_U^\perp x + E[\|P_U\epsilon\|_2^2] \\ &= x'P_U^\perp x + \text{trace}\{P_U E[\epsilon\epsilon']\} = x'P_U^\perp x + \sigma_0^2\beta. \end{aligned}$$

Explore 7.14 Because $X_n' u_i = \sigma_i v_i$, we simply take $v_i = X_n' u_i / \|X_n' u_i\|_2$ for $i = 1, \dots, r$.

Explore 7.15 The formula (7.73) for Oja's method has unit balance only when all elements of x_n have the same units, and in that case the units of η_n must be the inverse of the square of the units of x_n . Thus, normalizing the data seems desirable.

Problems

Problem 7.1 Determine how to find a low-rank matrix \hat{X} that best approximates a set of matrices $A_1, \dots, A_L \in \mathbb{R}^{M \times N}$ as follows:

$$\hat{X} = \arg \min_{X \in \mathbb{R}^{M \times N} : \text{rank}(X) \leq K} \frac{1}{L} \sum_{l=1}^L \|X - A_l\|_F^2.$$

Problem 7.2 Apply the multidimensional scaling (MDS) algorithm to the distance matrix $D = \begin{bmatrix} 0 & a & 2a \\ a & 0 & a \\ 2a & a & 0 \end{bmatrix}$. Do the work by hand – no JULIA. (You may use JULIA to check your answer.)

Hint: Look for a rank-1 outer product.

Plot (by hand) the resulting coordinates in 2D. Use $d = 2$ throughout the problem. Verify that your answer makes sense in light of the original distance matrix D .

Problem 7.3 Generalize the proximal operator (7.24) to complex numbers by showing that

$$\arg \min_{z \in \mathbb{C}} \frac{1}{2} |s - z|^2 + \beta |z| = [|z| - \beta]_+ e^{i \angle z}.$$

More generally, if $f: \mathbb{C} \mapsto \mathbb{R}$ has the property that $f(z) = f(|z|)$, that is, $f(z) = g(|z|)$, where $g: [0, \infty) \mapsto \mathbb{R}$, show that $\text{prox}_f(v) = \text{prox}_g(|v|) \text{sign}(v)$.

Problem 7.4 Consider the following regularized low-rank approximation method:

$$\hat{X} = \arg \min_{X \in \mathbb{R}^{M \times N}} \frac{1}{2} \|Y - X\|_F^2 + \beta \|X\|_*.$$

Determine an expression for the approximation error $\|\hat{X} - Y\|_F$. Simplify as much as possible.

Problem 7.5 Section 7.5 discusses the (convex) regularized low-rank approximation problem

$$\hat{X} = \arg \min_{X \in \mathbb{C}^{M \times N}} \frac{1}{2} \|Y - X\|_F^2 + \beta \|X\|_*.$$

Determine the solution when we replace the Frobenius norm with the spectral norm:

$$\hat{X} = \arg \min_{X \in \mathbb{C}^{M \times N}} \frac{1}{2} \|Y - X\|_2^2 + \beta \|X\|_*.$$

It suffices to solve it for the case where $\text{rank}(Y) = 1$.

Problem 7.6 For $Y \in \mathbb{R}^{M \times N}$, determine (analytically) the solution of the regularized low-rank approximation method:

$$\hat{X} = \arg \min_{X \in \mathbb{R}^{M \times N}} \|Y - X\|_* + \beta \frac{1}{2} \|X\|_F^2.$$

Think about your solution and consider whether it seems to be a good method for low-rank approximation.

Problem 7.7 Define $\mathbb{R}_+ \triangleq [0, \infty)$. A symmetric gauge function $\phi(x)$ is a mapping from \mathbb{R}^n into \mathbb{R}_+ that satisfies the following four properties for all $x, y \in \mathbb{R}^n$:

- $\phi(x) > 0$ if $x \neq 0$ (positivity).
- $\phi(\alpha x) = |\alpha| \phi(x)$ for all $\alpha \in \mathbb{R}$ (homogeneity).
- $\phi(x+y) \leq \phi(x) + \phi(y)$ (triangle inequality).
- $\phi(s_1 x_{[1]}, \dots, s_n x_{[n]}) = \phi(x)$ for all $s_k = \pm 1$ and for any permutation $(x_{[1]}, \dots, x_{[n]})$ of the elements of x (symmetry).

Some examples are:

- $\phi(x) = \|x\|_1$;
- $\phi(x) = \max_i |x_i| = \|x\|_\infty$;
- $\phi(x) = 7|x_{(1)}| + 5|x_{(2)}|$, where $x_{(1)}$ and $x_{(2)}$ denote the first- and second-largest elements of x in magnitude.

Such symmetric gauge functions are at the heart of many matrix norm properties. Let $\phi(\cdot)$ be any symmetric gauge function and define a matrix norm by

$$\|A\|_\phi = \phi(\sigma_1, \dots, \sigma_{\min(M, N)}),$$

where $\{\sigma_k\}$ denotes the singular values of an $M \times N$ matrix A .

- Verify that this definition indeed defines a proper matrix norm.
- Prove or disprove (by counterexample) that any such matrix norm $\|A\|_\phi$ is unitarily invariant.

Problem 7.8 The OptShrink method evaluates the function $D(z, S)$ and its derivative $D'(z, S)$ for a certain (rectangular) diagonal matrix S . If either dimension of this matrix is large, then either SS' or $S'S$ may become impractical to store and manipulate, at least if stored as a dense matrix. Section 7.6.2 shows how to compute $D(z, S)$ without forming SS' or $S'S$, providing a step towards making OptShrink suitable for larger problems. Derive a similarly efficient expression for $D'(z, S)$.

Problem 7.9 Related to the OptShrink method, solve, for $\theta > 0$,

$$w_* = \arg \min_{w \in \mathbb{R}} \|w \hat{u} \hat{v}' - \theta u v'\|_F$$

for $u, \hat{u} \in \mathbb{F}^M$ and $v, \hat{v} \in \mathbb{F}^N$, all unit norm vectors. Based on your result, is w_* bigger or smaller than θ ?

Problem 7.10 Revisit the training derivation in Section 7.7.1 for the case where the encoder and decoder are affine instead of linear, that is, $\phi(x) = Ex + b_1$, $\psi(z) = Dz + b_2$.

8 Special Matrices, Markov Chains, and PageRank

8.1 Introduction

This chapter contains topics related to matrices with special structures that arise in many applications. Section 8.2 discusses companion matrices that are a classic linear algebra topic. Section 8.3 constructs circulant matrices from a particular companion matrix and describes their signal processing applications. Section 8.4 discusses the closely related family of Toeplitz matrices. Section 8.5 describes the power iteration that is used later in the chapter for Markov chains. Sections 8.6 and 8.7 discuss non-negative matrices and their relationships to graphs, leading to the analysis of Markov chains in Section 8.8. The chapter ends with two applications: Google's PageRank method and spectral clustering using graph Laplacians in Section 8.9. These are largely standalone topics that are useful in practice but not used in subsequent chapters.

8.2 Companion Matrices

We previously defined the eigenvalues of a square matrix A to be the roots of the characteristic polynomial

$$\det(zI - A) = 0.$$

Here we work somewhat in the reverse direction by starting with a polynomial and then defining a matrix from it. For $n \in \mathbb{N}$, consider the monic polynomial

$$p(z) = z^n + c_{n-1}z^{n-1} + \cdots + c_1z + c_0, \quad z \in \mathbb{C}, \quad (8.1)$$

and now define the following $n \times n$ matrix, called a companion matrix of that polynomial:

$$A \triangleq \begin{bmatrix} -c_{n-1} & -c_{n-2} & \cdots & -c_1 & -c_0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \quad (8.2)$$

Q8.1 Which matrix class is this?

- A: Lower triangular B: Upper triangular C: Lower Hessenberg D: Upper Hessenberg
E: None of these

Example 8.1 For $n = 3$ we have

$$A = \begin{bmatrix} -c_2 & -c_1 & -c_0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

To examine the eigenvalues of A , determine the characteristic polynomial by evaluating the determinant using the usual minors:

$$\begin{aligned} \det(zI - A) &= \det \left\{ \begin{bmatrix} z + c_2 & c_1 & c_0 \\ -1 & z & 0 \\ 0 & -1 & z \end{bmatrix} \right\} \\ &= (z + c_2) \det \left\{ \begin{bmatrix} z & 0 \\ -1 & z \end{bmatrix} \right\} - c_1 \det \left\{ \begin{bmatrix} -1 & 0 \\ 0 & z \end{bmatrix} \right\} + c_0 \det \left\{ \begin{bmatrix} -1 & z \\ 0 & -1 \end{bmatrix} \right\} \\ &= (z + c_2)(z^2 + 1 \cdot 0) - c_1(-z - 0^2) + c_0((-1)^2 - 0z) \\ &= z^3 + c_2z^2 + c_1z + c_0 = p(z). \end{aligned}$$

So, the eigenvalues of the companion matrix A are exactly the roots of the monic polynomial whose (negative) coefficients correspond to the first row of A . This is not a coincidence; it is by design.

Explore 8.1 Examine the cases $n = 2$ and $n = 1$.

For the general $n \times n$ case, the same process yields:

$$\begin{aligned} \det(zI - A) &= \det \left\{ \begin{bmatrix} z + c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \\ -1 & z & 0 & \cdots & 0 \\ 0 & -1 & z & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & z \end{bmatrix} \right\} \\ &= (z + c_{n-1}) \det \underbrace{\left\{ \begin{bmatrix} z & 0 & \cdots & 0 \\ -1 & z & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & z \end{bmatrix} \right\}}_{(n-1) \times (n-1)} \end{aligned}$$

$$\begin{aligned}
 & -c_{n-2} \det \left\{ \left[\begin{array}{c|ccccc} -1 & 0 & 0 & \cdots & 0 \\ 0 & z & 0 & \cdots & 0 \\ 0 & -1 & z & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & z \end{array} \right] \right\}_{(n-1) \times (n-1)} \\
 & + c_{n-3} \det \left\{ \left[\begin{array}{cc|ccccc} -1 & z & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & z & 0 & \cdots & 0 \\ 0 & 0 & -1 & z & \cdots & 0 \\ \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & z \end{array} \right] \right\}_{(n-1) \times (n-1)} + c_{n-4} + \cdots \\
 & = (z + c_{n-1})z^{n-1} + c_{n-2}z^{n-2} + c_{n-3}z^{n-3} + \cdots \quad (8.3)
 \end{aligned}$$

The general pattern is that we have a block diagonal matrix for the c_{n-k} term, where the first block is $(k-1) \times (k-1)$ upper triangular and has determinant $(-1)^{k-1}$, and the second block is lower triangular and has determinant z^{n-k} , leading to a final overall contribution to the determinant of $c_{n-k}z^{n-k}$ for $k = 1, \dots, n$, because

$$\det \left\{ \begin{bmatrix} B & 0 \\ 0 & C \end{bmatrix} \right\} = \det(B) \det(C).$$

Thus,

$$\det(zI - A) = c_0 + c_1 z + \cdots + c_{n-1} z^{n-1} + z^n = p(z). \quad (8.4)$$

In words: the polynomial $p(z)$ is the characteristic polynomial of A . It is also its minimal polynomial (see Fact 8.4). Due to this property, the matrix A so defined is called a companion matrix of the monic polynomial $p(z)$.

Fact 8.1 The eigenvalues of the companion matrix A are exactly the roots of the monic polynomial whose (negative) coefficients correspond to the first row of A .

There are other rearrangements of A (transposes and other permutations) that have the same property.

8.2.1 Practical Implementation

To match the arrangement given above, use the one-line JULIA function

```
compan = c -> [transpose(reverse(-c)); [I zeros(length(c)-1)]]
```

like $A = \text{compan}(c)$, where c is the (column) vector with elements $(c_0, c_1, \dots, c_{n-1})$. (This code needs $n \geq 1$ to work.) However, in practice this matrix is perhaps used more for analysis than for implementation.

Alternatively, one can use the `companion` method in the `Polynomial` package, $A = \text{companion}(\text{Polynomial}(c))$, where here c is the vector with elements $(c_0,$

$c_1, \dots, c_{n-1}, 1$). If $c_n \neq 1$ then the companion function divides all elements by c_n to make a monic polynomial.

Explore 8.2 Why use transpose above rather than c' ?

8.2.2 Polynomial Matrix Functions

Fact 8.2 The Cayley–Hamilton theorem says that the characteristic polynomial $p(z)$ of any $N \times N$ matrix A has the property that

$$p(A) = \sum_{k=0}^N c_k A^k = c_0 I_N + \sum_{k=1}^N c_k A^k = \mathbf{0}_{N \times N}, \quad (8.5)$$

where we use the matrix power property (see Section 3.2.4) that $A^0 = I$ for a square matrix A .

Proof (for the case of diagonalizable matrices). If $A = V\Lambda V^{-1}$ then

$$\begin{aligned} p(A) &= \sum_{k=0}^N c_k V\Lambda^k V^{-1} = V \left(\sum_{k=0}^N c_k \Lambda^k \right) V^{-1} = V \operatorname{Diag} \left\{ \sum_{k=0}^N c_k \lambda_i^k \right\} V^{-1} \\ &= V \operatorname{Diag}\{p(\lambda_i)\} V^{-1} = \mathbf{0}, \end{aligned}$$

because each eigenvalue λ_i is a root of the characteristic polynomial. \square

The proof for nondiagonalizable matrices uses the Jordan normal form (3.7).

Definition The minimal polynomial is the monic polynomial $\mu(z)$ having *least degree* for which $\mu(A) = \mathbf{0}$ [115, p. 143].

Fact 8.3 A matrix is diagonalizable iff its minimal polynomial is a product of distinct factors.

Fact 8.4 The minimal polynomial of a companion matrix for $p(z)$ is $p(z)$, the characteristic polynomial. Thus, a companion matrix for $p(z)$ is diagonalizable iff $p(z)$ has distinct roots.

In general, the characteristic polynomial for an $N \times N$ matrix has the form $p(z) = (z - z_1)^{p_1} \cdots (z - z_K)^{p_K}$, where K is the number of distinct eigenvalues (roots) and where $p_1 + \cdots + p_K = N$. The corresponding minimal polynomial for an $N \times N$ matrix has the form $\mu(z) = (z - z_1)^{q_1} \cdots (z - z_K)^{q_K}$, where $1 \leq q_k \leq p_k$ and $1 \leq q_1 + \cdots + q_K \leq N$ [115, p. 143].

Example 8.2 Consider the matrix $A = \beta I_N$. The characteristic polynomial is $p(z) = (z - \beta)^N$, whereas the minimal polynomial is $\mu(z) = z - \beta = 1z^1 + (-\beta)z^0$. In this example,

$$\mu(A) = 1 \cdot A^1 + (-\beta)A^0 = A - \beta I = \mathbf{0}.$$

Q8.2 The degree of the minimal polynomial of an $N \times N$ matrix is always in the set $\{1, \dots, N\}$.

A: True

B: False

Explore 8.3 Given a polynomial $p(z)$, determine another very simple matrix for which the eigenvalues $\{\lambda_i\}$ of that matrix are exactly the roots $\{z_i\}$ of that polynomial.

Example 8.3 Consider the (nondiagonalizable) matrix $A = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$ that has eigenvalues $(\lambda, \lambda, \lambda)$ and characteristic polynomial $p(z) = (z - \lambda)^3$. Clearly $A - \lambda I \neq 0$, but one can verify that $(A - \lambda I)^2 = 0$. Thus, the minimal polynomial for this matrix is $\mu(z) = (z - \lambda)^2$.

Example 8.4 Consider the monic polynomial $p(z) = z^2 - 6z + 9 = (z - 3)^2$ with corresponding companion matrix $A = \begin{bmatrix} 6 & -9 \\ 1 & 0 \end{bmatrix}$ for which $A - 3I = \begin{bmatrix} 3 & -9 \\ 1 & -3 \end{bmatrix}$. The only options for the minimal polynomial are $\mu(z) = (z - 3)^2$ and $\mu(z) = (z - 3)$. Because $A - 3I \neq 0$, the minimal polynomial must be $\mu(z) = (z - 3)^2 = p(z)$. This is expected because $\mu(z) = p(z)$ for all companion matrices. Both this example and the preceding one illustrate that a minimal polynomial can have repeated roots.

8.2.3 Eigenvectors of Companion Matrices

One can find eigenvectors of any companion matrix by inspection (i.e., by guess and check). For $t \in \mathbb{C}$,

$$\begin{aligned} v = \begin{bmatrix} t^{n-1} \\ \vdots \\ t \\ 1 \end{bmatrix} \implies Av &= \begin{bmatrix} -c_{n-1} & -c_{n-2} & \cdots & -c_1 & -c_0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} t^{n-1} \\ \vdots \\ t \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -c_{n-1}t^{n-1} - \cdots - c_1t - c_0 \\ t^{n-1} \\ \vdots \\ t \end{bmatrix} \stackrel{?}{=} \begin{bmatrix} t^n \\ t^{n-1} \\ \vdots \\ t \end{bmatrix} = tv, \quad (8.6) \end{aligned}$$

where the inner equality holds iff $-c_{n-1}t^{n-1} - \cdots - c_1t - c_0 = t^n$, that is, $0 = t^n + c_{n-1}t^{n-1} + \cdots + c_1t + c_0 = p(t)$. In other words, if t is a root of the polynomial $p(z)$, then the corresponding vector v is an eigenvector of A , with eigenvalue t . Note that v is a nonzero vector by definition.

8.2.4 Vandermonde Matrices

For each root of $p(z)$, the companion matrix A has a corresponding (unit norm) eigenvector. A companion matrix A for $p(z)$ is diagonalizable iff $p(z)$ has no repeated roots. (Recall that a matrix A is diagonalizable iff its minimal polynomial is a product of distinct factors.)

If the roots $\{z_1, \dots, z_n\}$ of $p(z)$ are *distinct*, then the corresponding companion matrix A is diagonalizable,

$$V^{-1}AV = \text{Diag}\{z_1, \dots, z_n\},$$

where the eigenvector matrix V is an $n \times n$ Vandermonde matrix of the form

$$V = \begin{bmatrix} z_1^{n-1} & z_2^{n-1} & \cdots & z_n^{n-1} \\ z_1^{n-2} & z_2^{n-2} & \cdots & z_n^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ z_1 & z_2 & \cdots & z_n \\ 1 & 1 & \cdots & 1 \end{bmatrix}. \quad (8.7)$$

Fact 8.5 A Vandermonde matrix is invertible iff the z_k values are distinct.

Vandermonde matrices are often transposed and/or reversed versions of the matrix shown above.

Example 8.5 In signal processing, the most important Vandermonde matrix is the N -point DFT matrix, which is a transposed and reversed version of the above matrix with $z_k = e^{-i2\pi(k-1)/N}$.

◆◆ Rectangular Vandermonde matrices have been used as frames [103].

Q8.3 The companion matrix for the polynomial $p(z) = z^2 - 6z + 9$ is diagonalizable.
A: True B: False

Example 8.6 The companion matrix for the polynomial $p(z) = z^2 - 6z + 9$ is

$A = \begin{bmatrix} 6 & -9 \\ 1 & 0 \end{bmatrix}$. The two eigenvalues are both at $z = 3$ and

$A - 3I = \begin{bmatrix} 3 & -9 \\ 1 & -3 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -3 \end{bmatrix}$, so all eigenvectors are in $\text{span}^\perp\left(\begin{bmatrix} 1 \\ -3 \end{bmatrix}\right)$, that is, multiples of $v_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$. This companion matrix is not diagonalizable. Try this in JULIA:

```

using LinearAlgebra
A = [6 -9; 1 0]
(lam, V) = eigen(A)
values:
2-element Vector{ComplexF64}:
 3.0 - 3.725290298461914e-8im
 3.0 + 3.725290298461914e-8im
vectors:
2x2 Matrix{ComplexF64}:
 0.948683-0.0im   0.948683+0.0im
 0.316228+3.927e-9im 0.316228-3.927e-9im

```

The two columns of V are almost identical to $v_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix} / \sqrt{10}$. Oddly, $\text{rank}(V)$ returns 2. Even more oddly, $V * \text{Diagonal}(\text{lam}) * \text{inv}(V)$ returns a matrix that is quite close to A . But $\text{cond}(V)$ returns $2.68e8$ and $\text{svdvals}(V)$ gives $1.414, 5.27e-9$, showing that V is essentially singular.

Explore 8.4 What is happening here?

8.2.5 Kronecker Sum and Polynomial Roots

To further connect matrices and polynomials, first we need another matrix tool.

Definition The Kronecker sum of $M \times M$ matrix A with an $N \times N$ matrix B is the following $MN \times MN$ matrix [4, p. 143]:

$$A \oplus B = (I_N \otimes A) + (B \otimes I_M). \quad (8.8)$$

Wikipedia uses this definition:

$$A \oplus B = (A \otimes I_N) + (I_M \otimes B).$$

The two definitions are the same to within a permutation; that is, there is an $MN \times MN$ permutation matrix P such that

$$P((I_N \otimes A) + (B \otimes I_M))P' = (A \otimes I_N) + (I_M \otimes B).$$

Because eigenvalues are invariant to similarity transforms, both definitions have the same eigenvalues so the following properties hold for both definitions.

8.2.5.1 Properties

- Scaling: $\alpha(A \oplus B) = (\alpha A) \oplus (\alpha B)$.
- Associative: $(A \oplus B) \oplus C = A \oplus (B \oplus C)$.
- Not commutative: $A \oplus B \neq B \oplus A$ in general.

Explore 8.5 Prove the associative property of \oplus .

Explore 8.6 Find a counterexample to verify that \oplus is not commutative.

Fact 8.6 [4, Theorem 13.16] If \mathbf{A} has eigenvalues $\{\lambda_m, m = 1, \dots, M\}$ and \mathbf{B} has eigenvalues $\{\mu_n, n = 1, \dots, N\}$, then the MN eigenvalues of $\mathbf{A} \oplus \mathbf{B}$ are

$$\lambda_m + \mu_n, \quad m = 1, \dots, M, n = 1, \dots, N. \quad (8.9)$$

Proof (sketch). If $\mathbf{Ax} = \lambda x$ and $\mathbf{By} = \mu y$, then

$$\begin{aligned} (\mathbf{A} \oplus \mathbf{B})(y \otimes x) &= (I_N \otimes \mathbf{A})(y \otimes x) + (\mathbf{B} \otimes I_M)(y \otimes x) \\ &= (y \otimes (\mathbf{Ax})) + ((\mathbf{By}) \otimes x) \\ &= (y \otimes (\lambda x)) + ((\mu y) \otimes x) = (\lambda + \mu)(y \otimes x). \end{aligned} \quad \square$$

8.2.5.2 Application: Checking for Common Roots of Two Polynomials

If $p(z)$ and $q(z)$ are two monic polynomials (possibly of different degrees) having corresponding companion matrices \mathbf{A} and \mathbf{B} , then $p(z)$ and $q(z)$ share a common root iff \mathbf{A} and \mathbf{B} have a common eigenvalue, that is, iff there exists some λ_m and μ_n such that $\lambda_m = \mu_n$. In other words, $p(z)$ and $q(z)$ share a common root iff the matrix $\mathbf{A} \oplus (-\mathbf{B})$ has a zero eigenvalue, that is, is singular. Thus we can determine if two polynomials have a common root *without* performing any eigendecomposition (Problem 8.1).

Another application is finding zeros of a univariate equation $f(x) = 0$ [228].

8.3 Circulant Matrices

A special case of the companion matrix considered above corresponds to the simple monic polynomial

$$p(z) = z^N - 1. \quad (8.10)$$

Here, $c_0 = -1$ and all other coefficients are 0, so the corresponding companion matrix is simply

$$\mathbf{G}_N \triangleq \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \quad (8.11)$$

This special case is a circulant matrix. (Indeed, it is the parent of all circulant matrices, as shown below.)

Multiplying this matrix by a vector $\mathbf{x} \in \mathbb{F}^N$ has the effect of (circularly) shifting the elements by one index. If $\mathbf{y} = \mathbf{G}_N \mathbf{x}$, then $y_i = x_{\text{mod}_1(i-1, N)}$, $i = 1, \dots, N$, where mod_1 corresponds to JULIA's `mod1` function and is defined by

$$\text{mod}_1(n, N) = \begin{cases} N, & n = 0, \\ n, & n = 1, \dots, N - 1. \end{cases} \quad (8.12)$$

The eigenvalues of the matrix G_N are the roots of the polynomial $p(z) = z^N - 1$ in (8.10), that is, the solutions to $z^N - 1 = 0$. Each of the N solutions is called a root of unity:

$$z_k = e^{-j2\pi k/N}, \quad k = 0, \dots, N-1. \quad (8.13)$$

We can also view the eigenvalues (8.13) from an SP perspective. The first column of the matrix G_N is the N -point signal $0, 1, 0, \dots, 0$, which we write as $h[n] = \delta[n-1]$ where $\delta[n] = I_{(n=0)}$ denotes the Kronecker impulse signal, and where $n = 0, 1, \dots, N-1$ in SP conventions. This $h[n]$ is the impulse response of a one-sample circular shift operation. The N -point DFT of the signal $h[n] = \delta[n-1]$ is $H[k] = e^{-j2\pi k/N} = z_k$. In other words, one can determine the eigenvalues of G_N by computing the N -point DFT of the signal values in the first column of G_N . We show below that this convenient eigenvalue property generalizes to all circulant matrices.

The eigenvectors of the matrix G_N are of the Vandermonde matrix form. An eigenvector (of unit norm) having eigenvalue corresponding to the k th root z_k is

$$v_k \triangleq \frac{1}{\sqrt{N}} z_k^{1-N} \begin{bmatrix} z_k^{N-1} \\ z_k^{N-2} \\ \vdots \\ z_k \\ 1 \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} z_k^0 \\ z_k^{-1} \\ \vdots \\ z_k^{2-N} \\ z_k^{1-N} \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ w_k \\ \vdots \\ w_k^{N-2} \\ w_k^{N-1} \end{bmatrix}, \quad w_k \triangleq e^{j2\pi k/N}. \quad (8.14)$$

Note that the w_k for $k = 0, \dots, N-1$ are also (distinct!) roots of $p(z)$, spaced equally around the unit circle in the complex plane. The $N \times N$ matrix formed from all N of the eigenvectors is a scaled Vandermonde-type matrix:

$$Q \triangleq [v_0 \ \dots \ v_{N-1}] = \frac{1}{\sqrt{N}} \begin{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} & \dots & \begin{bmatrix} 1 \\ w_k \\ \vdots \\ w_k^{N-2} \\ w_k^{N-1} \end{bmatrix} & \dots & \begin{bmatrix} 1 \\ w_{N-1} \\ \vdots \\ w_{N-1}^{N-2} \\ w_{N-1}^{N-1} \end{bmatrix} \end{bmatrix}. \quad (8.15)$$

Because the roots $\{w_k\}$ are distinct, this Vandermonde-like matrix is invertible. But here we can say much more: the columns of Q here are orthonormal, so Q is a unitary matrix; see (4.44). Here Q' is called the orthonormal DFT or unitary DFT matrix. Thus,

$$G_N = Q \text{Diag}\{z_0, \dots, z_{N-1}\} Q', \text{ or equivalently: } Q' G_N Q = \text{Diag}\{z_0, \dots, z_{N-1}\}. \quad (8.16)$$

Multiplying matrix Q' by a vector x of length N corresponds to taking the N -point (orthonormal) DFT of x .

Explore 8.7 Verify that Q is a symmetric (not Hermitian!) matrix.

Now consider a general circulant matrix:

$$\mathbf{C} = \begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & & c_2 \\ \vdots & c_1 & c_0 & & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}. \quad (8.17)$$

Because \mathbf{G}_N is a circulant permutation matrix, powers of \mathbf{G}_N provide other shifts:

$$\mathbf{G}_N^2 = \mathbf{G}_N \mathbf{G}_N = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & 0 \end{bmatrix}. \quad (8.18)$$

In SP language, $y = \mathbf{G}_N^2 \mathbf{x}$ corresponds to $y[n] = \delta[n-1] \oplus \delta[n-1] \oplus x[n] = \delta[n-2] \oplus x[n]$. By defining $\mathbf{G}_N^0 = \mathbf{I}$, we write the general circulant matrix \mathbf{C} in terms of powers of \mathbf{G}_N as follows:

$$\mathbf{C} = c_0 \mathbf{I} + c_1 \mathbf{G}_N + c_2 \mathbf{G}_N^2 + \cdots + c_{N-1} \mathbf{G}_N^{N-1} = c_0 \mathbf{I} + \sum_{n=1}^{N-1} c_n \mathbf{G}_N^n = \sum_{n=0}^{N-1} c_n \mathbf{G}_N^n. \quad (8.19)$$

The matrix \mathbf{G}_N^n has the same (orthonormal) eigenvectors as \mathbf{G}_N because $\mathbf{G}_N^2 \mathbf{v}_k = \mathbf{G}_N(\mathbf{G}_N \mathbf{v}_k) = z_k^2 \mathbf{v}_k$. Thus we can diagonalize \mathbf{C} as follows:

$$\begin{aligned} \mathbf{Q}' \mathbf{C} \mathbf{Q} &= \mathbf{Q}' \left(\sum_{n=0}^{N-1} c_n \mathbf{G}_N^n \right) \mathbf{Q} = \sum_{n=0}^{N-1} c_n (\mathbf{Q}' \mathbf{G}_N^n \mathbf{Q}) = \sum_{n=0}^{N-1} c_n \text{Diag}\{z_0^n, \dots, z_{N-1}^n\} \\ &= \text{Diag} \left\{ \sum_{n=0}^{N-1} c_n z_0^n, \dots, \sum_{n=0}^{N-1} c_n z_{N-1}^n \right\} = \mathbf{\Lambda} = \text{Diag}\{\lambda_k\}, \quad \lambda_k \triangleq \sum_{n=0}^{N-1} c_n z_k^n. \end{aligned} \quad (8.20)$$

Rearranging, a unitary eigendecomposition of \mathbf{C} is

$$\mathbf{C} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}', \quad \lambda_k \triangleq \sum_{n=0}^{N-1} c_n z_k^n. \quad (8.21)$$

This decomposition holds for *any* circulant matrix \mathbf{C} . In other words, all $N \times N$ circulant matrices have the same unitary eigenvectors (the orthonormal DFT basis).

The k th eigenvalue of \mathbf{C} is

$$\lambda_k = \sum_{n=0}^{N-1} c_n z_k^n = \sum_{n=0}^{N-1} c_n e^{-i2\pi n k / N}, \quad k = 0, \dots, N-1, \quad (8.22)$$



which is the (ordinary, *not* orthonormal) DFT of the *first column* of \mathbf{C} . One can use the fast Fourier transform (FFT) to find all N eigenvalues with $O(N \log N)$ operations, by taking the FFT of the first column of the circulant matrix.

8.3.1 Relationship to DFT Properties from DSP

The eigendecomposition (8.21) is the linear algebra expression of the DFT property that N -point circular convolution, denoted by \otimes , corresponds to multiplying spectra:

$$h[n] \otimes x[n] \xrightarrow{\text{DFT}} H[k]X[k]. \quad (8.23)$$

The fact that all circulant matrices have the same unitary basis per (8.21) means that all circulant matrices commute (see Section 9.2.11; Problem 8.6), which is the linear algebra expression of the DFT property that circular convolution operations commute:

$$h[n] \otimes x[n] = x[n] \otimes h[n] \xrightarrow{\text{DFT}} H[k]X[k] = X[k]H[k]. \quad (8.24)$$

The following associative property of circular convolution can be proven using the fact that all circulant matrices commute:

$$h[n] \otimes (g[n] \otimes x[n]) = (h[n] \otimes g[n]) \otimes x[n]. \quad (8.25)$$

The identity matrix I is a circulant matrix, so $I = QIQ'$ where Q is the orthonormal DFT matrix defined in (8.15). The fact that the DFT of the Kronecker impulse signal $x[n] = [1 \ 0 \ \dots \ 0]$ is a flat spectrum $X[k] = 1$ is equivalent to the fact that the eigenvalues of I are all unity.

Q8.4 The number of distinct eigenvalues of the matrix G_6^3 is:

- A: 1 B: 2 C: 3 D: 4 E: 6

8.3.2 Practical Implementation

A circulant matrix is a special case of a Toeplitz matrix. To construct a circulant matrix in JULIA:

```
using ToeplitzMatrices
A = Circulant(1:3)
```

In JULIA, the resulting variable is a special type of matrix (somewhat like `Diagonal` or `Sparse`) that performs matrix–vector multiplication efficiently.

Directly multiplying an $N \times N$ matrix C by a length- N vector, $y = Cx$, via $y = C * x$ would require $O(N^2)$ operations. When C is circulant, we can compute the same product in $O(N \log_2 N)$ operations [229] using the (FFT) as follows:

$$y = \text{ifft}(\text{fft}(c) .* \text{fft}(x)) \quad \text{cf. } y = Q(\Lambda(Q'x)),$$

where c is the first column of C .

The resulting y will be the same as if we did $y = C * x$ except for some small numerical differences due to finite precision. These differences are rarely (if ever) important in practice.

One cautionary note is that if C and x are both real then of course $y = Cx$ is also real. However, the result of the above set of `fft` operations may end up with some very small imaginary part, depending on the implementation, so you might need to take the real part at the end.

The JULIA type `Circulant` in `ToeplitzMatrices.jl` takes care of this automatically by overloading the `*` operation. Other operations implemented elegantly include `inv` and `pinv`.

8.3.3 Spectral Properties of Circulant Matrices

Fact 8.7 Every circulant matrix is a normal matrix (Problem 8.6).

Q8.5 If \mathbf{C} is a circulant matrix, then its spectral norm is

$$\sigma_1 = \text{maximum}(\text{abs}, \text{fft}(\mathbf{C}[:, 1]))$$

(assume that using FFTW was invoked first).

A: True B: False

To avoid using an FFT, one can use a discrete version of Young's convolution inequality [230, 231] to show that $\|\mathbf{C}\mathbf{x}\|_p \leq \|\mathbf{C}_{:, 1}\|_1 \|\mathbf{x}\|_p$ when \mathbf{C} is circulant. In other words, all of the induced p -norms of a circulant matrix are bounded above by the easily computed value $\|\mathbf{C}_{:, 1}\|_1$. In particular, using $p = 2$ means that $\sigma_1(\mathbf{C}) \leq \|\mathbf{C}_{:, 1}\|_1$, a fact that is useful for learning filters [80], because we can compute $\|\mathbf{C}_{:, 1}\|_1$ with $O(N)$ operations.

Explore 8.8 Is the inequality $\|\mathbf{C}\mathbf{x}\|_\infty \leq \|\mathbf{C}_{:, 1}\|_1 \|\mathbf{x}\|_\infty$ tight?

Explore 8.9 Determine whether $\sigma_1(\mathbf{C}) = \|\mathbf{C}_{:, 1}\|_1$.

8.3.4 Inverting a Circulant Matrix

Any circulant matrix \mathbf{C} has eigendecomposition $\mathbf{C} = \mathbf{Q}\Lambda\mathbf{Q}'$. If the eigenvalues are nonzero, then \mathbf{C} is invertible and its inverse is

$$\mathbf{C}^{-1} = \mathbf{Q}\Lambda^{-1}\mathbf{Q}'. \quad (8.26)$$

In other words, multiplying the inverse of \mathbf{C} times a vector \mathbf{x} , as in $\mathbf{C}^{-1}\mathbf{x}$, can be done with $O(N \log N)$ FLOPs using an FFT, which is quite fast compared to a general invertible matrix. Specifically, $\mathbf{C}^{-1}\mathbf{x} = \mathbf{Q}(\Lambda^{-1}(\mathbf{Q}'\mathbf{x}))$. This property is useful for preconditioning and image deblurring.

8.4 Toeplitz Matrices

Circulant matrices are special cases of Toeplitz matrices. The broader family of Toeplitz matrices is of great interest in SP because Toeplitz matrix structure corresponds to linear shift-invariant convolution operations.

8.4.1 Toeplitz Matrix Multiplication with a Vector

As discussed in Section 2.3.1.2, convolution of a finite-length signal with a finite-length filter is equivalent to multiplying a Toeplitz matrix by a vector with the signal samples. Roughly:

$$h * x \equiv Hx \quad (8.27)$$

for an appropriate matrix H defined in (2.4) from the filter h .

Because of this equivalence, there are several methods for computing Hx for Toeplitz H . If $x \in \mathbb{F}^N$ and the filter length is L then ordinary convolution requires about $O(LN)$ multiplies, whereas a basic implementation of the matrix–vector product Hx would require $O((L + N - 1)N) = O(N^2)$ multiplies. When $L \ll N$ we can store H as a sparse matrix, in which case Hx requires $O(LN)$ multiplies.

When L is not small, it is more efficient to use zero-padding and perform convolution using an FFT-based approach. The `fftfilt` function in JULIA’s `DSP.jl` does this. That approach requires roughly $O(N \log N)$ multiplies. In terms of matrices, it is basically like extending H to make a bigger circulant matrix, zero-padding x as well, then doing circulant matrix–vector multiplication using FFT operations, and then trimming the result to the correct output size. JULIA’s `ToeplitzMatrices.jl` supports various options.

8.4.2 Inverting a Toeplitz Matrix

There are efficient methods for inverting Toeplitz matrices [232, 233, 234, 235, 236, 237].

8.4.3 Factoring a Toeplitz Matrix

If V is an $N \times K$ Vandermonde matrix with $K \leq N$ for which $V_{nk} = z_k^{n-1} = e^{i2\pi(n-1)f_k}$, where the $\{f_k\}$ are distinct, and $D = \text{Diag}\{d_1, \dots, d_K\} \in \mathbb{F}^{K \times K}$, with $d_k \neq 0$, then the following is an $N \times N$ rank- K Toeplitz matrix:

$$\begin{aligned} T &= VDV' = \sum_{k=1}^K d_k v_k v'_k \\ \text{because } T_{nm} &= \sum_{k=1}^K d_k V_{nk} V_{mk}^* = \sum_{k=1}^K d_k e^{i2\pi(n-1)f_k} e^{-i2\pi(m-1)f_k} = \sum_{k=1}^K d_k e^{i2\pi(n-m)f_k}. \end{aligned} \quad (8.28)$$

This fact is useful in some sampling/interpolation problems [238] and some imaging applications [239].

There is a partial converse of this fact. If T is a positive semidefinite Toeplitz matrix with rank K , then there exist V and D of the above form such that $T = VDV'$ where the elements of D are positive [240]. Without loss of generality we can sort them from largest to smallest.

Q8.6 This form is a compact SVD factorization of T .

A: True B: False

8.5 Power Iteration

For large matrices, performing a full eigendecomposition is impractical. Fortunately, sometimes we need only the largest-magnitude eigenvalue (and/or corresponding eigenvector). The power iteration is an iterative method for such tasks. It is a classic method with numerous applications, including spectral normalization for CNN training [241, 242], for example for image denoising [243]. Twitter uses the power iteration to show users recommendations of whom to follow [244].

Recall from (3.10) that if $A \in \mathbb{F}^{N \times N}$ is a diagonalizable matrix with eigendecomposition $A = V\Lambda V^{-1}$, then $A^k = V\Lambda^k V^{-1}$. Thus, for a vector $x_0 \in \mathbb{F}^N$,

$$A^k x_0 = V\Lambda^k \underbrace{V^{-1}x_0}_{\triangleq z} = \sum_{n=1}^N (\lambda_n^k z_n) v_n. \quad (8.29)$$

Now make some assumptions:

- Without loss of generality we order the eigenvalues with decreasing magnitudes: $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_N|$.
- Assume $\lambda_1 \neq 0$ to avoid the trivial case of $A = 0$.
- Assume $z_1 = [V^{-1}x_0]_1 \neq 0$, that is, the initial vector x_0 has a nonzero coordinate in the direction associated with the dominant eigenvalue λ_1 . Choosing x_0 at random ensures $z_1 \neq 0$ with probability 1.

Then it follows from the linear independence of the columns of V and the summation in (8.29) that $A^k x_0 \neq \mathbf{0}$ for all $k \in \mathbb{N}$, so we can normalize it to have unit norm and define

$$x_k \triangleq \frac{A^k x_0}{\|A^k x_0\|_2}, \quad k = 1, 2, \dots \quad (8.30)$$

This definition leads to the following simple recursion for $k = 0, 1, \dots$:

$$x_{k+1} = \frac{A^{k+1} x_0}{\|A^{k+1} x_0\|_2} = \frac{A(A^k x_0)}{\|A(A^k x_0)\|_2} = \frac{A(A^k x_0)/\|A^k x_0\|_2}{\|A(A^k x_0)/\|A^k x_0\|_2\|_2} = \frac{Ax_k}{\|Ax_k\|_2}.$$

For implementation we use this recursive form that is called the power iteration:

$$x_{k+1} = \frac{Ax_k}{\|Ax_k\|_2}, \quad k = 0, 1, \dots \quad (8.31)$$

Returning to the summation form (8.29), one can show with some algebra that

$$x_k \propto A^k x_0 = z_1 \lambda_1^k \left(v_1 + \sum_{n=2}^N \left(\left(\frac{\lambda_n}{\lambda_1} \right)^k \frac{z_n}{z_1} \right) v_n \right), \quad (8.32)$$

where v_n denotes a (unit-norm) eigenvector of A corresponding to λ_n , that is, the n th column of V .

Example 8.7 Consider the 1×1 matrix $A = i$. Here, $A^k x_0 = z_1 i^k$ so

$$x_k = A^k x_0 / \|A^k x_0\| = z_1 i^k / |z_1 i^k| = e^{i \angle z_1} i^k,$$

which does not converge due to the oscillations of the i^k term! Casually speaking, people say “the power iteration converges to v_1 ,” but this 1×1 example shows that such words are imprecise. In general, one cannot claim that $\|x_k - \alpha v_1\| \rightarrow 0$, no matter how we might pick α .

8.5.1 Convergence of the Power Iteration

However, if the first eigenvalue magnitude dominates all others, that is, if

$$|\lambda_n| < |\lambda_1|, n = 2, \dots, N,$$

then one can show that $\|e^{-i \angle z_1} (e^{-i \angle \lambda_1})^k x_k - v_1\|_2 \rightarrow 0$, because $e^{-i \angle z_1} z_1 = |z_1|$ so

$$e^{-i \angle z_1} (e^{-i \angle \lambda_1})^k x_k = \frac{|z_1| |\lambda_1|^k \left(v_1 + \sum_{n=2}^N \left(\left(\frac{\lambda_n}{\lambda_1} \right)^k \frac{z_n}{z_1} \right) v_n \right)}{\left\| z_1 \lambda_1^k \left(v_1 + \sum_{n=2}^N \left(\left(\frac{\lambda_n}{\lambda_1} \right)^k \frac{z_n}{z_1} \right) v_n \right) \right\|} \rightarrow v_1. \quad (8.33)$$

In other words, as k increases, x_k is approximately $e^{i \angle z_1} e^{i k \angle \lambda_1} v_1$, meaning that it is approximately v_1 times a unit-magnitude complex number. Another way of saying this is $\|P_{v_1}^\perp x_k\| \rightarrow 0$ as $k \rightarrow \infty$.

Under all the conditions assumed above, one can also show that the eigenvalue converges:

$$x_k' A x_k \rightarrow \lambda_1. \quad (8.34)$$

Here we do not need to worry about the extra phase term $e^{i \angle z_1} e^{i k \angle \lambda_1}$ because it cancels out due to x_k' and x_k :

$$(e^{i \phi} v_1)' A (e^{i \phi} v_1) = e^{-i \phi} e^{i \phi} v_1' (A v_1) = v_1' (\lambda_1 v_1) = \lambda_1. \quad (8.35)$$

- In words, if one eigenvalue magnitude dominates all others, and if $z_1 \neq 0$, then for large k the output of the power iteration x_k is approximately a unit-norm eigenvector corresponding to that largest-magnitude eigenvalue.
- Casually speaking we say “the power iteration converges to the eigenvector corresponding to the largest-magnitude eigenvalue,” but it is imprecise to say “the” eigenvector since we can always scale by -1 (or $e^{i \phi}$ more generally), and to make rigorous claims about convergence we must also think about the phase.
- The convergence rate of the power iteration is governed by $|\lambda_2/\lambda_1|$, so the bigger the “gap” between the first- and second-largest eigenvalue magnitudes, the faster the convergence.

- We have assumed A is diagonalizable here, but power iteration convergence analysis generalizes to non-diagonalizable matrices using the Jordan normal form. So, sufficient conditions are:
 - A is square;
 - $v_1 = [V^{-1}x_0]_1 \neq 0$, where V denotes the (linearly independent) generalized eigenvectors in the Jordan normal form, ordered so that $|\lambda_1|$ is largest;
 - $|\lambda_1|$ dominates the other eigenvalue magnitudes.
- For analyzing a Markov chain (Section 8.8), we will see that the dominant eigenvalue of the transition matrix P is unity. For any matrix where λ_1 is real and positive we have $e^{\lambda_1 k} = 1$, so there is no problematic oscillation term. In this case, under the sufficient conditions above we can say that the power iteration converges:

$$x_k \rightarrow e^{\lambda_1 k} v_1. \quad (8.36)$$

For Markov chains, we will also see that v_1 can be a nonnegative vector, so we can normalize it to sum to one, and if we initialize with a nonnegative vector x_0 that sums to one then $e^{\lambda_1 k} = 1$ and we can simply let $x_{k+1} = Px_k$ and conclude that $x_k \rightarrow v_1$.

- One can use a power iteration to compute the eigenvector corresponding to the smallest eigenvalue. This can be useful for finding null space vectors for the applications mentioned in Section 4.4.1 (Problem 3.7).

Q8.7 To determine the principal left singular vector u_1 when $\sigma_1(A) > \sigma_2(A)$, apply the power iteration to

- A: A' B: A C: AA' D: $A'A$ E: None of these

Example 8.8 Here is an illustration of a representative case where the power iteration fails to converge when the condition $[V^{-1}x_0]_1 \neq 0$ is not met, that is, when

$[V^{-1}x_0]_1 = 0$. Consider $A = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$, an asymmetric matrix having the eigendecomposition $A = V\Lambda V^{-1}$ with

$$V = \begin{bmatrix} 0 & 1/\sqrt{2} \\ 1 & -1/\sqrt{2} \end{bmatrix} = [v_1 \ v_2], \quad \Lambda = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad V^{-1} = \begin{bmatrix} 1 & 1 \\ \sqrt{2} & 0 \end{bmatrix}.$$

Now initialize the power iteration with $x_0 = \sqrt{2}v_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, for which

$[V^{-1}x_0]_1 = -1 \neq 0$, but $[V^{-1}x_0]_2 = 0$. In this case we have $Ax_0 = x_0$, so the power iteration does not converge to v_1 for this choice of x_0 .

8.5.2 Geršgorin Disk Theorem

Before proceeding to more special matrices, we state an important result about matrix eigenvalues. Recall that the spectral radius of a square matrix is its largest eigenvalue

magnitude: $\rho(\mathbf{A}) = \max_i |\lambda_i(\mathbf{A})|$. For $\mathbf{A} \in \mathbb{R}^{N \times N}$, the i th Geršgorin (row) disk is centered at a_{ii} and has radius given by the absolute sum of the off-diagonal elements of the i th row:

$$\mathcal{K}_i \triangleq \left\{ \lambda \in \mathbb{C}: |\lambda - a_{ii}| \leq r_i = \sum_{j \neq i} |a_{ij}| \right\}, \quad i = 1, \dots, N. \quad (8.37)$$

Fact 8.8 The Geršgorin disk theorem states that all eigenvalues of a matrix \mathbf{A} are contained in the union of the Geršgorin disks:

$$\lambda_i \in \bigcup_{j=1}^n \mathcal{K}_j. \quad (8.38)$$

More generally, if any union of J disks is disjoint from all the other $N - J$ disks, then that union contains exactly J eigenvalues of \mathbf{A} .

Example 8.9 If \mathbf{P} is a permutation matrix, then each row has one element that is 1 and all the others are zero. Thus, either $\mathcal{K}_i = \{1\}$ or \mathcal{K}_i is the unit disk. So all the eigenvalues of a permutation matrix lie inside the unit disk. (In fact they lie on the unit circle, because all singular values of \mathbf{P} are 1 and \mathbf{P} is normal so its singular values are the sorted absolute values of its eigenvalues.)

8.5.2.1 Row and Column Geršgorin Disks

The disks in (8.37) use the off-diagonal row sums. The eigenvalues of a matrix and its transpose (not Hermitian transpose!) are the same, so one can also define (“column”) disks using the off-diagonal column sums. The union of those column disks must also contain all the eigenvalues, so the eigenvalues must be in the intersections of those two unions.

Example 8.10 For the matrix $\mathbf{A} = \begin{bmatrix} 2 & 2 & 1 \\ 1 & 5 & 1 \\ 0 & 1 & 3 \end{bmatrix}$, Fig. 8.1(a) shows the row disks; they

contain the origin, so the matrix looks like it could be singular. Figure 8.1(b) shows the column disks; they do not contain the origin so the matrix must be invertible.

Q8.8 What is the best (i.e., lowest) upper bound for the spectral radius of \mathbf{A} ?

- A: -1 B: 1 C: 5 D: 7 E: 8

Example 8.11 Consider the 2×2 permutation matrix $\mathbf{P} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. This matrix is invertible because $\mathbf{P}^2 = \mathbf{I}_2$. Each Geršgorin disk is simply the unit disk centered at the origin. Thus, for a matrix to be invertible it is *sufficient* that the disk union does not cover the origin, but that is not a *necessary* condition for invertibility.

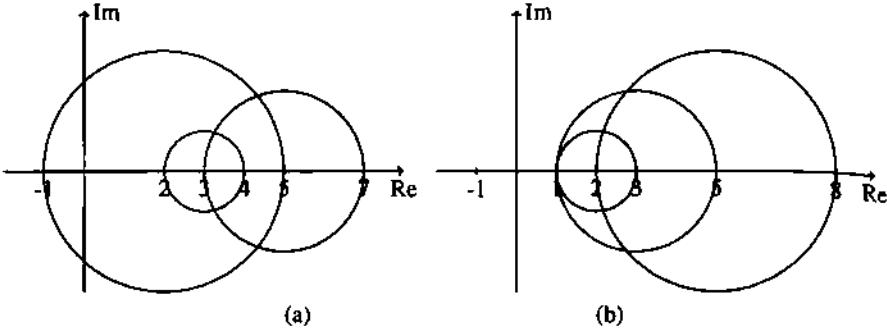


Figure 8.1 Geršgorin disk illustration.

8.5.2.2 General Spectral Radius Bounds

If $\lambda \in \mathcal{K}_i$, then applying the reverse triangle inequality to the Geršgorin disk theorem yields:

$$\begin{aligned} |\lambda| - |a_{ii}| &\leq ||\lambda| - |a_{ii}|| \leq |\lambda - a_{ii}| \leq r_i \\ \implies |\lambda| &\leq |a_{ii}| + r_i = |a_{ii}| + \sum_{j \neq i} |a_{ij}| = \sum_j |a_{ij}|. \end{aligned} \quad (8.39)$$

This inequality leads to the following upper bound on the spectral radius of any square matrix:

$$\rho(\mathbf{A}) = \max_i |\lambda_i| \leq \max_i \sum_j |a_{ij}| = \|\mathbf{A}\|_\infty. \quad (8.40)$$

Because \mathbf{A} and \mathbf{A}' have the same eigenvalue magnitudes,

$$\rho(\mathbf{A}) \leq \min \left(\max_i \sum_j |a_{ij}|, \max_j \sum_i |a_{ij}| \right) = \min(\|\mathbf{A}\|_\infty, \|\mathbf{A}\|_1). \quad (8.41)$$

Section 6.4.6 also provided these upper bounds because $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$ for any induced norm. Similarly, again using the reverse triangle inequality,

$$\begin{aligned} |a_{ii}| - |\lambda| &\leq ||\lambda| - |a_{ii}|| \leq |\lambda - a_{ii}| \leq r_i \implies |a_{ii}| - r_i \leq |\lambda| \\ \implies \min_i \{ \max(|a_{ii}| - r_i, 0) \} &\leq |\lambda|, \end{aligned} \quad (8.42)$$

from which we could obtain a (often loose) lower bound on ρ . There are several other lower bounds in the literature (see [245] and [115, Section 6.2]), including [246]:

$$\mathbf{A} \in \mathbb{R}^{N \times N} \implies \frac{1}{N} |\text{trace}(\mathbf{A})| \leq \rho(\mathbf{A}), \quad (8.43)$$

and [246] uses such bounds to analyze the stability of a recurrent neural network.

We turn now to matrices with quite simple upper and lower bounds.

8.6 Nonnegative Matrices and Graphs

We now turn to the special category of matrices whose elements are all real and non negative. These matrices have special eigenvalue and eigenvector properties (Section 8.7) that are important in their many applications.

- Such matrices can describe transition probabilities for Markov chains that model many random processes.
- Google's PageRank algorithm for ranking websites is built on a certain nonnegative matrix.
- Other applications include power control, commodity pricing, networks, and population growth [247].

We begin with some definitions.

8.6.1 Primitive Matrices

Definition A matrix is called a positive matrix iff all of its elements are real and positive.

Definition A matrix is called a nonnegative matrix iff all of its elements are real and nonnegative.

Definition A square nonnegative matrix is called a (left) stochastic matrix iff each of its columns sum to 1.

Mathematically, P is a stochastic matrix iff $P \in [0, 1]^{N \times N}$ and $\mathbf{1}_N' P = \mathbf{1}_N'$. Such a matrix typically is used to describe the transition matrix of a Markov chain; see Section 8.8.

Definition A matrix A is called a primitive matrix iff it is a square nonnegative matrix and A^m is a positive matrix for some $m \in \mathbb{N}$.

Q8.9 The matrix $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ is (choose the most specific answer):

- A: Square B: Nonnegative C: Primitive D: Positive E: None of these



The preceding definitions are unrelated to the terms positive definite and positive semidefinite matrices, except in some special cases like diagonal matrices.

Q8.10 Any positive semidefinite diagonal matrix is a nonnegative matrix. Any positive definite diagonal matrix is a positive matrix.

- A: (T,T) B: (T,F) C: (F,T) D: (F,F)

Q8.11 Every stochastic matrix is a square positive matrix. Every square positive matrix is a primitive matrix.

- A: (T,T) B: (T,F) C: (F,T) D: (F,F)



Figure 8.2 Venn diagram (for square matrices only).

Q8.12 The circulant generator matrix G_N in (8.11) for $N > 1$ is (choose the most specific answer):

- A: Square B: Nonnegative C: Primitive D: Positive E: None of these

Q8.13 Add stochastic matrices to Fig. 8.2.

(There are rectangular matrices that are nonnegative and/or positive, but those are not of interest here.)

8.6.1.1 Power Test for Primitive Matrices

Fact 8.9 An $N \times N$ nonnegative matrix A is a primitive matrix iff A^{N^2-2N+2} is a positive matrix [248]. (It suffices to use the binary matrix that has 1s where A is positive and 0s elsewhere.)

This test of whether a matrix is primitive is impractical for large N .

Example 8.12 The following code constructs an example matrix A (due to Helmut Wielandt) where A^{N^2-2N+1} is not positive yet A^{N^2-2N+2} is. This example is key to the “only if” part of Fact 8.9. See Fig. 8.3.

```
</> 8.1
N = 8
p = N^2 - 2N + 2
A = diagm(1 => ones(Int, N-1)); A[N,1:2] .= 1 # Wielandt's example
@assert !all(A^(p-1) .> 0) && all(A^p .> 0)
```

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

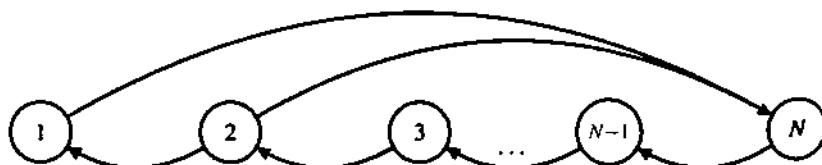


Figure 8.3 Wielandt's matrix and weighted directed graph.

Fact 8.10 If A is primitive with A^m being positive for some $m \in \mathbb{N}$, then A^n is positive $\forall n \geq m$.

Proof. When A^m is positive, then, because A is nonnegative, element i,j of $A^{m+1} = AA^m$ is positive unless the i th row of A were entirely zero, because $[A^{m+1}]_{i,j} = A_{i,:}[A^m]_{:,j}$. But if the i th row of A were entirely zero then so would be the i th row of every power of A , contradicting the assumption that A^m is positive. Thus A^{m+1} is positive, and by induction so is every higher power of A . \square

Explore 8.10 After studying the next section, express the above matrix property in terms of a graph property.

8.6.2 Weighted Directed Graphs

Square positive, nonnegative, stochastic, and primitive matrices have various special properties in terms of their eigenvalues and eigenvectors that are important in applications. We focus on those properties soon, but first we relate these matrices to directed graphs.

We can associate any square nonnegative matrix A with a corresponding weighted directed graph $G(A)$ that provides an intuitive visual depiction of the positive elements of A (see Fig. 8.3). A directed graph has nodes that are connected by arrows (or directed edges) that are labeled by weights. Our convention is that a_{ij} denotes the weight from node j to node i , with arrows *only* for positive weights. A node is also called a vertex or state, and is represented by a circle.

Example 8.13 Figure 8.4 shows the weighted directed graph $G(A)$ for the square,

$$\text{nonnegative matrix } A = \begin{bmatrix} 0 & 6 & 8 \\ 4 & 7 & 9 \\ 5 & 0 & 0 \end{bmatrix}.$$

We do not draw an arrow from node j to node i if $a_{ij} = 0$. We allow non-zero diagonal elements, so technically this is a weighted directed graph with loops.

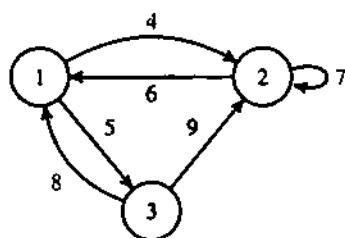


Figure 8.4 A weighted directed graph.

Explore 8.11 Think about what the graph would look like for a positive matrix.

Definition The weighted directed graph $\mathcal{G}(A)$ associated with an $N \times N$ square nonnegative matrix A has nodes numbered $1, \dots, N$, and a set of *distinct* weighted directed edges (arrows) consisting of at most N^2 tuples of the form (i, j, a_{ij}) where $i, j \in \{1, \dots, N\}$ and $a_{ij} > 0$.

Example 8.14 For $\mathcal{G}(A)$ in Fig. 8.4, the set of six distinct tuples is $\{(2, 1, 4), (3, 1, 5), (1, 2, 6), (2, 2, 7), (1, 3, 8), (2, 3, 9)\}$.

A directed graph looks visually like a “map” and we can imagine traveling from node to node around the diagram by following paths defined by the arrows.

Definition We say we can reach or access node i from node j in n steps if there is a path along n arrows (including loops) starting at j and ending at i (possibly passing through j or i along the way).

Example 8.15 In Fig. 8.4, we can reach node 3 from node 2 in two steps: $2 \rightarrow 1 \rightarrow 3$.

Q8.14 In Example 8.15, we can reach node 3 from node 2 in four steps in how many distinct ways?

- A: 0 B: 1 C: 2 D: 3 E: 4

8.6.2.1 Reachability or Accessibility

To determine mathematically (in matrix form) which nodes in $\mathcal{G}(A)$ we can reach from node j in one step, simply compute the matrix–vector product $Ae_j = A_{:,j}$ and examine which elements are positive.

Example 8.16 $Ae_2 = \begin{bmatrix} 0 & 6 & 8 \\ 4 & 7 & 9 \\ 5 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \\ 0 \end{bmatrix}$, so from node 2 we can reach nodes 1 and 2 in one step.

Fact 8.11 To determine which nodes we can reach in $\mathcal{G}(A)$ from node j in m steps, simply compute the product

$$A^m e_j \tag{8.44}$$

and then examine which elements are positive.

This is the mathematical version of what we do graphically when we travel along arrows in the diagram. The fact that A is square and nonnegative is key to establishing this property.

Example 8.17 $A^2 e_1 = A(Ae_1) = \begin{bmatrix} 0 & 6 & 8 \\ 4 & 7 & 9 \\ 5 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 64 \\ 73 \\ 0 \end{bmatrix}$, so from node 1 we can reach nodes 1 or 2 in two steps, but not node 3.

Fact 8.12 If A is a primitive matrix then $\exists m \in \mathbb{N}$ such that A^m is a positive matrix, so in $\mathcal{G}(A)$ one can reach any node i from any node j in m steps.

8.6.2.2 Adjacency Matrix

How we reach from one node to another node does not depend on the graph weights (other than the fact that $a_{ij} = 0$ is excluded from the graph). Only the presence or absence of an arrow matters. Thus, we often only need to consider the adjacency matrix of a (directed) graph. For directed graphs, an adjacency matrix need not be symmetric.



Example 8.18 For Example 8.17, the adjacency matrix is $M = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$. Note that $M_{ij} = \mathbb{I}_{\{a_{ij} \neq 0\}}$.

When illustrating an adjacency matrix using a (nonweighted) directed graph, we need not label the arrows because zero weights are not shown so all arrows correspond to a unity weight, like those illustrated in the next subsection.

8.6.3 Strongly Connected Graphs

Now we begin to relate matrix properties and graph properties.

Definition A directed graph is strongly connected if it is possible to reach any vertex/state/node from any other (in one or more steps).

Example 8.19 The directed graphs in Figs. 8.3 and 8.4 are strongly connected.

A strongly connected graph cannot have any transient or absorbing states, nor can it be partitioned into two or more separate graphs, as illustrated in Fig. 8.5.

Q8.15 Are these two statements true or false? For any primitive matrix A , the corresponding directed graph $\mathcal{G}(A)$ is strongly connected. If $\mathcal{G}(A)$ is strongly connected, then the corresponding matrix A is primitive.

A: (T,T)

B: (T,F)

C: (F,T)

D: (F,F)

Strongly connected graphs are especially important, so we want a suitable corresponding matrix type.



Node 3 is transient: after you leave it, you never return to it.



Node 4 is absorbing: after you enter it, you can never leave.



The graph partitions so the adjacency matrix can be permuted to be block diagonal.

Figure 8.5 Examples of graphs that are *not* strongly connected.

8.6.4 Irreducible Matrix

Definition A square matrix $A \in \mathbb{R}^{N \times N}$ (must be real) is called irreducible iff

$$\forall i, j, \exists m \in \mathbb{N} \text{ such that } [A^m]_{i,j} > 0. \quad (8.45)$$

(In general, m depends on i, j .) Otherwise, the matrix is called reducible.

There are other equivalent definitions of irreducible matrices.

Q8.16 Are these two statements true or false? Any primitive matrix is an irreducible matrix. The circulant generator matrix G_N is irreducible.

- A: (T,T) B: (T,F) C: (F,T) D: (F,F)

8.6.4.1 Properties of Irreducible Matrices

Fact 8.13 (Taussky's 1948 theorem [249, Theorem 1.11, p. 14]) If $A \in \mathbb{C}^{N \times N}$ is irreducibly diagonally dominant, then A is nonsingular. This property extends the sufficient condition for invertibility provided by Geršgorin's theorem.

Fact 8.14 If an $N \times N$ nonnegative matrix A is irreducible, then $(I+A)^{N-1}$ is a positive matrix [250, §8.2].

Fact 8.15 For a square nonnegative matrix A , the corresponding graph $\mathcal{G}(A)$ is strongly connected iff A (or equivalently its adjacency matrix) is irreducible.

The proof is essentially due to (8.44). In words, $[A^m]_{i,j} > 0$ means that we can reach node i from state j in m steps in the directed graph $\mathcal{G}(A)$.

We have seen that any primitive matrix is irreducible, but not the other way around in general. To fully describe how these two families of matrices are related, we need one more definition.

8.6.5 Matrix Period

Definition For a square nonnegative matrix, the period of the i th index is the greatest common divisor of all $m \in \mathbb{N}$ such that $[A^m]_{i,i} > 0$, if any such m exists.

Definition If the period of every index exists, and the periods all equal some $m > 1$, then we call m the period of A .

 However, we will *not* call such a matrix “periodic” because that has a different meaning (Problem 8.8).

Definition If the period of every index exists and is 1, then A is called aperiodic.

Otherwise, we will call it not aperiodic. Again, we will not call it periodic!

Example 8.20 Consider $A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$, for which $\mathcal{G}(A)$ is in Fig. 8.6(a). We have $A^m = A \forall m \in \mathbb{N}$, so the period of index 1 is simply 1, and index 2 does not have a period. Thus, this matrix is not aperiodic.

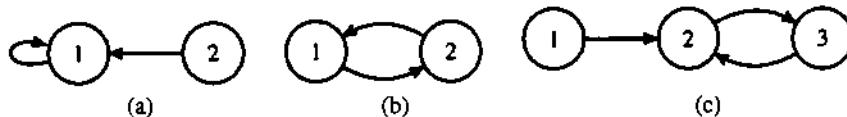


Figure 8.6 Three examples of not aperiodic matrices.

Example 8.21 Consider $A = \begin{bmatrix} 0 & 3 \\ 3 & 0 \end{bmatrix}$, for which $A^m = \begin{bmatrix} 3^m & 0 \\ 0 & 3^m \end{bmatrix}$ for m even and $A^m = \begin{bmatrix} 0 & 3^m \\ 3^m & 0 \end{bmatrix}$ for m odd, and $\mathcal{G}(A)$ is in Fig. 8.6(b). Thus, $[A^m]_{i,i} > 0$ for $m = 2, 4, 6, \dots$ for both $i = 1$ and $i = 2$, so the period is 2 for both indexes. We say this matrix has period 2. We also say that this matrix is not aperiodic.

Example 8.22 Consider $A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$, for which $A^m = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ for m even and $A^m = A$ for m odd, and $\mathcal{G}(A)$ is in Fig. 8.6(c). Thus, index 1 does not have a period, so this matrix is not aperiodic.

Example 8.23 Consider $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, for which $A^2 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$ and A^m is positive for all $m > 1$, and $\mathcal{G}(A)$ is in Fig. 8.7. The period of index 1 is $\gcd(1, 2, 3, \dots) = 1$ and the period of index 2 is $\gcd(2, 3, \dots) = 1$. This matrix is aperiodic.

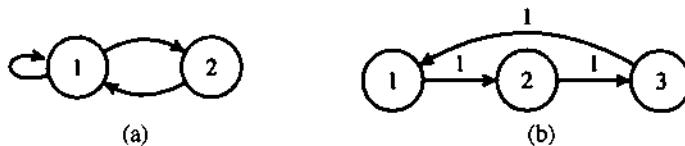


Figure 8.7 Graphs for an aperiodic matrix (a) and a matrix with period 3 (a).

Q8.17 Any square positive matrix is aperiodic.

A: True B: False

Fact 8.16 For a nonnegative irreducible matrix, the period of the i th index always exists and is the same for all i .

Again, if the value exceeds 1, then it is called the period of the matrix.

Example 8.24 The adjacency matrix corresponding to a loop of three states is

$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, for which $P^2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$, $P^3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $P^4 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, and $\mathcal{G}(P)$ is in Fig. 8.7(b). This matrix is nonnegative and irreducible, and it has period 3.

Fact 8.17 A square nonnegative matrix is primitive iff it is irreducible and aperiodic.

If A is primitive, then there is $K \in \mathbb{N}$ such that A^K is positive and hence A is irreducible. Furthermore, A^m is positive for all $m \geq K$ so the GCD must be 1 and hence A is aperiodic. The converse is more subtle.

Fact 8.18 If an nonnegative irreducible matrix A has at least one nonzero diagonal element, then A is primitive [250, p. 678].

Because then, from any state i we can return to that state in some m_i steps and also in $m_i + 1$ steps, so the i th period is 1 for all i , and hence the matrix is aperiodic.

Table 8.1. Relating matrix types and graph types.

| Matrix type | Graph type | Notes |
|-----------------------|-------------------------|--|
| Nonnegative | Weighted directed graph | Weights are positive matrix elements |
| Adjacency | Directed graph | No weights, just arrows (allow loops) |
| Transition/stochastic | Markov chain | Columns sum to 1 |
| Irreducible | Strongly connected | Can reach any node from any node |
| Primitive | (?!) | $\exists m \in \mathbb{N}$ such that can reach any node from any other node in m steps |
| Positive | Fully connected | One step between any nodes |

Example 8.25 However, having a nonzero diagonal element is not a necessary condition for a matrix P to be primitive. Consider the (transition) matrix

$$P = \begin{bmatrix} 0 & 1/2 & 1 \\ 1 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix}$$

with $G(P)$ as in Fig. 8.8. Here, P^5 is positive, so P is irreducible and aperiodic, and hence primitive, even though P has all-zero diagonals.

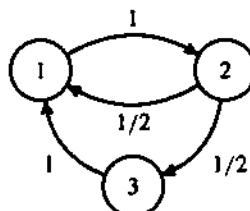
**Figure 8.8** A primitive case with zero diagonals.

Table 8.1 summarizes the relationships between square nonnegative matrix types and graph types.

8.7 Nonnegative Matrices and Perron–Frobenius Theorems

Now we examine eigenproperties, starting with the broadest category: square nonnegative matrices.

8.7.1 Perron–Frobenius for Square Nonnegative Matrices

Fact 8.19 If A is square and nonnegative, then we have the following bounds on its spectral radius [247]:

- Row sums provide bounds on the spectral radius:

$$\min_i \sum_j a_{ij} \leq \rho(A) \leq \max_i \sum_j a_{ij} = \|A\|_\infty.$$

- Column sums provide bounds on the spectral radius:

$$\min_j \sum_i a_{ij} \leq \rho(A) \leq \max_j \sum_i a_{ij} = \|A\|_1.$$

Combining yields the following bounds on the spectral radius of a nonnegative matrix:

$$\max\left(\min_i \sum_j a_{ij}, \min_j \sum_i a_{ij}\right) \leq \rho(A) \leq \min\left(\max_j \sum_i a_{ij}, \max_i \sum_j a_{ij}\right). \quad (8.46)$$

The upper bounds follow from the fact that $\rho(A) \leq \|A\|$ for any induced matrix norm. The proof of the lower bound is given in [251, p. 52] and [252, p. 37].

Example 8.26 For a permutation matrix, the row and column sums are all 1 so we know $\rho = 1$.

Fact 8.20 If P is a stochastic matrix, that is, $\mathbf{1}'P = \mathbf{1}'$, then $\rho(P) = 1$.

Proof. Clearly, one eigenvalue of P is 1, so we know $1 \leq \rho(P)$. Yet the upper bound in (8.46) is also 1, because the column sums are all 1. \square

The Perron–Frobenius theorem for any square, nonnegative matrix A states the following:

- $r \triangleq \rho(A) \geq 0$ is an eigenvalue of A . In other words, the largest-magnitude eigenvalue is a real, nonnegative eigenvalue. Think $\lambda_{(1)} = r \geq 0$, where $\lambda_{(1)}$ denotes the first eigenvalue when sorted by magnitude. We call r the Perron root or Perron–Frobenius eigenvalue.
- A has a (right) eigenvector v with nonnegative elements corresponding to that eigenvalue, that is, $Av = rv$.
- There likewise exists a left eigenvector w having all nonnegative elements for which $w'A = rw'$. We refer to such v and w as right and left Perron vectors, respectively.

Because A and A' have the same eigenvalues, the existence of the nonnegative left eigenvector follows directly from the existence of the nonnegative right eigenvector, because A' is also a nonnegative matrix.

The fact that the largest-magnitude eigenvalue is real and nonnegative takes some work to show, and does not follow from the Gershgorin disk theorem. One version uses the proof for positive matrices based on Gelfand's formula (6.81), followed by noting that any nonnegative matrix is a limit of positive matrices.

By the definition of spectral radius, $|\lambda_i(A)| \leq \rho(A)$, so for a square nonnegative matrix, $|\lambda_i(A)| \leq r = \lambda_{(1)}$.

This inequality is not strict and in general there can be multiple eigenvalues with the same magnitude. In fact, there can be multiple eigenvalues all equal to ρ , for

example for $A = I$. This matters because our convergence theory for the power iteration assumed there was one eigenvalue with dominant magnitude. So we cannot say much about powers of nonnegative matrices.

Likewise, there can be multiple noncolinear right nonnegative eigenvectors having an eigenvalue where $|\lambda_i| = r$. In other words, we cannot say anything about the uniqueness (or multiplicity) of an eigenvector having nonnegative elements. This means we cannot say much about the limiting behavior of Markov chains having merely nonnegative transition matrices.

Example 8.27 For the nonnegative matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ the eigenvalues are 1, 1, so $\rho = 1$.

Any nonzero vector of the form $\begin{bmatrix} x \\ y \end{bmatrix}$ with $x, y \geq 0$ is a (right and left) Perron vector.

Example 8.28 For $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ the eigenvalues are $+1, -1$, so $\rho = 1$. The vector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

is a (right and left) Perron vector; the vector $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$ is also an eigenvector whose eigenvalue, -1 , also has magnitude 1. Powers A^k do not converge. In this case, $\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$ is the unique nonnegative unit-norm eigenvector corresponding to the eigenvalue $\lambda = 1 = \rho$.

8.7.2 Perron–Frobenius for Nonnegative Irreducible Matrices

The Perron–Frobenius theorem for any (square) nonnegative irreducible matrix A states that:

- All the conclusions for (square) nonnegative matrices hold.
- $r \triangleq \rho(A) > 0$ is a (real) eigenvalue of A . In other words, the largest-magnitude eigenvalue is a real, *positive* eigenvalue. Think $\lambda_{(1)} = r > 0$. We call r the Perron root or Perron–Frobenius eigenvalue.
- That eigenvalue is simple (also termed unique), that is, has multiplicity one, meaning both its algebraic multiplicity is one (nonrepeated root of the characteristic polynomial) and its geometric multiplicity is one.
- If A has period m , then there are exactly m eigenvalues for which $|\lambda| = r = \rho(A)$.
- There exists a unit-norm eigenvector v with eigenvalue r , that is, $Av = rv$, where v has all *positive* elements.
- There likewise exists a unit-norm left eigenvector w having all positive elements for which $w'A = rw'$. We call this v and w the right and left Perron vectors, respectively.
- Uniqueness: All other (left and right) unit-norm eigenvectors have negative and/or non-real elements. So v and w are unique in that sense.

Recall that for a nonnegative irreducible matrix, the period of the i th index exists and is the same for all i .

- If that period is $m > 1$, then the matrix has period m and there are m eigenvalues where $|\lambda_i| = r$.
- If that period is $m = 1$, then the matrix is aperiodic and hence primitive, the case discussed next.

8.7.3 Perron–Frobenius for Primitive Matrices

To make stronger statements about uniqueness, we make stronger assumptions. The Perron–Frobenius theorem for any (square) primitive matrix A states that:

- All the conclusions for (square) nonnegative irreducible matrices hold.
- $r \triangleq \rho(A) > 0$ is a (real) eigenvalue of A . In other words, the largest-magnitude eigenvalue is a real, *positive* eigenvalue. Think $\lambda_{(1)} = r > 0$. We call r the Perron root or Perron–Frobenius eigenvalue.
- That eigenvalue has multiplicity one, both algebraic multiplicity and geometric multiplicity.
- Uniqueness of *magnitude*: For any other eigenvalue λ , $|\lambda| < r = \rho(A)$, that is, there is no other λ such that $|\lambda| = r$.
- There exists a unit-norm eigenvector v with eigenvalue r , that is, $Av = rv$, where v has all *positive* elements.
- There likewise exists a unit-norm left eigenvector w having all positive elements for which $w'A = rw'$.

We call this v and w the right and left Perron vectors, respectively.

- Uniqueness: All other (left and right) unit-norm eigenvectors have negative and/or nonreal elements. So v and w are unique in that sense.

Because a primitive matrix A has one eigenvalue whose magnitude dominates all others, the power iteration converges to the Perron vector (with no sign issues if initialized with a positive vector).

All of these properties hold for any (square) positive matrix because any such matrix is also primitive.

8.7.4 Perron–Frobenius for Stochastic Matrices

We will apply all the above definitions and properties to study Markov chains. The transition matrix P for a Markov chain is a stochastic matrix, meaning that its columns sum to one. In other words, $\mathbf{1}'P = \mathbf{1}$. As explained in Section 8.7, a stochastic matrix has $\rho(P) = 1$. All of the preceding Perron–Frobenius theory (for nonnegative, irreducible, or primitive matrices, whichever case is applicable) applies readily to any stochastic matrix simply by taking the special case that $\rho(P) = 1$.

Q8.18 Are these two statements true or false? There exists a 4×4 stochastic matrix having the four eigenvalues $\{\pm 1, \pm i\}$. There exists a 4×4 stochastic matrix for which every nonnegative vector in \mathbb{R}^4 is an eigenvector.

A: (T,T)

B: (T,F)

C: (F,T)

D: (F,F)

Explore 8.12 Give an example of an aperiodic stochastic matrix whose graph has a transient state.

Next, we apply these definitions and properties to study Markov chains.

8.8 Markov Chains

Definition A (first-order) Markov chain is a random process (or random sequence, that is, a sequence of random variables) X_1, X_2, \dots whose joint distribution is such that the conditional distribution of X_{k+1} given the values (states) of all preceding variables depends only on the preceding variable's value (state) and not how the process reached that state:

$$p(X_{k+1} = x | X_k = x_k, X_{k-1} = x_{k-1}, \dots, X_1 = x_1) = p(X_{k+1} = x | X_k = x_k). \quad (8.47)$$

Example 8.29 You start with \$2 and bet \$1 on the outcome of a fair coin toss. If you win, you then have \$3 and if you lose you then have \$1. And you continue until, with probability 1, you eventually lose all your money or you stop betting because you reach your goal of, say, \$5; see Fig. 8.9. If at time k you have, say \$3, then

$$\begin{aligned} p(X_{k+1} = x | X_k = 3, X_{k-1} = x_{k-1}, \dots, X_1 = x_1) &= p(X_{k+1} = x | X_k = 3) \\ &= \begin{cases} 1/2, & x = 4, \\ 1/2, & x = 2, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

In words, the next state will be \$2 or \$4, (with probability 1/2), regardless of how you reached the state of \$3.

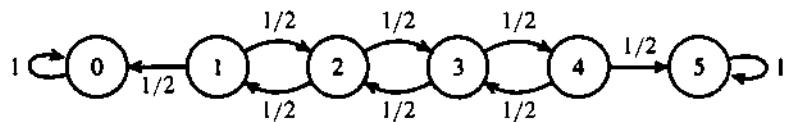


Figure 8.9 A Markov chain for coin tossing.

We focus on cases (like Example 8.29) where the random variables take a finite set of discrete values. In this case, without loss of generality we can use the values $\{1, \dots, N\}$, where N is the number of states (i.e., values). In this typical case, the statistical properties are governed by the transition probabilities:

$$p(X_{k+1} = i | X_k = j), \quad i, j = 1, \dots, N. \quad (8.48)$$

In the typical case where the distributions are independent of the “time” index k , we call it a time-homogeneous Markov chain, and we often describe the Markov chain by a weighted directed graph that illustrates the transition probabilities between states, for example, Fig. 8.9.

- Circles represent states.
- Arrows are labeled with transition probabilities.
- We draw arrows in such graphs only for nonzero transition probabilities.

Because our focus is matrix methods, not graphs or random processes, naturally we use an $N \times N$ matrix to describe these probabilities, called the transition matrix:

$$P_{i,j} \triangleq p(X_{k+1} = i | X_k = j), \quad i,j = 1, \dots, N. \quad (8.49)$$



In some texts the transpose of P is also called the transition matrix.

Example 8.30 See Fig. 8.10 for an example. The sum of the elements of each column is unity, that is, $\mathbf{1}'_3 P = \mathbf{1}'_3$. So P is a stochastic matrix.

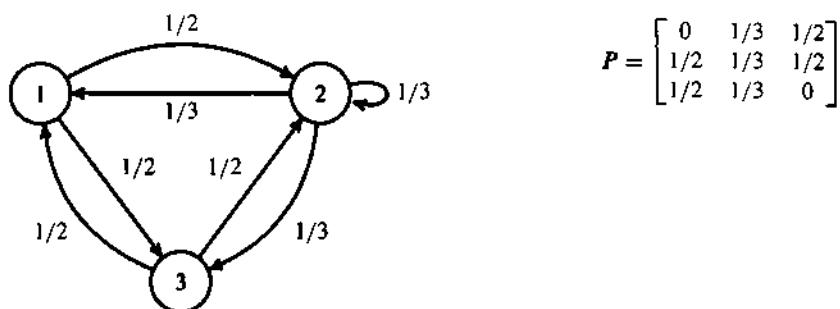


Figure 8.10 Markov chain example: graph and transition matrix.

Any $N \times N$ transition matrix P has useful properties.

- All transition matrices are square and nonnegative.
- Generalizing the previous example, the columns always sum to unity:

$$\mathbf{1}'_N P = \mathbf{1}'_N \text{ because } \sum_{n=1}^N p(X_{k+1} = n | X_k = j) = 1. \quad (8.50)$$

In words, the transition probabilities for leaving the j th state sum to one for every j .

Fact 8.21 By (8.50), an $N \times N$ transition matrix P always has $\mathbf{1}_N$ as a left eigenvector with eigenvalue 1.

- Because all columns sum to unity, $\rho(P) \leq 1$ so $|\lambda_i(P)| \leq 1$. (No eigenvalues are outside the unit disk.)
- Thus, $\rho(P) = 1$.

In general, there can be multiple nonnegative eigenvectors with their eigenvalue equal to unity.

Example 8.31 The following transition matrix has two linearly independent, unit-sum, nonnegative eigenvectors with eigenvalue 1:

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \implies v_1 = \begin{bmatrix} 1/2 \\ 1/2 \\ 0 \end{bmatrix} \text{ and } v_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \quad (8.51)$$

Q8.19 Are these two statements true or false? This P is irreducible. This P is primitive.
 A: (T,T) B: (T,F) C: (F,T) D: (F,F)

8.8.1 Equilibrium Distribution(s) of a Markov Chain

Markov chains are a rich subject with numerous applications and theoretical considerations. One particular aspect that we can examine using matrix methods is the existence of an equilibrium distribution (or stationary distribution or steady-state distribution).

Definition Let π denote an N -dimensional vector $\pi = \begin{bmatrix} \pi_1 \\ \vdots \\ \pi_N \end{bmatrix}$. We call π a stationary distribution (or equilibrium distribution or steady-state distribution) of a Markov chain having transition matrix P iff

$$\pi_n \geq 0, \quad n = 1, \dots, N, \quad \sum_{n=1}^N \pi_n = 1' \pi = 1, \quad P\pi = \pi. \quad (8.52)$$

In words, π has nonnegative elements that sum to unity and is an eigenvector of P with eigenvalue 1. Such a π is sometimes called a stochastic eigenvector. See Example 8.31 for two such π .

Because a transition matrix P is nonnegative and has $\rho(P) = 1$, the Perron–Frobenius theorem for nonnegative matrices ensures that there always exists such a (nonzero) eigenvector having nonnegative elements with eigenvalue 1, so we can normalize that eigenvector to sum to unity, establishing the existence of an equilibrium distribution.

However, that eigenvalue of 1 need not be unique and there can be multiple stochastic eigenvectors in general; see Example 8.31.

Fact 8.22 If P is irreducible, then the equilibrium distribution is unique. (This follows from the Perron–Frobenius theorem.)

The importance of an equilibrium distribution is the following. In general, by the law of total probability,

$$p(X_{k+1} = i) = \sum_{j=1}^N p(X_{k+1} = i | X_k = j) p(X_k = j). \quad (8.53)$$

The chain is in equilibrium or steady state when $p(X_{k+1} = n) = p(X_k = n) = \pi_n$, that is, when

$$\pi_i = \sum_{j=1}^N p(X_{k+1} = i \mid X_k = j) \pi_j = \sum_{j=1}^N P_{ij} \pi_j.$$

In matrix–vector form,

$$\boldsymbol{\pi} = \mathbf{P}\boldsymbol{\pi}. \quad (8.54)$$

In other words, an equilibrium distribution $\boldsymbol{\pi}$ is a stochastic eigenvector of the transition matrix \mathbf{P} associated with eigenvalue 1.



Caution: “steady state” does not mean “stuck in one state” in general, although one of the two equilibrium distributions for the preceding three-state example happens to have that property.

Example 8.32 Returning to the example in Fig. 8.10, the eigenvalues of \mathbf{P} are $\{1, -1/2, -1/6\}$ and \mathbf{P} is a primitive matrix. A (unit-norm) eigenvector associated with eigenvalue 1 is

$$\boldsymbol{v}_1 = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix} / \sqrt{17},$$

for which $\|\boldsymbol{v}_1\|_2 = 1$. This is not the normalization we want for a probability distribution. We want $\sum_n \pi_n = 1$, that is, $\mathbf{1}'_N \boldsymbol{\pi} = 1$. So define $\boldsymbol{\pi} = \boldsymbol{v}_1 / (\mathbf{1}'_N \boldsymbol{v}_1)$, which will also take care of any sign flip if needed, for which

$$\boldsymbol{\pi} = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix} / 7.$$

This stochastic eigenvector is the equilibrium distribution and its elements show the (equilibrium) probability of each state. State 2 is more likely, as is intuitive in light of the graph.

8.8.2 Limiting Distribution(s) of a Markov Chain

We can also use matrix methods to examine the limiting behavior or asymptotics of Markov chains. If we start the chain in some state (perhaps chosen at random, or perhaps not) and run the system for a long time, what is the probability of being in the n th state? Formally, we want to examine

$$\lim_{k \rightarrow \infty} p(X_k = n) \text{ for } n = 1, \dots, N. \quad (8.55)$$

Fact 8.23 If the transition matrix P is primitive, then it follows from the Perron–Frobenius theorem for primitive matrices [247] (see the proof sketch in (8.59)–(8.60)) that:

- the equilibrium distribution π_* is unique (as stated previously); and
- the limit of powers of P is a special rank-1 matrix:

$$\lim_{k \rightarrow \infty} P^k = P_* = \pi_* \mathbf{1}'_N = [\pi_* \quad \cdots \quad \pi_*]. \quad (8.56)$$

Thus, when P is primitive, regardless of the initial distribution,

$$\lim_{k \rightarrow \infty} p(X_k = n) = [\pi_*]_n, \quad (8.57)$$

that is, the limiting distribution of a Markov chain having a primitive transition matrix is the unique equilibrium distribution. Put another way, for any initial distribution π_1 ,

$$P^k \pi_1 \rightarrow \pi_*, \text{ that is, } \|P^k \pi_1 - \pi_*\| \rightarrow 0. \quad (8.58)$$

Example 8.33 For our running example in Fig. 8.10 with three states, the transition matrix is primitive.

```
</> 8.2 julia> P = [0 1/3 1/2; 1/2 1/3 1/2; 1/2 1/3 0]
      0.0  0.333333  0.5
      0.5  0.333333  0.5
      0.5  0.333333  0.0

julia> P^5
      0.270062  0.285751  0.301312
      0.428627  0.428498  0.428627
      0.301312  0.285751  0.270062
```

Q8.20 If a transition matrix P is a primitive matrix, then $\lim_{k \rightarrow \infty} P^k$ is a projection matrix.

A: True

B: False

Proof (of (8.56) in the case where P is diagonalizable, with magnitude-sorted eigenvalues).

$$\begin{aligned} P = V \Lambda V^{-1} \implies P^k &= V \Lambda^k V^{-1} \rightarrow P_* \triangleq V \text{Diag}\{1, 0, \dots, 0\} V^{-1} \\ &= V e_1 e_1' V^{-1} = v_1 e_1' V^{-1} \end{aligned} \quad (8.59)$$

as $k \rightarrow \infty$, because the Perron–Frobenius theorem says that there is only one eigenvalue $\lambda_{(1)} = 1$ and all others have $|\lambda_i| < 1$, so $\lambda_i^k \rightarrow 0$ as $k \rightarrow \infty$. So, we see that P_* is a rank-1 outer product.

Now, we know that $\mathbf{1}' P = \mathbf{1}'$, and it follows from the law of total probability that $\mathbf{1}' P^k = \mathbf{1}'$ for all $k \in \mathbb{N}$. Thus, in the limit we also have $\mathbf{1}' P_* = \mathbf{1}'$, so

$$\begin{aligned} \mathbf{1}' = \mathbf{1}' P_* &= \mathbf{1}' v_1 e'_1 V^{-1} = (\mathbf{1}' v_1) e'_1 V^{-1} \implies e'_1 V^{-1} = \mathbf{1}' / (\mathbf{1}' v_1) \\ &\implies P_* = \frac{v_1}{\mathbf{1}' v_1} \mathbf{1}' = \pi \mathbf{1}'. \end{aligned} \quad (8.60)$$

The proof for nondiagonalizable matrices uses the Jordan normal form (3.7) in a similar way. \square

Having P be primitive is sufficient, but not necessary, for $\{P^k\}$ to converge.

Example 8.34 Consider $P = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$, for which $P^k = P$ so the matrix sequence $\{P^k\}$ converges (immediately) to $\begin{bmatrix} 1 \\ 0 \end{bmatrix} [1 \ 1] = \pi_* \mathbf{1}_2'$ and the vector sequence $\{P^k \pi_1\}$ converges (immediately) to $\pi_* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ for any initial distribution π_1 . Yet P is not primitive.

8.8.3 Markov Chains with Strongly Connected Graphs

Example 8.35 The transition matrix corresponding to the loop of three states shown in Fig. 8.11(a) is

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \text{ for which } P^2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, P^3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, P^4 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

This matrix is nonnegative and irreducible, and has period 3. It has three eigenvalues whose magnitudes are 1. Its (only) equilibrium distribution is $\pi_* = \mathbf{1}_3 / 3$. However, the matrix power sequence $\{P^m\}$ does not converge.

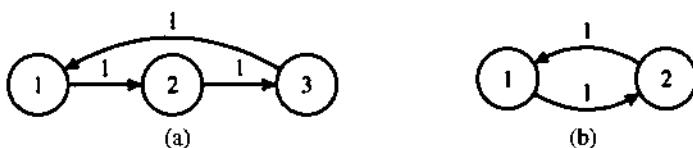


Figure 8.11 Matrix chains for a loop of three (a) or two (b) states that are strongly connected.

Example 8.36 Consider Fig. 8.11(b).

Q8.21 The corresponding transition matrix $P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ is primitive.

A: True

B: False

- The corresponding eigenvalues are $\{+1, -1\}$.
- This 2×2 matrix is irreducible, and its unit-norm Perron vector is $\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$.
- So the (unique) equilibrium distribution of this chain is $\pi_* = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$.

Q8.22 Let $\pi^{(n)} \triangleq P^{n-1}\pi^{(1)}$, where

$$\pi^{(1)} \triangleq \begin{bmatrix} P\{X_1 = 1\} \\ P\{X_1 = 2\} \end{bmatrix}$$

denotes an initial probability distribution for Fig. 8.11(b). Does $\pi^{(n)}$ approach the equilibrium distribution π_* as $n \rightarrow \infty$?

- A: Yes, for any $\pi^{(1)} \geq 0$ such that $1'_2 \pi^{(1)} = 1$.
- B: Yes, but only for certain $\pi^{(1)}$ choices.
- C: No, never, because P^n oscillates between $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.
- D: None of these.

As discussed in Section 8.6.3, a Markov chain with a strongly connected graph cannot have transient or absorbing states (see Fig. 8.5).

Explore 8.13 Express the existence of a transient state in terms of transition matrix P .

Fact 8.24 If a Markov chain has a strongly connected graph, then the following properties hold.

- Its transition matrix is irreducible.
- An equilibrium distribution exists and is unique and has all positive entries.
- The associated eigenvalue is 1 and is unique. (However, there can be other eigenvalues whose magnitude is 1.)

We *cannot* conclude that the Markov chain approaches in the limit that equilibrium distribution. Making the stronger assumption that the transition matrix is primitive ensures that limiting behavior.

Example 8.37 Figure 8.12 shows a Markov chain with a strongly connected graph and thus an irreducible transition matrix. From any state it takes exactly six steps to return to that state, no matter which path one takes. In other words, $[P^k]_{ii} = 1 > 0$ when $k = 6, 12, 18, \dots$ but is zero otherwise. Thus the transition matrix has period = 6, and hence P is not aperiodic, so P is not primitive.

8.8.4 Google's PageRank Method

This section uses the preceding ideas to summarize Google's PageRank method [17, 247, 253, 254] (US Patent 6285999). To rank web pages to present after a search, two natural criteria are:

- the relevance of each page (number of related terms, etc.);
- the "importance" (link popularity) of each web page.

Quantifying relevance seems challenging, but one way to quantify importance is to imagine making a random walk between pages, choosing the next page *at random* from all the out links on the current page. This process is a Markov chain having a

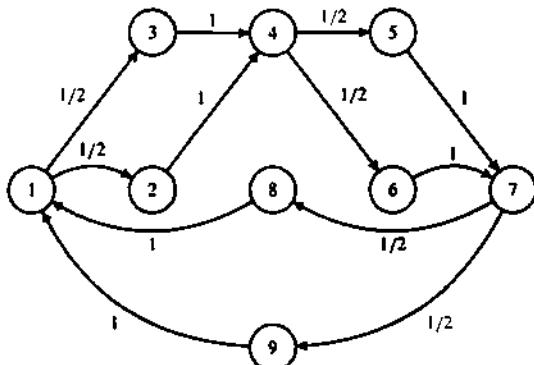


Figure 8.12 Markov chain with strongly connected graph.

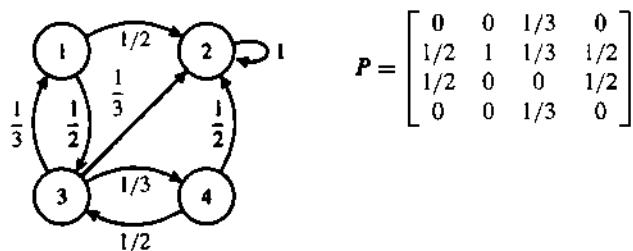


Figure 8.13 Out links example.

transition matrix as illustrated in Fig. 8.13:

$$p_{ij} = \begin{cases} 1, & i = j \text{ and page } j \text{ has no out links;} \\ 1/N_j, & i \neq j \text{ and page } j \text{ has } N_j \text{ out links, one of which is page } i; \\ 0, & \text{otherwise.} \end{cases} \quad (8.61)$$

The idea is that someone who hypothetically walked pages at random this way would tend to visit more important pages more often because important pages have many incoming links. So, to rank pages by importance we would like to find “the” equilibrium distribution of this Markov chain and then rank importance based on sorting the probabilities from largest to smallest.

In practice, the transition matrix P has columns that sum to one by construction, but the graph of web page out links is *not* strongly connected, so P is not irreducible. The web:

- has absorbing states (“dangling nodes”), for example PDF documents with no links and many business sites;
- is not strongly connected (e.g., it is probably not possible to reach umich.edu from whitehouse.gov).

Thus, there is not a unique equilibrium distribution.

One solution is to instead use a modified transition matrix for some $0 < \alpha < 1$:

$$\tilde{P}_\alpha \triangleq \alpha P + (1 - \alpha) \frac{1}{N} \mathbf{1}_N \mathbf{1}'_N = \alpha P + (1 - \alpha) \underbrace{\begin{bmatrix} 1/N & & 1/N \\ & \ddots & \\ 1/N & \cdots & 1/N \end{bmatrix}}_{\text{jump at random to any of the } N \text{ known web pages}}. \quad (8.62)$$

jump at random to any of the N known web pages \rightarrow

This matrix \tilde{P}_α is clearly positive by construction, so it has a unique equilibrium distribution π having all positive elements. This vector π has been called the \$25 billion eigenvector [17].

8.8.4.1 PageRank Algorithm

- Generate adjacency matrix by web crawling. It is (extremely) sparse, so store accordingly.
- Form the transition matrix P . Again, this is (extremely) sparse.
- Select a value for damping factor α . The original paper by Sergei Brin and Larry Page [254] used $\alpha = 0.85$, apparently based on how often users return to their bookmarks.
- Pick an initial distribution vector π_0 at random from the unit simplex (nonnegative entries and sums to 1).
- Perform a modified version of the power iteration where we keep the sum equal to one instead of the Euclidean norm equal to one:

$$\pi_{k+1} = \tilde{P}_\alpha \pi_k = \alpha P \pi_k + \frac{1 - \alpha}{N} \mathbf{1}_N. \quad (8.63)$$

- Perform multiple iterations of (8.63). The patent [253] mentions 2 to 100 iterations; [247] used 10–50.
- Use the final π vector (sorting?) to quantify the “importance” of web pages.

Example 8.38 Applying this method to the preceding four-state example with $\alpha = 0.85$ yields

$$\pi = \begin{bmatrix} 0.0634 \\ 0.7818 \\ 0.0914 \\ 0.0634 \end{bmatrix}.$$

State 2 is the most important and state 3 is the next; states 1 and 4 are tied for last. In this particular example we could have ranked states based on the number of incoming links. However, Page’s patent [253] argues that link counting is suboptimal in general; it is more informative if your web page is linked by other important web pages, not just by many (possibly spam) web pages.

8.8.4.2 PageRank Computation

A key step in the algorithm is multiplying the $N \times N$ sparse matrix P by a (nonsparse) vector: $P\pi_k$. Using ordinary matrix multiplication this would require N^2 operations, where N is in the trillions.

Fortunately, P is extremely sparse; most web pages link to just a few other pages, and the average number of outgoing links is probably roughly a constant that does not really scale with N . Using the fact that P is sparse, with an average of, say, $L \ll N$ links per page, the product $P\pi_k$ requires “just” $O(NL)$ operations. This is still enormous, requiring massive computation and considerable energy.

If N is about 100 trillion, then $1/N$ is about 10^{-14} , which is smaller than the machine precision (about 10^{-7}) of 32-bit floating point numbers, so apparently π_k must be stored as a 64-bit (double precision) vector. The number of bytes just to store π is then $8N$, which is over 700 petabytes. Surely Google must have some tricks to deal with this.

PageRank is independent of a user’s query so it can be done relatively infrequently instead of for every individual search. Numerous practical issues arise to deal with websites trying to boost scores artificially.

In summary, PageRank, one of the most influential algorithms of modern times, has its roots in Markov chains, directed graphs, positive matrices, and eigenvector computation using the power iteration.

8.9 Graph Laplacian and Spectral Clustering

- ◆◆ This section defines the graph Laplacian and describes some of its properties. Two (of many) applications that use this tool are spectral clustering and nonlinear dimensionality reduction using Laplacian eigenmaps.

8.9.1 Clustering

Given N unlabeled data vectors $x_1, \dots, x_N \in \mathbb{F}^M$, clustering is the process of partitioning those vectors into K “similar” groups. We previously introduced subspace clustering in Section 7.8.1. That clustering approach involves fairly challenging optimization problems. For data that is not well modeled by a union of subspaces, or when one would like to avoid the computational cost of subspace clustering, there are many alternatives. We focus here on an approach called spectral clustering that uses a weighted graph and its graph Laplacian. See [255] for a survey.

Example 8.39 As a concrete example, Fig. 8.14 shows $N = 20$ unlabeled images of handwritten digits, each of which can be considered to be a vector in \mathbb{R}^M where $M = 27 \cdot 28 = 756$. The goal of clustering would be to group together the “4” and “9” digits.

| | | | | | |
|---|---|---|---|---|---|
| 4 | 4 | 9 | 9 | 9 | 9 |
| 4 | 9 | 8 | 9 | 4 | 4 |
| 9 | 9 | 4 | 4 | 9 | 4 |
| 9 | 4 | 4 | 4 | 9 | 4 |

Figure 8.14 Unlabeled handwritten image data for unsupervised clustering.

- ④ Demo 8.1 applies spectral clustering to unlabeled handwritten digit images like those shown in Fig. 8.14.

8.9.2 Weighted Graph Based on Similarity

The first step in spectral clustering is to define a similarity function and a weighted graph based on it. We first select a function $s(\mathbf{x}_i, \mathbf{x}_j)$ that quantifies the “similarity” between each pair of data points. Typically the similarity function $s: \mathbb{R}^M \times \mathbb{R}^M \mapsto \mathbb{R}$ satisfies the symmetry property $s(\mathbf{x}, \tilde{\mathbf{x}}) = s(\tilde{\mathbf{x}}, \mathbf{x}) \forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^M$.

Example 8.40 A typical choice is the Gaussian similarity function

$$s(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/(2\sigma^2)},$$

where σ is a parameter that influences “neighborhood” width. This function is 1 (maximal similarity) if \mathbf{x}_i and \mathbf{x}_j are identical, and decreases monotonically as the distance between them increases.

We define a weighted graph, called a similarity graph, $G = (V, E)$ where vertex v_i represents a data point \mathbf{x}_i , and two vertices i and j are connected iff the nonnegative edge weight w_{ij} is nonzero, where w_{ij} is defined in terms of the similarity function values. The simplest choice lets each edge weight be the corresponding similarity value:

$$w_{ij} = s(\mathbf{x}_i, \mathbf{x}_j). \quad (8.64)$$

For a positive-valued similarity function like the Gaussian one, this choice leads to a fully connected graph. There are many other choices, such as connecting only data points that are sufficiently close, for example

$$w_{ij} = \begin{cases} s(\mathbf{x}_i, \mathbf{x}_j), & \|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon, \\ 0, & \text{otherwise,} \end{cases} \quad (8.65)$$

or including only the K -nearest neighbors [255].

Definition The $N \times N$ weight matrix W has elements w_{ij} .

8.9.3 Connected Components

Spectral clustering revolves around finding connected components of a graph.

Definition A subset \mathcal{A} of the graph is connected iff there is a path between any two vertices in \mathcal{A} along vertices in \mathcal{A} .

Definition A subset \mathcal{A} of the graph is a connected component iff \mathcal{A} is connected and there is *not* any path between \mathcal{A} and its complement.

8.9.4 Graph Laplacian

Having defined the weighted graph, the degree of the i th vertex is defined by

$$d_i \triangleq \sum_{j=1}^N w_{ij}. \quad (8.66)$$

The $N \times N$ diagonal matrix $D = \text{Diag}(d_i)$ is called the degree matrix. To understand the term “degree,” consider the case where the similarity function is the indicator function of two data points being within some ϵ of each other:

$$s(x_i, x_j) = \mathbb{I}_{\{\|x_i - x_j\| \leq \epsilon\}}. \quad (8.67)$$

In this case, the degree d_i is simply the number of neighbors of node i . But in general the degree is a nonnegative real number and need not be an integer.

There are several ways to define a graph Laplacian in terms of the weight matrix W and degree matrix D .

8.9.4.1 Unnormalized Graph Laplacian

The unnormalized graph Laplacian matrix is defined as $L \triangleq D - W$. It has some important properties:

- L is positive semidefinite (and hence symmetric with real, nonnegative eigenvalues).
- The smallest eigenvalue of L is 0, with corresponding eigenvector $\mathbf{1}$, because $[L\mathbf{1}]_i = d_i - [W\mathbf{1}]_i = d_i - \sum_j w_{ij} = 0$.
- For any $z \in \mathbb{F}^M$, we have

$$z'Lz = \frac{1}{2} \sum_{i,j=1}^N w_{ij}|z_i - z_j|^2, \quad (8.68)$$

which is nonnegative, verifying that L is positive semidefinite.

Proof (cf. (2.32)).

$$\begin{aligned} \mathbf{z}' \mathbf{L} \mathbf{z} &= \sum_{i=1}^N d_i |z_i|^2 - \sum_{i,j=1}^N w_{ij} z_i^* z_j \\ &= \frac{1}{2} \left(\sum_{i=1}^N d_i |z_i|^2 + \sum_{j=1}^N d_j |z_j|^2 - 2 \operatorname{real} \left[\sum_{i,j=1}^N w_{ij} z_i^* z_j \right] \right) = \frac{1}{2} \sum_{i,j=1}^N w_{ij} |z_i - z_j|^2. \end{aligned}$$

□

The following property of \mathbf{L} is crucial to spectral clustering methods because it relates connected components of the graph to the eigendecompositions of \mathbf{L} .

Fact 8.25 If the graph has K (distinct) connected components, say $\mathcal{A}_1, \dots, \mathcal{A}_K$, then the multiplicity of the zero eigenvalue is exactly K . Furthermore, eigenvectors corresponding to those K zero eigenvalues are the indicator vectors $\mathbf{1}_{\mathcal{A}_k}$, $k = 1, \dots, K$, for each of the connected components, defined by $(\mathbf{1}_{\mathcal{A}_k})_i = \mathbb{I}_{\{i \in \mathcal{A}_k\}}$.

See [255] for a proof.

8.9.4.2 Normalized Graph Laplacians

The literature has two versions of normalized graph Laplacian matrices:

$$\begin{aligned} L_{\text{symmetric}} &\triangleq \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}, \\ L_{\text{transition}} &\triangleq \mathbf{D}^{-1} \mathbf{L} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{W}. \end{aligned} \quad (8.69)$$

The $L_{\text{symmetric}}$ version is symmetric by definition. The (transpose of) the $\mathbf{D}^{-1} \mathbf{W}$ term in $L_{\text{transition}}$ corresponds to a transition matrix if we treat the graph as a Markov chain, because $\mathbf{D}^{-1} \mathbf{W} \mathbf{1} = \mathbf{D}^{-1} \mathbf{d} = \mathbf{1}$, that is, each row of $\mathbf{D}^{-1} \mathbf{W}$ sums to unity. Both $L_{\text{symmetric}}$ and $L_{\text{transition}}$ have the same (nonnegative) eigenvalues, and both have the key property that the number of zero eigenvalues is the number of connected components of the graph [255]. Furthermore, the eigenvectors of $L_{\text{transition}}$ corresponding to its zero eigenvalues are again the indicator vectors $\mathbf{1}_{\mathcal{A}_k}$ of the connected components. We focus on $L_{\text{transition}}$ hereafter because of its favorable statistical properties [255]. See [256] for more graph Laplacian theory.

8.9.5 Spectral Clustering Algorithm

Having defined all the ingredients, we can now summarize a spectral clustering algorithm, given a set of unlabeled data vectors $\mathbf{x}_1, \dots, \mathbf{x}_N$.

- Choose a similarity function $s(\mathbf{x}_i, \mathbf{x}_j)$.
- Compute an N -node weighted graph using the similarity function applied to all pairs of data points. Let \mathbf{W} and \mathbf{D} denote the corresponding weighting matrix and degree matrix.
- Compute the normalized graph Laplacian $L_{\text{transition}} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{W}$.

- Compute an eigendecomposition of the graph Laplacian, $L_{\text{transition}} = V \Lambda V^{-1}$, with eigenvalues in ascending order.
- Choose a number K of clusters to form, perhaps by examining the number of (nearly) zero eigenvalues of $L_{\text{transition}}$.
- Let $Y \triangleq [v_1 \ \cdots \ v_K]^\top = [y_1 \ \cdots \ y_N] \in \mathbb{R}^{K \times N}$ correspond to the eigenvectors of $L_{\text{transition}}$ having the K smallest nonzero eigenvalues.
- Apply K -means clustering to the columns of the matrix Y .

A computational challenge of spectral clustering is that it requires evaluating all N^2 pairs of distances between data points. The key benefit of spectral clustering is that often the K -means method works better when applied to the eigenvectors in Y than if applied directly to the original data [255].

For a complete illustration, see Demo 8.1. A cautionary note is that spectral clustering can be quite sensitive to the choice of similarity function and its parameter σ .

8.9.6 Laplacian Eigenmaps

Another application of graph Laplacians is Laplacian eigenmaps [201, 202], a type of nonlinear dimensionality reduction. This method assumes the data lies on a low-dimensional manifold and attempts to preserve local distances.



Demo 8.2 generates the Laplacian eigenmaps shown in Fig. 8.15(b) from images of rectangles having various orientations and widths, as seen in Fig. 8.15(a). These images lie on a 2D manifold.

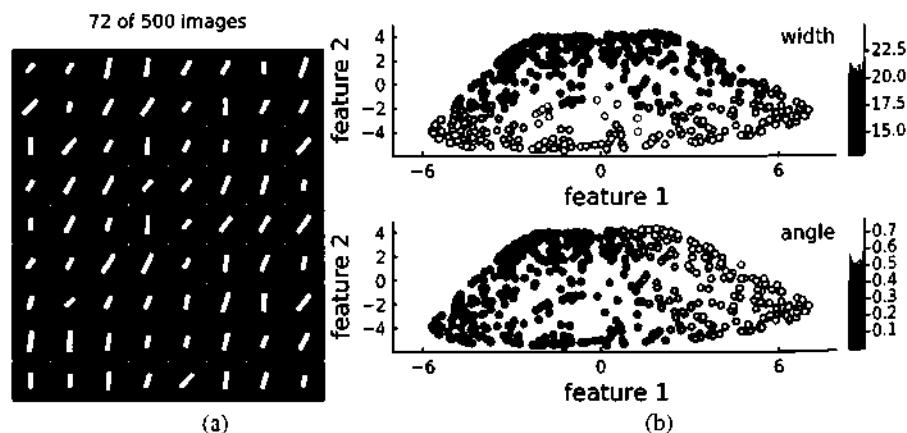


Figure 8.15 (a) Seventy-two of the 500 images of size 40×40 pixels in the Laplacian eigenmap demo. (b) Two-dimensional representation of the training images. One feature is quite correlated with rectangle width (top) and the other with orientation angle (bottom).

8.10 Summary

This chapter introduced just a few of the many important special matrices used in practice, culminating with Markov chains, the PageRank method, and spectral clustering. Another application of Perron–Frobenius theory is graph matching [257].

Table 8.2 summarizes the high-level relationships between a Markov chain transition matrix P , where $\mathbf{1}'P = \mathbf{1}'$, and its corresponding graph. Table 8.3 summarizes more detailed properties. When reviewing this chapter, consider these additional properties of square matrices:

- $\rho \geq 0$ versus $\rho > 0$;
- $\lambda_{(1)} = \rho$ versus $|\lambda_{(1)}| = \rho$;
- uniqueness of $\lambda_{(1)}$ (can any other $\lambda_i = \lambda_{(1)}$?);
- uniqueness of $|\lambda_{(1)}|$ (can any other $|\lambda_i| = |\lambda_{(1)}|$?);
- existence of $v_1 \geq 0$ versus existence of $v_1 > 0$.

Q8.23 The entries (i) and (ii) in Table 8.3 are:

- A: (n,n) B: (n,y) C: (y,n) D: (y,y)

Q8.24 For any Markov chain with an $N \times N$ circulant transition matrix, the vector $\pi = \mathbf{1}_N/N$ is an equilibrium distribution of the chain.

- A: True B: False

Q8.25 For any Markov chain with an $N \times N$ circulant transition matrix, the vector $\pi = \mathbf{1}_N/N$ is the unique equilibrium distribution of the chain.

- A: True B: False

Q8.26 For any Markov chain with an $N \times N$ circulant transition matrix, exactly one eigenvalue of that transition matrix has magnitude equal to 1.

- A: True B: False

Table 8.2 Relationships between a Markov chain transition matrix and its graph. Each row inherits the properties of the preceding rows.

| Transition matrix | Markov chain graph | Key eigenproperty |
|-------------------|--------------------------------|--|
| Nonnegative | Directed | Equilibrium distribution(s) exist with eigenvalue 1. |
| Irreducible | Strongly connected | Unique, positive equilibrium distribution π |
| Primitive | Strongly connected (and more?) | $P^k \rightarrow \pi \mathbf{1}'$ |
| Positive | “Fully” connected | |

Table 8.3 Summary of properties (yes/no) for Markov chain transition matrix P (square and nonnegative), where $1'P = 1'$. The matrix entries provide a counterexample illustrating why the entry would be “n.”

| Property | Any | Not aperiodic | Irreducible | Primitive | Positive |
|--|--|---|--|---|----------|
| $\rho(P) = 1$ | y | y | y | y | y |
| P has an eigenvalue equal to 1 | y | y | y | y | y |
| P has exactly one eigenvalue equal to 1 | — | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | y | y | y |
| P has exactly one eigenvalue whose magnitude is 1 | — | n | $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ | y | y |
| P has all nonnegative eigenvalues | — | n | n | $\begin{bmatrix} 0 & 0 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 1/2 \end{bmatrix}$ | (i) |
| P has a nonnegative eigenvector with eigenvalue $\rho(P)$ | y | y | y | y | y |
| P has a positive eigenvector with eigenvalue $\rho(P)$ | $\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$ | ? | y | y | y |
| P has a unique equilibrium distribution | — | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | y | y | y |
| Power iteration $\{P^k\pi\}$ converges to same limit $\forall\pi$ in N -simplex | — | n | $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ | y | y |
| $\{P^k\}$ converges to a rank-1 matrix | — | n | $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ | y | y |
| Can reach any state from any state | — | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | y | y | y |
| Graph is strongly connected | — | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | y | y | y |
| $\exists m \in \mathbb{N}$ such that can reach any state from any state in m steps | — | n | $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ | y | y |
| Can reach any state from any state in one step | — | n | n | $\begin{bmatrix} 0 & 0 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 1/2 \end{bmatrix}$ | (ii) |

Q8.27 If a monic polynomial has all negative coefficients (except for the leading coefficient), then the corresponding companion matrix has an eigenvector with all positive elements.

A: True

B: False

Q8.28 The companion matrix for the polynomial $p(x) = x^2 - 2x - 3$ has spectral radius:

A: 1

B: 2

C: 3

D: 4

E: 5

Figure 8.16 summarizes the relationships between many of the matrix types in this chapter. An important category that is missing from this diagram is transition matrices.

Q8.29 Complete the Venn diagram in Fig. 8.16 by adding transition matrices.

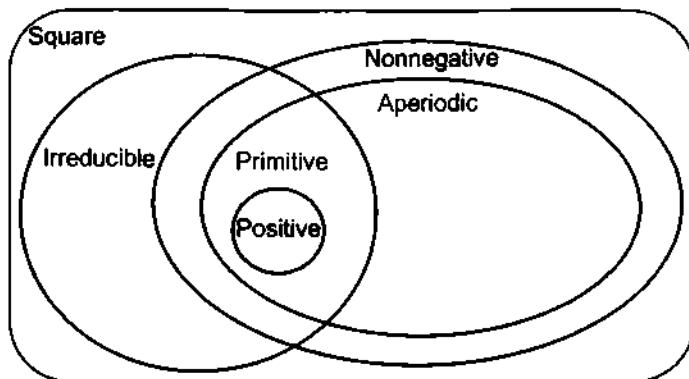


Figure 8.16 Venn diagram of square matrices.

Q8.30 Give an example of a nonzero matrix that is nonnegative but is in no other (square) category.

Solutions to Explorations

Explore 8.1 For $n = 2$ we have $A = \begin{bmatrix} -c_1 & -c_0 \\ 1 & 0 \end{bmatrix}$, which has characteristic polynomial $\det(zI - A) = \det\begin{bmatrix} z + c_1 & c_0 \\ -1 & z \end{bmatrix} = z^2 + zc_1 + c_0$. For $n = 1$ we have $A = [-c_0]$, which has characteristic polynomial $\det(zI_1 - A) = \det\{[z + c_0]\} = z + c_0$.

Explore 8.2 The polynomial coefficients c can be complex valued, and c' would take the complex conjugate of each coefficient, leading to a different polynomial.

Explore 8.3 The diagonal matrix having the roots $\{z_i\}$ on its diagonal. For example, $p(z) = z^2 - 7z + 10 = (z - 2)(z - 5)$, for which $C = \begin{bmatrix} 7 & -10 \\ 1 & 0 \end{bmatrix}$, $D = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}$. The companion matrix corresponds to the polynomial coefficients, whereas the diagonal matrix corresponds to the polynomial roots.

Explore 8.4 The numerical computation of the eigenvalues of A is imperfect, returning $3 \pm i\epsilon$ instead of $[3, 3]$.

Explore 8.5 A is $M \times M$, B is $N \times N$, C is $K \times K$.

$$\begin{aligned}(A \oplus B) \oplus C &= (I_K \otimes (A \oplus B)) + (C \otimes I_{MN}) \\ &= (I_K \otimes ((I_N \otimes A) + (B \otimes I_M))) + ((C \otimes I_N) \otimes I_M)\end{aligned}$$

$$\begin{aligned}
 &= (I_K \otimes I_N \otimes A) + (((I_K \otimes B) + (C \otimes I_N)) \otimes I_M) \\
 &= (I_{KN} \otimes A) + ((B \oplus C) \otimes I_M) \\
 &= A \oplus (B \oplus C).
 \end{aligned}$$

Explore 8.6 Consider $A = I_2$, $B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$.

Explore 8.7 $Q_{nk} = w_k^n = e^{i2\pi kn/N} = Q_{kn} = w_n^k$.

Explore 8.8 Yes. Taking $x = \text{sign} .(C_{:,1})$ achieves the upper bound because $\|x\|_\infty = 1$ and $\|Cx\|_1 = \|C_{:,1}\|_1 = \|C_{:,1}\|_1$. This equality is related to BIBO stability of LTI systems.

Explore 8.9 No. For $C = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$ we have $\sigma_1(C) = \sqrt{3} < 2 = \|C_{:,1}\|_1$.

Explore 8.10 If one can reach any node from any node in m steps, then one can also do so in any $n \geq m$ steps.

Explore 8.11 The graph is “fully” connected.

Explore 8.12 $P = \begin{bmatrix} 1/2 & 0 \\ 1/2 & 1 \end{bmatrix}$.

Explore 8.13 $\exists N \in \mathbb{N}$ such that $[P^n]_{ii} = 0 \forall n \geq N$.

Problems

Problem 8.1

- (a) Given two polynomials $p(z) = \sum_{i=0}^m \alpha_i z^i$ and $q(z) = \sum_{j=0}^n \beta_j z^j$, for $n \geq 0$ not necessarily equal to $m \geq 0$, how would you check whether p and q have a common (possibly complex) root? Assume $\alpha_m \neq 0$ and $\beta_n \neq 0$. You may use the determinant and/or the trace, but you may *not* explicitly compute the eigenvalues or eigenvectors of the respective companion matrices.

Hint: Use Section 8.2.5.

- (b) Test your approach using JULIA with some polynomials of your own design.

Problem 8.2 Enter the following JULIA code that makes some companion matrix examples.

```
</> 8.3 using LinearAlgebra: eigvals, I
n = rand(4:7)
a = randn(n+1) # random polynomial coefficients (a_0,a_1,...,a_n)
b = reverse(a) # reverse the coefficient order
# companion matrix maker:
compan = c -> [-transpose(reverse(c)); [I zeros(length(c)-1)]]
A = compan(a[1:end-1] / a[end])
B = compan(b[1:end-1] / b[end])
[eigvals(A) eigvals(B) 1 ./ eigvals(B)] # study array columns
```

What pattern do you see? Use `rand(ComplexF64, n)` to also try *complex* coefficients. Repeat multiple times to form a conjecture on how the eigenvalues of A and B are related. Prove your conjecture theoretically.

Problem 8.3 An algebraic number is a value in \mathbb{C} that is a root of some polynomial having *integer* coefficients. For example, $x = \sqrt{5}$ is an algebraic number because it is a root of the polynomial $p(x) = x^2 - 5$.

Consider $x_1 = -2 + i\sqrt{3}$ and $x_2 = 5 - \sqrt{7}$. These are algebraic numbers because they are roots of corresponding polynomials $p_1(x) = (x - (-2 + i\sqrt{3}))(x - (-2 - i\sqrt{3})) = x^2 + 4x + 7$ and $p_2(x) = (x - (5 - \sqrt{7}))(x - (5 + \sqrt{7}))$, respectively, having integer coefficients (when expanded). Now define $x_3 \triangleq x_1 + x_2$ and $x_4 \triangleq x_1x_2$. A theorem in algebra says that x_3 and x_4 are also algebraic because algebraic numbers form a field. In other words, there is a polynomial $p_3(x)$ with integer coefficients that has one root equal to x_3 , and a polynomial $p_4(x)$ with integer coefficients that has one root equal to x_4 . Use the Kronecker product and/or Kronecker sum to find those polynomials $p_3(x)$ and $p_4(x)$.

Hint: Start by finding the companion matrix with integer coefficients that has an eigenvalue equal to x_1 and another companion matrix with eigenvalue x_2 . You may use symbolic computation to find characteristic polynomials.

Problem 8.4 Let A denote an $N \times N$ Vandermonde matrix corresponding to the roots of the polynomial $z^N - 1$. Let B denote an $N \times N$ Vandermonde matrix corresponding to $\{e^{-i2\pi(n+1/2)/N} : n = 0, \dots, N-1\}$. Define the matrix $X \triangleq [aA \quad bB]$ for $a, b \in \mathbb{C}$.

- Determine whether the matrix X is a frame, a tight frame, a Parseval tight frame, or none of these.
- If it is some type of frame, determine its frame bound(s).
- Write a JULIA function with inputs a, b and $y \in \mathbb{C}^N$ that computes efficiently the minimum-norm solution to the LS problem

$$\arg \min_{x \in \mathbb{C}^{2N}} \|Xx - y\|_2.$$

Hint: For maximum efficiency, use a Vandermonde matrix form corresponding to the DFT. Aim for a solution with $O(N \log_2 N)$ computation!

Problem 8.5 The set $\{G_N^0, G_N^1, \dots, G_N^{N-1}\}$ forms a basis for the subspace of $N \times N$ circulant matrices in the vector space $\mathbb{F}^{N \times N}$ of all $N \times N$ matrices, where G_N denotes the “generator” for circulant matrices in (8.11).

- (a) Is this set an orthogonal basis for that subspace? Explain.
- (b) Is this set an orthonormal basis for that subspace? Explain.

Problem 8.6

- (a) Show that all circulant matrices (of the same size) commute.
- (b) Show that all circulant matrices are normal.
- (c) Determine the eigenvalues of the following $N \times N$ circulant matrix:

$$\mathbf{C} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & \ddots & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & -1 \end{bmatrix}.$$

Give a simple expression that holds for any $N \in \mathbb{N}$. Multiplying this matrix by a vector performs finite differences of the vector’s elements with periodic end conditions.

- (d) Check your expression for the previous part numerically for the case $N = 4$. Report the eigenvalues from your expression and from the `eigvals` command in JULIA.
- (e) Write a very short JULIA function that computes the nuclear norm of any circulant matrix \mathbf{C} without calling expensive functions like `eig` or `svd`. Suggestion: check your function using the previous part.
- (f) Optional. In image processing, matrices that are block circulant with circulant blocks (BCCB) are particularly important. Such matrices have the form $\mathbf{B} = \mathbf{C}_2 \otimes \mathbf{C}_1$, where \mathbf{C}_1 and \mathbf{C}_2 are both circulant matrices.
Express the eigenvalues of \mathbf{B} in terms of the first columns of \mathbf{C}_1 and \mathbf{C}_2 .
Challenge. Express the eigenvalues of \mathbf{B} in terms of the first column of \mathbf{B} .

Problem 8.7 Consider the matrix $\mathbf{A} = \mathbf{B} \oplus \mathbf{C}$, where \mathbf{B} is the $N \times N$ circulant matrix whose first column is $\mathbf{b} \in \mathbb{F}^N$, and \mathbf{C} is the $M \times M$ circulant matrix whose first column is $\mathbf{c} \in \mathbb{F}^M$. This problem shows how to use Fourier transforms to efficiently compute $\mathbf{y} = \mathbf{A}^{-1}\mathbf{x}$.

- (a) Show that

$$\mathbf{A} = \frac{1}{MN}(\mathbf{F}_N \otimes \mathbf{F}_M)'(\Lambda_b \oplus \Lambda_c)(\mathbf{F}_N \otimes \mathbf{F}_M),$$

where $\Lambda_b = \text{Diag}\{\mathbf{f}_b\}$ and $\Lambda_c = \text{Diag}\{\mathbf{f}_c\}$ are the diagonal matrices containing the DFTs of \mathbf{b} and \mathbf{c} , respectively, and \mathbf{F}_N is the $N \times N$ DFT matrix.

Hint: Use the fact that any $N \times N$ circulant matrix \mathbf{B} can be written as $\mathbf{B} = (1/N)\mathbf{F}'_N \Lambda_b \mathbf{F}_N$. Define $\mathbf{Q}_b = (1/\sqrt{N})\mathbf{F}'_N$ and $\mathbf{Q}_c = (1/\sqrt{M})\mathbf{F}'_M$, and apply your result from Problem 3.10.

(b) Show that

$$\mathbf{A}^{-1} = \frac{1}{MN} (\mathbf{F}_N \otimes \mathbf{F}_M)' (\mathbf{A}_b \oplus \mathbf{A}_c)^{-1} (\mathbf{F}_N \otimes \mathbf{F}_M).$$

- (c) Show that the $M \times N$ matrix $\mathbf{P} = \mathbf{1}_M \mathbf{f}'_b + \mathbf{f}_c \mathbf{1}'_N$, containing all pairwise sums of the entries of \mathbf{f}_b and \mathbf{f}_c (i.e., eigenvalues of \mathbf{B} and \mathbf{C}), satisfies the relationship $\mathbf{A}_b \oplus \mathbf{A}_c = \text{Diag}(\text{vec}(\mathbf{P}))$. In other words, the columns of \mathbf{P} stacked into a vector form the diagonal of the matrix $\mathbf{A}_b \oplus \mathbf{A}_c$.
- (d) Suppose $\mathbf{y} = \mathbf{A}^{-1} \mathbf{x}$, and let \mathbf{X} and \mathbf{Y} be the $M \times N$ matrices such that $\mathbf{x} = \text{vec}(\mathbf{X})$ and $\mathbf{y} = \text{vec}(\mathbf{Y})$. Use the above results and your answer to Problem 2.5 to argue that

$$\mathbf{Y} = \frac{1}{MN} \mathbf{F}'_M \left(\frac{\mathbf{F}_M \mathbf{X} \mathbf{F}_N^T}{\mathbf{P}} \right) \overline{\mathbf{F}_N}, \quad (8.70)$$

where the fraction denotes elementwise division, and $\overline{\mathbf{F}_N}$ denotes the (elementwise) complex conjugate of \mathbf{F}_N .

- (e) Explain why the JULIA code $\mathbf{Y} = \text{ifft}(\text{fft}(\mathbf{X}) ./ \mathbf{P})$ implements (8.70).

Problem 8.8 Some texts define a square matrix \mathbf{A} to be periodic iff $\mathbf{A}^{k+1} = \mathbf{A}$ for some $k \in \mathbb{N}$. If such a k exists and is the least such k , then k is called “the period” of the matrix. That definition is not equivalent to our definition in Section 8.6.5 of not aperiodic, even for nonnegative matrices. Consider $\mathbf{B} = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}$ that has period 2 but all powers of \mathbf{B} differ. Prove or disprove whether the two definitions are equivalent for stochastic matrices.

Problem 8.9 The mythical Michigan Wolverine eats something once a day, either cheese, grapes, or lettuce. Its daily dietary habits follow these rules:

- If it ate cheese today, tomorrow it will eat lettuce or grapes with equal probability.
 - If it ate grapes today, tomorrow it will eat grapes with probability 1/10, cheese with probability 4/10, and lettuce with probability 5/10.
 - If it ate lettuce today, it will not eat lettuce again tomorrow, but it will eat grapes with probability 4/10 or cheese with probability 6/10.
- (a) What is the long-term average probability of the Wolverine eating cheese, grapes, or lettuce on some day?
- (b) Is your answer the unique solution? Explain why or why not.

Problem 8.10 Let π be an N -dimensional vector having nonnegative elements that sum to one.

- (a) Prove (by example) that there exists a Markov chain with $\mathbf{P} \neq \mathbf{I}$ for which π is an equilibrium distribution. Specify the transition matrix \mathbf{P} for your example.
- (b) For that transition matrix \mathbf{P} , discuss whether the given π is the *unique* equilibrium distribution.
- (c) For that transition matrix \mathbf{P} , discuss whether the Markov chain power iteration $\pi_{k+1} = \mathbf{P}\pi_k$ is guaranteed to converge to π for any initial probability distribution π_1 .

Problem 8.11 Consider the Markov chain that has transition matrix $P = G_N + (1-p)(e_3 - e_2)e_1'$ for $N \geq 3$, where G_N denotes the circulant generator matrix, and $p \in (0,1)$. Determine the equilibrium distribution π_* of this Markov chain and discuss whether it is unique.

Hint: If needed, solve this numerically for a few small values of $N \geq 3$ to form a conjecture that you can then verify analytically. Your solution must be general for $N \geq 3$ and $p \in (0,1)$. For reference,

$$P = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ p & 0 & 0 & \cdots & 0 \\ 1-p & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.$$

Problem 8.12

- (a) Give an example of a rank-1 transition matrix $P \in \mathbb{R}^{4 \times 4}$ having equilibrium distribution $\pi = 1_4/4$.
- (b) Is your example the unique solution? Explain why or why not.

9 Optimization Basics and Logistic Regression

9.1 Introduction

Many of the preceding topics have involved optimization formulations: linear LS, Procrustes, low-rank approximation, multidimensional scaling. In all these cases we derived analytical solutions, like the pseudoinverse for minimum-norm LS problems and the truncated SVD for low-rank approximation. But often we need iterative optimization algorithms, for example if no closed-form minimizer exists, or if the analytical solution requires too much computation and/or memory (e.g., SVD for large problems).

To solve a problem like $\hat{x} = \arg \min_x f(x)$ via an iterative method, we start with some initial guess x_0 , and then the algorithm produces a sequence $\{x_k\}$ where hopefully the sequence converges to \hat{x} , meaning $\|x_k - \hat{x}\| \rightarrow 0$ for some norm $\|\cdot\|$ as $k \rightarrow \infty$, as discussed in Section 6.5.

This chapter describes the basics of gradient-based iterative optimization algorithms. Section 9.2 considers preconditioned gradient descent (PGD) for the linear LS problem. Along the way it introduces additional matrix concepts: matrix square root, more about positive (semi)definite matrices, commuting matrices, and majorization. PGD uses a fixed step size, whereas preconditioned steepest descent (PSD), as described in Section 9.3, uses a line search to determine the step size. Section 9.4 considers gradient descent and accelerated versions for general smooth convex functions. Section 9.5 applies gradient descent to the machine learning application of binary classification via logistic regression. Finally, Section 9.6 summarizes stochastic gradient descent.

9.2 Preconditioned Gradient Descent for LS

For the (possibly regularized) LS problem with $A \in \mathbb{F}^{M \times N}$ the cost function is quadratic:

$$\hat{x} = \arg \min_x f(x), \quad f(x) = \underbrace{\frac{1}{2} \|Ax - y\|_2^2}_{f: \mathbb{F}^N \mapsto \mathbb{R}}, \quad \nabla f(x) = \underbrace{A'(Ax - y)}_{\nabla f: \mathbb{F}^N \mapsto \mathbb{F}^N}. \quad (9.1)$$

Problem 5.5 considered the classical gradient descent (GD) iterative method of Cauchy [258]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k) = \mathbf{x}_k - \alpha \mathbf{A}'(\mathbf{A}\mathbf{x}_k - \mathbf{y}), \quad (9.2)$$

where convergence of the sequence $\{\mathbf{x}_k\}$ is ensured by choosing the step size α to satisfy

$$0 < \alpha < \frac{2}{\sigma_1(\mathbf{A}'\mathbf{A})} = \frac{2}{\sigma_1^2(\mathbf{A})}. \quad (9.3)$$

See (9.11) in Section 9.2.3. Now consider a generalization called preconditioned gradient descent (PGD):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{P} \nabla f(\mathbf{x}_k) = \mathbf{x}_k - \mathbf{P} \mathbf{A}'(\mathbf{A}\mathbf{x}_k - \mathbf{y}), \quad (9.4)$$

where \mathbf{P} is an $N \times N$ preconditioning matrix. Classical gradient descent simply uses $\mathbf{P} = \alpha \mathbf{I}_N$, but a single step size α may be suboptimal (and hard to choose), especially if different elements of \mathbf{x} have different units.

Several questions arise naturally here. What conditions on \mathbf{P} ensure convergence? Will some other $\mathbf{P} \neq \alpha \mathbf{I}$ provide faster convergence? Why does the step size (9.3) suffice? Answering these questions uses many matrix methods!

Explore 9.1 Revisit the units analysis in (2.20) and compare the constraints on the units of α in (9.2) and on \mathbf{P} in (9.4).

9.2.1 Tool: Matrix Square Root

We need another linear algebra tool first. (And more will come before we finish this topic.)

Definition We call \mathbf{S} a (matrix) square root of a square matrix \mathbf{A} iff $\mathbf{S}\mathbf{S} = \mathbf{A}$.

It is not unique: if \mathbf{S} is a square root of \mathbf{P} , then $-\mathbf{S}$ is also square root of \mathbf{P} .

Example 9.1 For the rotation matrix $\mathbf{R} = \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix}$ a square root is $\mathbf{S} = \begin{bmatrix} \cos(\phi/2) & \sin(\phi/2) \\ -\sin(\phi/2) & \cos(\phi/2) \end{bmatrix}$ because $\mathbf{S}\mathbf{S} = \mathbf{R}$. This example has an intuitive geometric interpretation.

Fact 9.1 If \mathbf{P} is positive semidefinite, that is, $\mathbf{P} \succeq \mathbf{0}$, then \mathbf{P} has a unique positive semidefinite square root.

Proof (of existence). $\mathbf{P} \succeq \mathbf{0} \implies \mathbf{P} = \mathbf{V}\Lambda\mathbf{V}'$, where the eigenvalues are all real and nonnegative, so let $\mathbf{S} = \mathbf{V}\Lambda^{1/2}\mathbf{V}'$ where $\Lambda^{1/2} \triangleq \text{Diag}\{\sqrt{\lambda_i}\}$. Clearly $\mathbf{S}\mathbf{S} = \mathbf{P}$ and $\mathbf{S} \succeq \mathbf{0}$. \square

Fact 9.2 If P is positive definite, $P \succ 0$, then P has a unique positive definite (thus invertible) square root.

Proof. Same as above, only now the eigenvalues are all positive. For uniqueness, see [115, p. 405]. \square

Definition When P is positive (semi)definite, the notation $P^{1/2}$ always denotes the unique positive (semi)definite square root of P . When being careful, we call $P^{1/2}$ the principal square root, but often we just call it “the square root” of P , just like people say “the square root of 9 is 3.”

For a square matrix A that is not positive (semi)definite, the notation $A^{1/2}$ means any matrix square root of A ; we should use that notation only if a matrix square root exists.

Q9.1 Every diagonalizable matrix has a matrix square root.

A: True **B: False**

Example 9.2 If $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ then $S = \begin{bmatrix} 1 & 1/2 \\ 0 & 1 \end{bmatrix}$ is a matrix square root of A .

Q9.2 If a square matrix has a matrix square root, then it is diagonalizable.

If A is any nondiagonalizable 2×2 matrix, then we can still factor it using the Jordan normal form as follows:

$$A = V \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} V^{-1}, \quad (9.5)$$

where V is an invertible matrix consisting of generalized eigenvectors. One can verify that if $\lambda \neq 0$, then a square root of this matrix is

$$S = V \begin{bmatrix} \sqrt{\lambda} & 1/(2\sqrt{\lambda}) \\ 0 & \sqrt{\lambda} \end{bmatrix} V^{-1}. \quad (9.6)$$

This Jordan normal form approach to find a matrix square root generalizes to any square matrix having nonzero eigenvalues (i.e., invertible). But not every matrix has a square root [259].

Example 9.3 The matrix $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ has no square root.

Q9.3 For $x \neq 0$, consider $A = xx' \succeq 0$. Which of the following is a matrix square root of A ?

$$A: \mathbf{x}\mathbf{x}' \quad B: \mathbf{x}\mathbf{x}'/\|\mathbf{x}\|_2 \quad C: \mathbf{x}\mathbf{x}'/\|\mathbf{x}\|_2^2 \quad D: \mathbf{x}\mathbf{x}'\|\mathbf{x}\|_2 \quad E: \mathbf{x}\mathbf{x}'\|\mathbf{x}\|_2^2$$

Q9.4 If Q is a unitary matrix, then Q has a matrix square root.

A: True B: False

Some matrices have an uncountably infinite set of square roots.

Example 9.4 For $A = I_2$, every matrix of the form $S_\phi = \begin{bmatrix} \cos \phi & \sin \phi \\ \sin \phi & -\cos \phi \end{bmatrix}$ is a square root of A , as are $\pm I_2$ and $\begin{bmatrix} \pm 1 & 0 \\ 0 & \mp 1 \end{bmatrix}$. Each S_ϕ is a unitary matrix consisting of rotation by ϕ followed by a sign flip of the second coordinate.



Caution. For scalars $\alpha, \beta \in \mathbb{C}$ we have the property that $\sqrt{\alpha\beta} = \sqrt{\alpha}\sqrt{\beta} = \sqrt{\beta}\sqrt{\alpha}$. This type of equality does *not* hold in general for matrices, that is, in general,

$$(AB)^{1/2} \neq A^{1/2}B^{1/2}.$$

Example 9.5 Consider $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$. These are both positive semidefinite, rank-1 matrices, and their principal square roots are

$$A^{1/2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad B^{1/2} = B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

Now $AB = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} = (AB)^{1/2}$, but

$$A^{1/2}B^{1/2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \neq (AB)^{1/2}.$$

9.2.1.1 Practical Implementation of Matrix Square Root



For a square matrix, the function that finds a matrix square root is `sqrt(A)`. This operation differs *completely* from the elementwise root `sqrt.(A)`. In MATLAB (and old JULIA versions) the operation is `sqrtm(A)`. Caution. If A is square but not diagonalizable then the output of `sqrt` can be meaningless.

Explore 9.2 Try `sqrt([0 1; 0 0])` in JULIA and `sqrtm([0 1; 0 0])` in MATLAB.

The JULIA manual says that `sqrt(A)` uses an eigendecomposition if A is Hermitian; otherwise, it uses a Schur decomposition [260].

9.2.2 Convergence Rate Analysis of PGD: First Steps

Hereafter we assume the preconditioner P in PGD (9.4) is positive definite, that is, $P \succ 0$, so that $P^{1/2}$ exists and is invertible. We denote its inverse by $P^{-1/2}$.

Let \hat{x} denote any minimizer of $f(x)$. That \hat{x} must satisfy the normal equations: $A'A\hat{x} = A'y$. For analysis purposes, replace $A'y$ in (9.4) with $A'A\hat{x}$:

$$\begin{aligned}
 x_{k+1} &= x_k - PA'(Ax_k - y) = x_k - P(A'Ax_k - A'y) \\
 &= x_k - P(A'Ax_k - A'A\hat{x}) = x_k - PA'A(x_k - \hat{x}) \\
 \implies x_{k+1} - \hat{x} &= x_k - \hat{x} - PA'A(x_k - \hat{x}) \\
 &= (I - PA'A)(x_k - \hat{x}) \\
 \implies \underbrace{P^{-1/2}(x_{k+1} - \hat{x})}_{\delta_{k+1}} &= P^{-1/2}(I - PA'A)(x_k - \hat{x}) \\
 &= (I - P^{1/2}A'AP^{1/2}) \underbrace{P^{-1/2}(x_k - \hat{x})}_{\delta_k} \\
 \implies \delta_{k+1} &= (I - P^{1/2}A'AP^{1/2})\delta_k = G\delta_k \\
 \delta_k &\triangleq P^{-1/2}(x_k - \hat{x}) \text{ (modified error vector)} \\
 G &\triangleq I - P^{1/2}A'AP^{1/2} \\
 \implies \delta_k &= G^k\delta_0. \tag{9.7}
 \end{aligned}$$

The matrix G "governs" the convergence of PGD.

9.2.2.1 PGD Convergence Condition Using Eigenvalues

Here, G is Hermitian, so $G = VAV'$ for some unitary V and Λ is real (but *not* nonnegative in general). Using matrix powers from (3.10), we express the modified error vector δ_k in terms of the initial error:

$$\delta_{k+1} = G\delta_k \implies \delta_k = G^k\delta_0 = V\Lambda^kV'\delta_0 \implies (V'\delta_k) = \Lambda^k(V'\delta_0). \tag{9.8}$$

Define $\tilde{\delta}_k \triangleq V'\delta_k$ to be the error coordinates in the V basis, then

$$\tilde{\delta}_{k+1} = \Lambda\tilde{\delta}_k \implies \tilde{\delta}_k = \Lambda^k\tilde{\delta}_0 = \text{Diag}\{\lambda_i^k\}\tilde{\delta}_0. \tag{9.9}$$

We have reduced the question of convergence of PGD down to seeing whether $\{\lambda_i^k\}$ is a decreasing geometric series. We need $-1 < \lambda_i < 1$ for all $i = 1, \dots, N$ to ensure that all error components diminish to zero.

Put concisely, a necessary and sufficient condition to ensure convergence of PGD from any initializer x_0 is:

$$\rho(G) = \rho(I - P^{1/2}A'AP^{1/2}) < 1, \text{ that is, } -1 < \text{eig}(I - P^{1/2}A'AP^{1/2}) < 1. \tag{9.10}$$

9.2.3 Classical GD: Step Size Bounds

Consider the classical case where $P = \alpha I$. Then the condition in (9.10) simplifies to

$$-1 < \text{eig}\{I - \alpha A'A\} < 1 \iff -1 < 1 - \alpha\sigma_i^2 < 1 \iff 0 < \alpha\sigma_i^2 < 2, \tag{9.11}$$

where $\{\sigma_i\}$ denotes the singular values of A .

The lower bound condition $0 < \alpha\sigma_i^2$ holds for all i if $0 < \alpha$ and if \mathbf{A} has full column rank (e.g., when Tikhonov regularization is used). So even though the analysis did not assume full rank at the start, if we want to make sure that all error modes diminish to zero, we must have full (column) rank matrix \mathbf{A} .

The upper bound condition $\alpha\sigma_i^2 < 2$ holds for all i if $\alpha\sigma_1^2 < 2$ because σ_1 is the largest, so we need $\alpha < 2/\sigma_1^2$.

Fact 9.3 In summary, we derived the step size condition (9.3) for convergence of classical GD for LS problems where \mathbf{A} has full column rank:

$$0 < \alpha < \frac{2}{\sigma_1^2(\mathbf{A})} = \frac{2}{\sigma_1(\mathbf{A}'\mathbf{A})}. \quad (9.12)$$

Computing the spectral norm for (9.12) with an SVD can be expensive. See Section 6.4.7 for practical step size alternatives that do not require an SVD.

9.2.4 Optimal Step Size for GD

Fact 9.4 For classical GD, where $\mathbf{P} = \alpha\mathbf{I}$ and \mathbf{A} has full column rank, the *optimal* step size (for fastest asymptotic convergence) is (Problem 9.1):

$$\alpha_* = \frac{2}{\sigma_1^2(\mathbf{A}) + \sigma_N^2(\mathbf{A})} = \frac{2}{\sigma_1(\mathbf{A}'\mathbf{A}) + \sigma_N(\mathbf{A}'\mathbf{A})} = \frac{2}{\sigma_1(\nabla^2 f(\hat{\mathbf{x}})) + \sigma_N(\nabla^2 f(\hat{\mathbf{x}}))}. \quad (9.13)$$

This step size makes $\rho(\mathbf{I} - \alpha\mathbf{A}'\mathbf{A})$ as small as possible, that is, so that its largest absolute eigenvalue is as close to zero as possible. That largest eigenvalue will be the mode that converges to zero the slowest, so it will ultimately govern the convergence rate.

However, the step size conditions (9.3) and (9.13) are unsatisfying in practice because if an LS problem is so large that we cannot use the SVD to compute the pseudoinverse, then probably we do not want to use an SVD (or `svds`) to find $\sigma_1(\mathbf{A})$ and perhaps also $\sigma_N(\mathbf{A})$.

Q9.5 If \mathbf{A} is unitary, what is the best step size α when $\mathbf{P} = \alpha\mathbf{I}$?

- A: 0 B: 1 C: 1.99 D: 2 E: None of these

Q9.6 Suppose $\mathbf{P} = \alpha(\mathbf{A}'\mathbf{A})^{-1}$ for some $\alpha \geq 0$. What is the best value of α so that the eigenvalues of \mathbf{G} are as small (in magnitude) as possible, so PGD converges as fast as possible?

- A: 0 B: 1/2 C: 1 D: $\sqrt{2}$ E: $2 - \epsilon$

9.2.5 Ideal Preconditioner for PGD

Before looking at more general choices for the preconditioner \mathbf{P} , we first explore the “ideal” \mathbf{P} .

Fact 9.5 The ideal preconditioner is $P_{\text{ideal}} = (A'A)^{-1}$.

This is because, for that choice,

$$G = I - P_{\text{ideal}}^{1/2} A' A P_{\text{ideal}}^{1/2} = I - (A'A)^{-1/2} A' A (A'A)^{-1/2} = I - I = 0,$$

so $\delta_{(1)} = G^T \delta_0 = 0 \delta_0 = 0$, that is, PGD converges in one iteration:

$$x_1 = x_0 - P_{\text{ideal}} A' (Ax_0 - y) = x_0 - (A'A)^{-1} A' (Ax_0 - y) = (A'A)^{-1} A' y = A^+ y.$$

Why not always use this ideal preconditioner? Inverting $A'A$ is expensive: it is as hard as the original LS problem. So we look for other preconditioners. Specifically, we would like to find a preconditioner $P \approx (A'A)^{-1}$ or $P^{-1} \approx A'A$ but where the inverse is much easier to compute. For that we need another tool.

9.2.6 Tool: Positive (Semi)Definiteness Properties

Recall that we define positive (semi)definiteness only for Hermitian matrices, and

- positive semidefinite: $A \succeq 0 \iff x' A x \geq 0 \forall x$;
- positive definite: $A \succ 0 \iff x' A x > 0 \forall x \neq 0$.

The following properties follow (A and B are all Hermitian here):

- $X' X \succeq 0$ for any matrix X , even nonsquare, per Fact 3.15.
- $A \succeq 0 \implies X' A X \succeq 0$ for any size-compatible matrix X .
- $A \succeq 0$ and $\alpha \geq 0 \implies \alpha A \succeq 0$.
- $A \succeq 0 \implies$ all eigenvalues of A are real and nonnegative.

Proof 1. If x is an eigenvector of A , $0 \leq x'(Ax) = x' \lambda x = \lambda \|x\|_2^2 \implies \lambda \geq 0$. \square

Proof 2. A Hermitian implies it has an orthogonal eigendecomposition $A = V \Lambda V'$, so $V \Lambda V' \succeq 0$. Using the second property above $\implies V'(V \Lambda V')V \succeq 0 \implies \Lambda \succeq 0 \implies e_j' \Lambda e_j \geq 0 \implies \lambda_j \geq 0$. \square

- $A \succ 0 \implies A$ invertible and $A^{-1} \succ 0$.
- $B \succeq A \iff B - A \succeq 0$ (trivial from definition of \succeq). This is called Loewner order.
- $B \succ A \iff B - A \succ 0$.
- $B \succeq A \succ 0 \iff A^{-1} \succeq B^{-1} \succ 0$.
- $B \succeq A \succ 0 \implies \gamma B \succ A \forall \gamma > 1$.
- $A \succ 0, B \succ 0 \implies BAB \succ 0$ (Problem 4.8).

Notice the pattern: start with a definition, then develop properties (especially addition and multiplication). The above properties relate to multiplication; now consider matrix addition.

Q9.7 If $A \succeq 0$ and $B \succeq 0$ have the same size, then $A + B \succeq 0$.

A: True

B: False

Q9.8 If $A \succ 0$ and $B \succeq 0$ have the same size, then $A + B \succ 0$.

A: True **B: False**

Q9.9 If $A \succ 0$ and $B \succeq 0$ are both $N \times N$ and $\sigma_1(A) > \sigma_1(B), \dots, \sigma_N(A) > \sigma_N(B)$, then $A \succ B$.

The above properties are all useful and important for GD for LS because we choose $P \succ 0$, and we assume A has full column rank so that $A'A \succ 0$.

9.2.7 General Preconditioners for PGD

Repeating (9.10), for convergence we want

$$-1 < \text{eig}\left\{I - P^{1/2} A' A P^{1/2}\right\} < 1. \quad (9.14)$$

Now the right-hand side of (9.14) is easy; assuming $A'A$ and P are positive definite,

$$\text{eig}\left\{I - P^{1/2}A'AP^{1/2}\right\} = 1 - \underbrace{\text{eig}\left\{P^{1/2}A'AP^{1/2}\right\}}_{> 0} < 1. \quad (9.15)$$

For the left-hand side of (9.14) we want

$$-1 < \text{eig}\left\{\mathbf{I} - \mathbf{P}^{1/2}\mathbf{A}'\mathbf{A}\mathbf{P}^{1/2}\right\} = 1 - \text{eig}\left\{\mathbf{P}^{1/2}\mathbf{A}'\mathbf{A}\mathbf{P}^{1/2}\right\},$$

that is (Problem 9.2),

$$\operatorname{eig}\left\{\mathbf{P}^{1/2}\mathbf{A}'\mathbf{A}\mathbf{P}^{1/2}\right\} < 2 \iff 2\mathbf{I} \succ \mathbf{P}^{1/2}\mathbf{A}'\mathbf{A}\mathbf{P}^{1/2} \iff 2\mathbf{P}^{-1} \succ \mathbf{A}'\mathbf{A}. \quad (9.16)$$

Definition We say that a (Hermitian) symmetric matrix A is a majorizer of (or majorizes) a (Hermitian) symmetric matrix B iff $A \succeq B$, that is, iff $A - B$ is positive semidefinite.

Summarizing: if βP^{-1} majorizes $A'A$ for some $0 < \beta < 2$, then convergence of PGD is ensured, because $\beta P^{-1} \succ A'A \implies 2P^{-1} \succ A'A$.

9.2.8 Diagonal Majorizer

Fact 9.6 (Diagonal majorizer). If B is (Hermitian) symmetric and $N \times N$, then (Problem 9.3)

$$\text{Diag}\{\|B\| \mathbf{1}_N\} \succ B. \quad (9.17)$$

where here $|B|$ denotes the elementwise absolute value of B , that is, `abs.(B)` in [JULIA](#).

Applying (9.17) with $B = A'A$ shows that $\text{Diag}\{|A'A| \mathbf{1}_N\} \succeq A'A$. Thus, when A has full column rank, $(2/\alpha)\text{Diag}\{|A'A| \mathbf{1}_N\} \succ A'A$ for any $0 < \alpha < 2$. Applying the condition in (9.16), we conclude that a valid preconditioner for PGD is $P^{-1} = \alpha^{-1} \text{Diag}\{|A'A| \mathbf{1}_N\}$. This choice leads to the diagonally preconditioned GD iteration:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha D A' (\mathbf{A}\mathbf{x}_k - \mathbf{y}), \quad D \triangleq \text{Diag}\{|A'A| \mathbf{1}_N\}^{-1}, \quad 0 < \alpha < 2. \quad (9.18)$$

Typically we use $1 \leq \alpha < 2$, and it is easy to adjust α in this range to find a good value.

Q9.10 If $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & -1 \end{bmatrix}$, what is the diagonal preconditioner D in (9.18)?

- A: $\begin{bmatrix} 1/3 & 0 \\ 0 & 1/6 \end{bmatrix}$ B: $\begin{bmatrix} 1/2 & 0 \\ 0 & 1/3 \end{bmatrix}$ C: $\begin{bmatrix} 1/2 & 0 \\ 0 & 1/5 \end{bmatrix}$ D: $\begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}$ E: None of these

The “normalization” by the inverse matrix D in (9.18) makes finding α much easier than in classical GD where suitable α values depend on the units of A .

The following code compares $\text{eig}\{I - P^{-1/2}A'AP^{-1/2}\}$ for classical GD with $P = \alpha_* I$ for the optimal step size α_* versus diagonally preconditioned GD with $\alpha = 1.3$ chosen empirically. The former has eigenvalues ± 0.5151 , and the latter has eigenvalues $(-0.3, 0.35)$, so at least in this example the diagonal preconditioner accelerates convergence, without requiring an eigendecomposition to find P .

```
<> 9.1 using LinearAlgebra
A = [1 0; 0 2; 1 -1]
@show A'A
alphabet = 2 / (sum(svdvals(A'A)[[1, end]])) # optimal GD step size
G1 = I - alphabet * (A' * A)
@show eigvals(G1)
D = Diagonal(1 ./ (abs.(A'A) * ones(2))) # diagonal preconditioner
alpha2 = 1.3
P2 = alpha2 * D
G2 = I - sqrt(P2) * (A' * A) * sqrt(P2)
@show eigvals(G2)
```

If one does not want to store the diagonal elements of D , then it suffices to use the maximum diagonal value in (9.17). Define

$$d_{\max} = \| |A'A| \mathbf{1}_N \|_{\infty}, \quad (9.19)$$

for which

$$D \succeq \frac{1}{d_{\max}} I. \quad (9.20)$$

Then the following version of GD for LS is guaranteed to converge for any $0 < \alpha < 2$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\alpha}{d_{\max}} A' (\mathbf{A}\mathbf{x}_k - \mathbf{y}). \quad (9.21)$$

Q9.11 For the previous example, what is the factor d_{\max} ?

- A: 1 B: 2 C: 3 D: 4 E: None

The D in (9.18) and d_{\max} in (9.19) are most useful in problems where A has nonnegative elements so that $|A'A| \mathbf{1}_N = A'(A\mathbf{1}_N)$, because then only matrix–vector multiplies are needed.

We have focused on the quadratic problem here, but the principles of diagonal majorization apply to using GD (and accelerated GD) in general for convex optimization of cost functions having Lipschitz gradients [261].

9.2.9 Preconditioning Illustration/Demo

 Demo 9.1 illustrates the benefits of preconditioning. See Fig. 9.1.

Q9.12 For the demo, the preconditioned LS cost function \tilde{f} relates to the nonpreconditioned LS cost function f via the relation $\tilde{f}(z) = f(g(z))$ for what function g ?

- A: $g(z) = P^{-1/2}z$ B: $g(z) = P^{1/2}z$ C: $g(z) = z$ D: $g(z) = Az$ E: $g(z) = \tilde{A}z$

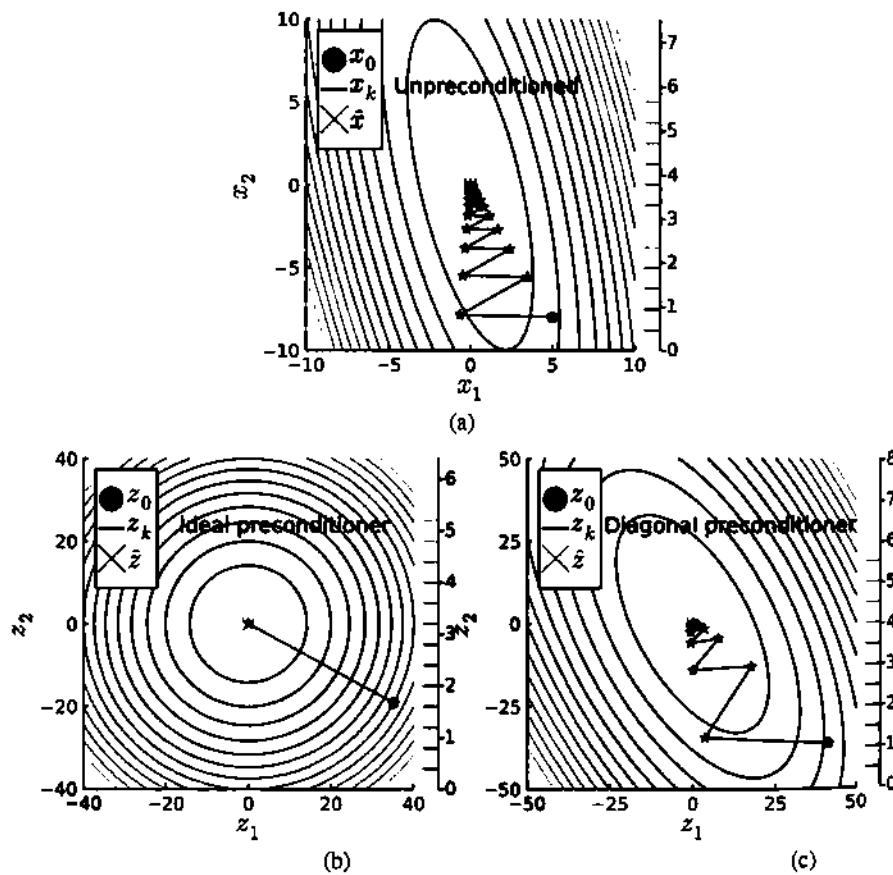


Figure 9.1 Convergence of GD (a) and PGD with ideal (b) and diagonal (c) preconditioners.

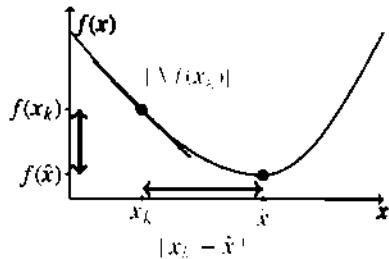


Figure 9.2 Convergence rate metrics.

An important type of preconditioner for signal processing is circulant approximations to Toeplitz systems [262].

9.2.10 Convergence Rates

There are many ways to assess the convergence rate of an iterative algorithm like PGD. Researchers study the following (see Fig. 9.2):

- $f(x_k) \rightarrow f(\hat{x})$
- $|\nabla f(x_k)| \rightarrow 0$
- $|x_k - \hat{x}| \rightarrow 0$
- $\|\delta_k\| \rightarrow 0$, $\delta_k \triangleq P^{-1/2} (x_k - \hat{x})$
- $\|\tilde{\delta}_k\| \rightarrow 0$, $\tilde{\delta}_k \triangleq V^* \delta_k$

Quantifying bounds on the rates of decrease of these quantities is an active research area. Even classical GD has relatively recent results [263] that tighten up the traditional bounds. A tight worst-case bound for GD (with fixed step size $\alpha = 1/L$) for the decrease of the cost function is $O(1/k)$ [263]:

$$f(x_k) - f(\hat{x}) \leq \frac{L \|x_0 - \hat{x}\|_2^2}{4k + 2}, \quad (9.22)$$

where L is the Lipschitz constant of the gradient $\nabla f(x)$. In contrast, Nesterov's fast gradient method has a worst-case cost function decrease at rate at least $O(1/k^2)$, which can be improved (and has been) by only a constant factor [264].

Example 9.6 Figure 9.3 illustrates how slow GD can converge for a simple LS problem with $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ and $y = \mathbf{0}$. This case used the optimal step size α_* for illustration. This slow convergence has been the impetus of thousands of papers on faster algorithms! The ellipses show the contours of the LS cost function $\|Ax - y\|$. Also shown is Nesterov's fast gradient method (FGM) method for comparison.

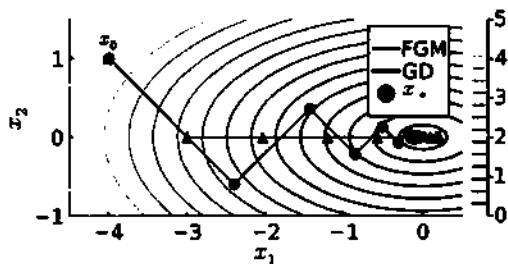


Figure 9.3 Convergence example for GD and FGM.

Q9.13 What is the optimal step size α_* in Example 9.6?

- A: 1/5 B: 2/5 C: 3/5 D: 1/3 E: 2/3

9.2.11 Tool: Commuting (Square) Matrices

Q9.14 Let X and Y denote square matrices of the same size. Which of the following conditions guarantee that X and Y commute, that is, $XY = YX$? Choose the most general correct combination of condition(s).

- (i) X diagonalizable
- (ii) Y diagonalizable
- (iii) X and Y have same eigenvectors, i.e., $Xv = \alpha v \iff Yv = \beta v$
- (iv) X (Hermitian) symmetric
- (v) Y (Hermitian) symmetric

- A: (i) and (ii)
 B: (i), (ii), and (iii)
 C: (i), (ii), (iii), and ((iv) or (v))
 D: (i), (ii), (iii), (iv), and (v)
 E: None of these suffice

Definition A set of matrices is simultaneously diagonalizable iff there is an invertible matrix V such that $V^{-1}AV$ is diagonal for every matrix A in the set. (Clearly all matrices in such a set are diagonalizable.)

Q9.15 All $N \times N$ companion matrices commute.

- A: True B: False

Q9.16 All $N \times N$ circulant matrices commute.

- A: True B: False

Q9.17 If X and Y are simultaneously diagonalizable, then

$$XY = YX = X^{1/2}YX^{1/2} = Y^{1/2}XY^{1/2}. \quad (9.23)$$

- A: True B: False

- ◆ Being simultaneously diagonalizable is a sufficient condition. A necessary (but not sufficient) condition for commuting matrices is that they are simultaneously triangularizable.

Example 9.7 The matrices $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ commute, but the first is not diagonalizable.

9.2.12 Monotonicity

Some algorithms get closer to the solution every iteration, and/or decrease the cost function every iteration, whereas other algorithms have no such guarantee.

For PGD, there are many quantities we can evaluate to see if they do (or do not) decrease monotonically, where to be precise we really mean “non-increasing” but usually we say “decreasing” for brevity:

- $f(\mathbf{x}_{k+1}) \stackrel{?}{\leq} f(\mathbf{x}_k)$ (cost function)
 - $\|\nabla f(\mathbf{x}_{k+1})\| \stackrel{?}{\leq} \|\nabla f(\mathbf{x}_k)\|$ (gradient norm)
 - $\|\mathbf{x}_{k+1} - \hat{\mathbf{x}}\| \stackrel{?}{\leq} \|\mathbf{x}_k - \hat{\mathbf{x}}\|$ (distance to solution in natural coordinates)
 - $\|\delta_{k+1}\| \stackrel{?}{\leq} \|\delta_k\|$, $\delta_k \triangleq P^{-1/2}(\mathbf{x}_k - \hat{\mathbf{x}})$ (distance to solution with a P -related coordinate transform)
 - $\|\tilde{\delta}_{k+1}\| \stackrel{?}{\leq} \|\tilde{\delta}_k\|$, $\tilde{\delta}_k \triangleq V' \delta_k$ (distance to solution under the V transform).

When one of the above inequalities holds (typically, with strict inequality for $x_k \neq \hat{x}$) we say the algorithm is monotonic or monotone, but to be complete we should also qualify that by saying in which sense it is monotone (e.g., cost function, gradient norm, distance to solution).

The easiest of the above list to analyze for PGD is the last one. Recall from (9.9) that $\tilde{\delta}_{k+1} = \Lambda \tilde{\delta}_k$. If $\tilde{\delta}_k \neq 0$, then because we choose P so that $-1 < \lambda_n < 1$ per (9.14), the distance to the solution diminishes monotonically in these coordinates:

$$\|\tilde{\delta}_{k+1}\|_2 = \|\mathbf{A}\tilde{\delta}_k\|_2 \leq \|\mathbf{A}\|_2 \|\tilde{\delta}_k\|_2 < \|\tilde{\delta}_k\|_2, \quad (9.24)$$

where the last inequality holds if $\|\tilde{\delta}_k\|_2$ is nonzero.

Q9.18 If $P \succ 0$ and $2P^{-1} \succ A'A \succ 0$, does $\|\delta_k\|_2$ decrease monotonically?

A: Yes

B: Not necessarily

Q9.19 Under the same conditions, when $x_k \neq \hat{x}$ does the distance $\|x_k - \hat{x}\|_2$ decrease monotonically? Choose the most general correct answer.

monotonically? Choose the most general correct answer.

A. Yes

B: Yes, if the right singular vectors of A are eigenvectors of P

C: Yes, if the left singular vectors of A are eigenvectors of P

D: Yes, if $P = \alpha I$ for suitably chosen α

E: No

With a similar approach we can analyze the norm of the gradient as follows:

$$\nabla f(\mathbf{x}) = A'(Ax - y) = A'A(\mathbf{x} - \hat{\mathbf{x}}),$$

$$\begin{aligned}\nabla f(\mathbf{x}_{k+1}) &= A'A(\mathbf{x}_{k+1} - \hat{\mathbf{x}}) = A'A(\mathbf{x}_k - PA'A(\mathbf{x}_k - \hat{\mathbf{x}}) - \hat{\mathbf{x}}) \\ &= A'A(I - PA'A)(\mathbf{x}_k - \hat{\mathbf{x}}) \\ &= (I - A'AP)A'A(\mathbf{x}_k - \hat{\mathbf{x}}) = (I - A'AP)\nabla f(\mathbf{x}_k) \quad (9.25)\end{aligned}$$

$$\implies \|\nabla f(\mathbf{x}_{k+1})\|_2 \leq \|I - A'AP\|_2 \|\nabla f(\mathbf{x}_k)\|_2. \quad (9.26)$$

Again, if P commutes with $A'A$, then that spectral norm equals the spectral radius, which is less than one, and the gradient norm is nonincreasing.

Example 9.8 If $A'A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$ and $P = \begin{bmatrix} 1/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix}$, then

$$2P^{-1} - A'A = \begin{bmatrix} 10 & -7 \\ -7 & 5 \end{bmatrix},$$

which is positive definite, so P is a valid preconditioner, for which $I - P^{1/2}A'AP^{1/2}$ has eigenvalues ± 0.9428 . But

$$I - A'AP = \begin{bmatrix} 0 & -4/3 \\ -2/3 & 0 \end{bmatrix},$$

which has spectral norm $4/3$. Nevertheless, $\|\nabla f(\mathbf{x}_k)\|_2$ does converge to zero,

because $(I - A'AP)^k = (8/9)^{k/2}I_2$ for k even and $(8/9)^{(k-1)/2} \begin{bmatrix} 0 & -4/3 \\ -2/3 & 0 \end{bmatrix}$ for k odd. However, the decrease is not monotone. If $\mathbf{x}_0 - \hat{\mathbf{x}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ then

$$\nabla f(\mathbf{x}_0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \nabla f(\mathbf{x}_1) = \begin{bmatrix} -4/3 \\ -2/3 \end{bmatrix},$$

and $\|\nabla f(\mathbf{x}_0)\|_2 = \sqrt{2} < \|\nabla f(\mathbf{x}_1)\|_2 = \sqrt{20}/3$.

There is a better approach to analyze the gradient norm decrease without assuming commutativity. Rewriting the recursion in (9.25):

$$\begin{aligned}\nabla f(\mathbf{x}_{k+1}) &= P^{-1/2}(I - P^{1/2}A'AP^{1/2})P^{1/2}\nabla f(\mathbf{x}_k) \\ \implies P^{1/2}\nabla f(\mathbf{x}_{k+1}) &= (I - P^{1/2}A'AP^{1/2})P^{1/2}\nabla f(\mathbf{x}_k) \\ \implies P^{1/2}\nabla f(\mathbf{x}_k) &= (I - P^{1/2}A'AP^{1/2})^k P^{1/2}\nabla f(\mathbf{x}_0).\end{aligned} \quad (9.27)$$

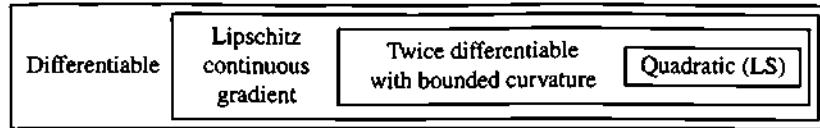


Figure 9.4 Venn diagram for differentiable convex functions.

Thus, as long as the spectral radius of $(I - P^{1/2}A'AP^{1/2})$ is less than unity, then the norm of $P^{1/2}\nabla f(x_k)$ will decrease monotonically. This property in turn ensures that $\|\nabla f(x_k)\|$ decreases to zero, though Example 9.8 illustrates that more conditions on P would be needed to show that decrease is monotonic. See also [265, Lemma 2].

9.3 Preconditioned Steepest Descent

Instead of using GD with a fixed step size α , an alternative is to use a line search to find the best step size *at each iteration*. This variation is called steepest descent (or GD with a line search). Here is how preconditioned steepest descent for a linear LS problem works:

$$\begin{aligned}
 d_k &= -P\nabla f(x_k) = -PA'(Ax_k - y) && \text{search direction} \\
 &\quad (\text{negative preconditioned gradient}) \\
 \alpha_k &= \arg \min_{\alpha} f(x_k + \alpha d_k) && \text{step size via a line search} \\
 x_{k+1} &= x_k + \alpha_k d_k && \text{update.}
 \end{aligned} \tag{9.28}$$

- Finding α_k for the LS problem is Problem 5.6.
- By construction, this iteration is guaranteed to decrease the cost function monotonically, $f(x_{k+1}) \leq f(x_k)$, with strict decrease unless x_k is already a minimizer, provided the preconditioner P is positive definite.
- Computing α_k takes some extra work, especially for nonquadratic problems. Often Nesterov's fast gradient method or the optimized gradient method (OGM) [264] are preferable because they do not require a line search (if the Lipschitz constant is available).

9.4 Gradient Descent for Smooth Convex Functions

So far we used (preconditioned) gradient descent (PGD) only for LS problems. We often need to solve more general (non-LS) unconstrained minimization problems: $\hat{x} = \arg \min_{x \in \mathbb{R}^N} f(x)$. What algorithm we use depends in part on the properties of the cost function $f: \mathbb{R}^N \mapsto \mathbb{R}$. See Fig. 9.4.

GD is applicable to the broader family of convex functions having gradients that are Lipschitz continuous. We call these smooth convex functions.

9.4.1 Lipschitz Continuity

Definition A function $f: \mathbb{F}^N \mapsto \mathbb{F}^M$ is called Lipschitz continuous iff there exists $L < \infty$ such that

$$\|f(x) - f(z)\|_\beta \leq L\|x - z\|_\alpha \quad \forall x, z \in \mathbb{F}^N \quad (9.29)$$

for some norm $\|\cdot\|_\alpha$ for \mathbb{F}^N and some norm $\|\cdot\|_\beta$ for \mathbb{F}^M .

The Euclidean norm is the default (i.e., $\alpha = \beta = 2$) unless otherwise specified. The smallest such L is called the *best* Lipschitz constant [266, p. 11]:

$$L_* = \sup_{x \neq z} \frac{\|f(x) - f(z)\|_\beta}{\|x - z\|_\alpha}. \quad (9.30)$$

Intuitively, the best Lipschitz constant characterizes how much a function's output can change when the input changes.

Explore 9.3 Does Lipschitz continuity of a function depend on the choice of α and β (9.29)?

Example 9.9 The ReLU function used in neural networks is $f(x) = \max(x, 0)$. By plotting this function we can see that the “rise over run” in (9.30) is upper-bounded by $L = 1$. Note that $f(x)$ is not differentiable, but it is still Lipschitz continuous. The best Lipschitz constant for this function is $L_* = 1$ because the supremum in (9.30) is achieved when $x = 7$ and $z = 5$, for example:

$$|f(x) - f(z)| / |x - z| = |f(7) - f(5)| / |7 - 5| = 1.$$

For optimization with gradient-based methods, we usually focus on Lipschitz continuity of the *gradient* of a cost function.

Definition A differentiable function $f: \mathbb{R}^N \mapsto \mathbb{R}$ has a Lipschitz continuous gradient if there exists $L < \infty$ such that

$$\|\nabla f(x) - \nabla f(z)\| \leq L\|x - z\| \quad \forall x, z \in \mathbb{R}^N. \quad (9.31)$$

Again, the Euclidean norm is the default here unless otherwise specified.

Example 9.10 For $f(x) = \frac{1}{2}\|Ax - y\|_2^2$ we have

$$\begin{aligned} \|\nabla f(x) - \nabla f(z)\|_2 &= \|A'(Ax - y) - A'(Az - y)\|_2 = \|A'A(x - z)\|_2 \\ &\leq \|A'A\|_2\|x - z\|_2. \end{aligned}$$

The upper bound is achieved when $x - z = v_1$, the leading right singular vector of A , so the best Lipschitz constant of ∇f is

$$L_* = \|A'A\|_2 = \|A\|_2^2 = \sigma_1^2(A) = \rho(A'A). \quad (9.32)$$

Fact 9.7 If $f: \mathbb{R}^N \mapsto \mathbb{R}$ is twice differentiable and if there exists $L < \infty$ such that its Hessian matrix has a bounded spectral norm,

$$\|\nabla^2 f(\mathbf{x})\|_2 \leq L \quad \forall \mathbf{x} \in \mathbb{R}^N, \quad (9.33)$$

then the gradient of $f(\mathbf{x})$ is Lipschitz continuous with Lipschitz constant L .

So, twice differentiability with bounded curvature is sufficient, but not necessary, for a function to have Lipschitz continuous gradient.

Proof. Using Taylor's theorem, the triangle inequality, and the definition of spectral norm:

$$\begin{aligned} \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{z})\|_2 &= \left\| \int_0^1 \nabla^2 f(\mathbf{x} + \tau(\mathbf{z} - \mathbf{x})) d\tau (\mathbf{x} - \mathbf{z}) \right\|_2 \\ &\leq \left(\int_0^1 \|\nabla^2 f(\mathbf{x} + \tau(\mathbf{z} - \mathbf{x}))\|_2 d\tau \right) \|\mathbf{x} - \mathbf{z}\|_2 \\ &\leq \left(\int_0^1 L d\tau \right) \|\mathbf{x} - \mathbf{z}\|_2 = L \|\mathbf{x} - \mathbf{z}\|_2. \end{aligned} \quad \square$$

Example 9.11 $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 \implies \nabla f(\mathbf{x}) = \mathbf{A}'(\mathbf{A}\mathbf{x} - \mathbf{y}) \implies \nabla^2 f = \mathbf{A}'\mathbf{A}$, so $\|\nabla^2 f\|_2 = \|\mathbf{A}'\mathbf{A}\|_2 = L_{\nabla f}$.

Explore 9.4 Must we use the spectral norm in (9.33)?

Q9.20 The best Lipschitz constant for the gradient of $f(\mathbf{x}) \triangleq \frac{1}{2} \mathbf{x}' \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \mathbf{x}$ is:

- A: 1 B: 2 C: 4 D: 5 E: 10

9.4.2 Convexity and Hessian

Fact 9.8 If $f(\mathbf{x})$ is twice differentiable, then $f(\mathbf{x})$ is convex iff its Hessian $\nabla^2 f(\mathbf{x})$ is positive semidefinite for all \mathbf{x} .

Example 9.12 $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 \implies \nabla^2 f(\mathbf{x}) = \mathbf{A}'\mathbf{A} \succeq 0 \implies f(\mathbf{x})$ is convex for any \mathbf{A} . One can also show convexity of this $f(\mathbf{x})$ directly from the definition of convex functions.

Fact 9.9 If $f(\mathbf{x})$ is twice differentiable, then $f(\mathbf{x})$ is strictly convex if its Hessian $\nabla^2 f(\mathbf{x})$ is positive definite for all \mathbf{x} .

Why not only if in the preceding fact? Because $f(\mathbf{x}) = x^4$ is strictly convex but $f'(0) = 0$.

Example 9.13 If A has full column rank, then $A'A$ is positive definite, so $f(x) = \frac{1}{2}\|Ax - y\|_2^2$ is strictly convex.

Q9.21 Suppose A is a wide matrix and consider the regularized LS cost function $f(x) \triangleq \frac{1}{2}\|Ax - y\|_2^2 + \beta \frac{1}{2}\|x\|_2^2$ where $\beta > 0$. Then f is a strictly convex function.

A: True B: False

Example 9.14 The Fair potential, used as an alternative to the 1-norm in many applications [267, 268, 269], is

$$\psi(x) = |x| - \log(1 + |x|). \quad (9.34)$$

From Fig. 9.5, it has the property of being roughly quadratic for $x \approx 0$ and roughly like $|x|$ for $|x| \gg 0$. For this function,

$$\dot{\psi}(x) = \frac{x}{1+|x|} \quad \text{and} \quad \ddot{\psi}(x) = \frac{1}{(1+|x|)^2} \leq 1. \quad (9.35)$$

The magenta line segment in Fig. 9.5 illustrates one possible x, z pair in (9.29). The

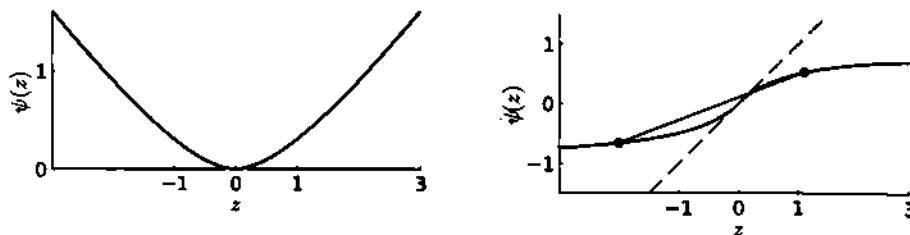


Figure 9.5 Fair potential and its derivative.

supremum in (9.29) considers all such pairs, and it is clear from the figure that the largest slope of ψ is when x, z both approach 0, corresponding to the red dashed line. The slope there is $\psi'(0)$. So the best Lipschitz constant of the derivative $\psi(\cdot)$ is $L_* = 1$, consistent with (9.33). Furthermore, its second derivative is nonnegative so it is a convex function.

Q9.22 Is the Fair potential ψ itself Lipschitz continuous?

A: No B: Yes with $L = 1/2$ C: Yes with $L = 1$ D: Yes with $L = 2$

9.4.3 GD Convergence Theorem

The GD convergence theorem states that if

- a convex function $f(x)$ has a (not necessarily unique) minimizer \hat{x} for which

$$-\infty < f(\hat{x}) \leq f(x) \quad \forall x \in \mathbb{R}^N, \quad (9.36)$$

- the gradient of $f(\mathbf{x})$ is Lipschitz continuous, that is,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{z})\|_2 \leq L \|\mathbf{x} - \mathbf{z}\|_2 \quad \forall \mathbf{x}, \mathbf{z} \in \mathbb{R}^N, \quad (9.37)$$

- and the (constant) step size α is chosen such that

$$0 < \alpha < 2/L, \quad (9.38)$$

then the GD iteration

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k) \quad (9.39)$$

has the following convergence properties [270, p. 207]:

- The cost function is monotonically nonincreasing: $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$. See [265] for a proof.
- For any minimizer $\hat{\mathbf{x}}$, the distance to that minimizer is nonincreasing: $\|\mathbf{x}_{k+1} - \hat{\mathbf{x}}\| \leq \|\mathbf{x}_k - \hat{\mathbf{x}}\|$.
- The gradient norm sequence $\{\|\nabla f(\mathbf{x}_k)\|_2\}$ is monotonically nonincreasing (see [265, Lemma 2]). Furthermore, if $\alpha \in (0, 1/L]$, then the gradient norm has a worst-case decrease of $O(1/k)$ [265, Theorem 2]:

$$\|\nabla f(\mathbf{x}_k)\|_2 \leq \frac{L}{Lak + 1} \|\mathbf{x}_0 - \hat{\mathbf{x}}\|_2. \quad (9.40)$$

This bound is tight and is achieved for a certain Huber function [265, Theorem 3].

- The sequence $\{\mathbf{x}_k\}$ converges to a minimizer of $f(\cdot)$.
- For $0 < \alpha \leq 1/L$, the cost function decrease is bounded [263, 271] by

$$f(\mathbf{x}_k) - f(\hat{\mathbf{x}}) \leq \frac{L\|\mathbf{x}_0 - \hat{\mathbf{x}}\|_2^2}{2} \max\left(\frac{1}{2kLa + 1}, (1 - La)^{2k}\right). \quad (9.41)$$

This upper bound is conjectured to also hold for $1/L < \alpha < 2/L$ and strong numerical evidence supports that conjecture [271]. A similar bound is established in [265, Theorem 4] for a particular sequence of step sizes α_k where $1/L < \alpha_k < 2/L$. The bounds (9.40) and (9.41) depend on the problem dimension N only via $\|\mathbf{x}_0 - \hat{\mathbf{x}}\|_2$.

GD is a first-order optimization method, meaning an iteration where the updates depend solely on cost function values and gradient values. The second-order methods like Newton's method also depend on the Hessian $\nabla^2 f$ of the cost function and have faster convergence rates, but the $N \times N$ Hessian is impractical to compute and store when N is large. Thus we focus on first-order methods here.

9.4.4 Nesterov's Fast Gradient Method

The $O(1/k)$ rate in (9.41) is quite slow. To improve on it, in 1983 Nesterov proposed a fast gradient method (FGM) that has the following efficient recursive form [272, 273]:

Initialize: $t_0 = 1, z_0 = x_0$

$$\begin{aligned} z_{k+1} &= x_k - \frac{1}{L} \nabla f(x_k) && \text{(usual GD update: "primary" sequence)} \\ t_{k+1} &= \frac{1}{2} \left(1 + \sqrt{1 + 4t_k^2} \right) && \text{(magic momentum factors)} \\ x_{k+1} &= z_{k+1} + \frac{t_k - 1}{t_{k+1}} (z_{k+1} - z_k) && \text{(update with momentum: "secondary" sequence)} \\ &= (1 + \gamma_k)z_{k+1} - \gamma_k z_k, & \gamma_k &= \frac{t_k - 1}{t_{k+1}} > 0. \end{aligned} \quad (9.42)$$

This method is easily implemented, but its convergence proof is nontrivial.

Nesterov showed an $O(1/k^2)$ convergence rate for the “primary” sequence $\{z_k\}$, greatly improving on (9.41). The cost function descent has the following upper bound [264, Eq. (3.5)]:

$$f(z_k) - f(\hat{x}) \leq \frac{L\|x_0 - \hat{x}\|_2^2}{2t_{k-1}^2} \leq \frac{2L\|x_0 - \hat{x}\|_2^2}{(k+1)^2}. \quad (9.43)$$

Nesterov [274, pp. 59–61] also constructed a simple quadratic cost function f with a tridiagonal Hessian matrix such that, for any general first-order method,

$$\frac{(1/32)L\|x_0 - \hat{x}\|_2^2}{(k+1)^2} \leq f(x_k) - f(\hat{x}). \quad (9.44)$$

Thus the $O(1/k^2)$ rate of FGM is “optimal” in a big O sense.

There is also a bound on the convergence rate of the “secondary” sequence $\{x_k\}$ [264, Eq. (5.5)]:

$$f(x_k) - f(\hat{x}) \leq \frac{L\|x_0 - \hat{x}\|_2^2}{2t_k^2} \leq \frac{2L\|x_0 - \hat{x}\|_2^2}{(k+2)^2}. \quad (9.45)$$

The bounds (9.43) and (9.45) are asymptotically tight [271].

Q9.23 To reach a cost within ϵ of the minimum $f(\hat{x})$, how many iterations are needed?
A: $O(1)$ B: $O(1/\sqrt{\epsilon})$ C: $O(1/\epsilon)$ D: $O(1/\epsilon^2)$ E: $O(1/\epsilon^4)$

The gap between 2 in (9.43) and 3/32 in (9.44) suggests we can do better, and we can.

9.4.5 Optimized Gradient Method

Inspired by numerical results in [263], Donghwan Kim derived an optimized gradient method (OGM) [264]. With a bit of rearranging of [275, p. 199], it has the following efficient recursive form.

With $t_0 = 1$,

$$z_{k+1} = x_k - \frac{1}{L} \nabla f(x_k) \quad (\text{gradient step})$$

$$\begin{aligned}
 t_{k+1} &= \frac{1}{2} \left(1 + \sqrt{1 + 4t_k^2} \right) && \text{(momentum factors)} \\
 \mathbf{x}_{k+1} &= \underbrace{\mathbf{x}_k - \frac{1 + t_k/t_{k-1}}{L} \nabla f(\mathbf{x}_k)}_{\text{over-relaxed GD}} + \underbrace{\frac{t_k - 1}{t_{k-1}} (\mathbf{z}_{k+1} - \mathbf{z}_k)}_{\text{FGM momentum}}. && (9.46)
 \end{aligned}$$

Remarkably, the only modification to FGM is the added ratio t_k/t_{k-1} . OGM reverts to Nesterov's FGM if one removes that new term. The new momentum term is similar to Güler's proximal point algorithm [276]. The convergence bound on the “primary” sequence [275, Eq. (25)] is

$$f(\mathbf{z}_k) - f(\hat{\mathbf{x}}) \leq \frac{L\|\mathbf{x}_0 - \hat{\mathbf{x}}\|_2^2}{4t_{k-1}^2} \leq \frac{1L\|\mathbf{x}_0 - \hat{\mathbf{x}}\|_2^2}{(k+1)^2}. \quad (9.47)$$

Of course the bound is $O(1/k^2)$, but the constant is twice better than that of Nesterov's FGM in (9.43). The secondary sequence $\{\mathbf{x}_k\}$ has a bound comparable to FGM, so use \mathbf{z}_k as the final output [275, Table 1].

One can show that $1 + t_k/t_{k+1} \rightarrow 2$ as $k \rightarrow \infty$, and recall that GD converges for a step size $\alpha < 2/L$, so the modification seems sensible, but proving that it works is complicated.

Y. Drori [277, Theorem 3] considered the class of general first-order methods and showed that, for $N > k$ (large-scale problems), where k is the number of iterations that will be run, *any* algorithm in that class has a function f with a lower bound on $f(\mathbf{x}_k) - f(\hat{\mathbf{x}})$ that matches the upper bound of OGM. Thus, OGM has optimal (worst-case) complexity among all first-order methods and no further improvements in worst-case performance are possible for the general class of smooth convex cost functions.

In general, neither FGM nor OGM guarantee nonincreasing cost functions, despite the bound $1/k^2$ being strictly decreasing. Optimal methods in general do not ensure that $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$ [274, p. 71].

9.4.6 Gradient Projection Method

In many applications, we seek the minimizer of a convex function $f(\mathbf{x})$ over a convex set C :

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in C} f(\mathbf{x}). \quad (9.48)$$

As mentioned in Section 4.9, this is called constrained optimization.

Fact 9.10 [270, p. 207] The conclusions about convergence for GD given above also hold for the gradient projection (GP) method

$$\mathbf{x}_{k+1} = \mathcal{P}_C(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)), \quad (9.49)$$

where $\mathcal{P}_C(\mathbf{v})$ denotes the projection of \mathbf{v} onto the set C per (4.47).

Example 9.15 The nonnegative least-squares (NNLS) problem is for the case where $f(\mathbf{x}) = \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2$ and $C = \mathbb{R}_+^N$.

Q9.24 If $f(\mathbf{x}) = \frac{1}{2} \left\| \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \mathbf{x} - \begin{bmatrix} 4 \\ -2 \end{bmatrix} \right\|^2$ and we apply GP for $C = \mathbb{R}_+^N$ with $\alpha = 1/L$ and $\mathbf{x}_0 = \mathbf{0}$, then $\mathbf{x}_1 = ?$

A: (0,0) B: (1,0) C: (1,-1) D: (0,-1) E: (1,1)

9.5 Machine Learning via Logistic Regression for Binary Classification

In a binary classification problem, we are given training feature vectors $\{\mathbf{v}_i\} \in \mathbb{R}^N$ and corresponding binary labels $\{y_i = \pm 1 : i = 1, \dots, M\}$. Given these training pairs (\mathbf{v}_i, y_i) , the training goal is to learn weights $\mathbf{x} \in \mathbb{R}^N$ such that $\langle \mathbf{x}, \mathbf{v}_i \rangle > 0$ if $y_i = +1$ and $\langle \mathbf{x}, \mathbf{v}_i \rangle < 0$ if $y_i = -1$, that is, such that $\langle \mathbf{x}, y_i \mathbf{v}_i \rangle > 0$, so that a reasonable classifier is $\text{sign}(\langle \mathbf{x}, \mathbf{v} \rangle)$.

Example 9.16 Classifying grayscale images of handwritten digits 0 and 1 is a simplification of handwritten digit recognition. See Figs. 9.6 and 9.7. In this example, we “hand crafted” the weights \mathbf{x} , hoping that $\text{sign}(\langle \mathbf{x}, \mathbf{v} \rangle)$ will be correct for a test image \mathbf{v} . For good statistical performance, we want to learn \mathbf{x} from training data. Learning is especially important for harder problems like classifying the digits 5 and 8. See Demo 9.2.

Q9.25 The decision boundary, that is, the set of points $\{\mathbf{v} \in \mathbb{R}^N : \langle \mathbf{x}, \mathbf{v} \rangle = 0\}$, is a (choose the most specific answer):

- A: subspace B: plane C: sphere D: convex set E: None of these

Section 5.9.4 described a linear LS approach to learning the classifier weights \mathbf{x} that Section 5.9.5 characterized in terms of empirical risk minimization (ERM). That ERM approach used a squared-error loss function that arguably is unnatural for binary classification problems. Here we consider more natural loss functions.

To learn the weights \mathbf{x} , we can minimize a cost function with a regularization parameter $\beta \geq 0$:

$$\hat{\mathbf{x}} \triangleq \arg \min_{\mathbf{x}} f(\mathbf{x}),$$

$$f(\mathbf{x}) = \sum_{m=1}^M h(\langle \mathbf{x}, y_m \mathbf{v}_m \rangle) + \beta \frac{1}{2} \|\mathbf{x}\|_2^2 = \mathbf{1}'_M h(\mathbf{A}\mathbf{x}) + \beta \frac{1}{2} \|\mathbf{x}\|_2^2, \quad (9.50)$$

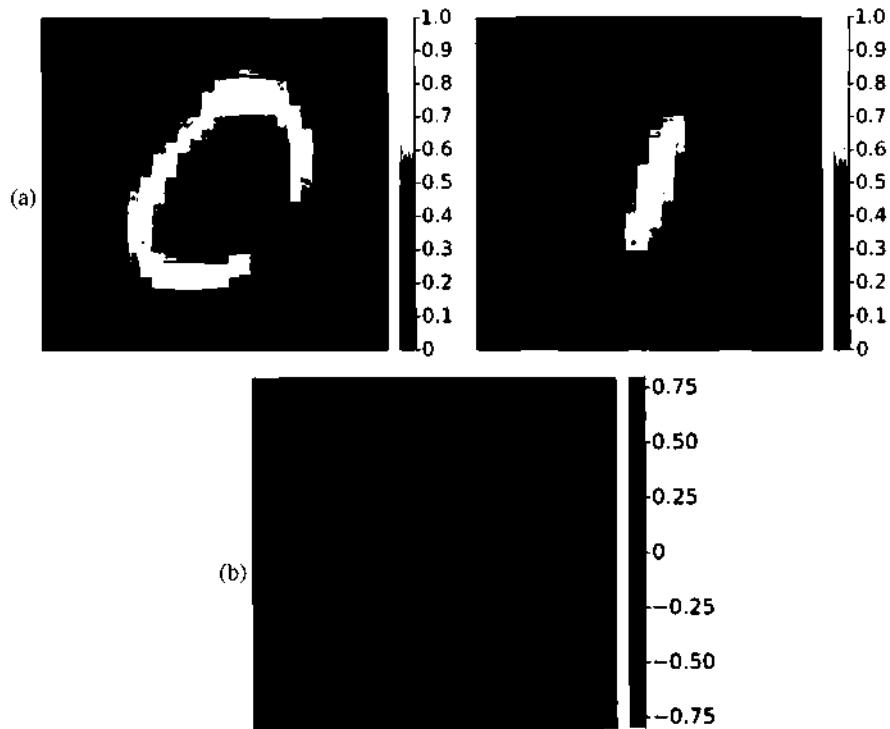


Figure 9.6 Handwritten digit classification using weight vector \mathbf{x} (b) defined (for illustration) as the difference between the two class means (a).

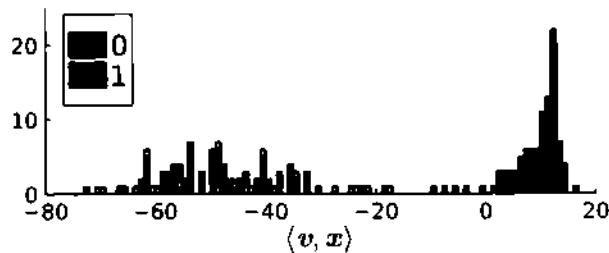


Figure 9.7 Inner products $(\mathbf{v}_i, \mathbf{x})$ using weight vector \mathbf{x} shown in Fig. 9.6 for 0 and 1 handwritten digits.

where the m th row of the $M \times N$ matrix \mathbf{A} is $y_m \mathbf{v}_m^T$, that is,

$$\mathbf{A} \triangleq \begin{bmatrix} y_1 \mathbf{v}_1^T \\ \vdots \\ y_M \mathbf{v}_M^T \end{bmatrix}.$$

A regularization term like $\|\mathbf{x}\|_2^2$ is especially important in the typical case where the feature vector dimension N is large relative to the sample size M . The 1-norm is also often used.

- ◆◆ **Explore 9.5** Referring to (5.77), the cost function (9.50) corresponds to ERM with what loss function?

We want the loss function h to discourage incorrect classification of the training data, that is, we want $\text{sign}(\langle \mathbf{x}, \mathbf{v}_i \rangle) = y_i$, or $\text{sign}(\langle \mathbf{x}, y_i \mathbf{v}_i \rangle) = 1$, or $\langle \mathbf{x}, y_i \mathbf{v}_i \rangle > 0 \forall i$. See Fig. 9.8.

- The 0–1 loss function $h(z) = \mathbb{I}_{\{z \leq 0\}}$ is natural because it counts how many training samples are misclassified, but it is nonconvex and nondifferentiable, so very difficult to use for optimization. Instead, one usually uses surrogate loss functions.
- The hinge loss function $h(z) = \max(1 - z, 0)$ is related to the soft-margin support-vector machine (SVM) and is convex, but nondifferentiable.
- The logistic loss function is convex and has a Lipschitz continuous derivative:

$$\begin{aligned} h(z) &= \log(1 + e^{-z}), \\ \dot{h}(z) &= -e^{-z} / (1 + e^{-z}) = -1 / (e^z + 1), \\ \ddot{h}(z) &= \frac{e^z}{(e^z + 1)^2} \in (0, \frac{1}{4}). \end{aligned} \quad (9.51)$$

It is suitable for gradient-based methods.

- The exponential loss function is convex and differentiable, but its derivative is not Lipschitz continuous. We will not consider it further.

For the logistic loss, the cost function is not quadratic, but it does have a Lipschitz continuous gradient.

For gradient-based optimization, we need the cost function gradient:

$$\begin{aligned} \underbrace{\nabla f(\mathbf{x})}_{\text{in } \mathbb{R}^N} &= \nabla \left(\mathbf{1}'_M h(\mathbf{A}\mathbf{x}) + \beta \frac{1}{2} \|\mathbf{x}\|_2^2 \right) = \left(\sum_{m=1}^M \nabla h(\mathbf{A}_m, \mathbf{x}) \right) + \beta \mathbf{x} \\ &= \left(\sum_{m=1}^M \mathbf{A}'_{m,:} \dot{h}(\mathbf{A}_m, \mathbf{x}) \right) + \beta \mathbf{x} = \mathbf{A}' \dot{h}(\mathbf{A}\mathbf{x}) + \beta \mathbf{x}. \end{aligned} \quad (9.52)$$

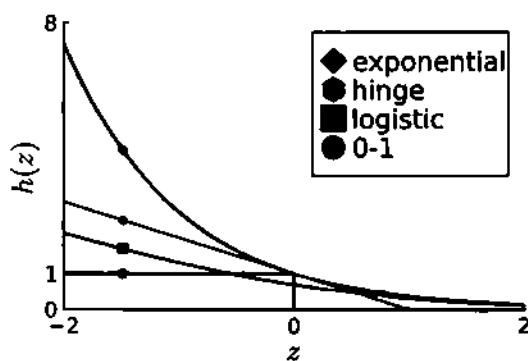


Figure 9.8 Binary classification loss functions.

The cost function Hessian matrix is

$$\begin{aligned}
 \nabla^2 f(x) &= \nabla^\top \nabla f(x) = \nabla^\top \left(\sum_{m=1}^M A'_{m,:} \hat{h}(A_m, x) + \beta I \right) \\
 &= \sum_{m=1}^M A'_{m,:} \hat{h}(A_m, x) A_{m,:} + \beta I \\
 &= A' \text{Diag}\{\hat{h}_\cdot(Ax)\} A + \beta I \\
 &= \underbrace{A'D(x)A}_{\succeq 0} + \beta \underbrace{I}_{\succ 0}, \quad D(x) \triangleq \text{Diag}\{\hat{h}_\cdot(Ax)\} \succ 0,
 \end{aligned} \tag{9.53}$$

where ∇^\top denotes a row gradient operator applied to a column vector.

Q9.26 f is a strictly convex function when ψ is the logistic loss function and $\beta > 0$.
 A: True B: False

To apply GD to the cost function (9.50), we need a Lipschitz constant for its gradient. Next we describe two different ways of deriving a bound.

Method 1. Start with the gradient expression (9.52):

$$\begin{aligned}
 \|\nabla f(x) - \nabla f(z)\|_2 &= \|A'\hat{h}_\cdot(Ax) + \beta x - A'\hat{h}_\cdot(Az) - \beta z\|_2 \\
 &= \|A'(\hat{h}_\cdot(Ax) - \hat{h}_\cdot(Az)) + \beta(x - z)\|_2 \\
 &\leq \|A'\|_2 \|\hat{h}_\cdot(Ax) - \hat{h}_\cdot(Az)\|_2 + \beta \|x - z\|_2 \\
 &\leq \|A'\|_2 L_h \|Ax - Az\|_2 + \beta \|x - z\|_2 \\
 &\leq L_h \|A'\|_2 \|A\|_2 \|x - z\|_2 + \beta \|x - z\|_2 \\
 &= (\|A'A\|_2 L_h + \beta) \|x - z\|_2 \\
 \implies L_{\nabla f} &= L_h \|A'A\|_2 + \beta.
 \end{aligned} \tag{9.54}$$

For the second inequality we used the fact that \hat{h} is Lipschitz:

$$\|\hat{h}_\cdot(s) - \hat{h}_\cdot(t)\|_2^2 = \sum_m |\hat{h}_\cdot(s_m) - \hat{h}_\cdot(t_m)|^2 \leq \sum_m L_h^2 |s_m - t_m|^2 = L_h^2 \|s - t\|_2^2. \tag{9.55}$$

Method 2. The Hessian majorizer leads to a Lipschitz constant bound for $\nabla f(x)$ that is useful for many iterative algorithms:

$$\begin{aligned}
 \nabla^2 f(x) &= A'DA + \beta I \preceq A' \left(\frac{1}{4} I \right) A + \beta I = \frac{1}{4} A'A + \beta I \\
 &\preceq \frac{1}{4} \|A'A\|_2 I + \beta I = \left(\frac{1}{4} \|A\|_2^2 + \beta \right) I.
 \end{aligned} \tag{9.56}$$

Thus, a Lipschitz constant bound that depends on the training data A and the regularization parameter β is

$$L \triangleq \frac{1}{4} \|A\|_2^2 + \beta. \tag{9.57}$$

Notice the process here: typically we do not try to find L numerically. Instead we analyze either ∇f or $\nabla^2 f$ and use properties and inequalities (analytically, on paper) to find a suitable L expressed in terms of the problem variables (in this case A and β).

9.5.1 Practical Implementation of Logistic Regression

Some practical aspects of logistic regression include the following:

- Include the constant “1” as an element of each feature vector v_i , that is, use $\tilde{v}_i \triangleq \begin{bmatrix} v_i \\ 1 \end{bmatrix} \in \mathbb{R}^{N+1}$. With this modification, the decision boundary is a subspace in \mathbb{R}^{N+1} but is a flat (or linear variety) in \mathbb{R}^N , that is, it need not intersect the origin.
- Normalizing each row of A to unit norm can help keep e^z from overflowing.
- Tuning β should use cross validation or other such tools from machine learning.
- The cost function is convex with Lipschitz gradient, so it is well suited for Nesterov’s fast gradient method (or OGM).
- When feature dimension N is very large, seeking a sparse weight vector x may be preferable; replace the Tikhonov regularizer $\|x\|_2^2$ with $\|x\|_1$ and then use FISTA [278] (or POGM [279]) for optimization.

Explore 9.6 Consider the cost function (9.50) and describe constraints (if any) on the units of v_i , x , and β .

9.5.2 Numerical Results: Logistic Regression

Example 9.17 Figure 9.9 illustrates logistic regression for binary classification for a synthetic example with labeled training data (green and blue points) where $N = 2$ and $M = 110$. The figure shows the initial decision boundary (red) and the ideal boundary (yellow) based on the Bayes classifier for the underlying data distribution. The magenta line is the final decision boundary $\{v \in \mathbb{R}^2 : \langle v, \hat{x} \rangle = 0\}$ after solving (9.50). See Demo 9.3.

9.6 Stochastic Gradient Descent

The cost function (9.50) for logistic regression is a sum of M terms, one for each data point. More generally, problems based on empirical risk minimization, including linear LS fitting, have cost functions of the form (5.77), that we can write generally like (5.78) as

$$\hat{x} = \arg \min_{x \in \mathbb{R}^N} f(x), \quad f(x) = \frac{1}{M} \sum_{m=1}^M f_m(x), \quad (9.58)$$

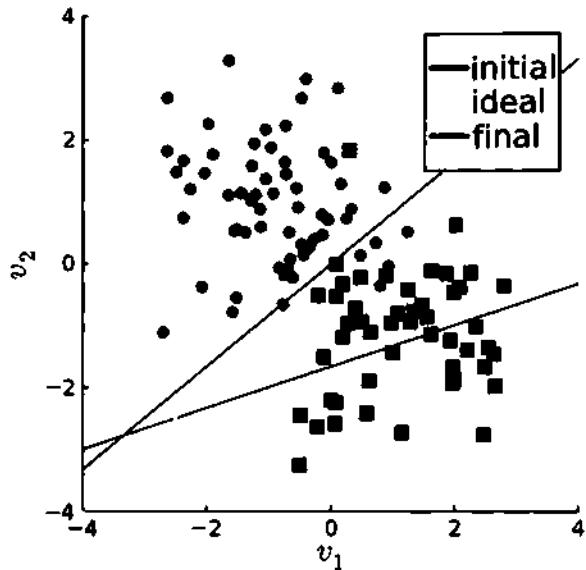


Figure 9.9 Logistic regression illustration.

where $f_m: \mathbb{F}^N \mapsto \mathbb{R}$, $m = 1, \dots, M$. Many DS–ML–SP applications involve cost functions of this additive form. For such cost functions, the GD update (9.39) becomes

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \frac{1}{M} \sum_{m=1}^M \nabla f_m(\mathbf{x}_k). \quad (9.59)$$

For applications where M is large, it can be expensive to compute the gradients ∇f_m for every m . The stochastic gradient descent (SGD) method reduces computation by approximating the gradient using just a batch of data each update [280, 281]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \frac{1}{|\mathcal{B}_k|} \sum_{m \in \mathcal{B}_k} \nabla f_m(\mathbf{x}_k), \quad (9.60)$$

where $\mathcal{B}_k \subset \{1, \dots, M\}$ is a random subset of the data. The “D” for descent might be misleading because the cost function f typically does not decrease every iteration. In machine learning settings, the step size α_k is called the learning rate and is often scheduled to change with iteration k . Those schedules involve hyperparameters that are chosen empirically, leading to the phenomena dubbed grad student descent [282]. The goal of automating that process led to many methods for choosing α_k adaptively, such as the Adam approach [283] that uses momentum and is widely used in machine learning.

9.7 Summary

This chapter has focused on gradient descent methods for optimization. Variations of GD, especially stochastic (sub)gradient methods like Adam [283], are used extensively

for training machine learning models. The logistic regression application considered in Section 9.5 has a convex and smooth cost function, where GD and accelerated GD have favorable convergence properties. Many modern machine learning models involve nonconvex and nonsmooth cost functions, for which algorithms like the alternating direction method of multipliers (ADMM) are used [284, 285]. Entire books are devoted to optimization methods for such problems, for example [9, 286]. This chapter barely scratches the surface of the field of optimization algorithms, but it illustrates how crucial matrix methods are to that field.

Solutions to Explorations

Explore 9.1 If $A = \text{Diag}\{\gamma_i\} X \text{Diag}\{\beta_j\}$, then x_j must have units $1/\beta_j$ so that Ax is valid, and $\gamma_i = \gamma$ must be constants and the units of each y_i must be γ so that the norm $\|Ax - y\|_2$ is well defined. (A weighted LS formulation would be needed if the y_i units differ.) The gradient vector $g = A'(Ax - y)$ then has elements with units $g_j = \beta_j \gamma^2$. For ordinary GD, we must have $\beta_j = \beta$ for all j and the units of α must be $(\gamma \beta)^{-2}$ so that the subtraction in (9.2) has consistent units.

For PGD, the preconditioner must have the form $P = \gamma^{-2} \text{Diag}(1/\beta_j) \tilde{P} \text{Diag}(1/\beta_j)$, where \tilde{P} is unitless.

Explore 9.2 Both return useless matrices, and neither throws an error! More recent JULIA ($\geq v1.5$ if not earlier) returns `zeros(2,2)`, whereas MATLAB returns `[0 Inf; 0 0]` with a warning.

Explore 9.3 The value of L depends on the choice of norm. But because all norms are equivalent per (6.10), if a function has Lipschitz continuity with respect to any particular choice of norms, it also does for all norms.

Explore 9.4 No, because of norm equivalence.

Explore 9.5 Ignoring the regularizer, the loss here is $L(\hat{y}, y) = Mh(y\hat{y})$.

Explore 9.6 Let α_n and γ_n denote the units of x_n and v_n , the n th elements of x and v , respectively. The inner product $\langle v_i, x \rangle$ term requires $\alpha_n = c/\gamma_n$, for some constant c . On the other hand, the regularizing norm $\|x\|_2$ requires that all elements of x have the same units. So in fact all elements of all v_i must have the same units (or be unitless). Also, the argument of the logistic function must be unitless (due to the exponential). So typically one first normalizes the features to have zero mean and (unitless) unit variance.

Problems

Problem 9.1

- (a) Derive the optimal step size α_* for classical gradient descent of the least-squares cost function $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{Ax} - \mathbf{y}\|_2^2$ in terms of the singular values of the (full column rank) matrix \mathbf{A} . Here, optimal means $\rho(\mathbf{I} - \alpha\mathbf{A}'\mathbf{A})$ is as small as possible.
- (b) For that optimal step size α_* , determine the spectral radius $\rho(\mathbf{I} - \alpha_*\mathbf{A}'\mathbf{A})$ that governs the convergence rate of the sequence $\{\mathbf{x}_k\}$ produced by the GD method. Express your answer in terms of the condition number of $\mathbf{A}'\mathbf{A}$.

Problem 9.2 Show that when \mathbf{P} is symmetric positive definite, $\text{eig}(\mathbf{P}^{1/2}\mathbf{A}'\mathbf{A}\mathbf{P}^{1/2}) < 2 \iff 2\mathbf{P}^{-1} \succ \mathbf{A}'\mathbf{A}$. This property is important for the preconditioned gradient descent algorithm for solving least-squares and related problems.

Problem 9.3 This problem constructs a simple and practical diagonal majorizer for use in iterative algorithms. An $N \times N$ square matrix \mathbf{A} is called diagonally dominant iff $|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|$ for $i = 1, \dots, N$.

- (a) Optional. Use the Geršgorin disk theorem to show that if \mathbf{H} is a (Hermitian) symmetric matrix that is diagonally dominant with $h_{ii} \geq 0$ for all i , then $\mathbf{H} \succeq \mathbf{0}$.
- (b) Show that if \mathbf{B} is an $N \times N$ Hermitian symmetric matrix, and $|\mathbf{B}|$ denotes the matrix consisting of the absolute values of the elements of \mathbf{B} , that is, `abs.(B)` in JULIA, and `1_N` is `ones(N)` in JULIA, then $\mathbf{D} \triangleq \text{Diag}\{|\mathbf{B}|1_N\} \succeq \mathbf{B}$.

Problem 9.4 Practical preconditioners are often built on approximations of a cost function's Hessian inverse. The initial design of such preconditioners often does not ensure convergence of PGD, so a step size α is usually needed. Specifically, for an LS problem we might have $\mathbf{P}_0 \approx (\mathbf{A}'\mathbf{A})^{-1}$ and then let $\mathbf{P} = \alpha\mathbf{P}_0$, where \mathbf{P}_0 is positive definite.

- (a) Determine the optimal (fixed) step size α_* that makes the preconditioned gradient descent (PGD) iteration $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\mathbf{P}_0\mathbf{A}'(\mathbf{Ax}_k - \mathbf{y})$ have the fastest possible convergence rate.

Hint: The answer depends on both \mathbf{A} and the preconditioner \mathbf{P}_0 .

- (b) Simplify your expression for α_* as much as possible in the special case where we use the ideal preconditioner $\mathbf{P}_0 = (\mathbf{A}'\mathbf{A})^{-1}$. Does your simple expression (a number) make sense?

Problem 9.5 Let ψ denote the (scaled) Fair potential function

$$\psi(t) \triangleq \delta^2 \left(\frac{|t|}{\delta} - \log \left(1 + \frac{|t|}{\delta} \right) \right),$$

with parameter $\delta > 0$.

- (a) Show that

$$\dot{\psi}(t) = \frac{t}{1 + |t|/\delta}.$$

Hint: At first glance, it may look like ψ is not differentiable due to the absolute value, but try examining $\dot{\psi}(t)$ separately for $t \geq 0$ and $t < 0$.

- (b) Show that the (best) Lipschitz constant of $\dot{\psi}$ is $\mathcal{L}_{\dot{\psi}} = 1$.

Hint: Recall that the best Lipschitz constant \mathcal{L}_g of a function $f: \mathbb{R} \mapsto \mathbb{R}$ is the smallest constant such that $|g(z) - g(x)| \leq \mathcal{L}_g|z - x|$ for all x, z .

Hint: If f is differentiable with bounded derivative, then $\mathcal{L}_f = \sup_t |\dot{f}(t)|$.

Hint: Show that $\ddot{\psi}(t) = \delta^2/(\delta + |t|)^2$. Again, consider $t \geq 0$ and $t < 0$ separately.

- (c) Determine the proximal operator for the (scaled) Fair potential on \mathbb{R}^N ; that is, for $v \in \mathbb{R}^N$ solve

$$\text{prox}_{\beta\psi}(v) = \arg \min_{x \in \mathbb{R}^N} \frac{1}{2} \|v - x\|_2^2 + \beta \mathbf{1}' \psi .(x).$$

Problem 9.6 An alternative to the logistic loss function for binary classifier design is the ℓ_2 -hinge loss function,

$$\psi(t) \triangleq (\max(1 - t, 0))^2.$$

- (a) Show that this loss function has a Lipschitz continuous derivative and determine its Lipschitz constant $\mathcal{L}_{\dot{\psi}}$.
(b) Now consider logistic regression using this loss function for binary classifier design using the optimization problem

$$\arg \min_x f(x), \quad f(x) = \sum_{m=1}^M \psi(y_m \langle x, v_m \rangle) + \beta \frac{1}{2} \|x\|_2^2.$$

Determine a Lipschitz constant $\mathcal{L}_{\nabla f}$ for the gradient of the overall cost function $f(x)$, where $x, v_m \in \mathbb{R}^N$ and $y_m = \pm 1$.

10 Matrix Completion and Recommender Systems

10.1 Introduction

This chapter discusses the important problem of matrix completion, where we know some, but not all, elements of a matrix and want to “complete” the matrix by filling in the missing entries. Obviously this problem is ill posed in general because one could assign arbitrary values to the missing entries, unless one assumes some model for the matrix elements. The most common model is that the matrix is low rank, an assumption that is reasonable in many applications [151].

Section 10.2 defines the problem and Section 10.3 describes an alternating projection approach for noiseless data. Section 10.4 discusses algorithms for the practical case of missing and noisy data. Section 10.5.1 extends the methods to consider the effects of outliers with the robust PCA method, and applies robust PCA to video foreground/background separation. Section 10.6 describes nonnegative matrix factorization, including the case of missing data.

A particularly famous application of low-rank matrix completion (LRMC) is the Netflix problem; this topic is also relevant to dynamic MR image reconstruction, and numerous other applications with missing data (incomplete observations).

Successful applications of LRMC include the following.

- recommender systems using collaborative filtering (Netflix, Spotify, Amazon, ...)
- imaging: denoising, reconstruction in medical, hyperspectral imaging
- anomaly detection in network flows
- source localization and target tracking in radar and sonar
- computer vision: background subtraction, object tracking, representing a single scene under varying illuminations
- environmental monitoring of soil and crop conditions, water contamination, and air pollution; also sensor calibration
- seismological activity and modal estimation in materials and human-made structures.

10.2 Measurement Model

If X is a latent $M \times N$ “low-rank” matrix with rank $r \leq \min(M, N)$, then

$$X = \sum_{k=1}^r \sigma_k u_k v_k' \quad (10.1)$$

Suppose we observe only *some* of the entries of X in a sampling set Ω , that is,

$$Y_{ij} = \begin{cases} X_{ij}, & (i,j) \in \Omega, \\ ?, & \text{otherwise,} \end{cases} \quad \Omega \subset \{1, \dots, M\} \times \{1, \dots, N\}, \quad (10.2)$$

where “?” denotes a value is “missing”; for example, the i th user has not rated the j th movie.

The key questions in this setting are the following:

- For a given (known) sampling set Ω , can we recover all of X from just the observed entries Y ? The answer will depend on the rank r , where typically $r \ll \min(M, N)$, and on the sampling set.
- How do we do it?
- How well does it work (e.g., in the presence of noise or if X is not exactly low rank)?

10.2.1 Practical Implementation

This type of topic is so important to modern data analysis problems that JULIA has its own data type, `Missing`, to represent missing values. Try this:

```
Y = [2 0; missing 4; 5 missing]
```

The type of `Y` is `3x2 Matrix{Union{Missing, Int64}}` because it is a 3×2 array of values that are either 64-bit integers or the special type `Missing`. We cannot use any particular value like “0” to represent a missing value because 0 might be one of the observed values!

Q10.1 What is the minimum possible rank of Y in this example?

- A: 0 B: 1 C: 2 D: 3 E: None of these

10.2.2 Sampling Conditions for LRMC

The first question to ask is whether it should even be possible to recover X from Y in the noiseless case. If X has rank r , then we can write X as the product of an $M \times r$ matrix (think $\tilde{U}_r = U_r \Sigma_r$, for example) with the transpose of an $N \times r$ matrix \tilde{V}_r . The number of degrees of freedom (DoF) in this representation looks to be $Mr + Nr = (M + N)r$. However, actually there are fewer DoF because once we fix \tilde{U}_r , there are constraints on the values that \tilde{V}_r can take and still have $\tilde{U}_r \tilde{V}_r' = X$. So, $(M + N)r$ is an upper bound on the DoF. The total number of elements of X is MN , so if the matrix is low rank, then $(M + N)r \ll MN$ so it seems plausible that having around $(M + N)r$ samples might suffice.

A more careful analysis of the DoF is

$$\text{DoF} = rM + (N - r)r = (M + N)r - r^2, \quad (10.3)$$

because the first (if we permute appropriately) r columns (of length M) are linearly independent, and the remaining $N - r$ columns depend linearly on those first r columns and we need r coefficients per column to describe the linear combination. If $r = N \leq M$ then $(M + N)r - r^2 = (M + N)N - N^2 = MN$, as expected. When $M = N$ and $r \leq N/2$, a lower bound on the number of samples is $4Nr - 4r^2$ [287].

If $M \approx N$, then the DoF $\approx 2Nr$. So we need at least $O(Nr)$ samples to have any chance of recovery (unless we have additional information such as knowing that U_r and/or V_r is sparse). Modern algorithms need $|\Omega| \geq O(Nr \text{polylog}(N))$ random samples under certain assumptions about X [186].

Q10.2 Approximately how many observations *per column* are the minimum needed to recover a rank- r low-rank matrix of size $N \times N$?

- A: r B: $2r$ C: r^2 D: N E: $2N$

10.2.3 Sampling Mask

Define an $M \times N$ binary sampling mask matrix M in terms of the sampling set Ω by

$$M_{ij} \triangleq \begin{cases} 1, & (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \quad (10.4)$$

In words, M is zero wherever we are missing a value in Y .

Example 10.1 For the earlier 3×2 example we have $M = \text{.!ismissing.}(Y)$, that is,

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

Hereafter, let $Y \in \mathbb{R}^{M \times N}$ be a matrix where

$$Y_{ij} = \begin{cases} X_{ij}, & (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \quad (10.5)$$

Now we *can* store a 0 in the locations of missing values because we have the separate mask M to keep track of which values are sampled and which are missing.

10.3 LRMC: Noiseless Case

10.3.1 Noiseless Problem Statement

The (noiseless) LRMC problem is to find a matrix \hat{X} that is low rank and that agrees with the measurements Y on Ω , that is, we want

$$M \odot \hat{X} = M \odot Y, \quad (10.6)$$

where \odot denotes elementwise multiplication, that is, $M .\star X$ in JULIA and MATLAB, also known as the Hadamard product or array multiplication (as opposed to matrix multiplication) and is applicable to arrays of any dimension. (In Python, $M * X$ is the Hadamard product for array objects in NumPy, but is ordinary matrix multiplication for matrix objects.)

One “ideal” (noiseless) optimization formulation for the LRMC problem [288] is

$$\hat{X} = \arg \min_{X} \text{rank}(X) \text{ such that } M \odot X = M \odot Y. \quad (10.7)$$

This is a nonconvex problem and is NP hard (impractical). No fast algorithm exists that is guaranteed to solve this problem formulation. Nevertheless, there are practical algorithms that often work reasonably well.

10.3.2 Alternating Projection Approach to LRMC

An alternative to (10.7) is to pick a rank $1 \leq K \ll \min(M, N)$ and seek a matrix \hat{X} that satisfies

$$\begin{aligned} \hat{X} \in C \cap D, \quad C &\triangleq \left\{ X \in \mathbb{F}^{M \times N} : \text{rank}(X) \leq K \right\}, \\ D &\triangleq \left\{ X \in \mathbb{F}^{M \times N} : M \odot X = M \odot Y \right\}. \end{aligned} \quad (10.8)$$

In words, we seek a matrix \hat{X} having rank at most K and that agrees with the measurements Y on Ω .

The projections onto convex sets (POCS) approach is a classic signal processing tool for finding points in the intersection of two or more convex sets.

Q10.3 Which (if any) of the two sets C and D above is convex?

- A: neither C nor D B: C but not D C: D but not C D: both C and D

To analyze the set D :

$$\begin{aligned} X, Z \in D &\implies M \odot X = M \odot Y \text{ and } M \odot Z = M \odot Y \\ &\implies M \odot (\alpha X + (1 - \alpha)Z) = \alpha M \odot X + (1 - \alpha)M \odot Z \\ &\quad = \alpha M \odot Y + (1 - \alpha)M \odot Y = M \odot Y \\ &\implies \alpha X + (1 - \alpha)Z \in D. \end{aligned} \quad (10.9)$$

Q10.4 The set \mathcal{D} is a subspace.

A: True

B: False

The alternating projection method alternates between projecting onto the sets C and \mathcal{D} :

$$X_{k+1} = \mathcal{P}_C(\mathcal{P}_{\mathcal{D}}(X_k)), \quad (10.10)$$

where $\mathcal{P}_{\mathcal{D}}(X)$ denotes the projection of its argument X onto the set \mathcal{D} ; that is, the closest point in \mathcal{D} :

$$\mathcal{P}_{\mathcal{D}}(X) = \arg \min_{Z \in \mathcal{D}} \|X - Z\|_F. \quad (10.11)$$

The algorithm designer gets to choose the norm for quantifying “closest.” Here we are using matrices so we work with the simplest choice, the Frobenius norm.

For the “data” set \mathcal{D} above, the projection operation is very simple:

$$[\mathcal{P}_{\mathcal{D}}(X)]_{i,j} = \begin{cases} Y_{i,j}, & (i,j) \in \Omega, \\ X_{i,j}, & \text{otherwise.} \end{cases} \quad (10.12)$$

Example 10.2 Suppose $(M, N) = (2, 1)$ and $M = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ so $Y = \begin{bmatrix} Y_1 \\ ? \end{bmatrix}$. Then

$\mathcal{D} = \{(X_1, X_2) \in \mathbb{R}^2 : X_1 = Y_1\}$ (see Fig. 10.1). For any point (X_1, X_2) , the projection onto \mathcal{D} is simply (Y_1, X_2) .

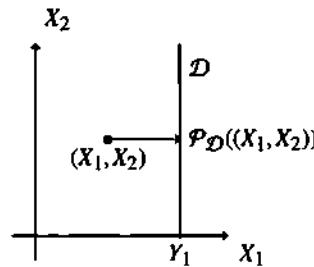


Figure 10.1 Illustration of set \mathcal{D} .

If M is a Boolean matrix, then we can implement this operation “in place” using JULIA logical indexing as follows: $X[M] . = Y[M]$.

Even though C is not convex, finding a projection onto C is fairly easy using Section 7.2:

$$\mathcal{P}_C(Z) = \arg \min_{L : \text{rank}(L) \leq K} \|Z - L\|_F = U_K \Sigma_K V'_K, \quad Z = U \Sigma V. \quad (10.13)$$

This is simply the low-rank approximation problem of (7.1). It requires an SVD of the input argument.

Q10.5 The projection $\mathcal{P}_C(\mathbf{Z})$ is unique for any $M \times N$ input matrix \mathbf{Z} .

- A: True B: False

Q10.6 Given $K = 1$, $\mathbf{Y} = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$, $\mathbf{M} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$, $\mathbf{X}_k = \begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}$, the POCS update

$\mathbf{X}_{k+1} = \mathcal{P}_C(\mathcal{P}_{\mathcal{D}}(\mathbf{X}_k))$ is:

- A: $\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$ B: $\begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$ C: $\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$ D: $\begin{bmatrix} 3 & 0 \\ 0 & 0 \end{bmatrix}$ E: $\begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}$

Explore 10.1 Describe $\mathcal{D} \cap \mathcal{C}$ for the preceding 2×2 example.



Demo 10.1 illustrates matrix completion using alternating minimization with 25% sampling. It also shows ISTA and FISTA; see Section 10.4.7, Section 10.4.10, and Fig. 10.2.

The alternating projection method works better than simply making a low-rank approximation to \mathbf{Y} , but it does not account for noise in the data, so we can do better.

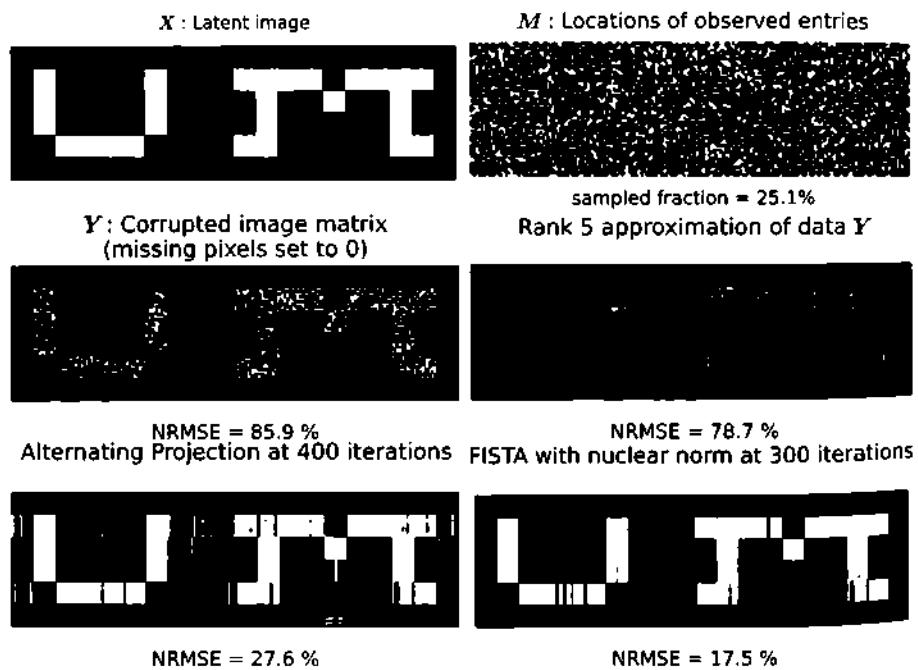


Figure 10.2 Results from matrix completion demo using alternating projection and FISTA (cf. Section 10.4.7).

10.3.2.1 Convergence of the Alternating Projection Method

Because the set of rank $\leq K$ matrices is nonconvex, it is hard to say much about the convergence of the alternating projection method. Choice of the initial guess X_0 can affect the convergence behavior.

Example 10.3 Consider $Y = \begin{bmatrix} 1 & 1 \\ ? & ? \end{bmatrix}$ with rank $K = 1$. If $X_0 = \begin{bmatrix} 1 & 1 \\ x & x \end{bmatrix}$ for any value x , then $X_k = X_0$ for all $k \in \mathbb{N}$. So in this case, the set of “optimal” solutions $C \cap D$ is uncountably infinite!

10.3.2.2 Choice of Rank

In practice one must choose the rank K to perform low-rank matrix completion. One can use cross validation or minimum description length (MDL) methods [289] among others [290]. See the survey papers [215, 291].

10.4 LRMC: Noisy Case

10.4.1 Noisy Problem Statement

In practice, there is often noise in the data, so the model (10.5) is unrealistic. A more realistic model is

$$Y_{ij} = \begin{cases} X_{ij} + \varepsilon_{ij}, & (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \quad (10.14)$$

Again, it is fine to store a 0 (or any other value) in the missing locations because we have the sampling mask M available. In this case, it would be unreasonable to insist on exact data equality, $M \odot X = M \odot Y$. So now we want to find \hat{X} where $M \odot \hat{X} \approx M \odot Y$, that is, $M \odot (\hat{X} - Y) \approx 0$, and also \hat{X} is low rank.

One possible LRMC formulation uses a (nonconvex) rank constraint:

$$\hat{X} = \arg \min_{\substack{X: \text{rank}(X) \leq K}} \|M \odot (X - Y)\|_F^2. \quad (10.15)$$

Another possible LRMC formulation uses a (nonconvex) rank regularizer:

$$\hat{X} = \arg \min_{X \in \mathbb{R}^{M \times N}} \|M \odot (X - Y)\|_F^2 + \beta \text{rank}(X). \quad (10.16)$$

The following LRMC optimization formulation is the convex relaxation of the rank regularizer, using instead a nuclear norm regularizer:

$$\hat{X} = \arg \min_X \frac{1}{2} \|M \odot (X - Y)\|_F^2 + \beta \|X\|_*. \quad (10.17)$$

There are fast and practical algorithms for all three of these formulations, and there are convergence guarantees for the convex formulation.

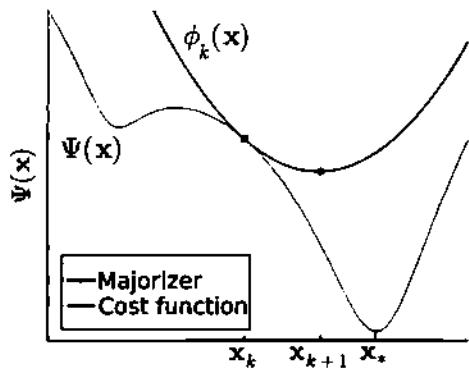


Figure 10.3 Majorizer example.

Unfortunately, the optimization tools presented in Chapter 9 (GD, PGD, PSD) are inapplicable to the above problems either because the cost function is not differentiable or it involves a nonconvex set. Even the cost function (10.17) that uses the (convex) nuclear norm $\|X\|_*$ is not differentiable. So we need an approach that is more sophisticated than gradient-based methods.

Q10.7 What is the nuclear norm $\|x\|_*$ of a 1×1 “matrix” x ?

- A: x B: $|x|$ C: $\mathbb{I}_{\{x \geq 0\}}$ D: $\max(0, x)$ E: x^2

10.4.2 Majorize–Minimize (MM) Iterations

To solve optimization problems like (10.17), we first consider majorize–minimize (MM) algorithms.

To solve an optimization problem like $\arg \min_{x \in X} \Psi(x)$ we first design a majorizer or surrogate function $\phi_k(x)$ that satisfies the following two conditions:

$$\begin{aligned}\Psi(x_k) &= \phi_k(x_k), \\ \Psi(x) &\leq \phi_k(x) \quad \forall x \in X\end{aligned}\tag{10.18}$$

(see Fig. 10.3). Then the MM algorithm update is simply

$$x_{k+1} = \arg \min_{x \in X} \phi_k(x).\tag{10.19}$$

Any MM algorithm will monotonically decrease the cost function because of the sandwich inequality:

$$\Psi(x_{k+1}) \leq \phi_k(x_{k+1}) \leq \phi_k(x_k) = \Psi(x_k).\tag{10.20}$$

Next we design an MM algorithm for LRMC problems like (10.17).

10.4.3 MM Methods for LRMC

First, define the complement of the mask matrix as follows (in JULIA this is `.!M`):

$$\tilde{M} \triangleq \mathbf{1}_M \mathbf{1}'_N - M, \quad \tilde{M}_{i,j} = \begin{cases} 0, & (i,j) \in \Omega, \\ 1, & (i,j) \notin \Omega. \end{cases} \quad (10.21)$$

We focus now on the quadratic data misfit term in (10.15)–(10.17):

$$q(X) \triangleq \|M \odot (X - Y)\|_F^2. \quad (10.22)$$

Now define the following quadratic function:

$$Q(X; Z) = \|X - Z + M \odot (Z - Y)\|_F^2 = \|X - (\tilde{M} \odot Z + M \odot Y)\|_F^2. \quad (10.23)$$

This function is a majorizer of $q(X)$ because

$$\begin{aligned} q(X) &= Q(X; X), \\ q(X) &\leq Q(X; Z) \quad \forall X, Z \in \mathbb{F}^{M \times N}. \end{aligned} \quad (10.24)$$

Proof of majorization inequality:

$$\begin{aligned} Q(X; Z) &= \|X - Z + M \odot (Z - Y)\|_F^2 \\ &= \|(\tilde{M} + M) \odot (X - Z) + M \odot (Z - Y)\|_F^2 \\ &= \|\tilde{M} \odot (X - Z) + M \odot (X - Y)\|_F^2 \\ &= \|\tilde{M} \odot (X - Z)\|_F^2 + \|M \odot (X - Y)\|_F^2 \\ &\geq \|M \odot (X - Y)\|_F^2 = q(X), \end{aligned}$$

where the fourth equality holds because $\tilde{M} \odot M = \mathbf{0}$. \square

♦♦ The design of the function (10.23) uses a second-order Taylor expansion of $q(X)$ about Z and then uses the fact that the elements of M are all 0 or 1. The above proof verifies that Q is a majorizer, which is all that is needed here.

Now we use the majorizer in (10.23) to develop a few different LRMC algorithms.

10.4.4 LRMC by Iterative Low-Rank Approximation

First consider LRMC using the rank constraint (10.15):

$$\hat{X} = \arg \min_{X : \text{rank}(X) \leq K} \|M \odot (X - Y)\|_F^2. \quad (10.25)$$

The MM algorithm update for this formulation is simply

$$\begin{aligned} X_{k+1} &= \arg \min_{X : \text{rank}(X) \leq K} Q(X; X_k) \\ &= \arg \min_{X : \text{rank}(X) \leq K} \|X - (\tilde{M} \odot X_k + M \odot Y)\|_F^2 \\ &= \arg \min_{X : \text{rank}(X) \leq K} \|X - \tilde{X}_k\|_F^2, \end{aligned}$$

$$\tilde{X}_k \triangleq \tilde{M} \odot X_k + M \odot Y = \begin{cases} Y_{i,j}, & (i,j) \in \Omega \\ [X_k]_{i,j}, & (i,j) \notin \Omega \end{cases} = \mathcal{P}_{\mathcal{D}}(X_k). \quad (10.26)$$

This algorithm alternates between the following two steps:

- Take the current guess X_k and replace all the values at sampled locations with the measurements from Y to get \tilde{X}_k . (As mentioned earlier, this can be done “in place” using $X[M] := Y[M]$.)
- Perform low-rank (rank at most K) approximation (using SVD) to \tilde{X}_k to get the next iterate X_{k+1} .

This process is exactly the same as the alternating projection method described earlier. So now we know that it is an MM method and thus it decreases the Frobenius norm cost function monotonically. (There is still no guarantee of convergence of $\{X_k\}$ to a global minimizer because the rank constraint set is nonconvex.)

10.4.5 LRMC by Iterative Singular Value Hard Thresholding

Now consider LRMC using the rank regularizer (10.16):

$$\hat{X} = \arg \min_{X \in \mathbb{R}^{M \times N}} \|M \odot (X - Y)\|_F^2 + \beta \operatorname{rank}(X). \quad (10.27)$$

The MM algorithm update for this formulation is simply

$$\begin{aligned} X_{k+1} &= \arg \min_{X \in \mathbb{R}^{M \times N}} \underbrace{Q(X; X_k)}_{\phi_k(x)} + \beta \operatorname{rank}(X) \\ &= \arg \min_{X \in \mathbb{R}^{M \times N}} \|X - (\tilde{M} \odot X_k + M \odot Y)\|_F^2 + \beta \operatorname{rank}(X) \\ &= \arg \min_{X \in \mathbb{R}^{M \times N}} \|X - \tilde{X}_k\|_F^2 + \beta \operatorname{rank}(X). \end{aligned} \quad (10.28)$$

This algorithm alternates between the following two steps:

- Take the current guess X_k and replace all the values at sampled locations with the measurements from Y to get \tilde{X}_k . (This can be done “in place.”)
- Apply singular value hard thresholding to \tilde{X}_k to get the next iterate X_{k+1} .

This MM method decreases the rank-regularized Frobenius norm cost function monotonically.

10.4.6 LRMC by Iterative Singular Value Soft Thresholding

Now consider LRMC using the convex nuclear norm regularizer (10.17):

$$\hat{X} = \arg \min_{X \in \mathbb{R}^{M \times N}} \frac{1}{2} \|M \odot (X - Y)\|_F^2 + \beta \|X\|_{**}. \quad (10.29)$$

Q10.8 Is this cost function (also in Demo 10.1) convex? Strictly convex?

- A: (T,T) B: (T,F) C: (F,T) D: (F,F)

The MM algorithm update for this formulation is simply

$$\begin{aligned} X_{k+1} &= \arg \min_{X \in \mathbb{R}^{M \times N}} \frac{1}{2} Q(X; X_k) + \beta \|X\|_* \\ &= \arg \min_{X \in \mathbb{R}^{M \times N}} \frac{1}{2} \|X - (\tilde{M} \odot X_k + M \odot Y)\|_F^2 + \beta \|X\|_* \\ &= \arg \min_{X \in \mathbb{R}^{M \times N}} \frac{1}{2} \|X - \tilde{X}_k\|_F^2 + \beta \|X\|_*. \end{aligned} \quad (10.30)$$

This algorithm alternates between the following two steps:

- Take the current guess X_k and replace all the values at sampled locations with the measurements from Y to get \tilde{X}_k . (This can be done “in place.”)
- Apply singular value soft thresholding (SVST) to \tilde{X}_k to get the next iterate X_{k+1} .

This MM method decreases its cost function monotonically. Because the cost function is convex, with some additional work one can show that the sequence (X_k) converges to a global minimizer.

10.4.7 Iterative Soft-Thresholding Algorithm

- ♦♦ In many modern applications, including LRMC, we have a composite cost function of the form

$$\arg \min_x f(x) + g(x), \quad (10.31)$$

where $f(x)$ is convex and has a Lipschitz gradient (smooth) but $g(x)$ is convex but not necessarily smooth.

To develop an algorithm for such problems, we first reinterpret the GD step for minimizing $f(x)$ as follows:

$$\begin{aligned} x_{k+1} &= \arg \min_x q(x; x_k) = x_k - \alpha \nabla f(x_k), \\ q(x; x_k) &\triangleq f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2\alpha} \|x - x_k\|_2^2. \end{aligned} \quad (10.32)$$

Proof (by completing the square).

$$\begin{aligned} q(x; x_k) &= f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2\alpha} \|x - x_k\|_2^2 \\ &= f(x_k) - \frac{\alpha}{2} \|\nabla f(x_k)\|_2^2 + \frac{1}{2\alpha} \left(\|\alpha \nabla f(x_k)\|_2^2 + 2 \langle \alpha \nabla f(x_k), x - x_k \rangle + \|x - x_k\|_2^2 \right) \\ &= \left(f(x_k) - \frac{\alpha}{2} \|\nabla f(x_k)\|_2^2 \right) + \frac{1}{2\alpha} \|x - (x_k - \alpha \nabla f(x_k))\|_2^2 \\ \implies x_{k+1} &= x_k - \alpha \nabla f(x_k). \end{aligned}$$

□

This alternate perspective on GD is the key to extending the GD approach to composite cost functions like (10.31). Consider the following iteration:

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg \min_{\mathbf{x}} \left\{ f(\mathbf{x}_k) + (\nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k) + \frac{1}{2\alpha} \|\mathbf{x} - \mathbf{x}_k\|_2^2 + g(\mathbf{x}) \right\} \\ &= \arg \min_{\mathbf{x}} \left\{ \frac{1}{2\alpha} \|\mathbf{x} - (\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))\|_2^2 + g(\mathbf{x}) \right\}. \end{aligned} \quad (10.33)$$

This is called the **iterative soft thresholding algorithm (ISTA)**, or the **iterative shrinkage thresholding algorithm**, or, more generally, the **proximal gradient method (PGM)**.

More concisely, ISTA uses the following two steps per iteration:

$$\begin{aligned} \tilde{\mathbf{x}}_k &\triangleq \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k) && \text{(usual GD step),} \\ \mathbf{x}_{k+1} &= \arg \min_{\mathbf{x}} \frac{1}{2\alpha} \|\mathbf{x} - \tilde{\mathbf{x}}_k\|_2^2 + g(\mathbf{x}) = \text{prox}_{\alpha g}(\tilde{\mathbf{x}}_k) && \text{(proximal step).} \end{aligned} \quad (10.34)$$

The first step is a conventional GD step. The second step is called a proximal operation; see Section 7.4.

The method is convergent [292] (in the sense of Section 6.5) when the convex part $f(\mathbf{x})$ is smooth, that is, its gradient $\nabla f(\mathbf{x})$ has Lipschitz constant L , and we choose $0 < \alpha < 2/L$. We do not need $f(\mathbf{x})$ to be quadratic! Convex and smooth (Lipschitz gradient) is sufficient. The LRMC iteration (10.30) is a special case.

Section 9.4.4 described Nesterov's FGM that converges faster than GD by using momentum. Similarly, FISTA [278] and POGM [279, 293] use momentum to converge faster than ISTA.

10.4.8 Debiasing the Nuclear Norm Effects

♦♦ Using the nuclear norm regularizer for low-rank matrix completion causes all of the singular values to be “shrunk” due to the SVST step. This causes a negative bias of the singular values that are larger than the threshold. After computing an initial low-rank estimate $\hat{\mathbf{X}}$ by solving (10.17), one can apply an additional debiasing step by solving a convex problem of the form [294, Eq. (3)]

$$\begin{aligned} \hat{\mathbf{X}} &= \arg \min_{\mathbf{X}} \frac{1}{2} \| \mathbf{M} \odot (\mathbf{X} - \bar{\mathbf{Y}}) \|_{\text{F}}^2 + \beta R(\mathbf{X}), \\ R(\mathbf{X}) &= \| \mathbf{X} \|_* - \text{trace} \left\{ \tilde{\mathbf{U}}_{\hat{r}}' \mathbf{X} \tilde{\mathbf{V}}_{\hat{r}} \right\}, \end{aligned} \quad (10.35)$$

where $\tilde{\mathbf{X}} = \tilde{\mathbf{U}} \tilde{\Sigma} \tilde{\mathbf{V}}'$ is an SVD of an initial estimate $\hat{\mathbf{X}}$ and \hat{r} is an estimate of the rank of the latent matrix \mathbf{X} , perhaps obtained from the rank of $\hat{\mathbf{X}}$. This approach is a convex approximation to using a (nonconvex) regularizer of the form $R_{\text{tail}}(\mathbf{X}) = \sum_{k=r+1}^{\min(M, N)} \sigma_k(\mathbf{X})$. Both the “tail singular value” regularizer and its approximate version in (10.35) avoid shrinking the larger singular values. One can apply (10.35) recursively, using the most recent estimate of \mathbf{X} as $\hat{\mathbf{X}}$. If the iterates converge, then the limit should be akin to a regularized solution using $R_{\text{tail}}(\mathbf{X})$.

10.4.9 Factorization Approaches

◆ The nuclear-norm regularized approach to LRMC in (10.17) has the benefit of being a convex optimization problem, but has the drawback of requiring that one store the entire matrix X , which can be challenging when both M and N are large. It also requires computing an SVD every iteration, which can be expensive for large matrices. An alternative approach is to represent the low-rank latent matrix in terms of a product of factors $X = UV'$, where $U \in \mathbb{F}^{M \times K}$ and $V \in \mathbb{F}^{N \times K}$ are both tall and much smaller than X in the usual case where the selected rank $K \ll \min(M, N)$. Then one estimates the two factors by solving the optimization problem [224, 225, 226]

$$\begin{aligned}\hat{U}, \hat{V} &= \arg \min_{U \in \mathbb{F}^{M \times K}, V \in \mathbb{F}^{N \times K}} f(U, V), \\ f(U, V) &= \frac{1}{2} \|M \odot (UV' - Y)\|_F^2.\end{aligned}\quad (10.36)$$

The cost function $f(U, V)$ is called biconvex because it is convex in U when V is held fixed and vice versa, but not jointly convex in the two arguments due to the product UV' . See also (6.64). Despite being nonconvex, it can have a reasonable optimization landscape [295, 296].

A Gauss–Newton matrix recovery (GNMR) approach for (10.36) uses the following iteration [297, Eq. (2.3)]:

$$\begin{bmatrix} U_{k+1} \\ V_{k+1} \end{bmatrix} = \arg \min_{U, V} \frac{1}{2} \|M \odot (U_k V' + U V'_k - U_k V'_k - Y)\|_F^2. \quad (10.37)$$

This convex minimization problem typically does not have a unique minimizer, so the GNMR method chooses the minimizer that minimizes $\|U\|_F^2 + \|V\|_F^2$, akin to the minimum-norm LS process described in Section 5.5.2. GNMR does not require any SVD operations and has favorable theoretical properties [297].

10.4.9.1 Matrix Sensing

Matrix completion is a special case of the more general matrix sensing problem where $y = \mathcal{A}(X) + \epsilon$ and $\mathcal{A}: \mathbb{F}^{M \times N} \mapsto \mathbb{F}^d$ is a sensing operator that maps an input matrix to an output vector. If the latent image X_0 is known to be low rank and if we have access to noiseless data $b = \mathcal{A}(X_0)$, then the natural approach to recovering X_0 is

$$\hat{X} = \arg \min_{X \in \mathbb{F}^{M \times N}} \text{rank}(X) \text{ such that } \mathcal{A}(X) = b. \quad (10.38)$$

This formulation is challenging due to the nonsmooth, nonconvex rank function. However, Recht et al. show [298, Theorem 3.3] that the convex relaxation

$$\hat{X} = \arg \min_X \|X\|_*, \text{ such that } \mathcal{A}(X) = b \quad (10.39)$$

exactly recovers X_0 under a restricted isometry property (RIP) assumption about \mathcal{A} of the form

$$(1 - \delta_r) \|X\|_F \leq \|\mathcal{A}(X)\|_2 \leq (1 + \delta_r) \|X\|_F \quad \forall X \text{ such that } \text{rank}(X) \leq r \quad (10.40)$$

for sufficiently small δ_r . Furthermore, if $K \geq \text{rank}(X_0)$, then [298, Lemma 5.1] shows that (10.39) is equivalent to the lower-memory approach

$$\hat{X} = \hat{L}\hat{R}',$$

$$(\hat{L}, \hat{R}) = \arg \min_{L \in \mathbb{F}^{M \times K}, R \in \mathbb{F}^{N \times K}} \frac{1}{2} (\|L\|_F^2 + \|R\|_F^2) \text{ such that } \mathcal{A}(LR') = b. \quad (10.41)$$

Q10.9 Is the optimization problem (10.41) convex?

- A: True B: False

10.4.10 Demo

An alternative approach to the LRMC optimization problem is the ADMM method [299]. ADMM requires a tuning parameter for fast convergence. When well tuned, ADMM can be faster than FISTA. Historically, the original work on this problem used an EM algorithm perspective [300].

 **Demo 10.2** illustrates matrix completion using ADMM. See Fig. 10.2.

10.5 Robust PCA and Video Foreground/Background Separation

10.5.1 Robust PCA

The SVD and PCA methods are known to be nonrobust to outliers in the data. The robust PCA method [165, 301] is designed to reduce the influence of outliers by modeling the data Y as the sum of a low-rank component L , a sparse component S , and additive noise Z , that is, $Y = L + S + Z \in \mathbb{F}^{M \times N}$. For that model, a reasonable optimization problem for estimating the two components L and S is

$$(\hat{L}, \hat{S}) = \arg \min_{L, S \in \mathbb{F}^{M \times N}} \Psi(L, S),$$

$$\Psi(L, S) \triangleq \frac{1}{2} \|L + S - Y\|_F^2 + \alpha \|L\|_* + \beta \|\text{vec}(S)\|_1, \quad (10.42)$$

where $\alpha, \beta \geq 0$ are regularization parameters that influence the rank of L and the sparsity of S . This is a convex composite cost function in (L, S) . A convenient optimization method for it is the proximal optimized gradient method (POGM) [279].

10.5.2 Video Foreground/Background Separation

One application of (10.42) is the problem of foreground/background separation in video sequences, for example in security camera footage. In this application, the “background” (the stationary scene in front of a fixed camera) is well modeled as low rank. It might even be constant, in which case $\text{rank}(L) = 1$, or if the illumination conditions

Original frame 81 | low-rank background | sparse foreground



Figure 10.4 Video foreground/background separation via robust PCA using video data from [304].

vary slowly over the video sequence then it might have some small rank that is larger than 1. The sparse component S can represent the “foreground,” that is, objects moving in and out of the scene. There is a large literature on such methods, ranging from the simpler case of a fixed camera [302] to the more challenging case of free-motion camera videos [47, 303].

- ➊ Demo 10.3 illustrates the separation of a video into background L and foreground S components; see Fig. 10.4. This process would be difficult for a single image, but is easier for a video sequence thanks to low-rank and sparse modeling.

10.6 Nonnegative Matrix Factorization

The factorization approach in (10.36) to matrix completion (and low-rank matrix approximation) has a nonunique solution because one can replace U with $\hat{U} = UT$ and V' with $\hat{V}' = T^{-1}V'$ and the product remains the same: $\hat{U}\hat{V}' = (UT)(T^{-1}V') = UV'$. This ambiguity reduces the interpretability of the columns of U and V . In some applications, it is reasonable to model U and V as having nonnegative elements. When U and V are nonnegative, their columns can be easier to interpret, and the product $X = UV' = \sum_k u_k v'_k$ describes how to “construct” the columns of X out of “parts” [305]. The nonnegativity constraint also aids uniqueness [306]. Constraining U and V to have nonnegative elements is called nonnegative matrix factorization (NMF). See [307, 308] for surveys of algorithms.

In the case of missing data, a typical NMF formulation is very similar to (10.36):

$$\hat{U}, \hat{V} = \underset{U \in \mathbb{F}_+^{M \times K}, V \in \mathbb{F}_+^{N \times K}}{\arg \min} f(U, V), \quad f(U, V) = \frac{1}{2} \|M \odot (UV' - Y)\|_F^2, \quad (10.43)$$

where $\mathbb{F}_+^{M \times K}$ denotes the set of $M \times K$ matrices having nonnegative elements. (If there is no missing data, then just remove the $M \odot$ from the cost function.)

JULIA’s `NMF.jl` provides several algorithms for solving (10.43), and variants that include regularizers for sparsity.

- ➋ Demo 10.4 applies NMF to handwritten digit images.

10.7 Summary

Matrix completion using low-rank models is a rich area with numerous applications. Low-rank matrix *approximation* is an easier problem and Chapter 7 provided nice SVD-based solutions. LRMC is a more challenging problem (due to the missing data) so algorithms for LRMC are typically iterative. Interestingly, LRMC algorithms usually use ingredients from low-rank matrix approximation. This pattern of using methods for simpler problems as part of some iterative approach for solving more complicated problems is quite common. For image processing applications, low-rank models are often applied to collections of image patches rather than to the entire image, for example [309, 310, 311, 312, 313]. This topic is a very active research area, with a growing body of work related to convergence and performance guarantees, for example [314, 315], heteroscedastic data [192], and very sparse data [316]. For a survey, see [317].

Solutions to Explorations

Explore 10.1 C is all matrices with rank at most 1. \mathcal{D} is all matrices of the form

$$X_a = \begin{bmatrix} a & 0 \\ 0 & 2 \end{bmatrix}$$

for $a \in \mathbb{F}$. The only choice of a for which X_a has rank at most 1 is $a = 0$, so $\mathcal{D} \cap C = \{X_0\}$. With a better initialization, POCS will converge to it.

Problems

Problem 10.1 The following matrix is known to be some constant plus a rank-1 matrix, but some values are missing:

$$A = \begin{bmatrix} 20 & 14 & w & 17 \\ 44 & x & 16 & y \\ 8 & 6 & z & 7 \end{bmatrix}.$$

Determine the values of w , x , y , and z . This is a simple variation on the matrix completion problem.

Problem 10.2 Write an iterative optimization algorithm based on alternating minimization for solving the factorized low-rank matrix completion problem (10.36) given inputs Y , M , the desired rank K , an initial guess of U_0 , and the number of iterations.

11 Neural Network Models

11.1 Introduction

Previous chapters have focused on regression and classification problems using fairly “classical” machine learning methods. This chapter focuses on artificial neural network or simply neural network (NN) models and methods. Although these methods have been studied for over 50 years [20, 318, 319], they have skyrocketed in popularity in recent years due to accelerated training methods, wider availability of large training sets, and the use of deeper networks that have significantly improved performance for many classification and regression problems.

Previous chapters emphasized subspace models. We have seen that subspaces are very useful for many applications, but they cannot model all types of signals. For example, images of a single person’s face (in a given pose) under different lighting conditions lie in a subspace [173]. However, a linear combination of face images from two *different* people will not look like a plausible face. Thus, all possible face images do not lie in a subspace. A manifold model is more plausible for images of faces (and handwritten digits) and other applications, and such models require more complicated algorithms.

Entire books are devoted to NN methods [320, 321]. This chapter just introduces the simplest methods, focusing on the role of matrices and nonlinear operations. Section 11.2 illustrates the benefits of nonlinearity. Section 11.3 describes the classic perceptron model for neurons and the multilayer perceptron. Section 11.4 describes the basics of NN training. Section 11.5 reviews convolutional neural network (CNN) models; such models are used widely in DS–ML–SP applications.

11.2 The Importance of Nonlinearity

The methods discussed in previous chapters have generally involved some form of linearity. Subspace models are explicitly linear. The logistic regression classifier in Section 9.5 used the inner product $\langle \mathbf{x}, \mathbf{v} \rangle$ that is a linear operation. Yet we have also seen some examples of nonlinearities. Section 5.2.3 described the benefits of using nonlinear functions of features in regression problems. The soft and hard thresholding operations in Section 7.4 are inherently nonlinear, as is the sign operation in the logistic classifier. Nonlinear functions are ubiquitous in NN models. The following simple

examples illustrate how nonlinear operations can be helpful in binary classification problems. The intuition here generalizes to numerous other DS-ML-SP problems.

Example 11.1 Consider supervised classifier learning using training data having the single feature (with offset) $(v_1, 1)$ shown in Fig. 11.1(a). For this data, a linear decision boundary is not possible. However, in this (simple, synthetic) example, nonlinear lifting from 1D to 2D enables a basic “linear” classifier by using $v = (v_1, v_2, 1) = (v_1, |v_1|, 1)$. If the feature weights are $x = (x_1, x_2, x_3) = (0, 1, -3)$ then the decision boundary is $\{(v_1, v_2, v_3) : 0 = x'v = 1v_2 - 3 = |v_1| - 3\}$, so we choose class1 when $|v_1| - 3 > 0$.

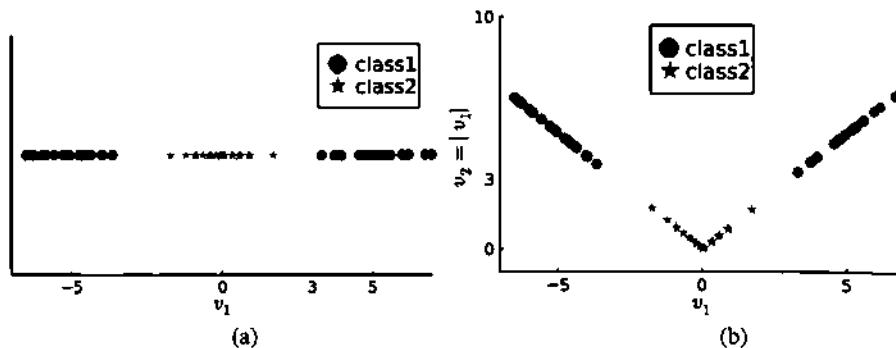


Figure 11.1 Training data for binary classification with 1D feature v_1 (a) lifted to 2D via $(v_1, v_2) = (v_1, |v_1|)$ (b). The classes have linear separability only after lifting.

Example 11.2 Figure 11.2 shows the same idea for 2D binary classification. Here it is impossible to separate the two classes with a linear decision boundary in the original parameter space, but lifting by adding one additional nonlinear “feature” enables linear separation: $v = (v_1, v_2, |v_1| + |v_2|, 1)$.

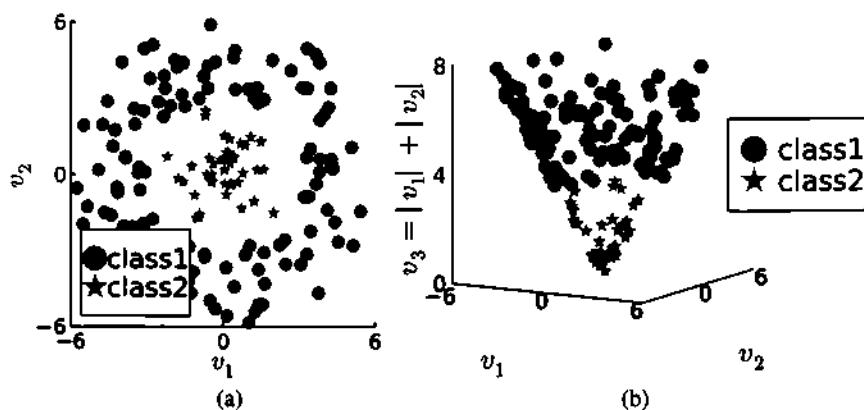


Figure 11.2 Training data for binary classification with 2D feature (v_1, v_2) (a) lifted to 3D (b) via $(v_1, v_2, v_3, 1) = (v_1, v_2, |v_1| + |v_2|, 1)$. The classes have linear separability only after lifting.

Q11.1 In this case, what shape is the decision boundary in \mathbb{R}^2 , assuming $\hat{x} = (0, 0, x_3, 4)$ with $x_3 \neq 0$?

- A: Circle B: Square C: Kite D: Half-plane E: None of these

In these examples we “hand crafted” the nonlinear lifting. In general the nonlinear combinations must be learned during training. Many NN models use the rectified linear unit (ReLU) as the nonlinearity:

$$\text{ReLU}(x) = \max(x, 0), \quad x \in \mathbb{R}. \quad (11.1)$$

The absolute value function used in these examples is related to the ReLU as follows:

$$|x| = \text{ReLU}(x) + \text{ReLU}(-x). \quad (11.2)$$

So an NN model defined with ReLUs could learn the lifting used in these examples.

11.3 Fully Connected NN Models

The section describes the basics of NN models. Actual biological brains have interconnections that provide feedback, whereas here we focus solely on feed-forward NN models.

11.3.1 Perceptron Model

The field of NN models began with efforts to emulate biological neurons with electronic circuits. The classic approach, that is still used extensively as a part of modern NN models, is the perceptron [318] that defines a function that takes a vector $x \in \mathbb{R}^N$ as input and produces a scalar output defined as some nonlinear function of an affine combination of the input vector values:

$$\underbrace{y}_{\text{output}} = \sigma \left(\underbrace{b + \sum_{j=1}^N w_j x_j}_{\text{affine}} \right) = \sigma(b + w^T x), \quad (11.3)$$

where the nonlinear function $\sigma: \mathbb{R} \mapsto \mathbb{R}$ is called the activation function. (It is unrelated to the singular values considered in earlier chapters.) The ReLU (11.1) is but one of numerous possible choices for the nonlinearity. Early NN models focused on sigmoid-shaped activation functions (hence the symbol σ), like the logistic function or the hyperbolic tangent,

$$\sigma(t) = \tanh(t/c) = \frac{e^{t/c} - e^{-t/c}}{e^{t/c} + e^{-t/c}} \quad (11.4)$$

for some $c \neq 0$.

Training such a model means learning the weight vector $w \in \mathbb{R}^N$ and the bias value $b \in \mathbb{R}$, and possibly some parameter(s) associated with the activation function σ , such as the scaling factor c in (11.4).

Biological brains have a multitude of interconnected neurons, and mathematical NN models also strive to emulate this. The first step is to generalize (11.3) to be a function with multiple output values, that is, an output vector $y \in \mathbb{R}^M$, instead of a scalar output:

$$\begin{aligned} y_i &= \sum_j \sigma(w_{ij}x_j + b_i) = \sigma(W_i \cdot x + b_i), \quad i = 1, \dots, M, \\ \implies y &= \sigma.(Wx + b), \end{aligned} \tag{11.5}$$

where JULIA's broadcast notation $f(\cdot)$ concisely expresses elementwise function application here. Clearly, matrix–vector multiplication and vector addition are central to this model.

11.3.2 Multilayer Perceptron NN Models

The multiple-input, multiple-output model (11.5) is a basic building block called a dense layer or fully connected layer. Useful NN models involve the concatenation (composition) of multiple layers, again emulating biological brains. When each layer is a perceptron of the form (11.5), the overall model is called a multilayer perceptron (MLP) model. The mathematical formula for an MLP model with L fully connected layers is

$$y = \sigma_L(W_L \cdots \sigma_3(W_3\sigma_2(W_2\sigma_1(W_1x + b_1) + b_2) + b_3) \cdots + b_L). \tag{11.6}$$

Often the matrix W_1 is tall, so that its output dimension is larger than the input dimension. This is called lifting the data into a higher dimension; Section 11.2 illustrated the potential benefits of such lifting. As L increases, the network becomes deeper, hence the term deep learning.

Training an MLP means learning all of the weight matrices W_1, \dots, W_L and bias vectors b_1, \dots, b_L for each layer, as well as any learnable parameters of the activation functions $\{\sigma_i\}$.

The innermost function $x \mapsto \sigma_1(W_1x + b_1)$ in (11.6) is called the input layer of the NN; the outermost function $x \mapsto \sigma_L(W_Lx + b_L)$ is the output layer, and the other functions are called hidden layers. Each nonlinear function σ_i is often the same scalar function, such as a ReLU (11.1), applied elementwise, except that often the output layer has a different function, for example simply the identity function.

Example 11.3 For the case $L = 3$, the MLP formula (11.6) translates into the following JULIA code:

```
</> 11.1 y = σ.(W3 * σ.(W2 * σ.(W1*x + b1) + b2) + b3)
```

11.3.3 Model Expressiveness

One might wonder if the MLP model (11.6) is sufficiently rich or expressive to perform useful operations. To address this, we first consider some properties of models of the form (11.6).

Let \mathcal{F}_N denote the class of affine functions from \mathbb{F}^N into \mathbb{F}^N . If $g \in \mathcal{F}_N$, then one can write $g(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ for some $\mathbf{W} \in \mathbb{F}^{N \times N}$ and $\mathbf{b} \in \mathbb{F}^N$. The class \mathcal{F}_N is closed under composition and addition, that is, if $g_1, g_2 \in \mathcal{F}_N$ then $g_2 \circ g_1 \in \mathcal{F}_N$ and $g_1 + g_2 \in \mathcal{F}_N$. This class corresponds to the (rarely used) case when the activation functions in (11.6) are all simply the identity function and the matrices \mathbf{W}_l are all the same size.

Now consider the (more common) case where the activation functions in (11.6) are all ReLU functions. In this case, the class of functions that is exactly representable by models of the form (11.6) is continuous piecewise linear functions [322], where the pieces are described by intersections of lines. This general function class is also closed under addition, composition, and also by maximization [323].

Representing piecewise functions exactly is a useful property, but in practice NN models are used to approximate functions that need not be piecewise linear. Fortunately, universal approximation theorems show that functions that are sufficiently regular can be well approximated by suitably designed and trained NN models [324, 325, 326]. For example, [327] shows that MLP with

a locally bounded piecewise continuous activation function can approximate any continuous function to any degree of accuracy if and only if the network's activation function is not a polynomial. [327, p. 861].

Such theory may not ensure good results in practical applications with limited training data, but provides at least some foundation for using NN models.

11.4 Training NN Models

An NN model like the MLP (11.6) is useful only if the learnable parameters (or weights) are suitably trained. This is the “learning” in machine learning. Here again, the ERM framework of Section 5.9.5 is useful. First collect all of the learnable parameters, like the weight matrices $\{\mathbf{W}_l\}$, into a vector $\boldsymbol{\theta}$, and define $y = h(\mathbf{x}; \boldsymbol{\theta})$ to be the model defined by (11.6). Given training pairs $\{(\mathbf{x}_m, y_m) : m = 1, \dots, M\}$, the empirical risk is

$$R_{\text{emp}}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{m=1}^M L(h(\mathbf{x}_m; \boldsymbol{\theta}), y_m), \quad (11.7)$$

where L is a loss function such as squared error for regression problems. Training the model is then a minimization problem of the form

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} R_{\text{emp}}(\boldsymbol{\theta}). \quad (11.8)$$

For classification problems, often the (negative) cross entropy loss is used for training. The cross entropy between two discrete probability distributions p and q is

$$H(p, q) = - \sum_{k=1}^K p_k \log q_k, \quad (11.9)$$

where K denotes the number of classes and $0 \log 0 = 0$. Here, typically p denotes a true class label, represented with one-hot encoding, where one element of p is one and the rest are zero, and q denotes a NN model output. For classification problems, often the final NN layer uses a softmax function so that the output vector represents a probability distribution (nonnegative values that sum to unity) suitable for (11.9). The softmax function here has domain \mathbb{R}^K and its codomain is the interior of the standard simplex:

$$\{(p_1, \dots, p_K) \in [0, 1]^K : \sum_k p_k = 1\}. \quad (11.10)$$

The softmax function is defined by $\sigma(v) = \exp.(v) / (\sum_k \exp(v_k))$. For binary classification, where $K = 2$, the (negative) cross entropy loss simplifies to

$$L(\hat{y}, y) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}), \quad (11.11)$$

assuming that $y \in \{0, 1\}$ and $\hat{y} \in [0, 1]$. The latter condition is ensured by using a logistic function $\sigma(x) = 1/(1 + e^{-x})$ as the final activation function.

The optimization problem (11.8) is challenging for multiple reasons. Typically, R_{emp} is a nonconvex function. The ReLU activation function (11.1) is not differentiable at zero. The MLP model (11.6) involves many layers, complicating evaluation of its gradient (or subgradient). The model (11.6) is a composition of L perceptron functions, and one must apply the chain rule to compute gradients of R_{emp} with respect to the model parameters θ . The backpropagation algorithm [328] provides a process for implementing the chain rule, also known as the reverse mode. Fortunately, modern machine learning software frameworks provide autodifferentiation tools that simplify implementation. See <https://github.com/JuliaDiff> for several such packages in JULIA. Another challenge is that the cost function (11.7) involves all the training data, which can be expensive to evaluate. The typical approach is to use SGD methods with data batches as discussed in Section 9.6 and (9.60).

11.4.1 Weight Regularization

Models with too many parameters relative to the amount of training data can overfit that data, leading to poor generalization to test data. One way to reduce overfitting in NN training is to modify the cost function (11.8) by adding a regularizer, just like we did in Section 5.6.5. The Tikhonov regularizer $\|\theta\|_2^2$ is closely related to NN training with weight decay [329]. The regularizer need not consider the bias vectors $\{b_l\}$ because those do not affect the Lipschitz constant of the MLP model (Problem 11.2).

 Demo 11.1 illustrates MLP training via JULIA autodifferentiation.

11.5 CNN Models

The MLP model (11.6) uses matrices W_l that are unstructured. Often we want some structure, notably to ensure a property like shift invariance, for example, we want a cat-vs-dog classifier to work the same regardless of where the object is located in the image [330]. Typically this desire means we want at least some layers of the NN model to be maps that are equivariant with respect to translation. Imposing such structure (e.g., requiring W_l to be Toeplitz) can also reduce memory needs. These considerations motivate convolutional neural network (CNN) models [331] that have layers involving convolution operations that act on one or more channels at once.

For imaging problems, the input to a CNN is typically either a single grayscale image $f[m, n]$, or (often) three color channels (RGB). So the input image has size $M \times N \times C_0$, where typically the number of input channels or “planes” is $C_0 = 1$ (grayscale) or $C_0 = 3$ (color). Denote this input array, or tensor, as $f_0[m, n; c]$, for $m = 1, \dots, M$, $n = 1, \dots, N$, $c = 1, \dots, C_0$.

The first layer of a typical CNN applies a set of filters to the input channels, along with some summations and a nonlinearity, producing a new set of images, typically with many more channels. Let $f_1[m, n; c]$ denote the output of the first layer, where now $c = 1, \dots, C_1$, that is, f_1 is an $M \times N \times C_1$ array (tensor), where typical values for C_1 are 64 or 128 or more. These C_1 channels no longer have direct physical meanings like RGB values, but rather are called feature maps. This process continues for $L \in \mathbb{N}$ layers, often combined with other operations such as downsampling or other versions of pooling and upsampling.

If the l th layer of a CNN is a convolutional layer, then the usual relationship between the C_l output channels and the C_{l-1} input channels is

$$f_l[m, n; c] = \sigma \left(b_{l,c} + \sum_{c'=1}^{C_{l-1}} h_l[m, n; c, c'] * f_{l-1}[m, n; c'] \right), \quad c = 1, \dots, C_l, \quad (11.12)$$

where $\sigma : \mathbb{R} \mapsto \mathbb{R}$ denotes a (usually nonlinear) activation function such as the ReLU (11.1). Each $b_{l,c}$ is called a bias. The set of filters $\{h_l[\cdot; c, c']\}$ and the biases $\{b_{l,c}\}$ are called the weights of the layer, and are learned from training data.

The “ $*$ ” notation in (11.12) represents 2D discrete-space convolution that generalizes the 1D convolution in (2.3). Here, we convolve with respect to the spatial coordinates m, n only, not along the channel dimension. In practice one must choose boundary conditions for finite-sized images, as illustrated in Fig. 2.5.

In (11.12), every input channel $f_{l-1}[\cdot; c']$ has the opportunity to influence every output channel $f_l[\cdot; c]$. Some models reduce the number of filters, essentially by constraining some of them to be zero, for example the groups option in pytorch.

Although (11.12) uses the convolution symbol, many (most?) software frameworks use 2D cross-correlation instead of convolution, to save the effort of reversing the filter. So a typical CNN is really a “correlational neural network” not a “convolutional neural network.”

Some CNN models use 1×1 convolution layers [332] that provide dimensionality reduction across feature (channel) dimensions. Those layers are essentially like a channelwise perceptron. But 3×3 filters are particularly common.

Let \mathcal{S} denote the discrete-space system (a CNN) described by cascading L layers of the form (11.12). For the following questions, ignore edge conditions (let $M = N = \infty$) and assume that $C_0 = C_L = 1$, so the CNN input is a single image $f_0[m, n; 1]$ and the CNN final output is a single image $f_L[m, n; 1]$. Assume that σ is the ReLU.

Explore 11.1 If each filter is 3×3 , then the l th layer in the model (11.12) has how many learnable weights?

Explore 11.2 Is the CNN system \mathcal{S} shift invariant or equivariant to translation?

Explore 11.3 Is the CNN system \mathcal{S} linear in general?

Q11.2 There are some choices of weights (other than zero) for which the CNN system \mathcal{S} is linear. (Assume that the final output layer does not have a ReLU.)

A: True

B: False

11.5.1 Matrix Representations

The convolution in (11.12) has a matrix representation $W_{l,c,c'} \in \mathbb{R}^{MN \times MN}$ for each c, c' pair. For zero boundary conditions that matrix is block Toeplitz with Toeplitz blocks (BTTB), with elements that depend on $h_l[m, n; c, c']$. Thus, one can relate $\text{vec}(f_l[m, n; c])$ to $\text{vec}(f_{l-1}[m, n; c])$ with an $MNC_l \times MNC_{l-1}$ matrix W_l that has the specific Toeplitz block form

$$W_l = \begin{bmatrix} W_{l,1,1} & & W_{l,1,C_l} \\ \vdots & & \vdots \\ W_{l,C_l,1} & \cdots & W_{l,C_l,C_{l-1}} \end{bmatrix}. \quad (11.13)$$

This form is never used in practical implementations; we include it here to describe the relationships between the general MLP model (11.6) and the CNN model (11.12).

For the typically small (e.g., 3×3) filters used in most CNN models, the matrices $W_{l,c,c'}$ are all banded and hence very sparse. Thus, CNN models are *not* fully connected. Despite this sparsity, deep networks often have redundancy after training, and low-rank approximations akin to those in Chapter 7 are useful for reducing NN model memory [329].

11.5.2 CNN Architectures

There is a large literature on different modifications to the basic CNN model (11.12). A few useful concepts include the following:

- skip connections and residual learning [333]

- strided convolutions that reduce memory and computation [334]
- convolutional attention mechanisms [335]
- filter constraints such as orthogonality [336]
- the U-net architecture that works across multiple image scales by downsampling and upsampling operations, with skip connections, is particularly popular in medical imaging applications [337].

See [338] for an exploration of 3×3 2D filters learned in a wide variety of applications.

See also JULIA’s `FluxML.jl` model zoo for many examples.

11.6 Summary

This chapter just scratches the surface of the vast literature on NN models. We focused on discriminative models that are trained end-to-end in a supervised fashion. There is also considerable interest in generative models [339] that are trained without supervision from unlabeled data. Those models require a probability background that is beyond the scope of this book, but they too build on the fundamentals discussed here.

Solutions to Explorations

Explore 11.1 $C_l(3^2 C_{l-1} + 1)$. The $+1$ is for the bias.

Explore 11.2 It is equivariant to translation, which is called shift invariant in the SP literature. (But not if we added any downsampling or upsampling layers.)

Explore 11.3 No, not in general, due to the activation function.

Problems

Problem 11.1 Let $f: \mathbb{F}^M \mapsto \mathbb{F}^K$ and $g: \mathbb{F}^N \mapsto \mathbb{F}^M$ be Lipschitz continuous functions with (best) Lipschitz constants \mathcal{L}_f and \mathcal{L}_g respectively.

- Define $h: \mathbb{F}^N \mapsto \mathbb{F}^K$ by $h(x) \triangleq f(g(x))$ and show that h is Lipschitz continuous with (best) Lipschitz constant $\mathcal{L}_h \leq \mathcal{L}_f \mathcal{L}_g$, or devise a simple counterexample where $\mathcal{L}_h > \mathcal{L}_f \mathcal{L}_g$.
- Prove that $\mathcal{L}_h = \mathcal{L}_f \mathcal{L}_g$ when $h = f \circ g$, where \circ denotes function composition, or devise a simple counterexample where $\mathcal{L}_h < \mathcal{L}_f \mathcal{L}_g$.

Problem 11.2 Determine a Lipschitz constant for the MLP model (11.6) in terms of the model parameters.

Hint: Use Problem 11.1. This Lipschitz constant is important as a regularizer for some NN applications [242].

12 Random Matrix Theory, Signal + Noise Matrices, and Phase Transitions

12.1 Introduction

Consider the low-rank signal-plus-noise model introduced in Section 7.5:

$$Y = \underbrace{X}_{\text{signal}} + \underbrace{Z}_{\text{noise}} \in \mathbb{F}^{M \times N}, \quad X = \sum_{k=1}^r \theta_k u_k v_k', \quad (12.1)$$

where the latent signal matrix X has rank $r \leq \min(M, N)$ with singular values $\theta_1 \geq \theta_2 \geq \dots$ and where $\{u_1, \dots, u_r\}$ and $\{v_1, \dots, v_r\}$ are each an orthonormal set of vectors. In applications like denoising and dimensionality reduction, we are interested in the properties of X , such as its SVD, but all we are given is the noisy matrix Y .

It is important to understand how the SVD components of Y relate to those of X in the presence of a random noise matrix Z . The field of random matrix theory (RMT) provides insights into those relationships, and this chapter summarizes some key results from RMT that help explain how the noise in Z perturbs the SVD components, by analyzing limits as M and N increase.

Informally, the distances between the singular values $\{\sigma_k(Y)\}$ and singular vectors $\{\hat{u}_k, \hat{v}_k\}$ of the noisy matrix Y and those of the latent matrix X are governed by the properties of the noise matrix Z . In particular, if $\|\theta_k\| \neq 0$ and if $\|Z\| \rightarrow 0$ in (12.1), then one might expect *qualitatively* that

$$\sigma_k(Y) \rightarrow \sigma_k(X) = \theta_k, \quad |\langle \hat{u}_k, u_k \rangle| \rightarrow 1, \quad |\langle \hat{v}_k, v_k \rangle| \rightarrow 1. \quad (12.2)$$

The tools from RMT discussed in this chapter provide *quantitative* relationships between the SVD of Y and that of X . The perturbations considered include roundoff error (Section 12.2), additive Gaussian noise (Section 12.3), outliers (Section 12.4), and missing data (Section 12.5).

12.1.1 Perturbation Bounds

We begin with simple inequalities that bound the effects of the noise matrix Z in (12.1) on the singular values. Specifically, by (6.71) we have [3, Corollary 8.6.2]

$$\begin{aligned} \sigma_k(Y) &= \sigma_k(X + Z) \leq \sigma_k(X) + \sigma_1(Z), \\ \sigma_k(X) &= \sigma_k(Y - Z) \leq \sigma_k(Y) + \sigma_1(Z) \\ \implies |\sigma_k(Y) - \sigma_k(X)| &\leq \sigma_1(Z) \leq \|Z\|_F. \end{aligned} \quad (12.3)$$

Thus, as $\|Z\|$ goes to zero, so does $|\sigma_k(Y) - \sigma_k(X)|$. In addition, the Wielandt–Hoffman theorem provides an aggregate bound for perturbations of all singular values [3, Theorem 8.6.4]:

$$\sum_{k=1}^{\min(M,N)} (\sigma_k(Y) - \sigma_k(X))^2 \leq \|Z\|_F^2. \quad (12.4)$$

There is an analogous (but messier) statement for perturbations of singular vectors [3, Theorem 8.6.5]. The perturbations of the singular vectors depend on both $\|Z\|$ and on the gaps between the singular values.

Because singular vectors have unit norm, that is, $\|\hat{u}\|_2 = \|u\|_2 = 1$, it follows that

$$\|\hat{u} - u\|_2^2 = 2 - 2 \operatorname{real}\{\langle \hat{u}, u \rangle\}. \quad (12.5)$$

So, a small distance $\|\hat{u} - u\|_2$ is equivalent to a large inner product $\operatorname{real}\{\langle \hat{u}, u \rangle\}$. That inner product is bounded above by 1 by the Cauchy–Schwarz inequality (2.34), so hereafter we simply analyze $|\langle \hat{u}, u \rangle|$ to assess similarity between singular vectors.

12.2 Roundoff Error

One natural use of the inequality (12.3) is analyzing the accuracy of numerical algorithms in the presence of errors induced by finite-precision computation, also known as roundoff error [340].

Example 12.1 The following code prints 1.0000001 for σ_1 , due to finite numerical precision, with $|\sigma_1(Y) - \sigma_1(X)| = |\sigma_1(Y) - 1| \approx 1.2 \cdot 10^{-7}$, where ideally $\sigma_1 = 1$.

```
</> 12.1 using LinearAlgebra: rank, svdvals
n = 100
T = Float32
u = ones(T, n) / T(sqrt(n))
v = ones(T, n) / T(sqrt(n))
Y = 1 * u * v' # theoretically rank-1 matrix
@assert rank(Y) == 1 # julia is aware of precision limits
sigma = svdvals(Y)
sigma[1], abs(sigma[1] - 1)
```

Figure 12.1 shows the singular values computed by that code. On a linear scale it is clear that $\sigma_1 \approx 1$ and the other singular values are nearly zero. When viewed on a semi-log scale, we see there are many (very small) σ_k values. In particular, $\sigma_2(Y) \approx 1.5 \cdot 10^{-8}$.

Note that *numerically* Y appears to have many nonzero singular values, even though $\operatorname{rank}(Y)$ returns 1. This is because JULIA’s `rank` function only counts singular values that are larger than a small threshold that depends on the matrix size and element precision (Problem 4.6).

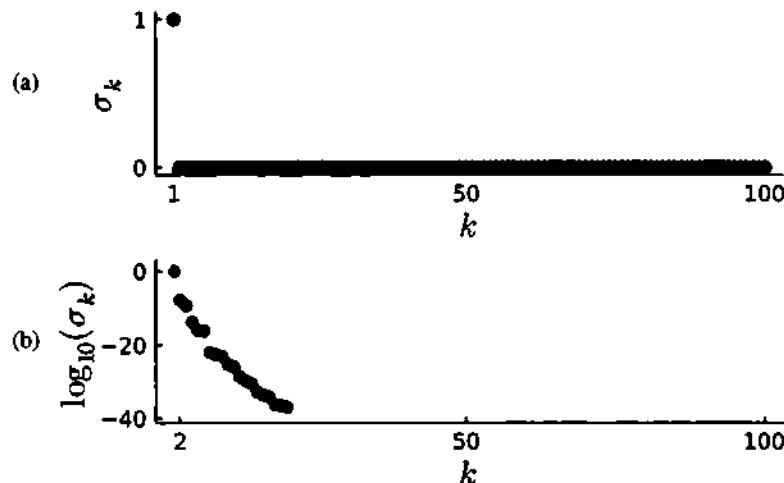


Figure 12.1 Numerically computed singular values on a linear scale (a) and a semi-log scale (b) for a matrix that ideally should have `rank = 1`.

Try `@less rank(Y)` to learn more. Specifically, the threshold is

$$\tau = \min(M, N) \varepsilon \sigma_1(Y), \quad (12.6)$$

where ε corresponds to the “machine epsilon” of the element type of the matrix. For 32-bit floating point values $\varepsilon = \text{eps}(\text{Float32}) = 2^{-23} \approx 1.2 \cdot 10^{-7}$ in JULIA, because the single precision floating-point format `Float32` uses a 24-bit significand or mantissa, 23 bits of which are stored explicitly.

A consequence of roundoff error and the choice of (12.6) is that there will be “numerical breakdown,” that is, if $\sigma_1(X) = 1$ but $\min(M, N)$ is so large that $\min(m, n) \varepsilon > 1$, then `rank(Y)` can return 0.

```
</> 12.2 using LinearAlgebra: rank
n = 1100 # > 1 / eps(Float16)
T = Float16
u = ones(T, n) / T(sqrt(n))
v = ones(T, n) / T(sqrt(n))
Y = u * v' # theoretically rank-1 matrix
rank(Y) # 0 !
```

This example used 16-bit floating point values. Such values are used in some machine learning applications to save memory. For 32-bit or better precision, we would need $N > 8.3$ million to observe the same effect. So such breakdown of the `rank` function is unlikely in typical practice, but nevertheless we see that when we represent a matrix numerically, the SVD we compute is only an approximation of the

exact SVD we would obtain if we did not have to quantize the matrix elements with finite precision.

Revisiting the basic perturbation bound in (12.3), we have

$$\left| \underbrace{\sigma_k(Y)}_{\text{numerical}} - \underbrace{\sigma_k(X)}_{\text{ideal}} \right| \leq \sigma_1(Z), \quad (12.7)$$

where here $\sigma_1(Z)$ is related to the “noise” due to finite precision (roundoff error). When $\sigma_1(Z)$ is very small, then we expect numerical results to be quite close to the ideal results. Using better precision (e.g., `Float64` rather than `Float32`) will decrease $\sigma_1(Z)$ and lead to better SVD approximations, at the price of requiring more memory.

In summary, $\sigma_1(Z)$ helps us bound the closeness of the SVD of Y with that of X . However, it is not immediately apparent how to determine $\sigma_1(Z)$ in this example, so we cannot directly apply (12.3) yet.

12.2.1 RMT for Roundoff Analysis

Motivated by the above discussion, we would like to understand how large or small $\sigma_1(Z)$ is for the roundoff error matrix. To proceed, we need to first recognize that roundoff error is signal dependent, so we must interpret (12.1) more carefully. Let $Q(x)$ denote the quantization function that maps a real number into the nearest floating-point number (for a given precision). Then we write $y_{ij} = Q(x_{ij})$ or, in JULIA notation, $Y = Q.(X)$, meaning that each element of the latent matrix X is independently quantized. With this definition, the roundoff error is $z_{ij} = y_{ij} - Q(x_{ij})$ and we interpret (12.1) as

$$Y = Q.(X) = X + (Q.(X) - X) = \underbrace{X}_{\text{signal}} + \underbrace{Z}_{\text{roundoff}} \in \mathbb{F}^{M \times N}. \quad (12.8)$$

In this expression, $Z = Q.(X) - X$ is a deterministic function of X , but if we think of X itself as a random matrix, then Z is also a random matrix.

Regardless of whether we think of Z as random, there is a useful bound on floating-point quantization error:

$$|z_{ij}| = |Q(x_{ij}) - x_{ij}| = |y_{ij} - x_{ij}| \leq \varepsilon |x_{ij}| / 2, \quad (12.9)$$

where ε corresponds to the relevant `eps`, for example `eps(Float32)` for 32-bit floating point numbers.

Hereafter, we assume that when X has random elements with zero mean and common variance ν_X , then we can model the roundoff error Z as an $M \times N$ random matrix with independent elements having mean 0 and variance $\nu_Z = c_0 \varepsilon^2 \nu_X$, where the constant c_0 depends on the distribution of the x_{ij} values integrated over the squared quantization error function $(Q(x) - x)^2$. An approximation given in [341] is $\nu_Z \approx 0.18 \cdot 2^{-2p} \nu_X = 0.18 \cdot 2^{-2} \varepsilon^2 \nu_X$, where p is the number of significand bits, implying that $c_0 \approx 0.18/4 = 0.045 \approx 1/22.22$. We found empirically that when the x_{ij} elements have a zero-mean, unit-variance uniform distribution, the constant is closer to $c_0 \approx 1/24$.

Table 12.1 Rounding error and Julia's `eps` function.

| Precision | Empirical variance | $\epsilon^2/24$ |
|-----------|----------------------|----------------------|
| Float16 | $4.0 \cdot 10^{-8}$ | $6.0 \cdot 10^{-8}$ |
| Float32 | $6.0 \cdot 10^{-16}$ | $5.9 \cdot 10^{-16}$ |
| Float64 | $2.1 \cdot 10^{-33}$ | $2.1 \cdot 10^{-33}$ |

The following code provides empirical verification of that assumption. For each precision level, the empirical variance due to roundoff error is quite similar to $\epsilon^2/24$, as seen in Table 12.1. In this code, $v_X = 1$.

```
using StatsBase: mean, var
N = 200
# very high-precision reference with var=1:
X = (2 * rand(BigFloat, N, N) .- 1) * sqrt(3)
for T in (Float16, Float32, Float64)
    Y = T.(X) # quantize
    Z = T.(Y - X) # quantization error
    @show T, mean(Z) # approximately 0
    v = var(Float64.(Z)) # sample variance
    @show (v, eps(T)^2/24) # empirical, predicted
end
```

We use this model to pin down the bound in (12.7). RMT in [342, Theorem 2] bounds the spectral norm of Z , assuming its elements are zero-mean and independent (not necessarily identically distributed) random variables:

$$\mathbb{E}[\sigma_1(Z)] \leq c_1 \left(\max_i \sqrt{\sum_j \mathbb{E}[z_{ij}^2]} + \max_j \sqrt{\sum_i \mathbb{E}[z_{ij}^2]} + \sqrt[4]{\sum_{i,j} \mathbb{E}[z_{ij}^4]} \right), \quad (12.10)$$

where c_1 is a universal constant that does not depend on M or N . If the elements of Z are zero mean and independent with common variance v_Z and kurtosis $\kappa_Z = \mathbb{E}[z_{ij}^4/v_Z^2]$, then (12.10) simplifies to

$$\begin{aligned} \mathbb{E}[\sigma_1(Z)] &\leq c_1 \sqrt{v_Z} \left(\sqrt{N} + \sqrt{M} + \kappa_Z^{1/4} (MN)^{1/4} \right) \\ &\leq c_1 (2 + \kappa_Z^{1/4}) \sqrt{v_Z \max(M, N)}. \end{aligned} \quad (12.11)$$

The distribution of floating-point quantization error is fairly complicated [343], as seen in Demo 12.1, so determining the fourth moments in (12.10) or (12.11) exactly may be challenging. Nevertheless, substituting the earlier assumption that $v_Z = c_0 \epsilon^2 v_X$ into (12.11) and simplifying yields the bound

$$\mathbb{E}[\sigma_1(\mathbf{Z})] \leq c_2 \varepsilon \sqrt{\nu_X \max(M, N)}, \quad c_2 \triangleq c_1 (2 + \kappa_Z^{1/4}) \sqrt{c_0}. \quad (12.12)$$

Now, (12.12) bounds only the *average* value of the spectral norm of \mathbf{Z} . Could individual cases be much larger than that average? If elements of the matrix \mathbf{Z} are modeled as *bounded*, independent random variables with maximum value z_{\max} , then Talagrand's concentration inequality (see [344] and [345, Theorem 2.1.13]) quantifies how likely it is that the spectral norm of \mathbf{Z} is near its expected value:

$$\mathbb{P}\{|\sigma_1(\mathbf{Z}) - \mathbb{E}[\sigma_1(\mathbf{Z})]| > \delta z_{\max}\} \leq c_3 e^{-c_4 \delta^2} \quad \forall \delta > 0, \quad (12.13)$$

where c_3 and c_4 are absolute constants. The following one-sided inequality follows immediately:

$$\mathbb{P}\{\sigma_1(\mathbf{Z}) > \mathbb{E}[\sigma_1(\mathbf{Z})] + \delta z_{\max}\} \leq c_3 e^{-c_4 \delta^2}. \quad (12.14)$$

Finally, combining with the bound in (12.12) yields

$$\mathbb{P}\left\{\sigma_1(\mathbf{Z}) > c_2 \varepsilon \sqrt{\nu_X \max(M, N)} + \delta z_{\max}\right\} \leq c_3 e^{-c_4 \delta^2}. \quad (12.15)$$

A subtle point here is that in practice rounding error is not bounded uniformly; the bound in (12.9) depends on x_{ij} . But (12.9) ensures that $z_{\max} \leq (\varepsilon/2)x_{\max}$, where $x_{\max} \triangleq \|X\|_{\max}$, leading to

$$\mathbb{P}\left\{\sigma_1(\mathbf{Z}) > c_2 \varepsilon \sqrt{\nu_X \max(M, N)} + \delta (\varepsilon/2)x_{\max}\right\} \leq c_3 e^{-c_4 \delta^2}. \quad (12.16)$$

Inspired by (12.6), consider the square case where $M = N$ and take

$$\delta = 2N\sigma_1(X)/x_{\max} \geq 2N,$$

because in that case $\delta(\varepsilon/2)x_{\max} = N\varepsilon\sigma_1(X) \approx \tau$. The right-hand side of (12.16) is bounded above by $c_3 e^{-c_4 4N^2}$ which vanishes as N increases. So we can conclude that the tolerance (12.6) in JULIA's `rank` function has the property that it is very unlikely that any noise singular values will be counted in its output.

A smaller value of δ can suffice in (12.16). Substituting the choice

$$\delta = 2\sqrt{\max(M, N)}\sigma_1(X)/x_{\max} \quad (12.17)$$

into (12.16) yields

$$\mathbb{P}\left\{\sigma_1(\mathbf{Z}) > c_2 \varepsilon \sqrt{\nu_X \max(M, N)} + \sqrt{\max(M, N)} \varepsilon \sigma_1(X)\right\} \leq c_3 e^{-4c_4 \max(M, N)}. \quad (12.18)$$

Now we have an exponentially small probability that noise singular values due to roundoff error are much larger than about $\varepsilon\sqrt{\max(M, N)}\sigma_1(X)$. Here, the tail probability again vanishes with increasing N , but at a slower rate. In practice, one must decide how conservative to be when choosing thresholds for noise singular values.

-  Demo 12.1 examines the effects of roundoff error on singular values and rank calculations, and compares the thresholds for singular values discussed above.

Explore 12.1 Verify the the inequality $\|X\|_2 \geq \|X\|_{\max}$ used in the derivation above.

12.3 Additive Noise

This section explores the SVD properties of the signal-plus-noise model $Y = X + Z$ when the matrix sizes are large, using RMT results. For simplicity we focus here on the simplest case where X is a rank-1 matrix. Specifically, suppose $X = \theta u v'$ with $\theta \neq 0$ and $\|u\|_2 \approx \|v\|_2 \approx 1$ so that $\text{rank}(X) = 1$. If the M elements of u are IID with mean 0 and variance $1/M$, then $E[\|u\|_2^2] = 1$, so, by the strong law of large numbers, when M is large then $\|u\|_2 \approx 1$. Similarly for v . Suppose also that the elements of $Z \in \mathbb{R}^{M \times N}$ are IID with $z_{ij} \sim N(0, 1/N)$. In other words, here we consider additive white Gaussian noise (AWGN) instead of rounding error.

We can consider each column of Y to be a sample of M -dimensional data:

$$y_j = \theta u v_j + z_j \in \mathbb{F}^M, \quad j = 1, \dots, N.$$

In Y we collect N such samples, or snapshots, of a rank-1 signal corrupted by noise. Our goal is to study the *estimates* of the rank-1 signal, computed using the first component of the SVD of Y :

$$Y \approx \sigma_1(Y) \hat{u} \hat{v}'.$$

Because the noise matrix Z is populated by random variables, it is a random matrix. Thus, Y is also a random matrix, $\sigma_1(Y)$ is a random variable, and \hat{u} and \hat{v} are random vectors. This section examines how well these random objects estimate the underlying signal components of X .

Because u and v have unit norm, the typical x_{ij} value is of order θ/\sqrt{MN} . Similarly, the typical z_{ij} value is of order $1/\sqrt{N}$. So the noise is larger than the signal, elementwise. The NRMSE between the noisy data Y and the latent matrix X , adapted from (7.37), is

$$\text{NRMSE} = \sqrt{\frac{E[\|Y - X\|_F^2]}{E[\|X\|_F^2]}} = \sqrt{\frac{E[\|Z\|_F^2]}{E[\|X\|_F^2]}} = \frac{\sqrt{M}}{\theta}. \quad (12.19)$$

For fixed θ , as M increases the NRMSE of the data Y worsens, which might look hopeless. But remarkably RMT shows we can estimate the signal components of X if θ is large enough, even as M grows, provided N also grows appropriately.

The bounds in (12.10) and (12.11) again apply if the elements z_{ij} are zero mean and independent. Furthermore, if the noise is bounded, then the concentration inequality (12.13) holds and we can again derive tail probability bounds like (12.15). If the noise is Gaussian, then it is not bounded, but it turns out that a Gaussian concentration inequality of the form (12.13) still holds [345, Theorem 2.1.12]. Furthermore, for IID Gaussian noise the kurtosis is $\kappa_Z = 3$, so (12.11) simplifies to

$$E[\sigma_1(Z)] \leq c_2 \sqrt{\nu_Z \max(M, N)}, \quad c_2 \triangleq c_1 (2 + 3^{1/4}). \quad (12.20)$$

The Gaussian analog of (12.13) is [345, Theorem 2.1.12]:

$$P\{|\sigma_1(Z) - E[\sigma_1(Z)]| > \delta \sqrt{\nu_Z}\} \leq c_3 e^{-c_4 \delta^2}. \quad (12.21)$$

The Gaussian version of (12.15) becomes

$$P\left\{\sigma_1(\mathbf{Z}) > c_2 \sqrt{v_Z \max(M, N)} + \delta \sqrt{v_Z}\right\} \leq c_3 e^{-c_4 \delta^2}. \quad (12.22)$$

Choosing $\delta = \sqrt{\max(M, N)}$ leads to the following Gaussian analog of (12.18):

$$P\left\{\sigma_1(\mathbf{Z}) > (c_2 + 1)\sqrt{v_Z \max(M, N)}\right\} \leq c_3 e^{-c_4 \max(M, N)}. \quad (12.23)$$

So, for both Gaussian noise and bounded noise one can derive tail probability bounds like (12.15) on the noise spectral norm. See [346, Theorem 4.4.5] for similar bounds that apply to the broad class of all sub-Gaussian distributions, that is, those whose tails decrease at least as fast as the tails of a Gaussian.

For AWGN we can say more about the behavior of the principal SVD components. The theoretical values of interest are given in [347, Eqs. (9)–(11)]. In our notation, as $M, N \rightarrow \infty$, with asymptotic ratio $M/N \rightarrow c \leq 1$, we have

$$\sigma_1(\mathbf{Y}) \xrightarrow{\text{a.s.}} \hat{\sigma}_{\text{theory}}(\theta, c) \triangleq \begin{cases} \frac{\sqrt{(1+\theta^2)(c+\theta^2)}}{\theta} & \text{if } \theta > c^{1/4}, \\ 1 + \sqrt{c} & \text{otherwise,} \end{cases} \quad (12.24)$$

$$|\hat{\mathbf{u}}' \mathbf{u}|^2 \xrightarrow{\text{a.s.}} \alpha^2(\theta, c) \triangleq \begin{cases} 1 - \frac{c(1+\theta^2)}{\theta^2(\theta^2+c)} & \text{if } \theta > c^{1/4}, \\ 0 & \text{otherwise,} \end{cases} \quad (12.25a)$$

$$|\hat{\mathbf{v}}' \mathbf{v}|^2 \xrightarrow{\text{a.s.}} \beta^2(\theta, c) \triangleq \begin{cases} 1 - \frac{\theta^2+c}{\theta^2(\theta^2+1)} & \text{if } \theta > c^{1/4}, \\ 0 & \text{otherwise.} \end{cases} \quad (12.25b)$$

Here, $\xrightarrow{\text{a.s.}}$ denotes almost sure convergence, indicating strong convergence of these random variables. These expressions convey the following key insights:

- *Deterministic limits.* The terms on the left-hand sides of (12.24) and (12.25) are random. However, asymptotically, they each converge to a *deterministic* value given by the right-hand side expressions. This is a remarkable result because as M and N increase, the number of random variables in the noise matrix \mathbf{Z} (and thus \mathbf{Y}) also increase. So one might expect that $\sigma_1(\mathbf{Y})$, $|\hat{\mathbf{u}}' \mathbf{u}|$, and $|\hat{\mathbf{v}}' \mathbf{v}|$ would become more unpredictable. However, (12.24) and (12.25) tell us that the exact opposite is true: as $M, N \rightarrow \infty$, the quantities of interest concentrate tightly around a *nonrandom* value that we can predict theoretically.
- *Biased estimates.* An important consideration for estimates is whether they are unbiased: if we get more data, do $\sigma_1(\mathbf{Y})$, $\hat{\mathbf{u}}$, $\hat{\mathbf{v}} \rightarrow \theta, \mathbf{u}, \mathbf{v}$? The expressions in (12.24) and (12.25) tell us that $\sigma_1(\mathbf{Y})$, $\hat{\mathbf{u}}$, and $\hat{\mathbf{v}}$ are *biased* estimates. In particular, we have $\hat{\sigma}_{\text{theory}}(\theta, c) > \theta$ so $\sigma_1(\mathbf{Y})$ is always positively biased (see Fig. 12.3). However, $\hat{\sigma}_{\text{theory}}(\theta, c)/\theta \rightarrow 1$ as $\theta \rightarrow \infty$, so the relative bias diminishes as SNR increases. If $\theta > c^{1/4}$, the angle between $\hat{\mathbf{u}}$ and \mathbf{u} (likewise $\hat{\mathbf{v}}$ and \mathbf{v}) is asymptotically a nonzero constant. In other words, the estimates $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ lie on cones around \mathbf{u} , \mathbf{v} (see Fig. 12.2(b)). When $\theta \leq c^{1/4}$, then $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ do not contain any information about

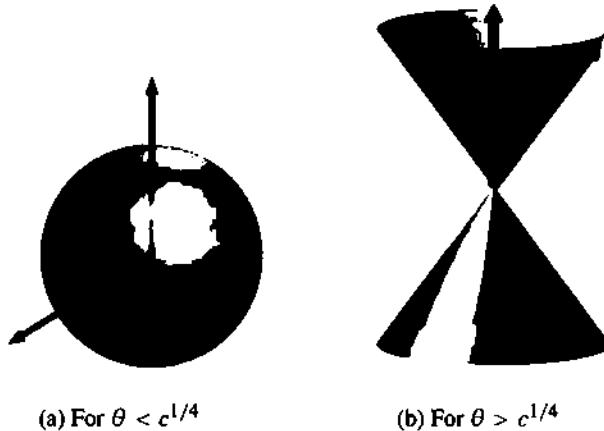


Figure 12.2 Each black arrow represents a signal vector u , and each red arrow represents a noisy estimate \hat{u} of u for a rank-1 signal + noise matrix. These figures illustrate the limits (12.25), for SNR values below (a) and above (b) the phase transition point $c^{1/4}$.

u , v , and are equivalent to selecting a vector randomly from an M - or N -dimensional sphere, respectively (see Fig. 12.2(a)).

- *Phase transition.* There is a phase transition in the SVD estimation performance around a critical SNR value given by

$$\theta_{\text{crit}} = c^{1/4}.$$

If the SNR is large enough (i.e., $\theta > \theta_{\text{crit}}$), then $\sigma_1(Y)$ converges to a value that depends on (and increases with) the true SNR. However, when the SNR drops below θ_{crit} , then $\sigma_1(Y)$ converges to $1 + \sqrt{c}$, a value independent of θ that is also the same value as when there is no signal (i.e., when $\theta = 0$). The behavior of our estimates $\sigma_1(Y)$, \hat{u} , and \hat{v} change when $\theta < \theta_{\text{crit}}$. In particular, \hat{u} and \hat{v} become uncorrelated with the latent u and v , so are useless.

To understand the practical impact of this phenomenon, consider the following example. Suppose we obtain measurements from M sensors (M -dimensional data, e.g., temperature sensors in a room, stock prices, etc.), and we take N different measurements (e.g., different times for temperatures, different days for stocks). This data collection will determine $c = M/N$. If $\theta > \theta_{\text{crit}} = c^{1/4}$, then $\sigma_1(Y)$, \hat{u} , and \hat{v} will be correlated with the underlying signal terms, θ , u , and v . We might be tempted to add more sensors, that is, to increase M to obtain more data without increasing N . However, by doing so we increase c . If we add too many sensors, then c will increase until $\theta < \theta_{\text{crit}}$, and \hat{u} and \hat{v} become essentially random: SVD-based analysis breaks down. An insight we gain here is that to compensate for increasing M we should also collect more data samples, that is, increase N .

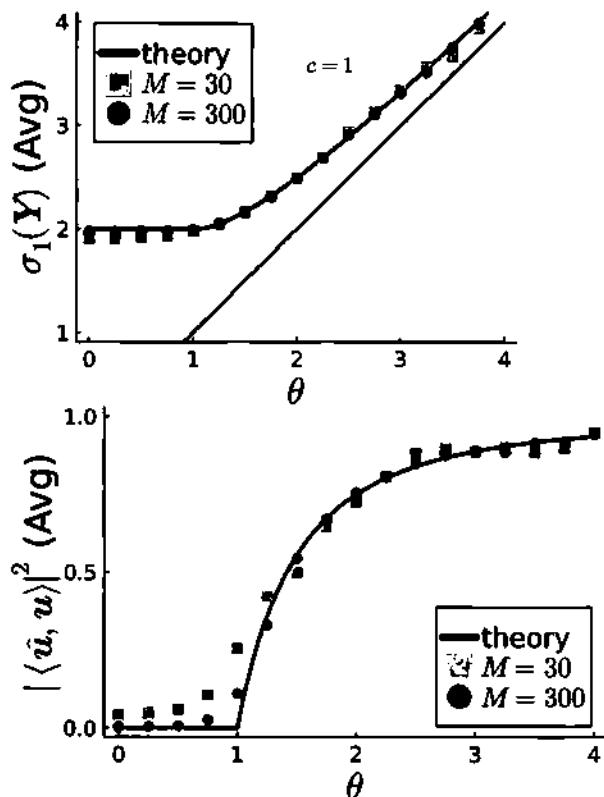


Figure 12.3 Comparisons of empirical results with RMT asymptotic predictions.

- ④ Demo 12.2 and Fig. 12.3 compare the theoretical predictions of (12.24) and (12.25) with empirical averages over 100 trials for $N \times N$ matrices with $N = 30$ and $N = 300$. As N increases, the sample averages approach the theoretical curves. The plot in the demo for $\langle \hat{v}, v \rangle$ is similar.

We see in (12.24) and in Fig. 12.3 from Demo 12.2 that when θ is small, the first singular value converges to $1 + \sqrt{c}$. RMT can explain what is special about this value. Consider the noise-only matrix Z where $z_{ij} \sim N(0, 1/N)$. The eigenvalues of the sample covariance ZZ'/N are random variables and their asymptotic behavior is described by the Marčenko–Pastur distribution [348]. The singular values of Z are also random variables, and one can use the relationship between singular values and eigenvalues to derive their asymptotic behavior. Specifically, as M, N increase with asymptotic ratio $c = M/N < 1$, the spectral measure of any randomly selected singular value of Z converges to the marginal distribution with density [347, §3.1]

$$p(x) = \frac{\sqrt{4c - (x^2 - 1 - c)^2}}{\pi cx} \mathbb{I}_{\{\sigma_+ \leq x \leq \sigma_+\}}, \quad (12.26)$$

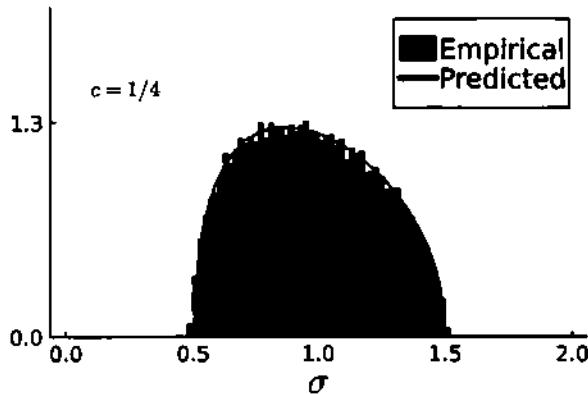


Figure 12.4 Marčenko-Pastur prediction of noise singular value distribution when $c = M/N = 100/400 = 1/4$, so $\sigma_{\pm} = (1/2, 3/2)$. The histogram aggregates data from 150 trials.

where $\sigma_{\pm} \triangleq 1 \pm \sqrt{c}$. Because $\sigma_+ = \hat{\sigma}_{\text{theory}}(0, c)$, we see from (12.24) that $\sigma_1(\mathbf{Z})$ converges to the right endpoint σ_+ of this distribution.

Demo 12.2 also compares the empirical distribution of the singular values of a noise-only matrix with the prediction given by (12.26); see Fig. 12.4.

An intriguing aspect of RMT is that the results extend well beyond the Gaussian distribution, a property known as universality [349, 350, 351, 352]. If we have an IID $M \times N$ matrix with elements z_{ij} having mean 0, variance $1/N$, and *bounded fourth moment*, then the asymptotic distribution (12.26) still applies, provided the distribution is not too sparse [316, 353]. Demo 12.2 illustrates a case where the z_{ij} values come from a $\pm 1/\sqrt{N}$ Bernoulli distribution, and the resulting plot looks just like Fig. 12.4. That demo also illustrates a “sparse” case where most of the z_{ij} values are zero, and (12.26) no longer applies.

We can also relate (12.26) to the analysis of rounding error in (12.11). The upper bound in (12.26) is

$$\sigma_+ = 1 + \sqrt{c} = 1 + \sqrt{M/N} = \sqrt{\nu_Z} (\sqrt{N} + \sqrt{M}) \leq 2\sqrt{\nu_Z \max(M, N)},$$

which is on the same order as the upper bound in (12.11). The upper limit σ_+ is an asymptotic result, whereas the upper bound (12.11) holds for any M, N , but the similarity is reassuring.

12.4 Outliers

A further extension of the signal + noise model $\mathbf{Y} = \mathbf{X} + \mathbf{Z}$ is to the case where the data is corrupted by outliers. Here we model this situation as

$$\mathbf{Y} = \underbrace{\mathbf{X}}_{\text{signal}} + \underbrace{\mathbf{Z}}_{\text{noise}} + \underbrace{\mathbf{S}}_{\text{outlier}} \in \mathbb{F}^{M \times N}, \quad (12.27)$$

where $X = \theta\mathbf{u}\mathbf{v}'$ again has elements with values of order θ/\sqrt{MN} and, as before, z_{ij} is zero mean with variance $1/N$. What is different here is that we consider a sparse outlier matrix S with elements modeled as follows:

$$s_{ij} = \begin{cases} q_{ij} & \text{with probability } p_N, \\ 0 & \text{with probability } 1 - p_N, \end{cases}$$

where q_{ij} is drawn IID from an unknown distribution with zero mean, variance v_Q , and finite fourth moment. Here we write p_N because the outlier probability might depend on the number of samples N . We consider the regime where v_Q is of order unity, so the outliers are usually quite distinct from the typical values of x_{ij} and z_{ij} .

With this model, $E[s_{ij}] = 0$, $\text{Var}\{s_{ij}\} = p_N v_Q$, and s_{ij} has a finite fourth moment. However, because the outliers are much larger than the typical $1/\sqrt{N}$ values of the z_{ij} elements, bounds of the form (12.10) on $\sigma_1(\mathbf{Z} + S)$ seem too loose to be helpful.

Nevertheless, assuming the universality conditions hold for S , that is, it is “not too sparse” [316], then asymptotic limits of the form in (12.24) and (12.25) still apply [354], provided we replace θ in the right-hand side of those expressions with an “effective SNR” $\bar{\theta}$ defined as follows:

$$\bar{\theta} \triangleq \lim_{N \rightarrow \infty} \frac{\theta}{\sqrt{p_N N v_Q + 1}}. \quad (12.28)$$

This expression provides a somewhat sobering insight. If v_Q is nonzero and p_N is a constant, even a seemingly small one like $1/1000$, then as $N \rightarrow \infty$ (with M also increasing and $M/N \rightarrow c$), the effective SNR $\bar{\theta}$ will diminish to zero and SVD/PCA will fail, meaning that $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ become uselessly uncorrelated with the latent components \mathbf{u} and \mathbf{v} . This property illustrates the lack of robustness of ordinary SVD/PCA to outliers.

Even though ordinary SVD/PCA can fail here, there are alternative methods that can handle outliers. One option is to use a robust PCA method as described in Section 10.5.1. Another option is to observe that in the regime considered here, the sparse outlier part has entries of order $\sqrt{v_Q}$, whereas the noise part has entries of order $1/\sqrt{N}$ (and the signal values x_{ij} are even smaller). Thus, a simple threshold can detect the outliers with high probability. After detecting the outliers, one can set the corresponding matrix elements to zero, leading to a problem akin to matrix completion discussed next.

 Demo 12.3 illustrates the effects of data outliers on low-rank approximation using the SVD.

12.5 Matrix Completion

Chapter 10 discussed methods for matrix completion (with and without noise) assuming the latent matrix X is low rank, for example $X = \theta\mathbf{u}\mathbf{v}'$. We can also use RMT tools to predict the performance of SVD-based methods for this application. Here we

consider the case where we set missing matrix values to zero and then compute the SVD of the data \mathbf{Y} . Letting \mathbf{M} denote the mask, we rewrite the model in (10.14) for the noisy case as

$$\begin{aligned}\mathbf{Y} &= \mathbf{M} \odot (\mathbf{X} + \mathbf{Z}_1) = \mathbf{p} \mathbf{X} + \mathbf{Z}, \quad \mathbf{Z} \triangleq \mathbf{Z}_2 + \tilde{\mathbf{Z}}_1 \\ \mathbf{Z}_2 &\triangleq \mathbf{M} \odot \mathbf{X} - \mathbf{p} \mathbf{X}, \quad \tilde{\mathbf{Z}}_1 \triangleq \mathbf{M} \odot \mathbf{Z}_1.\end{aligned}\quad (12.29)$$

For RMT analysis, we assume the mask \mathbf{M} is a random IID Boolean array where

$$m_{ij} = \begin{cases} 1 & \text{with probability } p, \\ 0 & \text{with probability } 1 - p. \end{cases}$$

The IID assumption for \mathbf{M} ensures that the sampling is “incoherent.” Matrix completion can fail if the pattern of missing elements has a certain structure. For example, if two entire rows of \mathbf{Y} were missing, it would be impossible to estimate the corresponding elements of \mathbf{u} even in the absence of noise.

The error matrix components $\tilde{\mathbf{Z}}_1$ and \mathbf{Z}_2 defined in (12.29) are defined such that they each have zero-mean elements. The AWGN matrix \mathbf{Z}_1 has zero-mean elements with variance $1/N$, so its masked version $\tilde{\mathbf{Z}}_1$ has IID zero-mean elements with variance p/N . The matrix \mathbf{Z}_2 describes the error due to missing values and it has elements that have zero mean (by construction) and that are either $(1-p)x_{ij}$ or $-px_{ij}$, where x_{ij} has values of order θ/\sqrt{MN} . These values are usually much smaller than the typical values of the AWGN matrix \mathbf{Z}_1 that has values of order $1/\sqrt{N}$.

If we think of the latent matrix \mathbf{X} as being a fixed nonrandom matrix, then the elements of \mathbf{Z}_2 are zero mean and statistically independent due to the IID mask, but they are not identically distributed because the variance (and fourth moment) of each element of \mathbf{Z}_2 depend on the x_{ij} values. Fortunately, the key bound (12.10) requires only zero mean and independence, not identical distributions. So one can follow similar steps to derive a bound on $\sigma_1(\mathbf{Z})$ that is similar to (12.23) [167, Eq. (37)]. That bound indicates that the signal components can stand out from the noise if θ is sufficiently large.

Furthermore, RMT describes how to extend (12.24) and (12.25) to the case of missing data set to zero. Of course, the SVD performance will degrade when some data is missing, and that degradation is characterized by an “effective SNR” defined by

$$\bar{\theta}_p = \sqrt{p} \theta. \quad (12.30)$$

With this definition, and assuming that \mathbf{u} and \mathbf{v} satisfy a “low-coherence” condition [167, Eq. (12)], the asymptotic limits for the first singular components (see [167, p. 3006, §VI] and [348]) are

$$\begin{aligned}\sigma_1(\mathbf{Y}) &\rightarrow \sqrt{p} \hat{\sigma}_{\text{theory}}(\bar{\theta}_p, c), \\ |\hat{\mathbf{u}}' \mathbf{u}|^2 &\rightarrow \alpha^2(\bar{\theta}_p, c), \\ |\hat{\mathbf{v}}' \mathbf{v}|^2 &\rightarrow \beta^2(\bar{\theta}_p, c),\end{aligned}\quad (12.31)$$

where the functions on the right-hand side were defined in (12.24) and (12.25), but now they are evaluated at the (lower) effective SNR $\bar{\theta}_p$. Applying (12.24), for large θ we have $\sigma_1(\mathbf{Y}) \approx p \theta$, as verified empirically in Demo 12.4.

Thus, we again have a phase transition where the SVD estimates \hat{u} and \hat{v} break down if $\bar{\theta}_p < c^{1/4}$, that is, if $\theta < c^{1/4}/\sqrt{p}$. As one might expect, when p decreases (more missing data), we need a larger signal amplitude θ to avoid breakdown.

-  Demo 12.4 compares the theoretical predictions given in (12.31) with empirical SVD performance, leading to results quite similar to Fig. 12.3 by replacing θ with $\bar{\theta}_p$. It also provides an image example.

For a more advanced method that can handle very sparse matrix completion, see [316].

12.6 Summary

This chapter has just scratched the surface of showing that RMT can provide quantitative predictions of how well SVD components from a noisy matrix Y relate to the corresponding components of a latent matrix X . We illustrated the effects of roundoff error, additive noise, outliers, and missing data.

How one might use such results will be application dependent. For example, the goal in signal denoising differs from the goals in dimensionality reduction. PCA is designed to learn components that “capture as much variance as possible,” which is sensible when referring to *signal* components. But the data Y has both signal and *noise* components in most practical applications, and generally one would like to reduce noise effects. Methods like OptShrink, discussed in Section 7.6.2, are designed using RMT principles to reduce noise effects.

It has often been said [355] that “there is nothing as practical as good theory.” Random matrix theory certainly fits this aphorism. For example, RMT has been used to design denoising methods for magnetic resonance images [356, 357]. This entire book is replete with equations (theory), essentially all of which have many DS–ML–SP applications.

Solutions to Explorations

Explore 12.1 Let i, j denote the index of the (or a) maximum absolute entry of X . Then $\|X\|_2 \geq \|Xe_j\|_2 = \|A_{:,j}\|_2 \geq |x_{ij}| = \|X\|_{\max}$.

Problems

Problem 12.1 Recently, several technology companies proposed an 8-bit floating point format for machine learning applications [358]. When $X = xx'$ and $x = \mathbf{1}_N/\sqrt{N}$ is stored in FP8 format, what value N would cause $\text{rank}(X)$ to return zero?

Problem 12.2 Low-rank signal-plus-noise-type matrices have many applications. For a rank-1 signal matrix, we model the signal-plus-noise matrix as $Y = \theta u v' + Z$, where $\theta \in \mathbb{R}_+$ is the signal-to-noise ratio; $u \in \mathbb{R}^M$ and $v \in \mathbb{R}^N$ are (arbitrary) unit-norm “signal” vectors, so that $X = \theta u v'$ is the “signal” matrix and Z is a noise-only matrix whose entries are IID Gaussian random variables with mean 0; and variance $1/N$. We can generate this numerically as follows.

```
</> 12.4 using LinearAlgebra: normalize
M,N = 100, 200
θ = 2.0
u = normalize(randn(M))
v = normalize(randn(N))
Z = randn(M,N) / sqrt(N)
Y = θ * u * v' + Z
```

Our goal is to ascertain numerically how the largest singular value of Y and its associated left and right singular vectors are related to their counterparts in the rank-1 signal matrix X . To that end, for $M = 100$ and $N = 200$, compute and plot, as three separate subplots, as a function of $\theta \in \text{range}(0.1, 5, 20)$, the empirical average, computed over 500 trials, of:

- (a) $\theta_1(Y)$, the largest singular value of Y ;
 - (b) $|\langle u_1(Y), u \rangle|^2$, where $u_1(Y)$ is the left singular vector of Y associated with $\theta_1(Y)$; and
 - (c) $|\langle v_1(Y), v \rangle|^2$, where $v_1(Y)$ is the right singular vector of Y associated with $\theta_1(Y)$.
- Superimpose on your empirically generated plots the asymptotic predictions from random matrix theory in (12.24) and (12.25) for each associated quantity. Do the theoretical plots match the experimentally observed values? Write down your observations and conclusions.

Repeat the experiment for $M = 400$ and $N = 400$ and submit the plots generated.

Problem 12.3 Section 12.3 mentions that the Marčenko–Pastur asymptotic theory works beyond Gaussian distributions. Modify the code in Demo 12.2 to compare empirical and predicted singular value distributions when the noise matrix Z has IID elements with a common (suitably wide) uniform distribution.

- [1] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, “Julia: A fresh approach to numerical computing,” *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.
- [2] E. Darve and M. Wootters, *Numerical linear algebra with Julia*. SIAM, 2021.
- [3] G. H. Golub and C. F. Van Loan, *Matrix computations*, 3rd ed. Johns Hopkins University Press, 1996.
- [4] A. J. Laub, *Matrix analysis for scientists and engineers*. SIAM, 2005.
- [5] L. Eldén, *Matrix methods in data mining and pattern recognition*. SIAM, 2007, errata: <http://users.mai.liu.se/larel04/matrix-methods/index.html>.
- [6] K. B. Petersen and M. S. Pedersen, *The matrix cookbook*. 2012. [Online]. Available: <http://matrixcookbook.com>
- [7] S. Boyd and L. Vandenberghe, *Introduction to applied linear algebra: Vectors, matrices, and least squares*. Cambridge University Press, 2017.
- [8] T. A. Driscoll and R. J. Braun, *Fundamentals of numerical computation: Julia edition*. SIAM, 2022. [Online]. Available: <https://fncbook.github.io/v1.0>
- [9] C. Heitzinger, *Algorithms with JULIA: Optimization, machine learning, and differential equations using the JULIA language*. Springer, 2022.
- [10] K. Lange, *Algorithms from THE BOOK*. SIAM, 2020.
- [11] S. H. Chan, “Introduction to probability for data science,” 2021. [Online]. Available: <https://probability4datascience.com>
- [12] C. T. Kelley, *Solving nonlinear equations with iterative methods: Solvers and examples in Julia*. SIAM, 2022.
- [13] J. Llacer and J. Nunez, “Iterative maximum likelihood and Bayesian algorithms for image reconstruction in astronomy,” in *Restoration of Hubble Space Telescope Images*, R. L. White and R. J. Allen, Eds. Space Telescope Science Institute, 1990, pp. 62–9.
- [14] L. Deng, “The MNIST database of handwritten digit images for machine learning research,” *IEEE Sig. Proc. Mag.*, vol. 29, no. 6, pp. 141–2, Nov. 2012.
- [15] B. Noble and J. W. Daniel, *Applied linear algebra*, 2nd ed. Prentice-Hall, 1977.
- [16] T. Kailath, *Linear systems*. Prentice-Hall, 1980.
- [17] K. Bryan and T. Leise, “The \$25,000,000,000 eigenvector: The linear algebra behind Google,” *SIAM Review*, vol. 48, no. 3, pp. 569–81, Sep. 2006.
- [18] A. N. Elmachtoub and C. F. Van Loan, “From random polygon to ellipse: An eigenanalysis,” *SIAM Review*, vol. 52, no. 1, pp. 151–70, 2010.
- [19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [20] F. Rosenblatt, *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*. Spartan, 1962. [Online]. Available: <http://www.dtic.mil/docs/citations/AD0256582>

- [21] C. Champlin, D. Bell, and C. Schocken, "AI medicine comes to Africa's rural clinics," *IEEE Spectrum*, vol. 54, no. 5, pp. 42–8, May 2017.
- [22] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, no. 2, pp. 4–22, Apr. 1987.
- [23] M. Anderson and T. S. Perry, "Large language and multimodal models don't come cheap," *IEEE Spectrum*, vol. 60, no. 7, p. 13, Jul. 2023.
- [24] J. Alman and V. V. Williams, "A refined laser method and faster matrix multiplication," in *Proc. ACM-SIAM Symp. on Discrete Alg. (SODA)*, 2021, pp. 522–39.
- [25] C. Choi, "DeepMind plays math and wins: A game-playing algorithm accelerates matrix multiplication," *IEEE Spectrum*, vol. 60, no. 1, pp. 14–15, Jan. 2023.
- [26] T. Davis, "Block matrix methods: Taking advantage of high-performance computers," 1998, university of Florida TR-98-204.
- [27] W. E. Roth, "On direct product matrices," *Bull. Amer. Math. Soc.*, vol. 40, no. 6, pp. 461–8, 1934. [Online]. Available: <https://projecteuclid.org:443/euclid.bams/1183497463>
- [28] A. Airola and T. Pahikkala, "Fast Kronecker product kernel methods via generalized vec trick," *IEEE Trans. Neural Net. Learn. Sys.*, vol. 29, no. 8, pp. 3374–87, Aug. 2018.
- [29] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix," *Ann. Math. Statist.*, vol. 20, no. 4, p. 621, Dec. 1949.
- [30] H. V. Henderson and S. R. Searle, "On deriving the inverse of a sum of matrices," *SIAM Review*, vol. 23, no. 1, pp. 53–60, Jan. 1981.
- [31] W. W. Hager, "Updating the inverse of a matrix," *SIAM Review*, vol. 31, no. 2, pp. 221–39, 1989.
- [32] W. Kahan, "Computing cross-products and rotations in 2- and 3-dimensional Euclidean spaces," 2016. [Online]. Available: <https://people.eecs.berkeley.edu/~wkahan/MathH110/Cross.pdf>
- [33] J. Kovacevic and A. Chebira, "Life beyond bases: The advent of frames (Part I)," *IEEE Sig. Proc. Mag.*, vol. 24, no. 4, pp. 86–104, Jul. 2007.
- [34] ———, "Life beyond bases: The advent of frames (Part II)," *IEEE Sig. Proc. Mag.*, vol. 24, no. 5, pp. 115–25, Sep. 2007.
- [35] A. G. Akritas, E. K. Akritas, and G. I. Malaschonok, "Various proofs of Sylvester's (determinant) identity," *Math. Comput. Simul.*, vol. 42, no. 4, pp. 585–93, Nov. 1996.
- [36] J. J. Sylvester, "On the equation to the secular inequalities in the planetary theory," *London, Edinburgh, and Dublin Phil. Mag. and J. of Sci.*, vol. 16, no. 100, pp. 267–9, 1883.
- [37] G. W. Stewart, "The decompositional approach to matrix computation," *Comp. Sci. Engin.*, vol. 2, no. 1, pp. 50–9, 2000.
- [38] P. B. Denton, S. J. Parke, T. Tao, and X. Zhang, "Eigenvectors from eigenvalues: A survey of a basic identity in linear algebra," *Bull. Amer. Math. Soc.*, vol. 59, pp. 31–58, 2022.
- [39] G. W. Stewart, "On the early history of the singular value decomposition," *SIAM Review*, vol. 35, no. 4, pp. 551–66, 1993.
- [40] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, and I. Yamazaki, "The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale," *SIAM Review*, vol. 60, no. 4, pp. 808–65, 2018.
- [41] A. P. Mosk, A. Lagendijk, G. Lerosey, and M. Fink, "Controlling waves in space and time for imaging and focusing in complex media," *Nature Photon.*, vol. 6, no. 5, pp. 283–92, 2012.

- [42] C. Jin, R. R. Nadakuditi, E. Michielssen, and S. C. Rand, "Iterative, backscatter-analysis algorithms for increasing transmission and focusing light through highly scattering random media," *J. Opt. Soc. Am. A*, vol. 30, no. 8, pp. 1592–602, Aug. 2013.
- [43] Y. Luo, Y. Zhao, J. Li, E. Cetintas, Y. Rivenson, M. Jarrahi, and A. Ozcan, "Computational imaging without a computer: Seeing through random diffusers at the speed of light," *eLight*, vol. 2, no. 1, p. 4, 2022.
- [44] M. J. Gander and G. Wanner, "From Euler, Ritz, and Galerkin to modern computing," *SIAM Review*, vol. 54, no. 4, pp. 627–66, 2012.
- [45] K. Fan, "On a theorem of Weyl concerning eigenvalues of linear transformations I," *Proc. Nat. Acad. Sci.*, vol. 35, no. 11, pp. 652–5, Nov. 1949.
- [46] B. Ghogogh, F. Karay, and M. Crowley, "Eigenvalue and generalized eigenvalue problems: Tutorial," 2019. [Online]. Available: <http://arxiv.org/abs/1903.11240>
- [47] B. E. Moore, C. Gao, and R. R. Nadakuditi, "Panoramic robust PCA for foreground–background separation on noisy, free-motion camera video," *IEEE Trans. Computational Imaging*, vol. 5, no. 2, pp. 195–211, Jun. 2019.
- [48] H. Weyl, "Inequalities between the two kinds of eigenvalues of a linear transformation," *Proc. Nat. Acad. Sci.*, vol. 35, no. 7, pp. 408–11, Jul. 1949.
- [49] N. Muller, L. Magaia, and B. M. Herbst, "Singular value decomposition, eigenfaces, and 3D reconstructions," *SIAM Review*, vol. 46, no. 3, pp. 518–45, 2004.
- [50] G. T. Gilbert, "Positive definite matrices and Sylvester's criterion," *Amer. Math. Monthly*, vol. 98, no. 1, pp. 44–6, 1991.
- [51] M.-D. Choi, "Tricks or treats with the Hilbert matrix," *Amer. Math. Monthly*, vol. 90, no. 5, pp. 301–12, 1983.
- [52] I. Lankham, B. Nachtergaele, and A. Schilling, *Linear algebra as an introduction to abstract mathematics*. World Scientific, 2016. [Online]. Available: https://www.math.ucdavis.edu/~anne/linear_algebra
- [53] G. Strang and C. Moler, "LU and CR Elimination," *SIAM Review*, vol. 64, no. 1, pp. 181–90, 2022.
- [54] G. W. Stewart, "A second order perturbation expansion for small singular values," *Linear Algebra Appl.*, vol. 56, pp. 231–5, Jan. 1984.
- [55] M. Stewart, "Perturbation of the SVD in the presence of small singular values," *Linear Algebra Appl.*, vol. 419, no. 1, pp. 53–77, Nov. 2006.
- [56] M. Rudelson and R. Vershynin, "Non-asymptotic theory of random matrices: Extreme singular values," in *Proc. Int. Congress of Math. (ICM 2010)*, 2011, pp. 1576–602.
- [57] G. Marsaglia, "Bounds for the rank of the sum of two matrices," Boeing Scientific Research Labs, Tech. Rep. 344, D1-82-0343, 1964. [Online]. Available: <https://apps.dtic.mil/sti/citations/AD0600471>
- [58] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization," *Proc. Nat. Acad. Sci.*, vol. 100, no. 5, pp. 2197–2202, Mar. 2003.
- [59] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. 34, no. 3, pp. 276–80, Mar. 1986.
- [60] M. Kaveh and A. Barabell, "The statistical performance of the MUSIC and the minimum-norm algorithms in resolving plane waves in noise," *IEEE Trans. Acoust. Sp. Sig. Proc.*, vol. 34, no. 2, pp. 331–41, Apr. 1986.

- [61] X.-L. Xu and K. M. Buckley, "Bias and variance of direction-of-arrival estimates from MUSIC, MIN-NORM, and FINE," *IEEE Trans. Sig. Proc.*, vol. 42, no. 7, pp. 1812–5, Jul. 1994.
- [62] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig, "ESPIRiT—an eigenvalue approach to autocalibrating parallel MRI: Where SENSE meets GRAPPA," *Mag. Res. Med.*, vol. 71, no. 3, pp. 990–1001, Mar. 2014.
- [63] K. H. Jin, D. Lee, and J. C. Ye, "A general framework for compressed sensing and parallel MRI using annihilating filter based low-rank Hankel matrix," *IEEE Trans. Comput. Imaging*, vol. 2, no. 4, pp. 480–95, Dec. 2016.
- [64] J. P. Haldar and K. Setsompop, "Linear predictability in MRI reconstruction: Leveraging shift-invariant Fourier structure for faster and better imaging," *IEEE Sig. Proc. Mag.*, vol. 37, no. 1, pp. 69–82, Jan. 2020.
- [65] R. A. Lobos, C.-C. Chan, and J. P. Haldar, "New theory and faster computations for subspace-based sensitivity map estimation in multichannel MRI," *IEEE Trans. Med. Imag.*, 2023.
- [66] G. Strang, "Bringing the SVD to life," *SIAM News*, vol. 39, no. 1, Jan. 2006. [Online]. Available: <https://archive.siam.org/news/news.php?id=828>
- [67] P. L. Combettes and H. J. Trussell, "Method of successive projections for finding a common point of sets in metric spaces," *J. Optim. Theory Appl.*, vol. 67, no. 3, pp. 487–507, Dec. 1990.
- [68] F. Galton, "Regression towards mediocrity in hereditary stature," *J. Anthropol. Inst. of Great Britain and Ireland*, vol. 15, pp. 246–63, 1886. [Online]. Available: <http://www.jstor.org/stable/2841583>
- [69] S. M. Stigler, "Gauss and the invention of least squares," *Ann. Statist.*, vol. 9, no. 3, pp. 465–74, May 1981.
- [70] T. P. Minka, "Old and new matrix algebra useful for statistics," 2000. [Online]. Available: <https://tminka.github.io/papers/matrix>
- [71] T. Adali, P. J. Schreier, and L. L. Scharf, "Complex-valued signal processing: The proper way to deal with impropriety," *IEEE Trans. Sig. Proc.*, vol. 59, no. 11, pp. 5101–25, Nov. 2011.
- [72] L. Sorber, M. V. Barel, and L. D. Lathauwer, "Unconstrained optimization of real functions in complex variables," *SIAM J. Optim.*, vol. 22, no. 3, pp. 879–98, 2012.
- [73] J. Liu, C. Garcia-Cardona, B. Wohlberg, and W. Yin, "First- and second-order methods for online convolutional dictionary learning," *SIAM J. Imaging Sci.*, vol. 11, no. 2, pp. 1589–628, Jan. 2018.
- [74] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK users' guide*, 3rd ed. SIAM, 1999.
- [75] Many, "The Moore–Penrose inverse," 2022, wikibook on Topics in Abstract Algebra / Linear Algebra. [Online]. Available: https://en.wikibooks.org/wiki/Topics_in_Abstract_Algebra/Linear_algebra
- [76] R. Penrose, "A generalized inverse for matrices," *Math. Proc. Camb. Phil. Soc.*, vol. 51, no. 3, pp. 406–13, Jul. 1955.
- [77] C. Börgers, *Introduction to numerical linear algebra*. SIAM, 2022.
- [78] Y. Long, J. A. Fessler, and J. M. Balter, "3D forward and back-projection for X-ray CT using separable footprints," *IEEE Trans. Med. Imag.*, vol. 29, no. 11, pp. 1839–50, Nov. 2010.

- [79] A. Sinha, P. Malo, and K. Deb, “A review on bilevel optimization: From classical to evolutionary approaches and applications,” *IEEE Trans. Evol. Comp.*, vol. 22, no. 2, pp. 276–95, 2018.
- [80] C. Crockett and J. A. Fessler, “Bilevel methods for image reconstruction,” *Found. Trends Sig. Proc.*, vol. 15, no. 2-3, pp. 121–289, 2022.
- [81] A. Gilyen, S. Lloyd, and E. Tang, “Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension,” 2018. [Online]. Available: <http://arxiv.org/abs/1811.04909>
- [82] L. Mor-Yossef and H. Avron, “Sketching for principal component regression,” *SIAM J. Matrix. Anal. Appl.*, vol. 40, no. 2, pp. 454–85, Jan. 2019.
- [83] P.-Å. Wedin, “Perturbation theory for pseudo-inverses,” *BIT Numerical Math.*, vol. 13, no. 2, pp. 217–32, Jun. 1973.
- [84] G. W. Stewart, “On the perturbation of pseudo-inverses, projections and linear least squares problems,” *SIAM Review*, vol. 19, no. 4, pp. 634–62, 1977.
- [85] ———, “Stochastic perturbation theory,” *SIAM Review*, vol. 32, no. 4, pp. 579–610, 1990.
- [86] L. Meng and B. Zheng, “The optimal perturbation bounds of the Moore–Penrose inverse under the Frobenius norm,” *Linear Algebra Appl.*, vol. 432, no. 4, pp. 956–63, Feb. 2010.
- [87] ———, “New multiplicative perturbation bounds of the Moore–Penrose inverse,” *Linear Multilinear Algebra*, vol. 63, no. 5, pp. 1037–48, 2015.
- [88] J. P. Haldar, “On ambiguity in linear inverse problems: Entrywise bounds on nearly data-consistent solutions and entrywise condition numbers,” *IEEE Trans. Sig. Proc.*, vol. 71, pp. 1083–92, 2023.
- [89] L. Ying, B. Liu, M. Steckner, G. Wu, M. Wu, and S.-J. Li, “A statistical approach to SENSE regularization with arbitrary k -space trajectories,” *Mag. Res. Med.*, vol. 60, no. 2, pp. 414–21, Aug. 2008.
- [90] H. Gu, B. Yaman, K. Ugurbil, S. Moeller, and M. Akcakaya, “Compressed sensing MRI with ℓ_1 -wavelet reconstruction revisited using modern data science tools,” in *Proc. Int. Conf. IEEE Engr. Med. Biol. Soc.*, 2021, pp. 3596–600.
- [91] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*. Kluwer, 1996.
- [92] P. C. Hansen, *Rank-deficient and discrete ill-posed problems: Numerical aspects of linear inversion*. SIAM, 1998.
- [93] S. F. D. Waldron, *An introduction to finite tight frames*. Springer, 2018.
- [94] R. J. Duffin and A. C. Schaeffer, “A class of nonharmonic Fourier series,” *Trans. Amer. Math. Soc.*, vol. 72, no. 2, pp. 341–66, Mar. 1952.
- [95] B. Adcock and D. Huybrechs, “Frames and numerical approximation,” *SIAM Review*, vol. 61, no. 3, pp. 443–73, 2019.
- [96] V. Palyan, X. Y. Han, and D. L. Donoho, “Prevalence of neural collapse during the terminal phase of deep learning training,” *Proc. Nat. Acad. Sci.*, vol. 117, no. 40, pp. 24 652–63, Oct. 2020.
- [97] R. R. Coifman and D. L. Donoho, “Translation-invariant denoising,” 1995.
- [98] J. Kovacevic and A. Chebira, “An introduction to frames,” *Found. Trends Sig. Proc.*, vol. 2, no. 1, pp. 1–94, 2008.
- [99] A. Rahimi, G. Zandi, and B. Daraby, “Redundancy of fusion frames in Hilbert spaces,” *Complex Anal. Operat. Theory*, vol. 10, no. 3, pp. 545–65, Mar. 2016.

- [100] R. Yin, T. Gao, Y. M. Lu, and I. Daubechies, "A tale of two bases: Local–nonlocal regularization on image patches with convolution framelets," *SIAM J. Imaging Sci.*, vol. 10, no. 2, pp. 711–50, Jan. 2017.
- [101] E. Kang and J. C. Ye, "Framelet denoising for low-dose CT using deep learning," in *Proc. IEEE Int. Symp. Biomed. Imag.*, 2018, pp. 311–4.
- [102] J. C. Ye, Y. Han, and E. Cha, "Deep convolutional framelets: A general deep learning framework for inverse problems," *SIAM J. Imaging Sci.*, vol. 11, no. 2, pp. 991–1048, Jan. 2018.
- [103] M. Akcakaya and V. Tarokh, "A frame construction and a universal distortion bound for sparse representations," *IEEE Trans. Sig. Proc.*, vol. 56, no. 6, pp. 2443–50, Jun. 2008.
- [104] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.
- [105] V. Vapnik, "Principles of risk minimization for learning theory," in *NeurIPS*, vol. 4, 1991, pp. 831–8. [Online]. Available: <https://proceedings.neurips.cc/paper/1991/hash/fb4d5fbbafdf976cfdc032e3bde78de5-Abstract.html>
- [106] A. H. Sayed and T. Kailath, "A state-space approach to adaptive RLS filtering," *IEEE Sig. Proc. Mag.*, vol. 11, no. 3, pp. 18–60, Jul. 1994.
- [107] W. Liu, I. Park, Y. Wang, and J. C. Principe, "Extended kernel recursive least squares algorithm," *IEEE Trans. Sig. Proc.*, vol. 57, no. 10, pp. 3801–14, Oct. 2009.
- [108] D. Angelosante, J. A. Bazerque, and G. B. Giannakis, "Online adaptive estimation of sparse signals: where RLS meets the ℓ_1 -norm," *IEEE Trans. Sig. Proc.*, vol. 58, no. 7, pp. 3436–47, Jul. 2010.
- [109] Y. Chen and A. O. Hero, "Recursive $\ell_{1,\infty}$ group lasso," *IEEE Trans. Sig. Proc.*, vol. 60, no. 8, pp. 3978–87, Aug. 2012.
- [110] Y. Chi, Y. C. Eldar, and R. Calderbank, "PETRELS: Parallel subspace estimation and tracking by recursive least squares from partial observations," *IEEE Trans. Sig. Proc.*, vol. 61, no. 23, pp. 5947–59, Dec. 2013.
- [111] L. M. Lesser, "Critical values and transforming data: Teaching statistics with social justice," *J. Statist. Edu.*, vol. 15, no. 1, Aug. 2017.
- [112] D. G. Luenberger, *Optimization by vector space methods*. Wiley, 1969.
- [113] A. Bjorck and G. H. Golub, "Numerical methods for computing angles between linear subspaces," *Math. Comput.*, vol. 27, no. 123, pp. 579–94, Jul. 1973.
- [114] V. Eijkhout and P. Vassilevski, "The role of the strengthened Cauchy–Buniakowski–Schwarz inequality in multilevel methods," *SIAM Review*, vol. 33, no. 3, pp. 405–19, Sep. 1991.
- [115] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 1985.
- [116] V.-S. Chellaboina and W. M. Haddad, "Is the Frobenius matrix norm induced?" *IEEE Trans. Auto. Control*, vol. 40, no. 12, pp. 2137–9, Dec. 1995.
- [117] S. F. Cotter, B. D. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Trans. Sig. Proc.*, vol. 53, no. 7, pp. 2477–88, Jul. 2005.
- [118] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit," *Signal Process.*, vol. 86, no. 3, pp. 572–88, Mar. 2006.
- [119] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Royal Statist. Soc. Ser. B*, vol. 68, no. 1, pp. 49–67, 2006.

- [120] S. Feizi and D. Tse, "Maximally correlated principal component analysis," 2017. [Online]. Available: <http://arxiv.org/abs/1702.05471>
- [121] D. Cullina, P. Mittal, and N. Kiyavash, "Fundamental limits of database alignment," in *Int. Symp. Inf. Theory*, 2018, pp. 651–5.
- [122] C. Xu, B. Yang, F. Guo, W. Zheng, and P. Poignet, "Sparse-view CBCT reconstruction via weighted Schatten p -norm minimization," *Optics Express*, vol. 28, no. 24, pp. 35 469–82, Nov. 2020.
- [123] J. von Neumann, "Some matrix-inequalities and metrization of matrix-space," *Tomsk. Univ. Rev.*, vol. 1, pp. 286–300, 1937.
- [124] L. Mirsky, "On the trace of matrix products," *Mathematische Nachrichten*, vol. 20, no. 3-6, pp. 171–4, 1959.
- [125] ——, "A trace inequality of John von Neumann," *Monatshefte für Mathematik*, vol. 79, no. 4, pp. 303–6, 1975.
- [126] E. M. de Sa, "Exposed faces and duality for symmetric and unitarily invariant norms," *Linear Algebra Appl.*, vol. 197, pp. 429–50, 1994.
- [127] A. Ruhe, "Perturbation bounds for means of eigenvalues and invariant subspaces," *BIT Numerical Math.*, vol. 10, no. 3, pp. 343–54, 1970.
- [128] R. Bhatia, *Matrix analysis*. Springer, 1997.
- [129] K. Ball, E. A. Carlen, and E. H. Lieb, "Sharp uniform convexity and smoothness inequalities for trace norms," *Invent. Math.*, vol. 115, pp. 463–82, 1994.
- [130] N. Srebro, J. Rennie, and T. Jaakkola, "Maximum-margin matrix factorization," in *NeurIPS*, vol. 17, 2004. [Online]. Available: https://papers.nips.cc/paper_files/paper/2004/hash/e0688d13958a19e087e123148555e4b4-Abstract.html
- [131] H. Weyl, "Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung)," *Math. Ann.*, vol. 71, no. 4, pp. 441–79, Dec. 1912.
- [132] K. Fan, "Maximum properties and inequalities for the eigenvalues of completely continuous operators," *Proc. Nat. Acad. Sci.*, vol. 37, no. 11, pp. 760–6, Nov. 1951.
- [133] A. J. Hoffman and H. W. Wielandt, "The variation of the spectrum of a normal matrix," *Duke Math. J.*, vol. 20, no. 1, pp. 37–9, 1953.
- [134] Z. Leka, "Some singular value inequalities via convexity," *Linear Multilinear Algebra*, vol. 67, no. 2, pp. 360–9, 2019.
- [135] R. Bhatia and F. Kittaneh, "Notes on matrix arithmetic–geometric mean inequalities," *Linear Algebra Appl.*, vol. 308, no. 1–3, pp. 203–11, Mar. 2000.
- [136] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Proc. Mach. Learn. Res.*, vol. 28, no. 3, pp. 1310–8, 2013. [Online]. Available: <http://proceedings.mlr.press/v28/pascanu13.html>
- [137] D. Gilton, G. Ongie, and R. Willett, "Neumann networks for inverse problems in imaging," *IEEE Trans. Comput. Imaging*, vol. 6, pp. 328–43, 2020.
- [138] I. Dokmanic and R. Gribonval, "Beyond Moore–Penrose part I: Generalized inverses that minimize matrix norms," 2017. [Online]. Available: <http://arxiv.org/abs/1706.08349>
- [139] P. H. Schonemann, "A generalized solution of the orthogonal Procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, Mar. 1966.
- [140] J. C. Gower and G. B. Dijksterhuis, *Procrustes problems*. Oxford University Press, 2004.
- [141] S. Ahmed and I. M. Jaimoukha, "A relaxation-based approach for the orthogonal Procrustes problem with data uncertainties," in *Proc. UKACC Int. Conf. Control*, 2012, pp. 906–11.

- [142] G. Wahba, "A least squares estimate of satellite attitude," *SIAM Review*, vol. 7, no. 3, p. 409, 1965.
- [143] F. C. Ghesu, B. Georgescu, S. Grbic, A. K. Maier, J. Hornegger, and D. Comaniciu, "Robust multi-scale anatomical landmark detection in incomplete 3D-CT data," in *Medical Image Computing and Computer-Assisted Intervention*, 2017, pp. 194–202.
- [144] S. Ravishankar and Y. Bresler, " ℓ_0 sparsifying transform learning with efficient optimal updates and convergence guarantees," *IEEE Trans. Sig. Proc.*, vol. 63, no. 9, pp. 2389–404, May 2015.
- [145] M. Journee, Y. Nesterov, P. Richtárik, and R. Sepulchre, "Generalized power method for sparse principal component analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 517–53, Mar. 2010. [Online]. Available: <http://www.jmlr.org/papers/v11/journee10a.html>
- [146] R. W. Lissitz, P. H. Schonemann, and J. C. Lingoes, "A solution to the weighted Procrustes problem in which the transformation is in agreement with the loss function," *Psychometrika*, vol. 41, no. 4, pp. 547–50, 1976.
- [147] M. A. Koschat and D. F. Swayne, "A weighted Procrustes criterion," *Psychometrika*, vol. 56, no. 2, pp. 229–39, Jun. 1991.
- [148] P. Batchelor and J. M. Fitzpatrick, "A study of the anisotropically weighted Procrustes problem [optical image-guided surgery application]," in *Proc. IEEE Workshop Math. Methods Biomed. Image Anal.*, 2000, pp. 212–8.
- [149] M. Contino, J. I. Giribet, and A. Maestripieri, "Weighted Procrustes problems," *J. Math. Anal. Appl.*, vol. 445, no. 1, pp. 443–58, Jan. 2017.
- [150] M. A. T. Figueiredo and R. D. Nowak, "Ordered weighted L1 regularized regression with strongly correlated covariates: Theoretical aspects," in *AISTATS*, 2016, pp. 930–8. [Online]. Available: <http://proceedings.mlr.press/v51/figueiredo16.html>
- [151] M. Udell and A. Townsend, "Why are big data matrices approximately low rank?" *SIAM J. Math. Data Sci.*, vol. 1, no. 1, pp. 144–60, 2019.
- [152] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learning Representations*, 2022. [Online]. Available: <https://openreview.net/forum?id=nZeVKeeFYf9>
- [153] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–8, 1936.
- [154] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms," *Quarterly J. Math.*, vol. 11, no. 1, pp. 50–9, Jan. 1960.
- [155] E. Schmidt, "Zur Theorie der linearen und nichtlinearen Integralgleichungen," *Math. Ann.*, vol. 63, no. 4, pp. 433–76, 1907.
- [156] R. B. Cattell, "The scree test for the number of factors," *Multivar. Behav. Res.*, vol. 1, no. 2, pp. 245–76, 1966.
- [157] E. Dobriban, "Permutation methods for factor analysis and PCA," *Ann. Statist.*, vol. 48, no. 5, pp. 2824–47, 10 2020.
- [158] D. Hong, Y. Sheng, and E. Dobriban, "Selecting the number of components in PCA via random signflips," 2020. [Online]. Available: <http://arkiv.org/abs/2012.02985>
- [159] W. Leeb and E. Romanov, "Optimal spectral shrinkage and PCA with heteroscedastic noise," *IEEE Trans. Info. Theory*, vol. 67, no. 5, pp. 3009–37, 2021.
- [160] M. Buehrer, K. P. Pruessmann, P. Boesiger, and S. Kozerke, "Array compression for MRI with large coil arrays," *Mag. Res. Med.*, vol. 57, no. 6, pp. 1131–9, Jun. 2007.

- [161] T. Zhang, J. M. Pauly, S. S. Vasanawala, and M. Lustig, "Coil compression for accelerated imaging with Cartesian sampling," *Mag. Res. Med.*, vol. 69, no. 2, pp. 571–82, Feb. 2013.
- [162] P. Drineas and I. C. F. Ipsen, "Low-rank matrix approximations do not need a singular value gap," *SIAM J. Matrix. Anal. Appl.*, vol. 40, no. 1, pp. 299–319, Jan. 2019.
- [163] Y.-L. Yu and D. Schuurmans, "Rank/norm regularization with closed-form solutions: Application to subspace clustering," in *Proc. 27th Conf. Uncertainty in AI*, 2011, pp. 778–85. [Online]. Available: <http://arxiv.org/abs/1202.3772>
- [164] P. Verboon and W. J. Heiser, "Resistant lower rank approximation of matrices by iterative majorization," *Comp. Statist. Data Anal.*, vol. 18, no. 4, pp. 457–67, Nov. 1994.
- [165] E. J. Candes, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. Assoc. Comput. Mach.*, vol. 58, no. 3, pp. 1–37, May 2011.
- [166] T.-H. Oh, Y.-W. Tai, J.-C. Bazin, H. Kim, and I. S. Kweon, "Partial sum minimization of singular values in robust PCA: Algorithm and applications," *IEEE Trans. Pattern Anal. Mach. Int.*, vol. 38, no. 4, pp. 744–58, 2016.
- [167] R. R. Nadakuditi, "OptShrink: An algorithm for improved low-rank signal matrix denoising by optimal, data-driven singular value shrinkage," *IEEE Trans. Info. Theory*, vol. 60, no. 5, pp. 3002–18, May 2014.
- [168] Y. Shitov, "The nonnegative rank of a matrix: Hard problems, easy solutions," *SIAM Review*, vol. 59, no. 4, pp. 794–800, 2017.
- [169] L. T. K. Hien, D. N. Phan, N. Gillis, M. Ahookhosh, and P. Patrinos, "Block Bregman majorization minimization with extrapolation," *SIAM J. Math. Data Sci.*, vol. 4, no. 1, pp. 1–25, 2022.
- [170] M. B. Cohen, J. Nelson, and D. P. Woodruff, "Optimal approximate matrix product in terms of stable rank," 2016. [Online]. Available: <http://arxiv.org/abs/1507.02268>
- [171] R. J. Woodham, "Photometric method for determining surface orientation from multiple images," *Optical Eng.*, vol. 19, no. 1, p. 191139, 1980.
- [172] H. Hayakawa, "Photometric stereo under a light source with arbitrary motion," *J. Opt. Soc. Am. A*, vol. 11, no. 11, pp. 3079–89, Nov. 1994.
- [173] R. Basri and D. Jacobs, "Photometric stereo with general, unknown lighting," in *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, vol. 2, 2001, pp. 374–81.
- [174] L. Wu, A. Ganesh, B. Shi, Y. Matsushita, Y. Wang, and Y. Ma, "Robust photometric stereo via low-rank matrix completion and recovery," in *ACCV*, 2011, pp. 703–17.
- [175] S. Ikehata, D. Wipf, Y. Matsushita, and K. Aizawa, "Robust photometric stereo using sparse regression," in *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, 2012, pp. 318–25.
- [176] M. Zhang and M. S. Drew, "Robust surface normal estimation via greedy sparse regression," *EURASIP J. Image Vid. Proc.*, vol. 2015, no. 1, p. 42, 2015.
- [177] I. Borg and P. J. F. Groenen, *Modern multidimensional scaling: Theory and applications*. Springer, 2005.
- [178] E. Peterfreund and M. Gavish, "Multidimensional scaling of noisy high dimensional data," *Applied Comput. Harmonic Anal.*, vol. 51, pp. 333–73, 2021.
- [179] E. Waltz, "The algorithm that mapped omicron shows a path forward," *IEEE Spectrum*, Feb. 2022. [Online]. Available: <https://spectrum.ieee.org/omicron-covid-variant>
- [180] J. J. Moreau, "Proximité et dualité dans un espace hilbertien," *Bulletin de la Société Mathématique de France*, vol. 93, pp. 273–99, 1965. [Online]. Available: http://www.numdam.org/item?id=BSMF_1965__93__273_0

- [181] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 123–231, 2013.
- [182] A. Beck, *First-order methods in optimization*. SIAM, 2017.
- [183] E. J. Candes, C. A. Sing-Long, and J. D. Trzasko, "Unbiased risk estimates for singular value thresholding and spectral estimators," *IEEE Trans. Sig. Proc.*, vol. 61, no. 19, pp. 4643–57, Oct. 2013.
- [184] K. Fan and A. J. Hoffman, "Some metric inequalities in the space of matrices," *Proc. Amer. Math. Soc.*, vol. 6, no. 1, pp. 111–6, Feb. 1955.
- [185] M. Fazel, H. Hindi, and S. Boyd, "A rank minimization heuristic with application to minimum order system approximation," in *Proc. Amer. Control Conf.*, vol. 6, 2001, pp. 4734–9.
- [186] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Trans. Info. Theory*, vol. 56, no. 5, pp. 2053–80, May 2010.
- [187] G. Golub, A. Hoffman, and G. Stewart, "A generalization of the Eckart–Young–Mirsky matrix approximation theorem," *Linear Algebra Appl.*, vol. 88, pp. 317–27, Apr. 1987.
- [188] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, 2014, pp. 2862–9.
- [189] Y. Xie, S. Gu, Y. Liu, W. Zuo, W. Zhang, and L. Zhang, "Weighted Schatten p -norm minimization for image denoising and background subtraction," *IEEE Trans. Image Proc.*, vol. 25, no. 10, pp. 4842–57, Oct. 2016.
- [190] X. Liu, X.-Y. Jing, G. Tang, F. Wu, and Q. Ge, "Image denoising using weighted nuclear norm minimization with multiple strategies," *Signal Processing*, vol. 135, pp. 239–52, Jun. 2017.
- [191] T. Blu and F. Luisier, "The SURE-LET approach to image denoising," *IEEE Trans. Image Proc.*, vol. 16, no. 11, pp. 2778–86, Nov. 2007.
- [192] W. E. Leeb, "Matrix denoising for weighted loss functions and heterogeneous signals," *SIAM J. Math. Data Sci.*, vol. 3, no. 3, pp. 987–1012, 2021.
- [193] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *J. Mach. Learn. Res.*, vol. 16, pp. 2859–900, 2015.
[Online]. Available: <http://jmlr.org/papers/v16/cunningham15a.html>
- [194] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–7, Jul. 2006.
- [195] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 2, no. 6, pp. 459–73, 1989.
- [196] P. J. B. Hancock, R. J. Baddeley, and L. S. Smith, "The principal components of natural images," *Netw. Comput. Neural Syst.*, vol. 3, no. 1, pp. 61–70, 1992.
- [197] I. Davies and S. Eglen, "[Re] the principal components of natural images," *ReScience C*, vol. 6, no. 3, p. 6, Oct. 2020.
- [198] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Int.*, vol. 35, no. 8, pp. 1798–828, Aug. 2013.
- [199] B. Scholkopf, A. Smola, and K.-R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–319, Jul. 1998.
- [200] M. Scholz, F. Kaplan, C. L. Guy, J. Kopka, and J. Selbig, "Non-linear PCA: A missing data approach," *Bioinformatics*, vol. 21, no. 20, pp. 3887–95, 2005.

- [201] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *NeurIPS*, vol. 14, 2001. [Online]. Available: <https://proceedings.neurips.cc/paper/2001/hash/f106b7f99d2cb30c3db1c3cc0fde9ccb-Abstract.html>
- [202] ———, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–96, 2003.
- [203] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–605, 2008. [Online]. Available: <http://jmlr.org/papers/v9/vandermaaten08a.html>
- [204] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–6, 2000.
- [205] Y. Wang, H. Huang, C. Rudin, and Y. Shaposhnik, "Understanding how dimension reduction tools work: An empirical approach to deciphering t-SNE, UMAP, triMap, and PaCMAP for data visualization," *J. Mach. Learn. Res.*, vol. 22, no. 201, pp. 1–73, 2021. [Online]. Available: <http://jmlr.org/papers/v22/20-1061.html>
- [206] P. A. Penczek, M. Kimmel, and C. M. T. Spahn, "Identifying conformational states of macromolecules by eigen-analysis of resampled cryo-EM images," *Structure*, vol. 19, no. 11, pp. 1582–90, Nov. 2011.
- [207] E. Bair, T. Hastie, D. Paul, and R. Tibshirani, "Prediction by supervised principal components," *J. Am. Statist. Assoc.*, vol. 101, no. 473, pp. 119–37, 2006.
- [208] E. Barshan, A. Ghodsi, Z. Azimifar, and M. Z. Jahromi, "Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds," *Pattern Recognition*, vol. 44, no. 7, pp. 1357–71, Jul. 2011.
- [209] A. Ritchie, C. Scott, L. Balzano, D. Kessler, and C. S. Sripada, "Supervised principal component analysis via manifold optimization," in *IEEE Data Science Workshop*, 2019, pp. 6–10.
- [210] Y. Ma, P. Niyogi, G. Sapiro, and R. Vidal, "Dimensionality reduction via subspace and submanifold learning," *IEEE Sig. Proc. Mag.*, vol. 28, no. 2, pp. 14–126, Mar. 2011.
- [211] R. Vidal, "Subspace clustering," *IEEE Sig. Proc. Mag.*, vol. 28, no. 2, pp. 52–68, Mar. 2011.
- [212] J. Lipor, D. Hong, Y. S. Tan, and L. Balzano, "Subspace clustering using ensembles of K -subspaces," *Inf. Inference*, vol. 10, no. 1, pp. 73–107, Nov. 2020.
- [213] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Patt. Anal. Mach. Int.*, vol. 27, no. 12, pp. 1945–59, Dec. 2005.
- [214] P. Comon and G. H. Golub, "Tracking a few extreme singular values and vectors in signal processing," *Proc. IEEE*, vol. 78, no. 8, pp. 1327–43, Aug. 1990.
- [215] L. Balzano, Y. Chi, and Y. M. Lu, "Streaming PCA and subspace tracking: The missing data case," *Proc. IEEE*, vol. 106, no. 8, pp. 1293–310, Aug. 2018.
- [216] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numer. Math.*, vol. 31, no. 2, pp. 111–29, 1978.
- [217] G. H. Golub, "Some modified matrix eigenvalue problems," *SIAM Review*, vol. 15, no. 2, pp. 318–34, 1973.
- [218] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra Appl.*, vol. 415, no. 1, pp. 20–30, May 2006.
- [219] E. Oja, "Simplified neuron model as a principal component analyzer," *J. Math. Bio.*, vol. 15, no. 3, pp. 267–73, Nov. 1982.

- [220] C. J. Li, M. Wang, H. Liu, and T. Zhang, "Near-optimal stochastic approximation for online principal component estimation," *Math. Program.*, vol. 167, no. 1, pp. 75–97, Jan. 2018.
- [221] N. Halko, P. Martinsson, and J. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–88, 2011.
- [222] J. A. Duersch and M. Gu, "Randomized projection for rank-revealing matrix factorizations and low-rank approximations," *SIAM Review*, vol. 62, no. 3, pp. 661–82, 2020.
- [223] D. Zhang and L. Balzano, "Global convergence of a Grassmannian gradient descent algorithm for subspace estimation," in *AISTATS*, vol. 51, 2016, pp. 1460–8. [Online]. Available: <https://proceedings.mlr.press/v51/zhang16b.html>
- [224] Y. Chi, Y. M. Lu, and Y. Chen, "Nonconvex optimization meets low-rank matrix factorization: An overview," *IEEE Trans. Sig. Proc.*, vol. 67, no. 20, pp. 5239–69, Oct. 2019.
- [225] S. Burer and R. D. C. Monteiro, "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization," *Math. Program.*, vol. 95, no. 2, pp. 329–57, Feb. 2003.
- [226] Y. Chen and Y. Chi, "Harnessing structures in big data via guaranteed low-rank matrix estimation: Recent theory and fast algorithms via convex and nonconvex optimization," *IEEE Sig. Proc. Mag.*, vol. 35, no. 4, pp. 14–31, Jul. 2018.
- [227] Z. Li, Y. Luo, and K. Lyu, "Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning," in *Proc. Int. Conf. Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=AH0s7Sm5H7R>
- [228] J. P. Boyd, "Finding the zeros of a univariate equation: Proxy rootfinders, Chebyshev interpolation, and the companion matrix," *SIAM Review*, vol. 55, no. 2, pp. 375–396, 2013.
- [229] J. W. Cooley and J. W. Tukey, "An algorithm for machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, Apr. 1965.
- [230] W. H. Young, "On the multiplication of successions of Fourier constants," *Proc. Roy. Soc. Lon. A, Math. and Phys. Sci.*, vol. 87, no. 596, pp. 331–9, 1912.
- [231] M. Unser and T. Blu, "Generalized smoothing splines and the optimal discretization of the Wiener filter," *IEEE Trans. Sig. Proc.*, vol. 53, no. 6, pp. 2146–59, Jun. 2005.
- [232] H. Akaike, "Block Toeplitz matrix inversion," *SIAM J. Appl. Math.*, vol. 24, no. 2, pp. 234–41, Mar. 1973.
- [233] M. Wax and T. Kailath, "Efficient inversion of Toeplitz-block Toeplitz matrix," *IEEE Trans. Acoust. Sp. Sig. Proc.*, vol. 31, no. 5, pp. 1218–21, Oct. 1983.
- [234] L. Rodman and T. Shalom, "On inversion of symmetric Toeplitz matrices," *SIAM J. Matrix. Anal. Appl.*, vol. 13, no. 2, pp. 530–49, 1992.
- [235] M. K. Ng, K. Rost, and Y.-W. Wen, "On inversion of Toeplitz matrices," *Linear Algebra Appl.*, vol. 348, no. 1, pp. 145–51, Jun. 2002.
- [236] I. Gohberg and G. Heinig, "Inversion of finite Toeplitz matrices," in *Convolution equations and singular integral operators*, L. Lerer, V. Olshevsky, and I. M. Spitkovsky, Eds. Springer, 2010, pp. 1–6.
- [237] V. Sukhoy and A. Stoytchev, "Generalizing the inverse FFT off the unit circle," *Nature Sci. Rep.*, vol. 9, no. 1, p. 14443, 2019.

- [238] H. G. Feichtinger, K. Gröchenig, and T. Strohmer, "Efficient numerical methods in non-uniform sampling theory," *Numerische Mathematik*, vol. 69, no. 4, pp. 423–40, Feb. 1995.
- [239] J. A. Fessler, S. Lee, V. T. Olafsson, H. R. Shi, and D. C. Noll, "Toeplitz-based iterative image reconstruction for MRI with correction for magnetic field inhomogeneity," *IEEE Trans. Sig. Proc.*, vol. 53, no. 9, pp. 3393–402, Sep. 2005.
- [240] Z. Yang, L. Xie, and P. Stoica, "Vandermonde decomposition of multilevel Toeplitz matrices with application to multidimensional super-resolution," *IEEE Trans. Info. Theory*, vol. 62, no. 6, pp. 3685–701, Jun. 2016.
- [241] Y. Yoshida and T. Miyato, "Spectral norm regularization for improving the generalizability of deep learning," 2017. [Online]. Available: <http://arxiv.org/abs/1705.10941>
- [242] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. Int. Conf. Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=B1QRgziT->
- [243] E. K. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *Proc. Int. Conf. Mach. Learn.*, 2019. [Online]. Available: <http://arxiv.org/abs/1905.05406>
- [244] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "WTR: The who to follow service at Twitter," in *Proc. Int. Conf. WWW*, 2013, pp. 505–14.
- [245] N. J. Higham and F. Tisseur, "Bounds for eigenvalues of matrix polynomials," *Linear Algebra Appl.*, vol. 358, no. 1, pp. 5–22, 2003.
- [246] B. G. Horne, "Lower bounds for the spectral radius of a matrix," *Linear Algebra Appl.*, vol. 263, pp. 261–73, Sep. 1997.
- [247] S. U. Pillai, T. Suel, and S. Cha, "The Perron–Frobenius theorem: Some of its applications," *IEEE Sig. Proc. Mag.*, vol. 22, no. 2, pp. 62–75, Mar. 2005.
- [248] H. Schneider, "Wielandt's proof of the exponent inequality for primitive nonnegative matrices," *Linear Algebra Appl.*, vol. 353, no. 1, pp. 5–10, Sep. 2002.
- [249] R. S. Varga, *Gershgorin and his circles*. Springer, 2004.
- [250] C. D. Meyer, *Matrix analysis and applied linear algebra*. SIAM, 2000.
- [251] H. Minc, *Nonnegative matrices*. New York: Wiley, 1988.
- [252] A. Berman and R. J. Plemmons, *Nonnegative matrices in the mathematical sciences*. SIAM, 1994.
- [253] L. Page, "Method for node ranking in a linked database," 1998, patent US6285999. [Online]. Available: <https://patents.google.com/patent/US6285999>
- [254] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Comp. Netw. ISDN Syst.*, vol. 30, no. 1, pp. 107–17, 1998.
- [255] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
- [256] M. Belkin and P. Niyogi, "Towards a theoretical foundation for Laplacian-based manifold methods," *J. Comp. Sys. Sci.*, vol. 74, no. 8, pp. 1289–308, Dec. 2008.
- [257] S. Feizi, G. Quon, M. Recamonde-Mendoza, M. Medard, M. Kellis, and A. Jadbabaie, "Spectral alignment of graphs," *IEEE Trans. Network Sci. Eng.*, vol. 7, no. 3, pp. 1182–97, 2020.
- [258] A. Cauchy, "Méthode générale pour la résolution des systèmes d'équations simultanées," *Comptes Rendu Sci. Paris*, vol. 25, pp. 536–8, 1847.
- [259] G. W. Cross and P. Lancaster, "Square roots of complex matrices," *Linear Multilinear Algebra*, vol. 1, no. 4, pp. 289–93, 1974.

- [260] A. Bjorck and S. Hammarling, "A Schur method for the square root of a matrix," *Linear Algebra Appl.*, vol. 52-53, pp. 127–40, Jul. 1983.
- [261] W. Zuo and Z. Lin, "A generalized accelerated proximal gradient approach for total-variation-based image restoration," *IEEE Trans. Image Proc.*, vol. 20, no. 10, pp. 2748–59, Oct. 2011.
- [262] R. H. Chan and M. K. Ng, "Conjugate gradient methods for Toeplitz systems," *SIAM Review*, vol. 38, no. 3, pp. 427–82, Sep. 1996.
- [263] Y. Drori and M. Teboulle, "Performance of first-order methods for smooth convex minimization: A novel approach," *Math. Program.*, vol. 145, no. 1-2, pp. 451–82, Jun. 2014.
- [264] D. Kim and J. A. Fessler, "Optimized first-order methods for smooth convex minimization," *Math. Program.*, vol. 159, no. 1, pp. 81–107, Sep. 2016.
- [265] M. Teboulle and Y. Vaisbourd, "An elementary approach to tight worst case complexity analysis of gradient based methods," *Math. Program.*, 2022.
- [266] Y. Benyaminini and J. Lindenstrauss, *Geometric nonlinear functional analysis: Volume 1*. AMS, 2000.
- [267] R. C. Fair, "On the robust estimation of econometric models," *Ann. Econ. Social Measurement*, vol. 2, pp. 667–77, Oct. 1974. [Online]. Available: <http://fairmodel.econ.yale.edu/rayfair/pdf/1974D.HTM>
- [268] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Comm. Statist. Theory Meth.*, vol. 6, no. 9, pp. 813–27, 1977.
- [269] K. Lange, "Convergence of EM image reconstruction algorithms with Gibbs smoothing," *IEEE Trans. Med. Imag.*, vol. 9, no. 4, pp. 439–46, Dec. 1990, corrections: vol. 10, no. 2, p. 288, June 1991.
- [270] B. T. Polyak, *Introduction to optimization*. Optimization Software Inc, 1987. [Online]. Available: <http://lab7.ipu.ru/eng/sel-papers.html>
- [271] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Smooth strongly convex interpolation and exact worst-case performance of first-order methods," *Math. Program.*, vol. 161, no. 1, pp. 307–45, Jan. 2017.
- [272] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," *Dokl. Akad. Nauk. USSR*, vol. 269, no. 3, pp. 543–7, 1983.
- [273] ——, "Smooth minimization of non-smooth functions," *Math. Program.*, vol. 103, no. 1, pp. 127–52, May 2005.
- [274] ——, *Introductory lectures on convex optimization: A basic course*. Springer, 2004.
- [275] D. Kim and J. A. Fessler, "On the convergence analysis of the optimized gradient methods," *J. Optim. Theory Appl.*, vol. 172, no. 1, pp. 187–205, Jan. 2017.
- [276] O. Güler, "New proximal point algorithms for convex minimization," *SIAM J. Optim.*, vol. 2, no. 4, pp. 649–64, 1992.
- [277] Y. Drori, "The exact information-based complexity of smooth convex minimization," *J. Complexity*, vol. 39, pp. 1–16, Apr. 2017.
- [278] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
- [279] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Exact worst-case performance of first-order methods for composite convex optimization," *SIAM J. Optim.*, vol. 27, no. 3, pp. 1283–313, Jan. 2017.
- [280] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–7, Sep. 1951.

- [281] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.
- [282] O. Gencoglu, M. van Gils, E. Guldogan, C. Morikawa, M. Süzen, M. Gruber, J. Leinonen, and H. Huttunen, "HARK side of deep learning – from grad student descent to automated machine learning," 2019. [Online]. Available: <http://arxiv.org/abs/1904.07633>
- [283] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [284] J. Eckstein and D. P. Bertsekas, "On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Math. Program.*, vol. 55, no. 1–3, pp. 293–318, Apr. 1992.
- [285] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.
- [286] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for machine learning*. MIT Press, 2011. [Online]. Available: <https://mitpress.mit.edu/books/optimization-machine-learning>
- [287] Z. Xu, "The minimal measurement number for low-rank matrix recovery," *Appl. Comput. Harmonic Anal.*, vol. 44, no. 2, pp. 497–508, Mar. 2018.
- [288] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Found. Comp. Math.*, vol. 9, no. 6, pp. 717–72, 2009.
- [289] I. Ramirez and G. Sapiro, "An MDL framework for sparse coding and dictionary learning," *IEEE Trans. Sig. Proc.*, vol. 60, no. 6, pp. 2913–27, Jun. 2012.
- [290] R. H. Keshavan and S. Oh, "A gradient descent algorithm on the Grassmann manifold for matrix completion," 2009. [Online]. Available: <http://arxiv.org/abs/0910.5260>
- [291] N. Vaswani, Y. Chi, and T. Bouwmans, "Rethinking PCA for modern data sets: Theory, algorithms, and applications," *Proc. IEEE*, vol. 106, no. 8, pp. 1274–6, Aug. 2018.
- [292] K. Bredies and D. A. Lorenz, "Linear convergence of iterative soft-thresholding," *J. Fourier Anal. Appl.*, vol. 14, no. 5, pp. 813–37, Dec. 2008.
- [293] D. Kim and J. A. Fessler, "Adaptive restart of the optimized gradient method for convex optimization," *J. Optim. Theory Appl.*, vol. 178, no. 1, pp. 240–63, Jul. 2018.
- [294] W. Miao, S. Pan, and D. Sun, "A rank-corrected procedure for matrix completion with fixed basis coefficients," *Math. Program.*, vol. 159, no. 1, pp. 289–338, 2016.
- [295] Z. Zhu, Q. Li, G. Tang, and M. B. Wakin, "Global optimality in low-rank matrix optimization," *IEEE Trans. Sig. Proc.*, vol. 66, no. 13, pp. 3614–28, Jul. 2018.
- [296] Y. Zhang, Q. Qu, and J. Wright, "From symmetry to geometry: Tractable nonconvex problems," 2020. [Online]. Available: <http://arxiv.org/abs/2007.06753>
- [297] P. Zilber and B. Nadler, "GNMR: A provable one-line algorithm for low rank matrix recovery," *SIAM J. Math. Data Sci.*, vol. 4, no. 2, pp. 909–34, 2022.
- [298] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.
- [299] J. Cai, E. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–82, 2010.
- [300] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 720–7. [Online]. Available: <https://dl.acm.org/doi/10.5555/3041838.3041929>

- [301] V. Chandrasekaran, S. Sanghavi, P. Parrilo, and A. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM J. Optim.*, vol. 21, no. 2, pp. 572–596, 2011.
- [302] T. Bouwmans, S. Javed, H. Zhang, Z. Lin, and R. Otazo, "On the applications of robust PCA in image and video processing," *Proc. IEEE*, vol. 106, no. 8, pp. 1427–57, Aug. 2018.
- [303] K. Gitman and L. Balzano, "Panoramic video separation with online Grassmannian robust subspace estimation," in *Proc. Int. Conf. Comp. Vision*, 2019, pp. 643–51.
- [304] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequievre, "A benchmark dataset for outdoor foreground/background extraction," in *ACCV Workshop*, 2012, pp. 291–300.
- [305] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–91, 1999.
- [306] X. Fu, K. Huang, N. D. Sidiropoulos, and W.-K. Ma, "Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications," *IEEE Sig. Proc. Mag.*, vol. 36, no. 2, pp. 59–80, Mar. 2019.
- [307] A. Cichocki, R. Zdunek, A. H. Phan, and S.-I. Amari, *Nonnegative matrix and tensor factorizations: Applications to exploratory multi-way data analysis and blind source separation*. Wiley, 2009.
- [308] G. Zhou, A. Cichocki, Q. Zhao, and S. Xie, "Nonnegative matrix and tensor factorizations: An algorithmic perspective," *IEEE Sig. Proc. Mag.*, vol. 31, no. 3, pp. 54–65, May 2014.
- [309] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion," in *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, 2010, pp. 1791–8.
- [310] J. D. Trzasko and A. Manduca, "Calibrationless parallel MRI using CLEAR," in *Proc. IEEE Asilomar Conf. Signals, Systems, and Comp.*, 2011, pp. 75–9.
- [311] W. Dong, G. Shi, and X. Li, "Nonlocal image restoration with bilateral variance estimation: A low-rank approach," *IEEE Trans. Im. Proc.*, vol. 22, no. 2, pp. 700–11, Feb. 2013.
- [312] W. Dong, G. Shi, X. Li, Y. Ma, and F. Huang, "Compressive sensing via nonlocal low-rank regularization," *IEEE Trans. Im. Proc.*, vol. 23, no. 8, pp. 3618–32, Aug. 2014.
- [313] S. Ravishankar, B. E. Moore, R. R. Nadakuditi, and J. A. Fessler, "Low-rank and adaptive sparse signal (LASSI) models for highly accelerated dynamic imaging," *IEEE Trans. Med. Imag.*, vol. 36, no. 5, pp. 1116–28, May 2017.
- [314] Y. Cai and S. Li, "Convergence and stability of iteratively reweighted least squares for low-rank matrix recovery," *Inverse Prob. Imaging*, vol. 11, no. 4, pp. 643–61, Aug. 2017.
- [315] R. Ge, C. Jin, and Y. Zheng, "No spurious local minima in nonconvex low rank problems: A unified geometric analysis," in *Proc. Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1233–42. [Online]. Available: <http://proceedings.mlr.press/v70/ge17a.html>
- [316] C. Bordenave, S. Coste, and R. R. Nadakuditi, "Detection thresholds in very sparse matrix completion," *Found. Comp. Math.*, 2022.
- [317] L. T. Nguyen, J. Kim, and B. Shim, "Low-rank matrix completion: A contemporary survey," *IEEE Access*, vol. 7, pp. 94215–37, 2019.
- [318] F. Rosenblatt, "The Perceptron – a perceiving and recognizing automaton," Aeronautical Laboratory, Cornell, Tech. Rep. 85–460-1, Jan. 1957.
- [319] ——, "The perceptron. A probabilistic model for information storage and organization in the brain," *Pysch. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.
- [320] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: <http://www.deeplearningbook.org>
- [321] J. C. Ye, *Geometry of deep learning: A signal processing perspective*. Springer, 2022.

- [322] C. Huang, "ReLU networks are universal approximators via piecewise linear or constant functions," *Neural Computation*, vol. 32, no. 11, pp. 2249–78, 2020.
- [323] G. Strang, "The functions of deep learning," *SIAM News*, vol. 51, no. 10, Dec. 2018. [Online]. Available: <https://sinews.siam.org/Details-Page/the-functions-of-deep-learning>
- [324] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Sig. Sys.*, vol. 2, no. 4, pp. 303–14, Dec. 1989.
- [325] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–7, 1991.
- [326] H. N. Mhaskar and T. Poggio, "Deep vs. shallow networks: An approximation theory perspective," *Anal. Appl.*, vol. 14, no. 06, pp. 829–48, Nov. 2016.
- [327] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–7, 1993.
- [328] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–6, Oct. 1986.
- [329] J. Shenouda, R. Parhi, K. Lee, and R. D. Nowak, "Vector-valued variation spaces and width bounds for DNNs: Insights on weight decay regularization," 2023. [Online]. Available: <http://arxiv.org/abs/2305.16534>
- [330] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Sig. Proc. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.
- [331] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980.
- [332] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. Int. Conf. Learning Representations*, 2014. [Online]. Available: <https://openreview.net/forum?id=yIxE6yojDR5yqX>
- [333] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, 2016, pp. 770–8.
- [334] R. Sunkara and T. Luo, "No more strided convolutions or pooling: A new CNN building block for low-resolution images and small objects," in *ECML PKDD*, 2022, pp. 443–59.
- [335] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. European Comp. Vision Conf.*, 2018, pp. 3–19.
- [336] J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu, "Orthogonal convolutional neural networks," in *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, 2020, pp. 11 502–12.
- [337] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–41.
- [338] P. Gavrikov and J. Keuper, "CNN filter DB: An empirical investigation of trained convolutional filters," in *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, 2022, pp. 19 044–54.
- [339] Z. Zhao, J. C. Ye, and Y. Bresler, "Generative models for inverse imaging problems: From mathematical foundations to physics-driven applications," *IEEE Sig. Proc. Mag.*, vol. 40, no. 1, pp. 148–63, 2023.

- [340] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, vol. 23, no. 1, pp. 5–48, Mar. 1991.
- [341] B. Widrow, J. Kollár, and M.-C. Liu, "Statistical theory of quantization," *IEEE Trans. Instr. Meas.*, vol. 45, no. 2, pp. 353–61, Apr. 1996.
- [342] R. Latała, "Some estimates of norms of random matrices," *Proc. Amer. Math. Soc.*, vol. 133, pp. 1273–82, 2005.
- [343] J. L. Barlow and E. H. Bareiss, "On roundoff error distributions in floating point and logarithmic arithmetic," *Computing*, vol. 34, no. 4, pp. 325–47, 1985.
- [344] M. Talagrand, "Concentration of measure and isoperimetric inequalities in product spaces," *Publ. Math. de l'Institut des Hautes Scientifiques*, vol. 81, no. 1, pp. 73–205, 1995.
- [345] T. Tao, *Topics in random matrix theory*. AMS, 2012, vol. 132. [Online]. Available: <https://terrytao.files.wordpress.com/2011/02/matrix-book.pdf>
- [346] R. Vershynin, *High-dimensional probability: An introduction with applications in data science*. Cambridge University Press, 2018, vol. 47. [Online]. Available: <https://www.math.uci.edu/~rvershyn/papers/HDP-book/HDP-book.html>
- [347] F. Benaych-Georges and R. R. Nadakuditi, "The singular values and vectors of low rank perturbations of large rectangular random matrices," *J. Multivar. Anal.*, vol. 111, pp. 120–35, Oct. 2012.
- [348] V. A. Marcenko and L. A. Pastur, "Distribution of eigenvalues for some sets of random matrices," *Math. USSR-Sbornik*, vol. 1, no. 4, p. 457, 1967.
- [349] L. Erdos and H.-T. Yau, "Universality of local spectral statistics of random matrices," *Bull. Amer. Math. Soc.*, vol. 49, no. 3, pp. 377–414, Jul. 2012.
- [350] T. Tao and V. H. Vu, "Random matrices: The universality phenomenon for Wigner ensembles," in *Modern aspects of random matrix theory*, V. H. Vu, Ed. AMS, 2014, pp. 121–72.
- [351] Z. D. Bai and J. W. Silverstein, "No eigenvalues outside the support of the limiting spectral distribution of large-dimensional sample covariance matrices," *Ann. Prob.*, vol. 26, no. 1, pp. 316–45, Jan. 1998.
- [352] J. W. Silverstein, "Strong convergence of the empirical distribution of eigenvalues of large dimensional random matrices," *J. Multivar. Anal.*, vol. 55, no. 2, pp. 331–9, Nov. 1995.
- [353] J. Huang, B. Landon, and H.-T. Yau, "Bulk universality of sparse random matrices," *J. Math. Phys.*, vol. 56, no. 12, p. 123301, 12 2015.
- [354] L. Erdos, A. Knowles, H.-T. Yau, and J. Yin, "Spectral statistics of Erdős–Rényi graphs II: Eigenvalue spacing and the extreme eigenvalues," *Comm. Math. Phys.*, vol. 314, no. 3, pp. 587–640, 2012.
- [355] A. G. Bedejan, "A note on the aphorism ‘there is nothing as practical as a good theory’," *J. Manag. Hist.*, vol. 22, no. 2, pp. 236–42, 2016.
- [356] J. Veraart, D. S. Novikov, D. Christiaens, B. Ades-aron, J. Sijbers, and E. Fieremans, "Denoising of diffusion MRI using random matrix theory," *NeuroImage*, vol. 142, pp. 394–406, Nov. 2016.
- [357] L. Vizoli, S. Moeller, L. Dowdle, M. Akcakaya, F. D. Martino, E. Yacoub, and K. Ugurbil, "Lowering the thermal noise barrier in functional brain mapping with magnetic resonance imaging," *Nature Comm.*, vol. 12, p. 5181, 2021.
- [358] A. Kuzmin, M. V. Baalen, Y. Ren, M. Nagel, J. Peters, and T. Blankevoort, "FP8 quantization: The power of the exponent," in *NeurIPS*, 2022. [Online]. Available: <https://openreview.net/forum?id=H3Gv7XEGzYV>

***K*-nearest neighbor**, 275
t-SNE, 274
t-distributed stochastic neighbor embedding, 274
JULIA programming language, 7
 2-norm, 42
 2D DFT, 59
 2D inverse DFT, 59
 802.11n Wi-Fi, 80
 absorbing, 305, 306, 319, 320
 activation function, 383, 387
 Adam, 361
 adaptive, 184, 277
 additive synthesis, 103
 additive white Gaussian noise, 396
 additively separable, 257, 258
 additivity, 203
 adjacency matrix, 14, 305, 306, 309
 adjoint, 20
 ADMM, 362
 affine, 11, 148, 270, 282, 385
 affine subspace, 161
 algebraic multiplicity, 70
 algebraic number, 331
 almost sure convergence, 397
 alternating direction method of multipliers, 362
 alternating projection, 369
 analysis operator, 169
 angle, 43
 between flats, 206
 between subspaces, 206, 233
 between vectors, 205
 principal, 206
 ANN, 269
 antipodal, 225
 antisymmetric, 91
 aperiodic, 307–309, 312, 313
 arctangent, 21
 arithmetic–geometric mean inequality, 217
 array, 12
 multiplication, 368
 slicing, 7
 artificial neural network, 191, 269, 381
 associative property, 30
 asymptotic, 316, 354
 autodifferentiation, 386
 autoencoder, 269
 AWGN, 396
 backpropagation, 270, 386
 banded, 388
 basis, 101–104, 125, 132, 179, 247, 248, 332
 basis extension theorem, 104
 basis matrix, 102
 basis vectors, 101, 193
 batch, 361
 Bayes classifier, 360
 BCCB, 20, 332
 beamforming, 80
 best fit, 187
 best Lipschitz constant, 232
 bias, 387
 bias vector, 270
 BIBO stability, 330
 biconvex, 377
 big O, 26, 354
 bilevel optimization, 162
 binary classification, 182, 356
 binary classifier design, 364
 block circulant, 20, 332
 block diagonal, 20, 285
 block lower triangular, 48
 block matrix, 19, 59
 block matrix multiplication, 33
 block matrix triangularization, 48
 block Toeplitz, 388
 block upper triangular, 48
 Boolean matrix, 369
 bounded curvature, 351
 broadcast, 37, 384
 BTTB, 388
 Burer–Monteiro factorization, 279
 canonical basis, 126
 cardinality, 64, 102, 105
 cascade, 30
 Cauchy–Schwarz inequality, 43, 203–206, 211, 391
 causal system, 16

- Cayley–Hamilton theorem, 286
 chain rule, 386
 channel estimation, 80
 characteristic equation, 50
 characteristic polynomial, 50, 53, 283–286
 Cholesky decomposition, 17, 63
 circulant matrix, 18, 67, 173, 290, 292, 293, 306,
 332, 346
 circular convolution, 293
 classification, 130, 279, 386
 classifier, 3
 closed, 130, 135
 closest, 190
 clustering, 322
 CNN, 19, 381, 387
 code, 269
 codomain, 257
 coil compression, 242
 collaborative filtering, 365
 color channel, 1
 column rank, 111, 170
 column space, 109
 column vector, 10
 column-major order, 27
 commutative property, 30, 53
 commuting matrices, 293, 332, 335, 347
 compact SVD, 75, 77, 121–123, 141, 151, 153,
 154, 156, 157, 160, 170, 181, 190, 193, 194,
 207, 213, 236, 241, 253, 254
 companion matrix, 283, 285–288, 290, 330, 346
 compatible, 207
 complement, 106, 108
 completed the square, 152
 complex conjugate, 4, 21
 complex coordinate space, 10
 complex number, 21
 complex-conjugate pair, 50
 composite cost function, 375
 compressed sensing, 162
 concatenation, 30
 condition number, 164, 165, 192, 363
 conformable, 30
 conjugate cogradient, 150
 conjugate gradient method, 167
 conjugate transpose, 20
 connected component, 324
 consistent, 206
 constrained optimization, 134, 355
 continuous, 10
 converge, 92, 218–220, 312, 335, 353
 convergence, 229
 convergence rate, 363
 convex, 136, 257, 262, 351, 352, 368, 374, 377
 convex combination, 135
 convex function, 134, 136, 148, 150, 200, 232, 349
 convex hull, 98
 convex relaxation, 262, 371
 convex set, 130, 134, 135, 142, 200, 232
 convolution, 16, 294, 387
 1×1 , 388
 convolutional layer, 387
 convolutional neural network, 16, 387
 coordinate, 102
 coordinate system, 102
 correlation coefficient, 203
 cost function, 134, 335
 countably infinite, 99
 counting measure, 198
 Courant–Fischer min–max principle, 79
 covariance matrix, 51
 cross entropy, 386
 cross product, 24
 cross validation, 168, 360, 371
 cross-correlation, 387
 CWR factorization, 111
 cyclic commutative property, 55
 damping factor, 321
 data, 12
 DCT, 101, 124
 decision boundary, 356, 382
 decoder, 269
 decomposition method, 48
 deep learning, 384
 defective, 68, 69
 degree, 50, 324
 degree matrix, 324
 degrees of freedom, 91, 366
 denoising, 279, 390
 dense layer, 384
 dense matrix, 18
 derivative, 123, 148
 determinant, 41, 45, 46, 52, 54, 55, 61, 284
 commutative property, 46
 scaling, 47
 transpose, 47
 DFT, 16, 67, 101, 126, 291, 331
 matrix, 288
 diagonal, 17, 69, 87, 193, 220, 301
 diagonal majorizer, 363
 diagonal matrix, 18
 diagonalizable, 66, 68–70, 73, 83, 175, 286, 288,
 296, 317, 337, 346
 diagonalize, 292
 diagonally dominant, 306, 363
 diagonally preconditioned GD, 343
 digital signal processing, 16
 dimension, 103, 104
 dimension theorem for vector spaces, 102
 dimensionality reduction, 96, 238, 244, 247, 257,
 269, 271, 272, 390
 direct sum, 106
 directed edge, 303
 directed graph, 14, 305

- discrete cosine transform, 101, 124
discrete Fourier transform, 16, 58
discrete sine transform, 124
discriminative, 389
displacement vector, 225
distance matrix, 250
distinct, 87
distributive property, 30, 53
divergence, 265
DoF, 366
dot product, 23, 38
downsampling, 387
DSP, 16
DST, 124
dynamical system, 13

Eckart–Young–Mirsky, 245
economy SVD, 77, 122, 153, 154, 170
eigendecomposition, 63–65, 69, 92
eigenfunction, 123
eigenvalue, 13, 14, 50, 53, 55, 61, 62, 64, 80, 91,
 92, 94, 139, 283, 285, 291, 297, 331
 algorithm, 17
eigenvector, 14, 51, 62, 64, 65, 86, 87, 91, 92, 94,
 139, 287, 291
elementwise, 36
EM algorithm, 378
empirical covariance matrix, 273
empirical measure, 183
empirical risk minimization, 183, 356, 360
encoder, 269
equilibrium distribution, 315, 316, 319
equivalent, 200, 214
equivariant, 18, 387–389
ERM, 183, 356, 358
error, 161
ETF, 172
Euclidean norm, 43, 44, 190, 197, 198
Euclidean space, 9
 complex, 10
Euler's formula, 21
exists, 4

factor analysis, 240, 271
factorization, 94
Fair potential, 352, 363
fast Fourier transform, 292
fast gradient method, 353
feature map, 387
feature vector, 13
FEM, 124
FFT, 292, 293
FGM, 353, 376
Fibonacci number, 13
field, 8, 60, 331
finite difference, 17, 124, 173, 332
finite-dimensional, 104

finite-element method, 124
first-order, 353
FISTA, 360, 376
flat, 161
floating-point number, 393
floating-point operation, 26
floating-point precision, 164
FLOPs, 26
FOIL, 222
for all, 4
foreground/background separation, 2, 378
forgetting factor, 185
Fourier series, 97, 203
frame, 169, 170, 174, 192, 288, 331
frame bound, 169, 331
Frobenius inner product, 204, 205, 207, 211, 212
Frobenius norm, 92, 93, 197, 204, 207, 216, 221,
 222, 229, 233, 238, 369
full column rank, 190, 192
full matrix, 18
full rank, 41, 85, 112
full SVD, 77, 122, 154, 157, 194
fully connected graph, 323
fully connected layer, 384
function composition, 131, 389
fundamental theorem of algebra, 50
fundamental theorem of linear algebra, 120

Gauss–Newton matrix recovery, 377
Gaussian elimination, 17, 41, 63, 151
GD, 188, 218, 336
Gelfand's formula, 217, 310
generalized eigenvalue, 81
generalized eigenvector, 298, 337
generalized inverse, 221, 229, 236
generalized PCA, 277
generative, 389
geodesic distance, 257
geometric multiplicity, 70
Gershgorin disk, 299, 310, 363
Givens rotation, 44
GNMR, 377
golden ratio, 13
GP, 355
GPCA, 277
Grace Wahba, 222
grad student descent, 361
gradient, 148–150, 233, 349
gradient descent, 188, 191, 218, 233, 336,
 349, 363
gradient projection, 355
Gram matrix, 17, 45, 51, 100, 151, 177, 252
Gram–Schmidt, 64, 279
graph
 connected subset, 324
graph Laplacian, 116, 322, 324, 326
grayscale, 12

- greatest common divisor, 307
 group sparsity, 208
 Hölder's inequality, 198, 205, 212
 Haar wavelet, 173
 Hadamard product, 34, 368
 handwritten digit recognition, 3, 133, 177, 356
 hard thresholding, 259
 Heaviside step function, 259
 Helmut Wielandt, 302
 Hermitian, 21, 80, 176, 203
 Hermitian transpose, 20, 22, 150
 Hessian, 353
 Hessian matrix, 351, 359
 heteroscedastic, 188
 hidden layer, 384
 Hilbert matrix, 101
 Hilbert–Schmidt inner product, 204
 Hilbert–Schmidt norm, 207
 hinge loss, 358
 Hoffman–Wielandt inequality, 215
 homographic transformation, 82
 Householder reflection, 44
 hull, 98
 hyperbolic tangent, 383
 hyperparameter, 261
 hyperplane, 96
 idempotent, 60, 131, 132, 157, 174–176, 196, 227
 identity, 8
 identity function, 132
 identity matrix, 18, 30
 if and only if, 4
 IID, 183
 ill posed, 365
 image (range), 109
 image compression, 238
 image deblurring, 1, 294
 image registration, 222
 impulse response, 16, 291
 inclusion-exclusion principle, 107
 incremental, 277
 incremental SVD, 278
 independently and identically distributed, 183
 indicator function, 16, 198
 induced, 208, 209, 219, 310
 induced matrix norm, 208, 209, 211, 217, 218
 induced norm, 204, 232
 induction, 73, 135
 infinite dimensional, 97, 104
 infinity norm, 198
 information retrieval, 13
 inner product, 23, 31, 101, 203–205, 232, 233, 248, 250
 inner product space, 203
 input layer, 384
 integer programming, 277
 integration by substitution, 46
 intersection, 105
 invariance, 18
 invariant, 224
 inverse, 8
 inverse DFT, 59
 invertible, 40, 41, 46, 53, 68, 73, 100, 147, 288, 299
 invertible matrix, 60
 involution, 22
 irreducible, 306–309, 311, 315, 319, 320
 ISTA, 376
 ISVD, 278
 iterative shrinkage thresholding algorithm, 376
 iterative soft thresholding algorithm, 376
 Jacobian matrix, 46
 Jensen's inequality, 234
 Jordan block, 69
 Jordan normal form, 55, 63, 69, 286, 298, 318, 337
 Jordan–Wielandt matrix, 216
 Juan Gris, 2
K-nearest neighbor, 130
 kernel (nullspace), 116
 kernel PCA, 274
 Khatri–Rao, 34
 Kronecker impulse, 291
 Kronecker product, 34, 331
 Kronecker sum, 93, 289, 331
 kurtosis, 394
 Ky Fan norm, 210, 212, 213
 Lambertian surface, 250
 landmark registration, 222
 Laplace expansion, 48
 Laplacian eigenmap, 274, 326
 latent, 1, 263, 269, 366
 law of total probability, 315
 LDA, 24
 learning rate, 361
 least-squares, 187, 188, 190, 191, 232, 237, 363
 left eigenvector, 14
 left inverse, 40, 155, 170, 221
 left singular vector, 74, 82, 87, 90
 lifting, 382, 384
 line search, 189, 349
 linear, 23, 109, 388
 linear algebra, 23
 linear combination, 98, 99, 135
 linear discriminant analysis, 24
 linear least-squares, 146, 191, 224, 244, 245
 linear map, 11, 12, 108
 linear operation, 12, 14, 15, 131
 linear operator, 11, 208
 linear regression, 144
 linear separability, 382
 linear span, 98
 linear subspace, 96

- linear time-invariant, 16
 linear transform, 11, 12, 108
 linear variety, 161, 221
 linearly dependent, 81, 99, 100, 114
 linearly independent, 41, 68, 97, 99, 100, 104, 110, 125, 139, 147, 179, 247
 link graph matrix, 14
 Lipschitz constant, 232, 233, 364, 389
 Lipschitz continuous, 232, 349–353, 358, 389
 Lipschitz function, 232
 LLS, 146, 192
 Loewner order, 88, 341
 logical implication, 4, 6
 logistic function, 358, 359, 383, 386
 logistic regression, 183, 360, 364
 long-term average probability, 333
 loss function, 183, 191, 364
 low-rank, 242, 253, 365
 low-rank approximation, 165, 238, 245, 248, 270, 279, 281
 low-rank matrix completion, 365
 lower bound on the number of samples, 367
 lower Hessenberg, 17
 lower triangular, 17, 285
 lower–upper (LU) decomposition, 41
 LRMC, 365
 LTI, 16
 LU decomposition, 63
 machine learning, 3
 majorization, 335
 majorize, 342, 372, 373
 majorize–minimize, 372
 Manhattan norm, 198
 manifold, 326, 381
 mantissa, 392
 maps to, 5
 Marčenko–Pastur, 399, 404
 Markov chain, 283, 298, 301, 312, 313, 327, 328, 333, 334
 matched filter, 24
 matrix, 12–14, 22, 149, 269, 314
 tall, 18, 75, 76, 121, 123, 129
 wide, 18, 75, 76, 121, 122
 matrix 2-norm, 78
 matrix addition, 341
 matrix calculus, 148
 matrix completion, 262, 365, 380, 401
 matrix decomposition, 24, 30
 matrix determinant lemma, 49
 matrix Euclidian norm, 207
 matrix exponential, 36, 73
 matrix factorization, 213
 matrix inversion lemma, 41, 184
 matrix multiplication, 30, 34, 109, 368
 matrix norm, 197, 206, 217, 219, 229, 235
 $\ell_{p,q}$, 208
 matrix power, 53, 73
 matrix product, 113
 matrix sensing, 262, 377
 matrix square root, 210, 335, 337, 338
 matrix–matrix product, 29, 31
 matrix–vector multiplication, 15, 35, 384
 matrix–vector product, 26, 27
 max norm, 198, 207, 235
 maximum column sum matrix norm, 210
 maximum row sum matrix norm, 209
 MDL, 371
 MDS, 238, 250, 254, 257
 MIMO, 80
 min–max theorem, 81
 minimal polynomial, 69, 285–288
 minimum 2-norm solution, 190
 minimum description length, 371
 minimum norm, 160
 minimum-norm LS solution, 162, 186
 Minkowski inequality, 198
 Minkowski sum, 105
 minor, 48
 MLP, 384
 MM, 372
 model, 365
 modulo, 15
 modulus, 21
 momentum, 361
 monic polynomial, 50, 283, 285
 monomial, 50, 100
 monotone, 347, 353
 monotonicity, 215
 Moore–Penrose pseudoinverse, 154, 155, 197, 221, 229
 MSE, 264
 multi-input multi-output, 80
 multidimensional scaling, 238, 250, 254, 279, 281
 multilayer perceptron, 384
 multiple dispatch, 37
 multiplication, 55
 multiplication by a scalar, 9
 multiplicity, 53, 311, 312
 MUSIC algorithm, 116
 nearest neighbor, 275
 nearest subspace, 276
 nearest-subset classifier, 131
 necessary condition, 160
 Netflix problem, 365
 Neumann series, 220
 neural network, 145, 381
 neuron, 383
 Newton’s method, 353
 NMF, 379
 NN, 381

- NNLS, 356
node, 303
noncausal, 19
nonconvex, 239, 368
nonconvex function, 386
nonlinear dimensionality reduction, 274, 322, 326
nonlinear PCA, 274
nonnegative, 308–310, 314
nonnegative least-squares, 356
nonnegative matrix, 301–303, 306, 309, 310
nonnegative matrix factorization, 249, 379
nonnegative orthant, 135
nonsingular, 40, 140, 306
norm, 42, 199, 204, 220, 231, 232
 ℓ_p , 198
 ℓ_∞ , 198
ordered weighted ℓ_1 , 229
sup, 198
uniform, 198
weighted, 198
normal, 66–68, 81, 83, 84, 87, 91, 94, 115, 217, 218, 235, 332
normal equation, 150, 158, 160, 186, 271
normal matrix, 66, 294
normalized root mean-squared difference, 264
normalized root mean-squared error, 264
not aperiodic, 307, 333
NP hard, 368
NRMSD, 264
NRMSE, 264
nuclear norm, 210, 212–214, 236, 262, 332, 371, 374
nuclear norm regularizer, 376
null space, 115–117, 119, 140, 141, 157, 193
nullity, 117, 118
numerical stability, 244
oblique projection, 175
ODE, 73
OGM, 349, 354
Oja’s method, 279
one-hot encoding, 386
online, 184, 277
operator norm, 78, 94, 197, 206, 208, 209, 229
optimization, 78, 134, 335
optimized gradient method, 349, 354
OptShrink, 265, 282, 403
ordinary differential equation, 73
orthogonal, 42, 44, 52, 55, 61, 65, 70, 97, 100, 107, 125, 190
orthogonal basis, 125–127, 332
orthogonal complement, 107, 108, 141, 180, 227, 252
orthogonal matrix, 43, 44, 66, 67, 74, 126, 127, 176
orthogonal polynomials, 147
orthogonal Procrustes problem, 222, 224, 228, 229
orthogonal projection, 133, 142, 235
orthogonal projection matrix, 176, 177, 181
orthogonal projector, 157, 176, 181
orthogonal set, 42
orthogonal wavelet transform, 101
orthogonality principle, 132, 161, 179, 180
orthonormal, 42, 61, 100, 272, 291
orthonormal K -frame, 45
orthonormal basis, 65, 126, 132, 133, 179, 181, 248, 275, 276, 332
orthonormal columns, 45, 155, 156
orthonormal DFT, 291
orthonormal rows, 155, 156
orthonormal set, 42, 44
orthonormal vectors, 81
outer product, 24, 25, 33, 47, 57, 109, 110, 113, 118, 128, 208
matrix, 51, 58
outlier, 201, 378, 400
output layer, 384
over-determined system, 154
OWL, 229
OWT, 101
PageRank, 14, 283, 301, 319, 327
parallel processing, 31
parallelogram law, 204
Parseval tight frame, 171–174, 177, 186, 192, 331
Parseval’s identity, 45, 202
partial fraction expansion, 91
PCA, 133, 249, 271, 272
pentadiagonal, 17
perceptron, 24, 383
period, 307, 308, 311
periodic, 333
periodic end conditions, 332
periodic functions, 97
permutation group, 215
permutation matrix, 46, 66
permutation method, 242
perpendicular, 42, 161
Perron root, 310–312
Perron vector, 310–312
Perron–Frobenius eigenvalue, 310–312
Perron–Frobenius theorem, 310–312, 315
PGD, 335, 336, 349
PGM, 376
phase transition, 398
photometric stereo, 2, 250
pilot signal, 80
pivoting, 63
Plancherel theorem, 202
POCS, 368
POGM, 360, 376, 378
polar decomposition, 224
polar form, 21
polylog, 367
polynomial regression, 147

- pooling, 387
 poorly conditioned, 164
 positive definite, 63, 88, 89, 140, 203, 231, 301, 335, 337, 338, 341, 349, 351
 positive matrix, 301, 302, 304, 306, 308, 309, 312, 321
 positive semidefinite, 88, 89, 94, 95, 140, 193, 196, 210, 216, 224, 295, 301, 324, 336, 338, 341, 351
 power iteration, 16, 92, 296, 298, 311, 312, 333
 preconditioned gradient descent, 336, 363
 preconditioned steepest descent, 335, 349
 preconditioner, 294, 336, 338
 predict, 144
 primitive, 302, 303, 305–309, 312, 315–319
 primitive matrix, 301, 302, 305, 309, 312, 317
 principal component analysis, 13, 249, 271
 principal component regression, 166
 principal eigenvector, 16
 principal minors, 89
 principal square root, 337, 338
 Procrustes, 225, 236, 250
 Procrustes problem, 197, 222, 225, 237
 projection, 130, 131, 135, 177, 178, 196, 200, 249, 355, 369
 projection matrix, 131, 132, 174, 177, 317
 projections onto convex sets, 368
 projective transformation, 82
 proximal gradient method, 376
 proximal operator, 257, 259, 364, 376
 PSD, 335
 pseudoinverse, 157, 162, 170, 188, 195, 221
 push-through identity, 41
 QR decomposition, 64, 85, 151, 158
 quadratic, 149, 335
 query, 14
 random matrix theory, 265, 390
 random walk, 319
 range, 109, 110, 115, 139, 141, 177, 193
 range space, 119, 140
 rank, 75, 85, 93, 96, 111–114, 139, 140, 193, 249, 279
 rank constraint, 371
 rank regularizer, 371
 rank-1 approximation, 244, 245
 rank-1 matrix, 24
 rank-1 update, 41, 49, 91, 185
 rational function, 60, 278
 ray, 225
 Rayleigh quotient, 81, 273
 Rayleigh–Ritz theorem, 81
 real coordinate space, 9
 recommender system, 365
 rectangular diagonal matrix, 17, 74, 156, 267
 rectified linear unit, 383
 recurrent neural network, 217, 300
 recursive, 184, 277
 recursive least-squares, 184
 reducible, 306
 regularization parameter, 167, 279
 regularized low-rank approximation, 281
 regularized LS, 140, 191, 192, 233
 regularizer, 261
 relative complement, 5, 53
 ReLU, 383
 residual, 146, 160, 161, 187, 201
 restricted isometry property, 377
 reverse mode, 386
 reverse triangle inequality, 200, 300
 RGB, 1, 387
 ridge regression, 165, 167
 right eigenvector, 14, 90
 right inverse, 40, 155, 221
 right singular vector, 74, 82, 87
 RIP, 377
 risk, 183, 264
 RLS, 184, 185, 277
 RMT, 390
 robust PCA, 378
 robust regression, 202
 root of unity, 291
 roots, 285
 rotation
 improper, 72
 matrix, 67, 72, 75
 roundoff error, 207, 391, 395
 row gradient, 359
 row rank, 110, 111, 113
 row space, 109
 row vector, 10
 sample covariance matrix, 272
 sampling mask, 367
 sandwich inequality, 372
 SAT exam, 189
 scaling, 203
 scatter matrix, 51
 Schatten norm, 207, 210–212, 215, 216
 Schur complement, 40
 Schur decomposition, 64, 338
 Schur norm, 207
 Schwarz, 43, 204
 scores, 273
 scree plot, 240, 242
 second-order, 353
 seminorm, 198, 199, 230
 semiorthogonal matrix, 45
 semiunitary, 45
 sensor localization, 250
 set difference, 5, 53
 SGD, 361, 386
 Sherman–Morrison formula, 41

- Sherman–Morrison–Woodbury identity, 41
 shift invariant, 18, 387–389
 shrink, 256, 261
 sigmoid, 383
 signal-to-noise ratio, 80
 significand, 392, 393
 similar, 62, 68, 69, 91
 similarity function, 323
 similarity graph, 323
 simplex equiangular tight frame, 172
 simultaneously diagonalizable, 69, 346, 347
 simultaneously triangularizable, 347
 single precision, 392
 singular matrix, 51, 64, 299
 singular value, 74, 75, 80, 81, 92, 94, 169, 192, 233
 singular value decomposition, 17, 24, 63, 69, 74
 singular value hard thresholding, 261, 374
 singular value soft thresholding, 262, 375
 singular vector, 74, 94, 248
 sketching, 279
 skew-symmetric, 62
 smooth function, 349
 SNR, 80
 social justice, 189
 soft thresholding, 258, 262
 softmax, 386
 source localization, 238
 span, 98, 99, 102, 105
 spark, 114
 sparse matrix, 18
 sparse subspace clustering, 277
 sparsity, 160, 258
 spectral clustering, 116, 283, 322, 323, 325, 327
 spectral norm, 78, 93, 197, 209, 214, 216, 233,
 245, 281, 294, 351
 spectral normalization, 296
 spectral radius, 217, 229, 235, 298, 300, 309, 310,
 363
 spectral theorem, 64–66
 specular reflection, 250
 square, 17, 75, 303, 309, 314
 square root, 336, 337
 SSC, 277
 stable rank, 249, 250
 standard basis, 101, 104, 126
 standard simplex, 386
 state, 303, 313
 stationary distribution, 315
 stationary point, 148
 statistical learning theory, 183
 steady state, 315, 316
 steepest descent, 189, 349
 Stein’s unbiased risk estimate, 168, 264
 step size, 336, 353, 363
 Stiefel manifold, 127, 212, 225, 228, 237, 276, 279
 stochastic eigenvector, 315, 316
 stochastic gradient descent, 361
 stochastic matrix, 301, 302, 309, 310, 312, 314
 streaming, 184, 277
 strictly convex, 136, 200, 235, 257, 351, 352, 359
 strong law of large numbers, 396
 strongly connected, 305, 306, 318–320
 sub-Gaussian, 397
 subadditive, 114, 197
 submultiplicative, 206–209, 213, 216, 219, 229,
 235
 subset, 96, 105
 subspace, 96, 98, 103, 105, 108, 109, 116, 131,
 132, 135, 139, 141, 177, 179
 sum, 105
 subspace clustering, 277, 322
 subspace learning, 277, 279
 subspace sum, 106
 subspace union, 106
 sum of outer products, 36
 supervised learning, 168, 182, 191, 277
 supervised PCA, 276
 support-vector machine, 358
 supremum, 198
 SURE, 264
 surface normals, 250
 surrogate function, 372
 surrogate loss functions, 358
 SVD, 63, 64, 69, 74, 86, 92–94, 114, 143, 156,
 157, 167, 181, 193, 221–223, 227, 229, 239,
 248, 270, 273, 279, 374
 SVM, 358
 SVST, 262, 375, 376
 Sylvester’s criterion, 89
 Sylvester’s determinant theorem, 49
 Sylvester’s rank inequality, 113
 symmetric, 21, 97
 symmetric gauge, 261, 282
 synthesis operator, 169
 system of linear equations, 13, 143
 Talagrand’s concentration inequality, 395
 Taylor’s theorem, 351, 373
 tensor, 387
 term-document matrix, 13
 thin SVD, 77
 thresholding, 261
 tight, 79, 94, 209, 211, 234, 235
 tight frame, 44, 170–172, 174, 193, 331
 tight upper bound, 79
 Tikhonov regularization, 165, 167, 192
 time invariance, 18
 time-homogeneous Markov chain, 314
 Toeplitz, 18, 293–295, 387
 total variation, 199
 trace, 54, 55, 60–62
 trace norm, 210
 tracking, 277

- Tracy-Singh, 34
training, 270
transient, 305, 306, 313, 319
transition matrix, 301, 309, 312, 314, 315, 317, 327, 328, 333, 334
transition probabilities, 313
translation, 225
transpose, 9, 20, 22
triangle inequality, 142, 197, 200, 206, 212, 213, 215, 230, 351
tridiagonal, 17
trigonometric identities, 124
truncated SVD, 164, 165, 167, 240, 256
tune, 167
tuples, 304
TV, 199
twice differentiable, 351
U-net, 389
uncorrelated, 230
uncountably infinite, 64, 74, 78, 99
under-determined, 151, 158
union of subspaces, 105, 139, 277, 322
unique, 75, 243, 244, 311, 317, 370
unit norm, 42
unit vector, 24, 25
unitarily invariant, 140, 202, 216, 221, 229, 245, 251, 261, 282
unitary, 41, 44, 52, 61, 65, 66, 74, 115, 118, 140, 186, 248
unitary DFT, 291
unitary eigendecomposition, 66, 67, 176
unitary invariance, 45, 82, 235
unitary matrix, 44, 126, 172, 291
units, 35, 47, 54, 60
unity, 8
universal approximation theorem, 385
universality, 400
unnormalized graph Laplacian, 324
unsupervised, 269, 277
updating, 277
upper Hessenberg, 17
upper triangular, 17, 103, 285
upsampling, 387
Vandermonde, 288, 291, 295, 331
variational, 79
vec operator, 34
vec trick, 34, 58, 60
vector, 7, 9, 10, 12, 22
vector 2-norm, 198
vector addition, 6, 9
vector norm, 197, 219, 229, 232
vector space, 8, 9, 12, 23, 35, 96, 108, 204, 206, 232
Venn diagram, 177, 192, 196
vertex, 303
video, 2, 378
volume of the parallelepiped, 46
von Neumann's trace inequality, 211, 212
Wahba's problem, 222
weak duality, 95
weakly differentiable, 264
weight decay, 386
weight matrix, 324
weighted directed graph, 303, 314
weighted graph, 322, 323
weighted least-squares, 188
weights, 387
Weinstein–Aronszajn identity, 49
Weyl's inequality, 82, 215, 239, 247
Wielandt–Hoffman theorem, 391
Wirtinger calculus, 150
Young's convolution inequality, 294
zero vector, 9