

Mathematical Prerequisites

Linear algebra, calculus, and matrix decompositions

Last modified: 2026-10-22

i Source and scope

Consolidated from Kameron Decker Harris' 2022 weekly notes and Logan Sizemore's Spring 2025 math review lecture.

This appendix fixes the notation the rest of the notes use and directs you to the worked calculations, interactive widgets, and exercises in the lab's shared prerequisite site: *Math for Data Science*^a. The probability foundations are in Probability, and the statistical prerequisites are in Statistics.

Everything here is standard. Goodfellow et al. [1], chapter 2, covers linear algebra, and chapter 4 covers derivatives, gradients, and gradient-based optimization. Bishop [2] covers matrix identities in Appendix C, and Boyd and Vandenberghe [3] covers mathematical background in Appendix A.

^a<https://morrison-lab.github.io/mds/>

Almost every model in these notes is built out of a handful of operations on vectors and matrices, together with the derivatives of functions that take those objects as input. *Math for Data Science*¹, the lab's shared mathematics prerequisites, defines each of them with worked examples. This appendix links to those definitions rather than repeating them, and for the main ones says where these notes use them.

None of it is specific to machine learning. For a fuller treatment of the linear algebra, see chapter 2 of Goodfellow, Bengio and Courville [1].

1 Notation

Every model in this course takes numbers in and gives numbers out, and the numbers arrive as vectors (lists of numbers). One email is a vector of word counts; one patient is a vector of measurements. This chapter fixes the notation for those vectors, for the functions that map a vector to a number or to another vector, and for the derivative that tells us which direction to adjust a model.

Three conventions hold throughout the notes:

- A single real number² (a scalar) is written in plain italic: x , w , b .
- A column vector³ of p real numbers⁴, an $p \times 1$ array, is written with $\mathbf{v}_x$ ($x \in \mathbb{R}^{p \times 1}$) or $\mathbf{vec}\{\mathbf{x}\}$. Vectors are columns unless something says otherwise.
- A **matrix** — a rectangle of real numbers with m rows and n columns — is a capital letter: $W \in \mathbb{R}^{m \times n}$, or $\mathbf{X}$ for data matrices.

Scalars, vectors and matrices in these notes have real entries, with one exception: a covariate vector $\tilde{x}$ can mix numbers and categories, because a raw covariate may be categorical, such as a car's country of origin. A feature function encodes it as numbers, which gives its feature vector, a column vector⁵ in $\mathbb{R}^{p \times 1}$.

¹<https://morrison-lab.github.io/mds/>

²<https://morrison-lab.github.io/mds/number-systems.html#def-real-numbers>

³<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-column-vector>

⁴<https://morrison-lab.github.io/mds/number-systems.html#def-real-numbers>

⁵<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-column-vector>

Other sources write $\mathbb{R}^p$ for $\mathbb{R}^{p \times 1}$, treat a vector as a row, or allow complex entries or categorical entries in a vector.

x_i is the i th entry of $\tilde{x}$, and W_{ij} is the entry of W in row i and column j . Row index first, always.

These notes use the shared mathematics macros from [Morrison-Lab/macros](https://github.com/Morrison-Lab/macros)⁶, so their notation matches the notation chapters of *Math for Data Science*⁷, *Probability for Data Science*⁸, and *Regression Models for Epidemiology*⁹. We write column vectors with $\backslash\mathbf{v}\mathbf{x}$ ($\tilde{x}$), which gives the same typography across all of the lab's notes repositories.

Table 1: Notation used in this book

symbol	meaning	LaTeX
$\tilde{x} \in \mathbb{R}^{p \times 1}$	column vector ¹⁰ of p real numbers ¹¹	$\backslash\mathbf{v}\mathbf{x}$ $\backslash\mathbf{in}$ $\backslash\mathbf{reals}^{\backslash\mathbf{ninputvars}}$ $\backslash\mathbf{times} \ 1\}$
$\mathbf{X} \in \mathbb{R}^{n \times p}$	data matrix with n observations and p features	$\backslash\mathbf{m}\mathbf{X}$ $\backslash\mathbf{in}$ $\backslash\mathbf{reals}^{\backslash\mathbf{nobs}}$ $\backslash\mathbf{times} \ \mathbf{ninputvars}\}$
$x^\top, \tilde{x}^\top$	transpose of a scalar or vector	$\backslash\mathbf{tp}\{x\}, \backslash\mathbf{tp}\{\mathbf{v}\mathbf{x}\}$
$x \cdot y$	inner product of vectors x and y	$\backslash\mathbf{dprod}\{\mathbf{v}\mathbf{x}\}\{\mathbf{v}\mathbf{y}\}$
$\ \tilde{x}\ _2$	Euclidean (ℓ_2) norm / length of vector	$\backslash\mathbf{norm}\{\mathbf{v}\mathbf{x}\}_2$
$\mathbb{E}, \mathbb{E}[\cdot]$	$\tilde{x}$ expectation of a random variable	$\backslash\mathbf{E}\mathbf{p}, \backslash\mathbf{Exp}\mathbf{f}\{\backslash\mathbf{c}\mathbf{d}\mathbf{ot}\}$
$\text{Var}(\cdot)$	variance of a random variable	$\backslash\mathbf{V}\mathbf{ar}\{\backslash\mathbf{c}\mathbf{d}\mathbf{ot}\}$
ω	odds ¹² $\pi/(1 - \pi)$ of a probability π	$\backslash\mathbf{odds}$
η	log-odds ¹³ , the logarithm of the odds	$\backslash\mathbf{logodds}$
	ω ; also the linear component η of a logistic model	
ω	odds ratio ¹⁴ : the odds in one group divided by the odds in a reference group, written ω (the ratio symbol with the odds as subscript); OR is its abbreviation in running text	$\backslash\mathbf{ror}, \backslash\mathbf{OR}$
$\stackrel{\text{def}}{=}$	defined as	$\backslash\mathbf{eqdef}$
$\mathbb{1}_A(x), \mathbb{1}(P)$	indicator function: 1 if condition holds, 0 otherwise	$\backslash\mathbf{indic}\{A\}(x), \backslash\mathbf{indic}\mathbf{p}\{P\}$
$\perp\!\!\!\perp$	independent	$\backslash\mathbf{ind}$
$\therefore$	therefore, thus	$\backslash\mathbf{tf}, \backslash\mathbf{therefore}$

For logical connectives, quantifiers, set notation, and proofs, see the *Math for Data Science* Notation page¹⁵ and *Probability for Data Science* Notation page¹⁶.

1.1 Where different notation conventions come from

Machine learning, statistics, computer science, and econometrics developed their notations independently before converging on modern data science. Students reading across these literatures encounter different symbols for the same objects. Knowing where these conventions originate explains the choices:

⁶<https://github.com/Morrison-Lab/macros>

⁷<https://morrison-lab.github.io/mds/notation.html>

⁸<https://morrison-lab.github.io/pds/notation.html>

⁹<https://morrison-lab.github.io/rme/chapters/notation.html>

¹⁰<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-column-vector>

¹¹<https://morrison-lab.github.io/mds/number-systems.html#def-real-numbers>

¹²<https://morrison-lab.github.io/pds/probability-basics.html#def-odds>

¹³<https://morrison-lab.github.io/pds/probability-basics.html#def-log-odds>

¹⁴<https://morrison-lab.github.io/sds/categorical-tests.html#def-odds-ratio>

¹⁵<https://morrison-lab.github.io/mds/notation.html>

¹⁶<https://morrison-lab.github.io/pds/notation.html>

- **Weights $\tilde{w}$, b vs. coefficients $\vec{\beta}, \beta_0$:** In artificial intelligence and neural networks [1], [2], a linear function is written $f(x) = \tilde{w} \cdot x + b$ (or $Wx + \vec{b}$), where $\tilde{w}$ contains synaptic *weights* and b is the *bias* shifting the activation threshold. In classical statistics and econometrics [4], [5], regression models trace back to Gauss and Legendre, using Greek letters ($\vec{\beta}$) for unknown population parameters, and either Roman letters ($\vec{b}$) or a hat on the parameter ($\hat{\beta}$) for sample estimates. This chapter uses $\tilde{w}$ and b after Goodfellow et al. [1], while Linear Regression and subsequent chapters use the traditional regression notation $\vec{\beta}$ and β_0 (matching *Regression Models for Epidemiology* and Hastie et al. [4]). They represent the exact same mathematical linear model.
- **Dimension (D vs. p) and sample size (N vs. n):** Computer science and machine learning texts typically denote the feature dimension by D (for *dimension*) or d , with N data points [1], [2]. Statistical literature traditionally denotes the sample size by n and the number of predictors by p (for *parameters* or *predictors*) [4], [6].
- **Targets vs. responses:** Neural networks and pattern recognition texts often denote the supervised label by a target t or y [2], [7], whereas statistics denotes the outcome as the response or dependent variable y .
- **Rosetta stones across disciplines:** Ripley [7, Sec. 1.1], Table 1.1 provides an early systematic glossary mapping parallel terminology between statistics and neural networks (inputs vs. predictors, outputs vs. responses, weights vs. coefficients, training vs. fitting). In economics, Varian [5] discusses the complementary perspectives and terminology of machine learning and econometrics (such as features vs. regressors and explanatory variables). Breiman [8] and Efron and Hastie [6] provide deeper historical perspective on how algorithmic modeling and data modeling traditions diverged and recombined.

Notation across sources¹⁷ sets our notation beside Brian Hutchinson’s and Kameron Decker Harris’ lecture notes and five widely used texts, one row per concept.

1.2 Vectors and inner products

Math for Data Science covers:

- Column vectors¹⁸ and the transpose¹⁹. Both *Math for Data Science* and these notes write a vector with a tilde under the letter, using the shared macro $\tilde{x}$ (and transpose $\tilde{x}^\top$).
- Orthogonal vectors²⁰.
- The outer product²¹, and the rank of an outer product²².

The inner product of two column vectors of the same length multiplies them entry by entry and adds up the results, so it takes two vectors in and returns a single number. *Math for Data Science*²³ defines it, with a worked example, as the dot product²⁴. These notes write it as

$$\tilde{a} \cdot \tilde{b} = \sum_{i=1}^p a_i b_i \quad (1)$$

It is also called the *dot product* or the *scalar product* [9]. The three names are interchangeable. Which one you meet depends on whether the author came to the subject through linear algebra, through vector calculus, or through machine learning. These notes say inner product.

Almost every model in this course computes a score that way: a vector of weights, one per feature, and a vector of feature values, joined by their inner product.

¹⁷<https://morrison-lab.github.io/macros/notation-across-sources.html>

¹⁸<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-column-vector>

¹⁹<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-transpose>

²⁰<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-orthogonal-vectors>

²¹<https://morrison-lab.github.io/mds/linear-algebra-inner-products.html#def-outer-product>

²²<https://morrison-lab.github.io/mds/linear-algebra-inner-products.html#thm-outer-product-rank>

²³<https://morrison-lab.github.io/mds/>

²⁴<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-dot-product>

A vector's length, or ℓ_2 norm, is the square root of its inner product (Equation 1) with itself. *Math for Data Science* defines it as the Euclidean norm²⁵. These notes write it as

$$\begin{aligned}\|a\|_2 &= \sqrt{a \cdot a} \\ &= \sqrt{\sum_{i=1}^p a_i^2}\end{aligned}\tag{2}$$

It is also called a vector's *magnitude*. The subscript 2 distinguishes it from the other ℓ_p norms, and an unsubscripted $\|a\|$ almost always means this one.

In two dimensions that is the Pythagorean theorem, and in p dimensions it is the same formula with more terms. Norms return when we penalize large weights in Linear Regression.

Math for Data Science provides a worked exercise:

- Computing an inner product²⁶

i Watch

- Sanderson's video on vectors^a [10, Ch. 1], and on the dot product as a projection, from 0:51^b [10, Ch. 9].
- Hutchinson's Linear Function Basics^c (24 min) covers inner products, norms and hyperplanes [11]. The video site requires a login, which enrolled students find in the syllabus.

^ahttps://www.youtube.com/watch?v=fNk_zzaMoSs

^b<https://www.youtube.com/watch?v=LyGKycYT2v0&t=51s>

^chttps://facultyweb.cs.wvu.edu/~hutchib2/video_lectures/data371/#linear_function_basics

i Read

GeeksforGeeks explains:

- the dot product^a [12]
- vector norms^b [13]

^a<https://www.geeksforgeeks.org/maths/dot-product/>

^b<https://www.geeksforgeeks.org/maths/vector-norms/>

1.3 Matrices and the matrix product

Math for Data Science covers:

- Rank²⁷.
- The inverse²⁸ and invertible matrices²⁹.
- How matrix arithmetic compares with arithmetic on numbers³⁰: products do not commute, a nonzero matrix may have no inverse, and a factor can be cancelled only if it is invertible³¹.
- Diagonal³², symmetric³³ and orthogonal³⁴ matrices, and the fact that an orthogonal matrix preserves length³⁵.

²⁵<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-euclidean-norm>

²⁶<https://morrison-lab.github.io/mds/linear-algebra-vectors.html#exr-inner-product>

²⁷<https://morrison-lab.github.io/mds/linear-algebra-subspaces.html#def-rank>

²⁸<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-matrix-inverse>

²⁹<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-invertible-matrix>

³⁰<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#rem-matrix-vs-scalar-arithmetic>

³¹<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#thm-invertible-cancel>

³²<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-diagonal-matrix>

³³<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-symmetric-matrix>

³⁴<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-orthogonal-matrix>

³⁵<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#thm-orthogonal-norm>

- The trace³⁶ and its cyclic property³⁷.
- The matrix inner product³⁸, computed entry by entry³⁹, and the Frobenius norm⁴⁰. Unsupervised Learning measures the error of a truncated decomposition with the Frobenius norm.

A matrix stores a whole dataset or a whole layer of a model.

Math for Data Science defines the matrix product⁴¹ AB . The product exists only when the inner dimensions agree: A must be $m \times n$ and B must be $n \times p$, and the result is $m \times p$. Entry (i, j) of the product is the inner product (Equation 1) of row i of A with column j of B [14].

Definition 1.1 (The shape rule). Write the two shapes side by side, and read off

$$(m \times n)(n \times p) \rightarrow m \times p \quad (3)$$

The inner pair must match, and the outer pair is the answer's shape.

Example 1.1 (Checking the shapes in the normal equations). Let D be an $n \times p^*$ design matrix and $y \in \mathbb{R}^{n \times 1}$ a response column vector. Use the shape rule on each product in $\hat{\beta} = (D^\top D)^{-1} D^\top y$:

- $D^\top D$ is $(p^* \times n)(n \times p^*)$. The inner pair n, n matches, so the product is $p^* \times p^*$ and can be inverted.
- $D^\top y$ is $(p^* \times n)(n \times 1)$. The inner pair matches, so the product is $p^* \times 1$.
- $(D^\top D)^{-1} D^\top y$ is $(p^* \times p^*)(p^* \times 1)$, which is $p^* \times 1$: one coefficient for each column of D , as it should be.

Reading the product in the other order fails the check. DD would be $(n \times p^*)(n \times p^*)$, and its inner pair p^*, n does not match unless $n = p^*$, so the formula cannot be written that way.

The shape rule checks any expression in these notes quickly. If the shapes do not match, the expression is wrong, and we find that out without computing a single number.

Math for Data Science provides a worked exercise:

- Applying the shape rule⁴²

i Watch

- Sanderson draws the matrix product as a composition^a of two linear maps [10, Ch. 4].
- Starmer explains the matrix algebra of neural networks^b [15].

^a<https://www.youtube.com/watch?v=XkY2DOUCWMU>

^b<https://www.youtube.com/watch?v=ZTt9gsGcdDo>

i Read

GeeksforGeeks explains matrix multiplication^a [16].

^a<https://www.geeksforgeeks.org/maths/multiplying-matrices/>

2 Linear functions

Every linear model in these notes is built out of linear and affine maps. *Math for Data Science* covers their definitions, geometry, and exercises:

³⁶<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-trace>

³⁷<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#thm-trace-cyclic>

³⁸<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-matrix-inner-product>

³⁹<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#thm-matrix-inner-product-entries>

⁴⁰<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-frobenius-norm>

⁴¹<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#def-matrix-mult>

⁴²<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#exr-shape-rule>

- Linear functions⁴³, which send $\vec{0}$ to $\vec{0}$.
- Affine functions⁴⁴ of the form $f(\vec{x}) = W\vec{x} + \vec{b}$, and single-output affine functions⁴⁵ $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$. These notes follow standard machine learning usage and call both **linear**; Linear Regression keeps the same convention and says so at the same point.
- Lines, planes, and hyperplanes⁴⁶, including an interactive plane tilting widget⁴⁷. Linear Regression defines the general hyperplane.

i Watch

Sanderson draws linear transformations as matrices^a [10, Ch. 3].

^a<https://www.youtube.com/watch?v=kYB8IZa5AuE>

i Read

GeeksforGeeks explains linear transformations^a [17].

^a<https://www.geeksforgeeks.org/maths/linear-transformation/>

Worked exercises in Math for Data Science:

- Writing affine functions in matrix form⁴⁸
- Classifying linear, affine, and nonlinear functions⁴⁹

3 Derivatives and gradients

Fitting a model means adjusting parameters until some measure of its error is small, guided by the gradient. The Gradient Descent chapter uses it. *Math for Data Science* covers calculus foundations and gradient checks:

- The one-variable derivative⁵⁰ and rules for computing derivatives⁵¹.
- Interactive widget: Linear Approximation Explorer⁵².
- The vector derivative and gradient⁵³ $\nabla_{\vec{w}} f(\vec{w})$.
- Checking a gradient by its shape⁵⁴: the shape rule applied to calculus.
- Interactive widget: Gradient Explorer⁵⁵.
- The matrix derivative⁵⁶ of a scalar-valued function of a matrix.

Worked exercises in Math for Data Science:

- Finding flat points and critical points⁵⁷
- Computing partial derivatives and gradient vectors by hand⁵⁸
- Finding the gradient of a linear function⁵⁹

⁴³<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#def-linear-map>

⁴⁴<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#def-affine-map>

⁴⁵<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#eq-linear-scalar>

⁴⁶<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#def-hyperplane>

⁴⁷<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#fig-tilt-a-plane>

⁴⁸<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#exr-write-in-matrix-form>

⁴⁹<https://morrison-lab.github.io/mds/linear-algebra-matrices.html#exr-which-are-linear>

⁵⁰<https://morrison-lab.github.io/mds/calculus-derivatives.html#def-differentiable>

⁵¹<https://morrison-lab.github.io/mds/calculus-derivatives.html#sec-derivatives>

⁵²<https://morrison-lab.github.io/mds/calculus-derivatives.html#fig-linear-approx>

⁵³<https://morrison-lab.github.io/mds/vector-calculus.html#def-vector-derivative>

⁵⁴<https://morrison-lab.github.io/mds/vector-calculus.html#sec-gradient-shape>

⁵⁵<https://morrison-lab.github.io/mds/vector-calculus.html#fig-gradient-explorer>

⁵⁶<https://morrison-lab.github.io/mds/vector-calculus.html#def-matrix-derivative>

⁵⁷<https://morrison-lab.github.io/mds/calculus-derivatives.html#exr-flat-point>

⁵⁸<https://morrison-lab.github.io/mds/vector-calculus.html#exr-partials-by-hand>

⁵⁹<https://morrison-lab.github.io/mds/vector-calculus.html#exr-gradient-of-linear>

i Watch

- Khan Academy defines the gradient^a and draws it as a vector field^b [18].
- Hutchinson’s Gradients Refresher^c (17 min) covers gradients and partial derivatives [11].
- Starmer explains the chain rule^d [19].

^ahttps://www.youtube.com/watch?v=U7HQ_G_N6vo

^bhttps://www.youtube.com/watch?v=U7HQ_G_N6vo&t=235s

^chttps://facultyweb.cs.wvu.edu/~hutchib2/video_lectures/data371/#gradients

^d<https://www.youtube.com/watch?v=wl1myxrtQHQ>

i Read

GeeksforGeeks explains:

- partial derivatives^a [20]
- the gradient^b [21]

^a<https://www.geeksforgeeks.org/maths/partial-derivative/>

^b<https://www.geeksforgeeks.org/data-science/gradient/>

4 Decompositions

Math for Data Science covers:

- Eigenvalues and eigenvectors⁶⁰, the spectral theorem⁶¹ and the eigendecomposition⁶² of a symmetric matrix. Linear Regression uses the eigendecomposition of $X^T X$ to build a usable solution when $X^T X$ is not invertible.
- The existence of the singular value decomposition (SVD)⁶³, the SVD⁶⁴ itself, and how an SVD of X gives an eigendecomposition of $X^T X$ ⁶⁵. Unsupervised Learning truncates the SVD to get principal component analysis.

Math for Data Science writes the SVD as $\mathbf{UDV}^T$. These notes write USV^T instead, because p is the number of features throughout these notes. Both *Math for Data Science* and these notes write an eigendecomposition as $Q\Lambda Q^T$ (or $\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$), where Q is the orthogonal matrix of eigenvectors [22].

i Watch

- Sanderson’s video on eigenvectors and eigenvalues^a [10, Ch. 14], and Brunton’s overview of the SVD^b and mathematical overview of the SVD^c [23].
- Avati’s linear-algebra review covers eigenvalues, from 11:15^d, and quadratic forms, from 23:10^e [24], Lecture 2.

^a<https://www.youtube.com/watch?v=PFDu9oVAE-g>

^b<https://www.youtube.com/watch?v=gXbThCXjZFM>

^c<https://www.youtube.com/watch?v=nbBvuunVfco&t=175s>

^d<https://www.youtube.com/watch?v=b0HvwszmqcQ&t=675s>

^e<https://www.youtube.com/watch?v=b0HvwszmqcQ&t=1390s>

⁶⁰<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-eigenvalue>

⁶¹<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#thm-spectral>

⁶²<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-eigendecomposition>

⁶³<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#thm-svd>

⁶⁴<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-svd>

⁶⁵<https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#thm-svd-evd>

i Read

GeeksforGeeks explains:

- eigenvalues and eigenvectors^a [25]
- the singular value decomposition^b [26]

^a<https://www.geeksforgeeks.org/engineering-mathematics/eigen-values/>

^b<https://www.geeksforgeeks.org/data-science/singular-value-decomposition-svd/>

5 Further reading

- Goodfellow et al. [1], Chapters 2 and 4, on linear algebra and optimization
- Boyd and Vandenberghe [3], Appendix A
- Starmer [27], on matrix algebra for neural networks

References

- [1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016. Available: <https://www.deeplearningbook.org/>
- [2] C. M. Bishop, *Pattern recognition and machine learning*. in Information science and statistics. New York: Springer, 2006.
- [3] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441⁶⁶.
- [4] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, 2nd ed. Springer, 2009. Available: <https://hastie.su.domains/ElemStatLearn/>
- [5] H. R. Varian, “Big data: New tricks for econometrics,” *Journal of Economic Perspectives*, vol. 28, no. 2, pp. 3–28, 2014, doi: 10.1257/jep.28.2.3⁶⁷.
- [6] B. Efron and T. Hastie, *Computer age statistical inference: Algorithms, evidence, and data science*. Cambridge: Cambridge University Press, 2016. doi: 10.1017/CBO9781316576533⁶⁸.
- [7] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge: Cambridge University Press, 1996. doi: 10.1017/CBO9780511812651⁶⁹.
- [8] L. Breiman, “Statistical modeling: The two cultures,” *Statistical Science*, vol. 16, no. 3, 2001, doi: 10.1214/ss/1009213726⁷⁰.
- [9] Wikipedia contributors, “Dot product — Wikipedia, the free encyclopedia.” 2026. Available: https://en.wikipedia.org/wiki/Dot_product
- [10] G. Sanderson, “Essence of linear algebra.” 3Blue1Brown, video series on YouTube, 2016. Available: https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFItgF8hE_ab
- [11] B. Hutchinson, “DATA 471/571 (machine learning) and CSCI 481/581 (deep learning) video lectures.” Western Washington University. Accessed: Sep. 28, 2026. [Online]. Available: https://facultyweb.cs.wvu.edu/~hutchib2/video_lectures/data371/
- [12] GeeksforGeeks, “Dot product of two vector.” GeeksforGeeks, tutorial article, 2026. Available: <https://www.geeksforgeeks.org/maths/dot-product/>
- [13] GeeksforGeeks, “Vector norms.” GeeksforGeeks, tutorial article, 2026. Available: <https://www.geeksforgeeks.org/maths/vector-norms/>
- [14] Wikipedia contributors, “Matrix multiplication — Wikipedia, the free encyclopedia.” 2026. Available: https://en.wikipedia.org/wiki/Matrix_multiplication
- [15] J. Starmer, “Essential matrix algebra for neural networks, clearly explained.” StatQuest, video on YouTube, 2023. Available: <https://www.youtube.com/watch?v=ZTt9gsGcdDo>

⁶⁶<https://doi.org/10.1017/CBO9780511804441>

⁶⁷<https://doi.org/10.1257/jep.28.2.3>

⁶⁸<https://doi.org/10.1017/CBO9781316576533>

⁶⁹<https://doi.org/10.1017/CBO9780511812651>

⁷⁰<https://doi.org/10.1214/ss/1009213726>

- [16] GeeksforGeeks, “Multiplying matrices.” GeeksforGeeks, tutorial article, 2026. Available: <https://www.geeksforgeeks.org/maths/multiplying-matrices/>
- [17] GeeksforGeeks, “Linear transformation.” GeeksforGeeks, tutorial article, 2026. Available: <https://www.geeksforgeeks.org/maths/linear-transformation/>
- [18] Khan Academy, “Gradient 1: Partial derivatives, gradient, divergence, curl.” Video on YouTube, 2008. Available: https://www.youtube.com/watch?v=U7HQ_G_N6vo
- [19] J. Starmer, “The chain rule, clearly explained.” StatQuest, video on YouTube, 2020. Available: <https://www.youtube.com/watch?v=w1myxrtQHQ>
- [20] GeeksforGeeks, “Partial derivative.” GeeksforGeeks, tutorial article, 2026. Available: <https://www.geeksforgeeks.org/maths/partial-derivative/>
- [21] GeeksforGeeks, “Gradient.” GeeksforGeeks, tutorial article, 2025. Available: <https://www.geeksforgeeks.org/data-science/gradient/>
- [22] Wikipedia contributors, “Eigendecomposition of a matrix — Wikipedia, the free encyclopedia.” 2026. Available: https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix
- [23] S. L. Brunton, “Singular value decomposition (SVD).” Video series on YouTube, 2020. Available: <https://www.youtube.com/watch?v=gXbThCXjZFM>
- [24] A. Avati, “CS229: Machine learning, summer 2019.” Stanford Online, lecture videos on YouTube, 2019. Available: https://www.youtube.com/playlist?list=PLoROMvody4rNH7qL6-efu_q2_bPuy0adh
- [25] GeeksforGeeks, “Eigenvalues and eigenvectors.” GeeksforGeeks, tutorial article, 2026. Available: <https://www.geeksforgeeks.org/engineering-mathematics/eigen-values/>
- [26] GeeksforGeeks, “Singular value decomposition (SVD).” GeeksforGeeks, tutorial article, 2025. Available: <https://www.geeksforgeeks.org/data-science/singular-value-decomposition-svd/>
- [27] J. Starmer, *The StatQuest illustrated guide to neural networks and AI: With hands-on examples in PyTorch*. Independently published, 2025. Available: <https://statquest.org/statquest-store/>