

Core Math for ML/Stats (PCA, SPCA, PLS, Tensor Algebra)

Md Shahriar Forhad (<https://github.com/Shahriar88>)

Contents

1 Notation & Symbols (with Examples)	1
2 Einstein Summation Notation: Explanation and Examples	2
2.1 Vector Example: Dot Product	2
2.2 Matrix–Vector Multiplication	2
2.3 Matrix–Matrix Multiplication	3
2.4 Kronecker Delta	3
2.5 Tensor Contraction Example	3
2.6 Mixed Upper and Lower Indices	4
2.7 Levi–Civita Symbol and the Cross Product	4
2.8 Example from Machine Learning: Linear Layer	5
2.9 Summary of Rules	5
3 Core Statistical Building Blocks	5
3.1 Mean and variance	5
3.2 Gaussian density	5
3.3 Bayes' rule	6
4 Supervised Learning: Models, Losses, Optimizers	6
4.1 Linear regression (OLS / Ridge / Lasso)	6
4.2 Logistic regression (binary) & Softmax (multi-class)	6
4.3 Gradient-based optimization	6
4.4 Maximum Likelihood & Kullback–Leibler Divergence	6
4.5 Loss summary	7
5 SVM and Neural Networks (Quick)	7
6 Bias–Variance and Cross-Validation	7
7 PCA, Rank, Eigenvalues: From Matrices to Tensors	7
7.1 Rank	7
7.2 Eigenvalues/eigenvectors	7
7.3 PCA via covariance and SVD	8
7.4 Does PCA depend only on X ?	8
8 Tensor Algebra and Decompositions	8
8.1 Tensor regression (example)	8
8.2 HOSVD / Tucker (Tensor PCA)	8
8.3 CP/PARAFAC	8
8.4 Where the products appear	8

9 How-To: Khatri–Rao and $\mathbf{1}_k$	9
9.1 Khatri–Rao step-by-step (column-wise Kronecker)	9
9.2 What $\mathbf{1}_k$ does (not cumulative)	9
10 Supervised PCA (SPCA) and Partial Least Squares (PLS)	9
10.1 Supervised PCA (SPCA)	9
10.2 Partial Least Squares (PLS)	9
10.3 PCA vs SPCA vs PLS (at a glance)	10
11 Task Map: Where Each Equation is Used	10
12 Identities Reference (handy)	10
13 Mini Worked Examples (Hand-Check Size)	10
13.1 Khatri–Rao and $\mathbf{1}_k$ aggregation	10
13.2 PCA via SVD	11
13.3 SPCA toy intuition	11
14 Appendix: Python Snippets (shapes-first)	11

List of Figures

List of Tables

1 Frequently used operators with concrete examples.	1
2 Common losses and where they're used.	7
3 Supervised vs. unsupervised component methods.	10
4 Quick map from math objects to ML tasks.	10
5 Frequently used algebraic identities.	10

1 Notation & Symbols (with Examples)

Unless stated, vectors are bold lowercase (\mathbf{x}), matrices bold uppercase (\mathbf{X}), tensors calligraphic (\mathcal{X}).

Linear algebra & tensor operators

Symbol	Name	Description & Example
$\text{vec}(A)$	Vectorization	Stacks entries of A column-wise into a single column vector. Example: $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \Rightarrow \text{vec}(A) = [1, 3, 2, 4]^\top$.
\odot	Khatri–Rao product	Column-wise Kronecker: if $A \in \mathbb{R}^{m \times k}, B \in \mathbb{R}^{n \times k}$, then $A \odot B \in \mathbb{R}^{mn \times k}$ with $(A \odot B)_{:,j} = A_{:,j} \otimes B_{:,j}$. Example with $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}: A \odot B = \begin{bmatrix} 5 & 12 \\ 7 & 16 \\ 15 & 24 \\ 21 & 32 \end{bmatrix}$.
\otimes	Kronecker product	Full expansion: $(A \otimes B) = (a_{ij}B)_{ij}$. Example: $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix}$, then $A \otimes B = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}$.
\circ	Hadamard product	Elementwise product: $(A \circ B)_{ij} = A_{ij}B_{ij}$. Example: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \circ \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 10 & 40 \\ 90 & 160 \end{bmatrix}$.
$\langle A, B \rangle$	Inner product	Sum of elementwise products: $\langle A, B \rangle = \sum_{ij} A_{ij}B_{ij} = \text{tr}(A^\top B)$. Example with A, B above: $\langle A, B \rangle = 70$.
$\ \cdot\ _F$	Frobenius norm	$\ A\ _F = \sqrt{\sum_{ij} A_{ij}^2} = \sqrt{\langle A, A \rangle}$. Example: $\left\ \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \right\ _F = \sqrt{30}$.
$\mathbf{1}_k$	Vector of ones	$\mathbf{1}_k = [1, \dots, 1]^\top \in \mathbb{R}^k$. Used to sum/aggregate columns: $M\mathbf{1}_k$ yields row sums of M .

Table 1: Frequently used operators with concrete examples.

Useful identities. $\text{vec}(AXB) = (B^\top \otimes A)\text{vec}(X)$, $(A \odot B)^\top (A \odot B) = (A^\top A) \circ (B^\top B)$.

Extra numeric examples (quick check)

- **Inner/Frobenius:** With A, B as above, $\langle A, B \rangle = 70$, $\|A\|_F = \sqrt{30}$.
- **Aggregation with $\mathbf{1}_k$:** If $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, $M\mathbf{1}_3 = \begin{bmatrix} 6 \\ 15 \end{bmatrix}$ (row sums). Not a cumulative sum.

2 Einstein Summation Notation: Explanation and Examples

Einstein summation notation is a compact way to write sums over indices. It is especially useful in tensor algebra, differential geometry, and modern machine learning.

Main Rule

If an index appears **exactly twice** in a single term (once as a subscript or superscript and once again), then it is **implicitly summed** over its range.

For example,

$$a_i b_i$$

is shorthand for

$$\sum_i a_i b_i.$$

There is no need to write the summation symbol \sum explicitly.

2.1 Vector Example: Dot Product

Consider two vectors \mathbf{a} and \mathbf{b} in \mathbb{R}^n with components a_i and b_i .

Standard notation

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i.$$

Einstein notation

$$\mathbf{a} \cdot \mathbf{b} = a_i b_i.$$

Here the index i is repeated, so it is understood that we sum over $i = 1, \dots, n$.

2.2 Matrix–Vector Multiplication

Let A be an $m \times n$ matrix with entries A_{ij} , and let \mathbf{x} be a vector in \mathbb{R}^n with components x_j . The result $\mathbf{y} = A\mathbf{x}$ is a vector in \mathbb{R}^m with components y_i .

Standard notation

$$y_i = \sum_{j=1}^n A_{ij} x_j.$$

Einstein notation

$$y_i = A_{ij} x_j.$$

Explanation:

- j is a repeated index \Rightarrow sum over j .
- i appears only once $\Rightarrow y_i$ is the i -th component of the result.

2.3 Matrix–Matrix Multiplication

Let A be an $m \times p$ matrix with entries A_{ik} and B be a $p \times n$ matrix with entries B_{kj} . Their product $C = AB$ is an $m \times n$ matrix with entries C_{ij} .

Standard notation

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

Einstein notation

$$C_{ij} = A_{ik}B_{kj}.$$

Explanation:

- k is repeated \Rightarrow sum over k .
- i and j are free indices $\Rightarrow C$ has indices (i, j) .

2.4 Kronecker Delta

The Kronecker delta δ_{ij} is defined by

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

Identity action on a vector

For a vector x_j , we have

$$\delta_{ij}x_j = x_i.$$

Standard notation

$$\delta_{ij}x_j = \sum_j \delta_{ij}x_j = x_i.$$

Here the sum over j “picks out” the i -th component.

Einstein notation

$$\delta_{ij}x_j = x_i.$$

2.5 Tensor Contraction Example

Consider a rank-3 tensor T_{ijk} and vectors v_j and w_k . Define

$$u_i = T_{ijk}v_jw_k.$$

Standard notation

$$u_i = \sum_j \sum_k T_{ijk}v_jw_k.$$

Einstein notation

$$u_i = T_{ijk}v_jw_k.$$

Explanation:

- j is repeated \Rightarrow sum over j .
- k is repeated \Rightarrow sum over k .
- i is free \Rightarrow the result u_i is a vector.

2.6 Mixed Upper and Lower Indices

In many physics applications (especially relativity), one distinguishes between *covariant* (lower) and *contravariant* (upper) indices.

Scalar from a contravariant and a covariant vector

Let A^i be contravariant components and B_i covariant components. Then

$$A^i B_i = \sum_i A^i B_i.$$

Einstein notation:

$$A^i B_i.$$

Since i appears once up and once down, it is summed over.

Linear map acting on a vector

Let $T^i{}_j$ be a (1,1)-tensor (a linear map) and v^j a vector. Define

$$u^i = T^i{}_j v^j.$$

Standard notation:

$$u^i = \sum_j T^i{}_j v^j.$$

Einstein notation:

$$u^i = T^i{}_j v^j.$$

Here j is repeated and thus summed; i is free.

2.7 Levi–Civita Symbol and the Cross Product

The Levi–Civita symbol ϵ_{ijk} in three dimensions is defined by

$$\epsilon_{ijk} = \begin{cases} +1, & \text{if } (i, j, k) \text{ is an even permutation of } (1, 2, 3), \\ -1, & \text{if } (i, j, k) \text{ is an odd permutation of } (1, 2, 3), \\ 0, & \text{if any two indices are equal.} \end{cases}$$

Given vectors a_j and b_k , the cross product $\mathbf{a} \times \mathbf{b}$ has components

$$(\mathbf{a} \times \mathbf{b})_i = \epsilon_{ijk} a_j b_k.$$

Standard notation

$$(\mathbf{a} \times \mathbf{b})_i = \sum_j \sum_k \epsilon_{ijk} a_j b_k.$$

Einstein notation

$$(\mathbf{a} \times \mathbf{b})_i = \epsilon_{ijk} a_j b_k.$$

Again, j and k are repeated and therefore summed.

2.8 Example from Machine Learning: Linear Layer

Consider a linear layer (fully connected layer) in machine learning:

$$y_i = W_{ij}x_j + b_i,$$

where W_{ij} is a weight matrix, x_j is the input vector, and b_i is the bias vector.

Forward pass

$$y_i = W_{ij}x_j + b_i.$$

Here j is summed over, i is free.

Gradient with respect to weights

Let L be a loss function depending on y_i . The gradient of L with respect to W_{ij} is

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial y_i} x_j.$$

Note:

- i and j both appear once on the left-hand side, so the gradient is a matrix with indices (i, j) .
- On the right-hand side, there is *no* repeated index, so there is no implied sum here: this formula gives each component of the gradient.

2.9 Summary of Rules

- If an index appears exactly twice in a term, it is summed over (dummy index).
- Indices that appear only once in a term are free indices and label the components of the result.
- Expressions should not contain an index repeated more than twice in a single term; such expressions are considered ambiguous or invalid in Einstein notation.

3 Core Statistical Building Blocks

3.1 Mean and variance

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i, \quad \text{Var}(X) = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2.$$

Use: standardization, bias-variance analysis, Gaussian likelihoods.

3.2 Gaussian density

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right).$$

Use: regression noise models, generative modeling, conjugacy.

3.3 Bayes' rule

$$p(y | x) = \frac{p(x | y)p(y)}{p(x)}.$$

Use: naive Bayes, Bayesian regression, probabilistic inference.

4 Supervised Learning: Models, Losses, Optimizers

4.1 Linear regression (OLS / Ridge / Lasso)

Model: $\mathbf{y} = \mathbf{X}\beta + \varepsilon$, loss $L(\beta) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\beta\|_2^2$. Closed-form OLS: $\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ (if full rank).

$$\text{Ridge: } L = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2, \quad \text{Lasso: } L = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1.$$

Task: regression. **Notes:** multicollinearity; sparsity.

4.2 Logistic regression (binary) & Softmax (multi-class)

Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$, probability $\hat{p}(y=1 | x) = \sigma(x^\top w + b)$.

Cross-entropy loss:

$$L = -\frac{1}{n} \sum_{i=1}^n [y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)].$$

Softmax: $\text{softmax}(z)_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$, multi-class loss $L = -\frac{1}{n} \sum_i \log \hat{p}_{i,y_i}$. **Task:** classification.

4.3 Gradient-based optimization

Update: $\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t)$.

Variants: SGD, momentum, Adam, RMSProp.

4.4 Maximum Likelihood & Kullback–Leibler Divergence

$$\hat{\theta} = \arg \max_{\theta} \sum_{i=1}^n \log p(x_i | \theta), \quad D_{\text{KL}}(p \| q) = \sum_x p(x) \log \frac{p(x)}{q(x)}.$$

Use: derive losses; variational inference (VI) / variational autoencoders (VAEs).

The Kullback–Leibler divergence measures how one probability distribution diverges from another. For distributions p and q , it is defined as:

$$D_{\text{KL}}(p \| q) = \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{q(x)} \right].$$

The double vertical bar “ $\|$ ” denotes divergence from ... to ..., that is, it separates the two probability distributions p and q .

In Variational Autoencoders (VAEs). A key quantity in VAEs is:

$$D_{\text{KL}}(q(z|x) \| p(z)),$$

read as “the Kullback–Leibler divergence of $q(z|x)$ relative to $p(z)$ ”.

It measures how much the encoder’s posterior distribution $q(z|x)$ differs from the prior $p(z)$.

- $q(z|x)$: the encoders posterior (distribution of latent variables given an input x);
- $p(z)$: the prior distribution, usually a standard normal $\mathcal{N}(0, I)$.

Minimizing this divergence encourages the latent space to follow the standard normal prior, resulting in a smoother, more continuous latent representation.

Intuition.

$D_{KL}(q(z|x) \parallel p(z))$ = how much information is lost when $p(z)$ is used to approximate $q(z|x)$.

In words:

Minimizing D_{KL} makes the encoders latent distribution $q(z|x)$ behave more like the prior $p(z)$.

4.5 Loss summary

Loss	Formula	Task	Notes
MSE	$\frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$	Regression	Smooth, sensitive to outliers
MAE	$\frac{1}{n} \sum_i y_i - \hat{y}_i $	Regression	Robust to outliers
Cross-entropy	$-\sum_i y_i \log \hat{y}_i$	Classification	Likelihood-based
Hinge	$\max(0, 1 - y_i \hat{y}_i)$	Margin cls. (SVM)	Maximizes margin
Huber	piecewise quad./linear	Regression	Robust & smooth

Table 2: Common losses and where they're used.

5 SVM and Neural Networks (Quick)

SVM (soft-margin):

$$\min_{w,b} \frac{1}{2} \|w\|_2^2 + C \sum_i \max(0, 1 - y_i(w^\top x_i + b)).$$

Neural layer: $h^{(\ell)} = f(W^{(\ell)} h^{(\ell-1)} + b^{(\ell)})$; **Use:** classification or regression depending on final activation.

6 Bias–Variance and Cross-Validation

$$\mathbb{E}[(\hat{f}(x) - f(x))^2] = \underbrace{(\text{Bias})^2}_{\text{systematic error}} + \underbrace{\text{Var}}_{\text{sensitivity}} + \text{Noise}.$$

K -fold CV error = $\frac{1}{K} \sum_{k=1}^K L_k$. **Use:** model selection & hyperparameters.

7 PCA, Rank, Eigenvalues: From Matrices to Tensors

7.1 Rank

$\text{rank}(A) = \#$ linearly independent columns (or rows). Determines identifiability of OLS; caps $\#$ nonzero PCs.

7.2 Eigenvalues/eigenvectors

$Av = \lambda v$. In PCA, eigenvectors of covariance give principal directions and eigenvalues give explained variance.

7.3 PCA via covariance and SVD

With centered $X \in \mathbb{R}^{n \times d}$:

$$\Sigma = \frac{1}{n} X^\top X, \quad \Sigma v_k = \lambda_k v_k.$$

Equivalently SVD: $X = U\Sigma V^\top$; columns of V are PCs and Σ^2/n are variances.

Short equivalence note (PCA \leftrightarrow SVD) $X^\top X = V\Sigma^2 V^\top$ so eigenvectors of the covariance are the right singular vectors of X , and eigenvalues are σ_i^2 .

7.4 Does PCA depend only on X ?

Yes (standard PCA is unsupervised; depends solely on X via its covariance). Supervised variants below incorporate y .

8 Tensor Algebra and Decompositions

8.1 Tensor regression (example)

Scalar response:

$$y_i = \alpha + \left\langle \text{vec}(\tilde{\mathbf{B}}), \text{vec}(\tilde{\mathbf{S}}_i) \right\rangle + \sigma \epsilon_i = \alpha + \left\langle (\tilde{B}_3 \odot \tilde{B}_2 \odot \tilde{B}_1) \mathbf{1}_k, \text{vec}(\tilde{\mathbf{S}}_i) \right\rangle + \sigma \epsilon_i.$$

$\tilde{B}_1, \tilde{B}_2, \tilde{B}_3$ are factor matrices; \odot aligns columns across modes; $\mathbf{1}_k$ aggregates k rank-1 components.

8.2 HOSVD / Tucker (Tensor PCA)

$$\mathcal{X} \approx \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3, \quad X_{(n)} = U_n S_n \left(\bigotimes_{m \neq n} U_m \right)^\top.$$

Kronecker (\otimes) ties other-mode bases in matricized form.

8.3 CP/PARAFAC

$$\mathcal{X} \approx \sum_{r=1}^R a_r \circ b_r \circ c_r, \quad \text{vec}(\mathcal{X}) \approx (C \odot B \odot A) \mathbf{1}_R.$$

Use: compact multiway factorization; \odot is the natural column-aligned expansion.

8.4 Where the products appear

- **Kronecker \otimes :** separable covariances ($\Sigma_t \otimes \Sigma_s$), matrix-variate normals, vectorization identities $\text{vec}(AXB) = (B^\top \otimes A)\text{vec}(X)$.
- **Khatri–Rao \odot :** CP/Tucker updates, tensor regression design matrices, feature interaction with aligned ranks.
- **Hadamard \circ :** elementwise weighting, covariance identities, attention masks.

9 How-To: Khatri–Rao and $\mathbf{1}_k$

9.1 Khatri–Rao step-by-step (column-wise Kronecker)

Let $A = [a_1 \ a_2 \ \dots \ a_K] \in \mathbb{R}^{I \times K}$, $B = [b_1 \ b_2 \ \dots \ b_K] \in \mathbb{R}^{J \times K}$.

$$A \odot B = [a_1 \otimes b_1 \ a_2 \otimes b_2 \ \dots \ a_K \otimes b_K] \in \mathbb{R}^{(IJ) \times K}.$$

Example: $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$:

$$(A \odot B)_{:,1} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \otimes \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \\ 15 \\ 21 \end{bmatrix}, \quad (A \odot B)_{:,2} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \otimes \begin{bmatrix} 6 \\ 8 \end{bmatrix} = \begin{bmatrix} 12 \\ 16 \\ 24 \\ 32 \end{bmatrix}.$$

9.2 What $\mathbf{1}_k$ does (not cumulative)

$\mathbf{1}_k$ is a column of ones used to *sum* components, not to produce cumulative sums. If $Z = [z_1 \ \dots \ z_K]$ collects K rank-1 terms, then $Z\mathbf{1}_k = \sum_{j=1}^K z_j$ (aggregation). For tensors, $(C \odot B \odot A)\mathbf{1}_K$ sums rank-1 parts into a single vector.

10 Supervised PCA (SPCA) and Partial Least Squares (PLS)

10.1 Supervised PCA (SPCA)

Find directions v in X that correlate with y :

$$v = \arg \max_{\|v\|=1} \text{Cov}^2(Xv, y).$$

A common formulation solves the eigenproblem on

$$C = X^\top yy^\top X \quad (\text{or } X^\top YY^\top X \text{ for multi-response}), \quad Cv = \lambda v.$$

Use: supervised dimensionality reduction when only components predictive of y are desired. For classification, let Y be one-hot labels.

10.2 Partial Least Squares (PLS)

Find paired latent scores (t, u) with $t = Xw$, $u = yq$ maximizing covariance:

$$\max_{w,q} \text{Cov}^2(Xw, yq), \quad \text{with orthogonality across components.}$$

One PLS1 iteration (single response):

$$w \propto X^\top y, \quad t = Xw, \quad q = \frac{y^\top t}{t^\top t}, \quad p = \frac{X^\top t}{t^\top t}, \quad X \leftarrow X - tp^\top, \quad y \leftarrow y - tq.$$

Final regression: $\hat{y} = XW(P^\top W)^{-1}Q^\top$.

Use: high-dimensional regression where predictors are collinear; chemometrics, genomics.

Method	Uses y ?	Objective	Output	Typical Task
PCA	No	$\max_v \text{Var}(Xv)$	Unsupervised PCs	Compression/denoising
SPCA	Yes	$\max_v \text{Cov}^2(Xv, y)$	Supervised PCs	Predictive features
PLS	Yes	$\max_{w,q} \text{Cov}^2(Xw, yq)$	Latent scores (t, u)	Regression

Table 3: Supervised vs. unsupervised component methods.

Equation/Concept	Primary Use	Notes
Linear regression (OLS/Ridge/Lasso)	Regression	L2 for stability, L1 for sparsity
Logistic/Softmax + Cross-Entropy	Classification	Probabilistic interpretation
MLE / Log-likelihood	Estimation	Derives many learning objectives
KL divergence	Regularization / VI	Distance between distributions
PCA (SVD)	Unsupervised DR	Depends only on X
SPCA / PLS	Supervised DR	Aligns components with y
SVM (hinge)	Classification	Margin maximization
Gradient descent/Adam	Optimization	Ubiquitous in deep learning
Kronecker \otimes	Multiway lin. maps, covariances	Separable structures
Khatri–Rao \odot	Tensor decompositions	Column-aligned Kronecker
Hadamard \circ	Elementwise modeling	Masks/weights, covariance algebra

Table 4: Quick map from math objects to ML tasks.

Identity	Comment
$\text{vec}(AXB) = (B^\top \otimes A)\text{vec}(X)$	Vectorization + Kronecker
$(A \odot B)^\top (A \odot B) = (A^\top A) \circ (B^\top B)$	Appears in CP/ALS updates
$\ A\ _F^2 = \langle A, A \rangle = \text{tr}(A^\top A)$	Frobenius norm
$X^\top X = V\Sigma^2 V^\top$ (SVD of X)	PCASVD equivalence

Table 5: Frequently used algebraic identities.

10.3 PCA vs SPCA vs PLS (at a glance)

11 Task Map: Where Each Equation is Used

12 Identities Reference (handy)

13 Mini Worked Examples (Hand-Check Size)

13.1 Khatri–Rao and $\mathbf{1}_k$ aggregation

Let $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$. Then $A \odot B = \begin{bmatrix} 5 & 12 \\ 7 & 16 \\ 15 & 24 \\ 21 & 32 \end{bmatrix}$. If each column is a rank-1 component, multiplying by $\mathbf{1}_2$ sums the two components: $(A \odot B)\mathbf{1}_2 = \begin{bmatrix} 17 \\ 23 \\ 39 \\ 53 \end{bmatrix}$.

13.2 PCA via SVD

Centered $X = U\Sigma V^\top$. The top k principal directions are columns of $V_{:,1\dots k}$; the projected scores are $Z = X V_{:,1\dots k} = U_{:,1\dots k} \Sigma_{1\dots k}$.

13.3 SPCA toy intuition

If y varies mainly with a subset of features, $X^\top y y^\top X$ emphasizes those directions even when overall variance in X is driven by others.

14 Appendix: Python Snippets (shapes-first)

The following compact demo reproduces our sessions computations (rank, PCA, SPCA, PLS, Kronecker, Khatri–Rao, Hadamard, plus vec/inner/Frobenius). Center X (and y) for PCA/SPCA/PLS.

```
import numpy as np
from numpy.linalg import matrix_rank, lstsq, norm
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import PLSRegression

# OPTIONAL: SciPy for KhatriRao (fallback to manual if unavailable)
try:
    from scipy.linalg import khatri_rao as scipy_khatri_rao
    HAS SCIPY = True
except Exception:
    HAS SCIPY = False

# ----- Toy data -----
X = np.array([
    [0.0, 1.0, 2.0],
    [1.0, 2.0, 3.0],
    [2.0, 3.0, 4.0],
    [3.0, 4.0, 5.0],
])
y = np.array([1.0, 3.0, 5.0, 7.0])

A = np.array([[1, 2],
              [3, 4]])
B = np.array([[5, 6],
              [7, 8]])

# ----- Rank -----
print("rank(A) =", matrix_rank(A), " | rank(X) =", matrix_rank(X))

# ----- PCA (unsupervised; depends only on X) -----
Xc = X - X.mean(axis=0, keepdims=True)
pca = PCA(n_components=2)
Z_pca = pca.fit_transform(Xc) # scores (n x k)
V_pca = pca.components_.T # loadings (d x k)
print("\nPCA loadings V:\n", V_pca)
print("PCA scores Z:\n", Z_pca)
print("PCA explained variances:", pca.explained_variance_)

# ----- SPCA (simple eigen formulation) -----
yc = y - y.mean()
C_spca = Xc.T @ np.outer(yc, yc) @ Xc # d x d
```

```

evals, evecs = np.linalg.eigh(C_spca) # symmetric eig
idx = np.argsort(evals)[::-1]
V_spca = evecs[:, idx[:2]]
Z_spca = Xc @ V_spca
print("\nSPCA directions:\n", V_spca)
print("SPCA scores:\n", Z_spca)

# ----- PLS (supervised; maximizes covariance with y) -----
pls = PLSRegression(n_components=2)
T_pls, U_pls = pls.fit_transform(Xc, yc)
print("\nPLS X-scores T:\n", T_pls)
print("PLS Y-scores U:\n", U_pls)
print("PLS X-weights W:\n", pls.x_weights_)

# ----- Least Squares (OLS) -----
beta, residuals, rank, svals = lstsq(X, y, rcond=None)
print("\nOLS beta:\n", beta)
print("OLS residual sum of squares:", residuals.sum() if residuals.size else 0.0)

# ----- Kronecker, KhatriRao, Hadamard -----
kron_AB = np.kron(A, B)
print("\nKronecker A B:\n", kron_AB)

def khatri_rao(A, B):
    m, k1 = A.shape
    n, k2 = B.shape
    assert k1 == k2
    out = np.zeros((m*n, k1), dtype=np.result_type(A, B))
    for k in range(k1):
        out[:, k] = np.kron(A[:, k], B[:, k])
    return out

A2 = np.array([[1, 2],
              [3, 4]])
B2 = np.array([[5, 6],
              [7, 8]])
KR = scipy_khatri_rao(A2, B2) if HAS SCIPY else khatri_rao(A2, B2)
print("\nKhatriRao A B:\n", KR)

H = A * B
print("\nHadamard A B:\n", H)

# ----- vec, inner product, Frobenius, ones -----
vecA_col_major = A.reshape(-1, order='F') # vec in column-major sense
inner_AB = np.tensordot(A, B, axes=2)
froA = norm(A, 'fro')
ones2 = np.ones((2, 1))
print("\nvec(A) (col-major):", vecA_col_major)
print("A,B =", inner_AB, " == tr(A^T B)")
print("||A||_F =", froA)
print("1_k (k=2):\n", ones2)

```

Index

- activation function, 7
- Adam, 6
- Bayes' rule, 6
- Bayesian regression, 6
- bias–variance decomposition, 5, 7
- chemometrics, 9
- classification, 6
- conjugacy, 5
- contravariant, 4
- covariance matrix, 8
- covariant, 4
- CP decomposition, 8
- cross-entropy, 6, 7
- cross-validation, 7
- not* cumulative, 2
- eigenvalues, 7
- eigenvectors, 7
- Einstein summation, 2
- examples, 10
 - numeric, 2
- Frobenius norm, 1
- full rank, 6
- Gaussian
 - density, 5
 - likelihood, 5
- generative modeling, 5
- genomics, 9
- gradient descent, 6
- Gram matrix, 1
- Hadamard identity, 1
- Hadamard product, 1, 8
- hinge loss, 7
- HOSVD, 8
- Huber loss, 7
- hyperparameter tuning, 7
- identities, 10
- inference, 6
- inner product, 1
- Khatri–Rao product, 1, 8–10
- KL divergence, 6
- Kronecker product, 1, 8
- lasso, 6
- linear algebra, 1
- linear regression, 6
- logistic regression, 6
- loss
 - cross-entropy, 6
- loss functions, 7
- machine learning, 2
- MAE, 7
- matrix, 1
- maximum likelihood, 6
- mean, 5
- model selection, 7
- momentum, 6
- MSE, 7
- multicollinearity, 6
- naive Bayes, 6
- neural networks, 7
- normal distribution, 5
- notation, 1
- optimization, 6
- OLS (ordinary least squares), 6
- PARAFAC, 8
- PCA, 7, 11
 - SVD equivalence, 8
 - unsupervised, 8
 - vs SPCA vs PLS, 10
- PLS, 9
- posterior, 6
- prior, 6
- Python code, 11
- rank, 7
- ridge regression, 6
- RMSProp, 6
- SGD, 6
- sigmoid, 6
- softmax, 6
- sparsity, 6
- SPCA, 9, 11
- standardization, 5
- statistics
 - core, 5
- supervised learning, 6
- SVD, 7, 8, 11
- SVM, 7

symbols, 1
task map, 10
tensor, 1
tensor algebra, 1, 2, 8
tensor decomposition, 8
tensor PCA, 8
tensor regression, 8
trace, 1
Tucker decomposition, 8

variance, 5
variational autoencoder, 6
variational inference, 6
vector, 1
 $\mathbf{1}_k$, 9, 10
 $\mathbf{1}_k$ (vector of ones), 1
`vec` (vectorization), 1
vectorization identity, 1