Linear Algebra Fundamentals for AI in Science and Engineering

MEAM/EE/CBE/MSE 4600 — Prerequisites Review (2 Lectures)

Instructor: Paris Perdikaris

Reference: Murphy, K.P. (2022). *Probabilistic Machine Learning: An Introduction*, Chapter 7

Lecture 1: Vectors, Matrices, and Matrix Operations

1.1 Why Linear Algebra Matters for Machine Learning

Linear algebra is the mathematical backbone of modern machine learning. Nearly every ML algorithm can be expressed in terms of matrix and vector operations:

- **Data representation**: Datasets are matrices where rows are samples and columns are features
- **Model parameters**: Neural network weights are matrices and vectors
- **Transformations**: Linear layers compute $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$
- **Optimization**: Gradients are vectors; Hessians are matrices
- **Dimensionality reduction**: PCA uses eigendecomposition; autoencoders use matrix factorization
- **Probabilistic models**: Covariance matrices characterize multivariate Gaussians

Efficient numerical libraries (NumPy, PyTorch, JAX) are built around fast matrix operations, making linear algebra fluency essential for implementing and understanding ML algorithms.

1.2 Vectors

1.2.1 Definition and Notation

A **vector** $\mathbf{x} \in \mathbb{R}^n$ is an ordered list of n real numbers, written as a column:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Notation conventions:

- Bold lowercase letters denote vectors: $\mathbf{x}, \mathbf{y}, \mathbf{w}$
- x_i or $[\mathbf{x}]_i$ denotes the i-th element
- $\mathbf{0}$ is the vector of all zeros
- $\mathbf{1}$ is the vector of all ones

Special vectors:

- **Unit vector (one-hot)**: $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 in position i

Intuition: Think of a vector as a point in n-dimensional space, or equivalently, as an arrow from the origin to that point.

1.2.2 Vector Operations

Addition (elementwise):

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

Scalar multiplication:

$$c\mathbf{x} = \begin{bmatrix} cx_1 \\ \vdots \\ cx_n \end{bmatrix}$$

Geometric intuition: Adding vectors is "tip-to-tail" placement; scalar multiplication stretches/shrinks the arrow.

1.3 Vector Spaces and Subspaces

1.3.1 Vector Spaces

A **vector space** is a collection of vectors that is closed under addition and scalar multiplication. The

space $\mathbb{R}^n$ is the canonical example.

1.3.2 Linear Independence

A set of vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ is **linearly independent** if no vector can be written as a linear combination of the others.

Equivalently, the only solution to:

$$\alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \cdots + \alpha_k \mathbf{x}_k = \mathbf{0}$$

is $\alpha_1 = \alpha_2 = \cdots = \alpha_k = 0$.

Intuition: Linearly independent vectors point in "genuinely different" directions. In 2D, two vectors are linearly independent unless one is a scaled version of the other.

1.3.3 Span and Basis

The **span** of vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ is the set of all their linear combinations:

$$\text{span}(\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}) = \left\{ \mathbf{v} : \mathbf{v} = \sum_{i=1}^{k} \alpha_i \mathbf{x}_i,\ \alpha_i \in \mathbb{R} \right\}$$

A **basis** $\mathcal{B}$ for $\mathbb{R}^n$ is a set of n linearly independent vectors that span $\mathbb{R}^n$.

Standard basis: $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$

Key fact: Any vector in $\mathbb{R}^n$ can be uniquely expressed as a linear combination of basis vectors.

1.4 Matrices

1.4.1 Definition and Notation

A **matrix** $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a 2D array with m rows and n columns:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Notation:

- Bold uppercase: $\mathbf{A}, \mathbf{B}, \mathbf{W}$
- A_{ij} or $[\mathbf{A}]_{ij}$: entry in row i, column j
- $\mathbf{A}_{i,:}$: the i-th row (as a column vector after transpose)
- $\mathbf{A}_{:,j}$: the j-th column

Viewing a matrix:

- As n column vectors stacked horizontally: $\mathbf{A} = [\mathbf{a}_1 | \mathbf{a}_2 | \cdots | \mathbf{a}_n]$
- As m row vectors stacked vertically

1.4.2 The Transpose

The **transpose** $\mathbf{A}^\top \in \mathbb{R}^{n \times m}$ flips rows and columns:

$$[\mathbf{A}^\top]_{ij} = A_{ji}$$

Properties:

$$(\mathbf{A}^\top)^\top = \mathbf{A}, \quad (\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top, \quad (\mathbf{A} + \mathbf{B})^\top = \mathbf{A}^\top + \mathbf{B}^\top$$

A **symmetric matrix** satisfies $\mathbf{A} = \mathbf{A}^\top$ (must be square).

1.4.3 Tensors

A **tensor** generalizes matrices to higher dimensions:

- 1D tensor = vector
- 2D tensor = matrix
- 3D tensor: $\mathbf{A}_{ijk}$ (e.g., batch of images: batch × height × width)
- 4D tensor: common in CNNs (batch × channels × height × width)

1.5 Linear Maps and Matrix-Vector Multiplication

1.5.1 Matrices as Linear Transformations

1.5.1 Matrices as Linear Transformations

A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ defines a **linear map** $f : \mathbb{R}^n \to \mathbb{R}^m$ via:

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

Linearity means:

$$\mathbf{A}(\mathbf{x} + \mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y}, \quad \mathbf{A}(c\mathbf{x}) = c(\mathbf{A}\mathbf{x})$$

1.5.2 Two Perspectives on $\mathbf{y} = \mathbf{A}\mathbf{x}$

Perspective 1: Row-wise (inner products)

Each component y_i is the inner product of row i of $\mathbf{A}$ with $\mathbf{x}$:

$$y_i = \mathbf{a}_i^\top \mathbf{x} = \sum_{j=1}^{n} A_{ij} x_j$$

Perspective 2: Column-wise (linear combination)

$\mathbf{y}$ is a linear combination of the columns of $\mathbf{A}$, with coefficients from $\mathbf{x}$:

$$\mathbf{y} = x_1 \mathbf{a}_{:,1} + x_2 \mathbf{a}_{:,2} + \cdots + x_n \mathbf{a}_{:,n} = \sum_{j=1}^{n} x_j \mathbf{A}_{:,j}$$

ML Insight: In neural networks, $\mathbf{y} = \mathbf{W}\mathbf{x}$ computes a weighted combination of learned feature directions (columns of $\mathbf{W}$).

1.5.3 Range and Nullspace

Range (Column Space):

$$\text{range}(\mathbf{A}) = \{\mathbf{v} \in \mathbb{R}^m : \mathbf{v} = \mathbf{A}\mathbf{x} \text{ for some } \mathbf{x} \in \mathbb{R}^n\}$$

The set of all vectors "reachable" by $\mathbf{A}$.

Nullspace (Kernel):

$$\text{nullspace}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{0}\}$$

Vectors that $\mathbf{A}$ maps to zero.

Rank-Nullity Theorem:

$$\text{rank}(\mathbf{A}) + \dim(\text{nullspace}(\mathbf{A})) = n$$

1.6 Matrix Multiplication

1.6.1 Definition

For $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$, the product $\mathbf{C} = \mathbf{A}\mathbf{B} \in \mathbb{R}^{m \times p}$ has entries:

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

Requirement: Number of columns in $\mathbf{A}$ = number of rows in $\mathbf{B}$.

1.6.2 Four Perspectives on Matrix Multiplication

1. Entry-wise (inner products):

$$C_{ij} = \mathbf{a}_i^\top \mathbf{b}_j \quad (\text{row } i \text{ of } \mathbf{A} \text{ dotted with column } j \text{ of } \mathbf{B})$$

2. Column-wise:

$$\mathbf{C}_{:,j} = \mathbf{A}\mathbf{b}_j \quad (\text{each column of } \mathbf{C} \text{ is } \mathbf{A} \text{ times a column of } \mathbf{B})$$

3. Row-wise:

$$\mathbf{C}_{i,:}^\top = \mathbf{B}^\top \mathbf{a}_i \quad (\text{each row of } \mathbf{C} \text{ from rows of } \mathbf{A})$$

4. Outer product sum:

$$\mathbf{C} = \sum_{k=1}^{n} \mathbf{a}_k \mathbf{b}_k^\top$$

$$\mathbf{C} = \sum_{k=1} \mathbf{A}_{:,k} \mathbf{B}_{k,:}^\top = \sum_{k=1} \mathbf{a}_{:,k} \mathbf{b}_k^\top$$

Sum of rank-1 matrices (outer products).

1.6.3 Properties

- **Associative**: $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$
- **Distributive**: $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$
- **NOT commutative**: $\mathbf{AB} \neq \mathbf{BA}$ in general

1.6.4 Inner and Outer Products

Inner product (dot product) of $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y} = \sum_{i=1}^{n} x_i y_i \in \mathbb{R}$$

Outer product of $\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^n$:

$$\mathbf{x}\mathbf{y}^\top \in \mathbb{R}^{m \times n}, \quad [\mathbf{x}\mathbf{y}^\top]_{ij} = x_i y_j$$

1.7 Vector and Matrix Norms

1.7.1 Vector Norms

A **norm** $\|\mathbf{x}\|$ measures the "size" or "length" of a vector.

p-norm:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Common norms:

Norm	Formula	Interpretation

ℓ_2 (Euclidean)	$\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$	Geometric length
ℓ_1 (Manhattan)	$\|\mathbf{x}\|_1 = \sum_i$	x_i
ℓ_∞ (Max)	$\|\mathbf{x}\|_\infty = \max_i$	x_i
ℓ_0 (pseudo-norm)	$\|\mathbf{x}\|_0 = \#\{i : x_i \neq 0\}$	Number of nonzeros

ML Applications:

- ℓ_2 regularization (ridge) encourages small weights
- ℓ_1 regularization (lasso) encourages sparse weights
- ℓ_0 "norm" counts features (used conceptually in feature selection)

Key identity:

$$\|\mathbf{x}\|_2^2 = \mathbf{x}^\top \mathbf{x}$$

1.7.2 Matrix Norms

Frobenius norm (treats matrix as a long vector):

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2} = \sqrt{\text{tr}(\mathbf{A}^\top \mathbf{A})}$$

Spectral norm (induced ℓ_2 norm):

$$\|\mathbf{A}\|_2 = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \sigma_{\max}(\mathbf{A})$$

The largest singular value—measures maximum "stretching" by $\mathbf{A}$.

1.8 Similarity, Distance, and Projections

1.8.1 Cosine Similarity

Cosine similarity measures the cosine of the angle between two vectors, capturing their directional alignment regardless of magnitude:

$$\text{cos\_sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

Properties:

- Range: $[-1, 1]$
- $\text{cos\_sim} = 1$: vectors point in the same direction (parallel)
- $\text{cos\_sim} = 0$: vectors are orthogonal (perpendicular)
- $\text{cos\_sim} = -1$: vectors point in opposite directions (anti-parallel)

Geometric interpretation: If θ is the angle between $\mathbf{x}$ and $\mathbf{y}$:

$$\cos(\theta) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$$

Why cosine similarity is so useful in ML:

Application	Why Cosine Works
Word embeddings	Similar words have similar directions in embedding space
Document similarity	Normalizes for document length (long docs aren't automatically "closer")
Image retrieval	Feature vectors with similar orientations represent similar content
Recommendation systems	User/item embeddings compared by direction, not magnitude
Attention mechanisms	Scaled dot-product attention uses unnormalized version

Example: In word2vec, $\text{cos\_sim}(\text{"king"}, \text{"queen"}) \approx 0.7$ while $\text{cos\_sim}(\text{"king"}, \text{"apple"}) \approx 0.1$.

1.8.2 Relationship Between Dot Product, Norms, and Cosine

These three quantities are intimately connected:

$$\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos(\theta)$$

This means:

- **Dot product** = (length of $\mathbf{x}$) × (length of $\mathbf{y}$) × (cosine of angle)
- For **unit vectors** ($\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$): dot product = cosine similarity

ML Practice: Many systems normalize embeddings to unit length, so dot product and cosine similarity become identical—and dot products are faster to compute!

1.8.3 Distance Metrics

Distance measures how far apart two vectors are. Common choices:

Euclidean distance (ℓ_2):

$$d_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Squared Euclidean distance (often used to avoid the square root):

$$d_2^2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2 = \sum_{i=1}^{n} (x_i - y_i)^2$$

Manhattan distance (ℓ_1):

$$d_1(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^{n} |x_i - y_i|$$

Cosine distance:

$$d_{\cos}(\mathbf{x}, \mathbf{y}) = 1 - \cos\_\text{sim}(\mathbf{x}, \mathbf{y})$$

Relationship between Euclidean distance and cosine similarity:

For **unit vectors** $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$:

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = \|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 - 2\mathbf{x}^\top\mathbf{y} = 2 - 2\cos(\theta) = 2(1 - \text{cos\_sim}(\mathbf{x}, \mathbf{y}))$$

Implication: For normalized vectors, minimizing Euclidean distance is equivalent to maximizing cosine similarity!

1.8.4 Vector Projection

The **projection** of vector $\mathbf{y}$ onto vector $\mathbf{x}$ gives the component of $\mathbf{y}$ in the direction of $\mathbf{x}$:

Scalar projection (length of shadow):

$$\text{comp}_{\mathbf{x}}(\mathbf{y}) = \frac{\mathbf{x}^\top\mathbf{y}}{\|\mathbf{x}\|_2} = \|\mathbf{y}\|_2\cos(\theta)$$

Vector projection (the actual projected vector):

$$\text{proj}_{\mathbf{x}}(\mathbf{y}) = \frac{\mathbf{x}^\top\mathbf{y}}{\|\mathbf{x}\|_2^2}\mathbf{x} = \frac{\mathbf{x}^\top\mathbf{y}}{\mathbf{x}^\top\mathbf{x}}\mathbf{x}$$

Intuition: Drop a perpendicular from the tip of $\mathbf{y}$ onto the line defined by $\mathbf{x}$.

Projection onto a subspace: Given matrix $\mathbf{A}$ with linearly independent columns, the projection of $\mathbf{y}$ onto the column space of $\mathbf{A}$ is:

$$\text{proj}_{\mathbf{A}}(\mathbf{y}) = \mathbf{A}(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{y}$$

The matrix $\mathbf{P} = \mathbf{A}(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top$ is called the **projection matrix**.

ML Connection: Least squares finds the projection of the target $\mathbf{y}$ onto the column space of the design matrix $\mathbf{X}$.

1.8.5 Orthogonality

Two vectors are **orthogonal** (perpendicular) if:

$$\mathbf{x}^\top\mathbf{y} = 0 \quad \Leftrightarrow \quad \cos(\theta) = 0 \quad \Leftrightarrow \quad \theta = 90^\circ$$

Orthonormal vectors: orthogonal AND unit length ($\|\mathbf{x}\|_2 = 1$).

ML Importance:

ML Importance:

- PCA finds orthogonal directions of maximum variance
- Many regularization techniques encourage orthogonality
- Orthogonal weight matrices preserve gradient magnitudes (help with training)

1.8.6 Similarity in High Dimensions: A Caution

In high-dimensional spaces, some intuitions break down:

Concentration of distances: As $n \to \infty$, all pairwise distances become similar:

$$\frac{\max_{i,j} d(\mathbf{x}_i, \mathbf{x}_j) - \min_{i,j} d(\mathbf{x}_i, \mathbf{x}_j)}{\min_{i,j} d(\mathbf{x}_i, \mathbf{x}_j)} \to 0$$

Random vectors are nearly orthogonal: Two random vectors in $\mathbb{R}^n$ have $\cos(\theta) \approx 0$ with high probability for large n.

Practical implication: Cosine similarity often works better than Euclidean distance in high dimensions because it focuses on direction rather than absolute distance.

1.8.7 Application: Scaled Dot-Product Attention

The **attention mechanism** in Transformers (GPT, BERT, etc.) is fundamentally based on similarity:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

where:

- $\mathbf{Q}$ (queries), $\mathbf{K}$ (keys), $\mathbf{V}$ (values) are matrices
- $\mathbf{Q}\mathbf{K}^\top$ computes **all pairwise dot products** (similarity scores)
- Division by $\sqrt{d_k}$ prevents dot products from becoming too large
- Softmax converts similarities to attention weights (probabilities)

Connection to cosine similarity: If queries and keys were normalized, $\mathbf{Q}\mathbf{K}^\top$ would give cosine similarities. The scaling by $\sqrt{d_k}$ partially compensates for the lack of normalization.

1.9 Special Matrices

1.9 Special Matrices

1.8.1 Diagonal Matrices

$$\mathbf{D} = \text{diag}(d_1, d_2, \ldots, d_n) = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}$$

Properties:

- $\mathbf{Dx}$ scales each component: $[\mathbf{Dx}]_i = d_i x_i$
- $\mathbf{D}^{-1} = \text{diag}(1/d_1, \ldots, 1/d_n)$

Identity matrix: $\mathbf{I} = \text{diag}(1, 1, \ldots, 1)$ satisfies $\mathbf{IA} = \mathbf{AI} = \mathbf{A}$

1.8.2 Symmetric Matrices

$\mathbf{A}$ is **symmetric** if $\mathbf{A} = \mathbf{A}^\top$.

Ubiquitous in ML: Covariance matrices, Gram matrices, Hessians.

1.8.3 Orthogonal Matrices

A square matrix $\mathbf{U}$ is **orthogonal** if its columns are orthonormal:

$$\mathbf{U}^\top \mathbf{U} = \mathbf{U}\mathbf{U}^\top = \mathbf{I}$$

Key property: $\mathbf{U}^{-1} = \mathbf{U}^\top$ (transpose is the inverse!)

Geometric meaning: Orthogonal matrices represent rotations (if $\det(\mathbf{U}) = 1$) or reflections (if $\det(\mathbf{U}) = -1$). They **preserve lengths and angles**:

$$\|\mathbf{Ux}\|_2 = \|\mathbf{x}\|_2$$

1.8.4 Positive Definite Matrices

A symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is:

- **Positive definite (PD)** if $\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$
- **Positive semi-definite (PSD)** if $\mathbf{x}^\top \mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x}$

Equivalent characterizations (for symmetric $\mathbf{A}$):

- $\mathbf{A}$ is PD $\Leftrightarrow$ all eigenvalues are positive
- $\mathbf{A}$ is PSD $\Leftrightarrow$ all eigenvalues are non-negative

ML Importance:

- Covariance matrices are always PSD
- Hessians at local minima must be PSD
- Quadratic forms $\mathbf{x}^\top \mathbf{A}\mathbf{x}$ with PD $\mathbf{A}$ are strictly convex (bowl-shaped)

Gram matrix: For any $\mathbf{A} \in \mathbb{R}^{m \times n}$, the matrix $\mathbf{A}^\top \mathbf{A}$ is always PSD.

1.10 Scalar Properties of Matrices

1.9.1 Trace

The **trace** of a square matrix is the sum of diagonal elements:

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} A_{ii}$$

Properties:

$$\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{A}^\top), \quad \text{tr}(\mathbf{A} + \mathbf{B}) = \text{tr}(\mathbf{A}) + \text{tr}(\mathbf{B}), \quad \text{tr}(c\mathbf{A}) = c \cdot \text{tr}(\mathbf{A})$$

Cyclic property (very useful!):

$$\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{BCA}) = \text{tr}(\mathbf{CAB})$$

Trace trick (converts quadratic form to trace):

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = \text{tr}(\mathbf{x}^\top \mathbf{A}\mathbf{x}) = \text{tr}(\mathbf{A}\mathbf{x}\mathbf{x}^\top)$$

Connection to eigenvalues:

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i$$

1.9.2 Determinant

The **determinant** $\det(\mathbf{A})$ or $|\mathbf{A}|$ measures the "volume scaling factor" of the linear transformation.

Properties:

$$|\mathbf{A}^\top| = |\mathbf{A}|, \quad |\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|, \quad |c\mathbf{A}| = c^n|\mathbf{A}|$$

$$|\mathbf{A}^{-1}| = 1/|\mathbf{A}|, \quad |\mathbf{A}| = 0 \Leftrightarrow \mathbf{A} \text{ is singular}$$

Connection to eigenvalues:

$$\det(\mathbf{A}) = \prod_{i=1}^{n} \lambda_i$$

ML Application: Log-determinant appears in Gaussian likelihood: $\log p(\mathbf{x}) \propto -\frac{1}{2}\log|\mathbf{\Sigma}|$

1.9.3 Rank

The **rank** of $\mathbf{A}$ is the dimension of its column space (= dimension of row space).

Properties:

- $\text{rank}(\mathbf{A}) \leq \min(m, n)$
- **Full rank**: $\text{rank}(\mathbf{A}) = \min(m, n)$
- $\text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$

1.9.4 Condition Number

The **condition number** measures numerical stability:

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| = \frac{\sigma_{\max}}{\sigma_{\min}}$$

- $\kappa(\mathbf{A}) \geq 1$ always
- **Well-conditioned**: κ close to 1
- **Ill-conditioned**: large κ (nearly singular)

Practical impact: High condition number means small errors in $\mathbf{b}$ cause large errors in solving $\mathbf{Ax} = \mathbf{b}$.

1.11 Summary of Lecture 1

Concept	Key Formula/Fact	ML Relevance
Vector in $\mathbb{R}^n$	Column of n numbers	Feature vectors, gradients
Matrix in $\mathbb{R}^{m \times n}$	m rows, n columns	Weight matrices, data matrices
Linear independence	No redundant vectors	Feature selection
$\mathbf{y} = \mathbf{Ax}$	Linear combination of columns	Neural network layers
ℓ_2 norm	$\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top \mathbf{x}}$	Regularization, distance
Cosine similarity	$\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$	Embeddings, attention, retrieval
Vector projection	$\frac{\mathbf{x}^\top \mathbf{y}}{\mathbf{x}^\top \mathbf{x}} \mathbf{x}$	Least squares, PCA
Symmetric PD	$\mathbf{x}^\top \mathbf{Ax} > 0$	Covariances, convex objectives
Orthogonal	$\mathbf{U}^\top = \mathbf{U}^{-1}$	Rotations, preserves geometry

---

Lecture 2: Matrix Decompositions and Calculus

2.1 Matrix Inversion

2.1.1 Definition

The **inverse** of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, if it exists, satisfies:

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

Existence: $\mathbf{A}^{-1}$ exists $\Leftrightarrow \det(\mathbf{A}) \neq 0 \Leftrightarrow \mathbf{A}$ is full rank.

Properties:

$$(\mathbf{A}^{-1})^{-1} = \mathbf{A}, \quad (\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}, \quad (\mathbf{A}^{-1})^{\top} = (\mathbf{A}^{\top})^{-1}$$

2.1.2 The 2×2 Case

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \implies \mathbf{A}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

2.1.3 Pseudo-Inverse

For non-square or rank-deficient matrices, the **Moore-Penrose pseudo-inverse** $\mathbf{A}^{\dagger}$ generalizes inversion.

Tall matrix ($m > n$, full column rank):

$$\mathbf{A}^{\dagger} = (\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top} \quad (\text{left inverse: } \mathbf{A}^{\dagger}\mathbf{A} = \mathbf{I})$$

Wide matrix ($m < n$, full row rank):

$$\mathbf{A}^{\dagger} = \mathbf{A}^{\top}(\mathbf{A}\mathbf{A}^{\top})^{-1} \quad (\text{right inverse: } \mathbf{A}\mathbf{A}^{\dagger} = \mathbf{I})$$

ML Application: Least squares solution is $\hat{\mathbf{x}} = \mathbf{A}^{\dagger}\mathbf{b}$.

2.2 Eigenvalue Decomposition (EVD)

2.2.1 Eigenvalues and Eigenvectors

For a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, a scalar λ is an **eigenvalue** and $\mathbf{u} \neq \mathbf{0}$ is the corresponding **eigenvector** if:

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$$

Intuition: Eigenvectors are special directions that $\mathbf{A}$ merely scales (by λ) rather than rotates.

Finding eigenvalues: Solve the **characteristic equation**:

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0$$

This is a degree-n polynomial in λ.

2.2.2 Diagonalization

If $\mathbf{A}$ has n linearly independent eigenvectors, we can write:

$$\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^{-1}$$

where:

- $\mathbf{U} = [\mathbf{u}_1 | \mathbf{u}_2 | \cdots | \mathbf{u}_n]$ (columns are eigenvectors)
- $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ (diagonal of eigenvalues)

2.2.3 Symmetric Matrices (The Important Case)

For **symmetric** $\mathbf{A} = \mathbf{A}^\top$:

1. All eigenvalues are **real**
2. Eigenvectors are **orthonormal**: $\mathbf{U}^\top \mathbf{U} = \mathbf{I}$

Spectral decomposition:

$$\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$$

Interpretation: $\mathbf{A}$ is a weighted sum of rank-1 projection matrices.

Inverse:

$$\mathbf{A}^{-1} = \mathbf{U} \mathbf{\Lambda}^{-1} \mathbf{U}^\top = \sum_{i=1}^{n} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^\top$$

2.2.4 Positive Definiteness via Eigenvalues

For symmetric $\mathbf{A}$:

- $\mathbf{A}$ is **positive definite** $\Leftrightarrow$ all $\lambda_i > 0$
- $\mathbf{A}$ is **positive semi-definite** $\Leftrightarrow$ all $\lambda_i \geq 0$

Proof sketch: Let $\mathbf{y} = \mathbf{U}^\top \mathbf{x}$. Then:

$$\mathbf{x}^\top \mathbf{A} \mathbf{x} = \mathbf{x}^\top \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top \mathbf{x} = \mathbf{y}^\top \mathbf{\Lambda} \mathbf{y} = \sum_{i=1}^{n} \lambda_i y_i^2$$

2.2.5 Geometry of Quadratic Forms

The quadratic form $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$ with PD symmetric $\mathbf{A}$ defines **elliptical level sets**.

- **Eigenvectors** $\mathbf{u}_i$: principal axes of the ellipse
- **Eigenvalues** λ_i: determine axis lengths ($\propto 1/\sqrt{\lambda_i}$)
- **Condition number** $\kappa = \lambda_{\max}/\lambda_{\min}$: ellipse elongation

ML Insight: For Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, constant-probability contours are ellipses with axes along eigenvectors of $\mathbf{\Sigma}$.

2.3 Singular Value Decomposition (SVD)

2.3.1 Definition

Any matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ (not necessarily square!) can be decomposed as:

$$\mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$$

where:

- $\mathbf{U} \in \mathbb{R}^{m \times m}$: orthogonal, columns are **left singular vectors**
- $\mathbf{S} \in \mathbb{R}^{m \times n}$: diagonal with **singular values** $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$
- $\mathbf{V} \in \mathbb{R}^{n \times n}$: orthogonal, columns are **right singular vectors**
- $r = \text{rank}(\mathbf{A}) \leq \min(m, n)$

Expanded form:

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$$

Sum of rank-1 matrices, ordered by importance.

2.3.2 Thin (Economy) SVD

In practice, we often compute the **thin SVD**:

- If $m > n$: $\mathbf{U} \in \mathbb{R}^{m \times n}, \mathbf{S} \in \mathbb{R}^{n \times n}, \mathbf{V} \in \mathbb{R}^{n \times n}$
- If $m < n$: $\mathbf{U} \in \mathbb{R}^{m \times m}, \mathbf{S} \in \mathbb{R}^{m \times m}, \mathbf{V} \in \mathbb{R}^{n \times m}$

2.3.3 Connection to EVD

$$\mathbf{A}^\top \mathbf{A} = \mathbf{V} \mathbf{S}^\top \mathbf{S} \mathbf{V}^\top \implies \text{eigenvectors of } \mathbf{A}^\top \mathbf{A} = \mathbf{V}, \text{ eigenvalues } = \sigma_i^2$$

$$\mathbf{A} \mathbf{A}^\top = \mathbf{U} \mathbf{S} \mathbf{S}^\top \mathbf{U}^\top \implies \text{eigenvectors of } \mathbf{A} \mathbf{A}^\top = \mathbf{U}, \text{ eigenvalues } = \sigma_i^2$$

2.3.4 Low-Rank Approximation

The **truncated SVD** with top k singular values:

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top = \mathbf{U}_k \mathbf{S}_k \mathbf{V}_k^\top$$

Eckart-Young Theorem: $\mathbf{A}_k$ is the **best rank-k approximation** to $\mathbf{A}$:

$$\mathbf{A}_k = \arg\min_{\text{rank}(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_F$$

Approximation error:

$$\|\mathbf{A} - \mathbf{A}_k\|_F^2 = \sum^{r} \sigma_i^2$$

$$\sum_{i=k+1}$$

ML Applications:

- **PCA**: Projects data onto top k right singular vectors of centered data
- **Latent semantic analysis**: Low-rank approximation of document-term matrices
- **Image compression**: Store only top singular values/vectors
- **Recommender systems**: Matrix factorization methods

2.3.5 Pseudo-Inverse via SVD

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{S}^\dagger\mathbf{U}^\top$$

where $\mathbf{S}^\dagger = \text{diag}(1/\sigma_1, \ldots, 1/\sigma_r, 0, \ldots, 0)$.

2.4 Other Important Decompositions

2.4.1 Cholesky Decomposition

Any **symmetric positive definite** matrix can be factored as:

$$\mathbf{A} = \mathbf{L}\mathbf{L}^\top$$

where $\mathbf{L}$ is lower triangular with positive diagonal entries.

Applications:

- **Efficient solving**: $\mathbf{A}\mathbf{x} = \mathbf{b}$ becomes two triangular solves
- **Sampling from Gaussian**: If $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, then $\mathbf{x} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- **Log-determinant**: $\log|\mathbf{A}| = 2\sum_i \log L_{ii}$

2.4.2 QR Decomposition

Any $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$ can be factored as:

$$\mathbf{A} = \mathbf{Q}\mathbf{R}$$

where $\mathbf{Q} \in \mathbb{R}^{m \times n}$ has orthonormal columns and $\mathbf{R} \in \mathbb{R}^{n \times n}$ is upper triangular.

Application: Numerically stable least squares solving.

2.5 Solving Linear Systems

2.5.1 Three Cases

Consider $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{m \times n}$:

Case	Condition	Geometric Picture	Solution
Square	$m = n$, full rank	n hyperplanes, one intersection	$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$
Underdetermined	$m < n$	Infinitely many solutions	Min-norm: $\mathbf{x} = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{b}$
Overdetermined	$m > n$	No exact solution	Least squares: $\mathbf{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b}$

2.5.2 Least Squares (Overdetermined)

Problem: Find $\hat{\mathbf{x}}$ minimizing:

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

Solution (normal equations):

$$\mathbf{A}^\top\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^\top\mathbf{b} \implies \hat{\mathbf{x}} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b}$$

Interpretation: Project $\mathbf{b}$ onto the column space of $\mathbf{A}$. The residual $\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}$ is orthogonal to all columns of $\mathbf{A}$.

2.6 Matrix Calculus

2.6.1 Gradients of Scalar Functions

2.6.1 Gradients of Scalar Functions

For $f : \mathbb{R}^n \to \mathbb{R}$, the **gradient** is:

$$\nabla f(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \in \mathbb{R}^n$$

Geometric meaning: $\nabla f(\mathbf{x})$ points in the direction of steepest ascent.

2.6.2 Essential Gradient Identities

These formulas appear constantly in ML derivations:

Linear forms:

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{a}^\top \mathbf{x}) = \mathbf{a}$$

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{b}^\top \mathbf{A}\mathbf{x}) = \mathbf{A}^\top \mathbf{b}$$

Quadratic forms:

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{A}\mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}$$

For **symmetric** $\mathbf{A}$:

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{A}\mathbf{x}) = 2\mathbf{A}\mathbf{x}$$

Squared norm:

$$\frac{\partial}{\partial \mathbf{x}} \|\mathbf{x}\|_2^2 = \frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{x}) = 2\mathbf{x}$$

Least squares objective:

$$f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \implies \nabla f = 2\mathbf{A}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$$

2.6.3 The Jacobian

For $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, the **Jacobian** is the $m \times n$ matrix of partial derivatives:

$$\mathbf{J}_\mathbf{f}(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Row i is the gradient of f_i.

Chain rule for vector functions:

$$\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{f}(\mathbf{x})) \implies \mathbf{J}_\mathbf{h}(\mathbf{x}) = \mathbf{J}_\mathbf{g}(\mathbf{f}(\mathbf{x})) \cdot \mathbf{J}_\mathbf{f}(\mathbf{x})$$

ML Application: Backpropagation computes Jacobians layer by layer.

2.6.4 The Hessian

For $f : \mathbb{R}^n \to \mathbb{R}$, the **Hessian** is the $n \times n$ matrix of second derivatives:

$$\mathbf{H}_f = \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

The Hessian is always symmetric (if f is twice continuously differentiable).

Connection to optimization:

- $\mathbf{H} \succ 0$ (positive definite) at $\mathbf{x}^* \implies$ local minimum
- $\mathbf{H} \prec 0$ (negative definite) at $\mathbf{x}^* \implies$ local maximum
- $\mathbf{H}$ indefinite $\implies$ saddle point

Hessian of quadratic form:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} \implies \mathbf{H}_f = \mathbf{A} + \mathbf{A}^\top$$

2.6.5 Matrix Derivatives

For $f : \mathbb{R}^{m \times n} \to \mathbb{R}$, the derivative is:

$$\frac{\partial f}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial f}{\partial X_{11}} & \cdots & \frac{\partial f}{\partial X_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial X_{m1}} & \cdots & \frac{\partial f}{\partial X_{mn}} \end{bmatrix}$$

Useful identities:

$$\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{A}\mathbf{X}\mathbf{B}) = \mathbf{A}^\top \mathbf{B}^\top$$

$$\frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{X}^\top \mathbf{A}) = \mathbf{A}$$

$$\frac{\partial}{\partial \mathbf{X}} \log |\mathbf{X}| = \mathbf{X}^{-\top}$$

2.7 Data Preprocessing with Linear Algebra

2.7.1 Standardization

Given data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ (rows = samples):

Centering: Subtract the mean from each column:

$$\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}_N \bar{\mathbf{x}}^\top$$

where $\bar{\mathbf{x}} = \frac{1}{N} \mathbf{X}^\top \mathbf{1}_N$ is the column-wise mean.

Standardization: Center and scale to unit variance:

$$\text{standardize}(\mathbf{X}) = (\mathbf{X} - \mathbf{1}_N \boldsymbol{\mu}^\top) \text{diag}(\boldsymbol{\sigma})^{-1}$$

2.7.2 Whitening (Decorrelation)

PCA whitening: Transform data so covariance becomes identity:

$$\mathbf{W}_{\text{PCA}} = \mathbf{D}^{-1/2}\mathbf{E}^\top$$

where $\boldsymbol{\Sigma} = \mathbf{E}\mathbf{D}\mathbf{E}^\top$ is the eigendecomposition of the covariance.

After transformation $\mathbf{y} = \mathbf{W}_{\text{PCA}}\mathbf{x}$:

$$\text{Cov}[\mathbf{y}] = \mathbf{I}$$

2.8 Summary: Key Decompositions Comparison

Decomposition	Applies to	Form	Key Property
EVD	Square $\mathbf{A}$	$\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}$	Diagonalizes $\mathbf{A}$
Spectral	Symmetric $\mathbf{A}$	$\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$	Orthonormal eigenvectors
SVD	Any $\mathbf{A}$	$\mathbf{U}\mathbf{S}\mathbf{V}^\top$	Best low-rank approx.
Cholesky	Symmetric PD	$\mathbf{L}\mathbf{L}^\top$	Efficient solving
QR	Any tall $\mathbf{A}$	$\mathbf{Q}\mathbf{R}$	Orthonormal columns

---

2.9 Essential Formulas Reference Card

Norms & Distances

- $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top\mathbf{x}}$
- $\|\mathbf{A}\|_F = \sqrt{\text{tr}(\mathbf{A}^\top\mathbf{A})}$
- $d_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$

Similarity & Projection

Similarity & Projection

- Cosine similarity: $\text{cos\_sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$
- Dot product identity: $\mathbf{x}^\top \mathbf{y} = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos(\theta)$
- Vector projection: $\text{proj}_{\mathbf{x}}(\mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\mathbf{x}^\top \mathbf{x}} \mathbf{x}$
- For unit vectors: $\|\mathbf{x} - \mathbf{y}\|_2^2 = 2(1 - \text{cos\_sim}(\mathbf{x}, \mathbf{y}))$

Trace & Determinant

- $\text{tr}(\mathbf{A}) = \sum_i \lambda_i$
- $\det(\mathbf{A}) = \prod_i \lambda_i$
- $\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{BCA}) = \text{tr}(\mathbf{CAB})$

EVD (Symmetric)

- $\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top = \sum_i \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$
- $\mathbf{A}^{-1} = \mathbf{U} \mathbf{\Lambda}^{-1} \mathbf{U}^\top$

SVD

- $\mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^\top = \sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$
- $\mathbf{A}^\dagger = \mathbf{V} \mathbf{S}^\dagger \mathbf{U}^\top$

Gradients

- $\nabla_{\mathbf{x}} (\mathbf{a}^\top \mathbf{x}) = \mathbf{a}$
- $\nabla_{\mathbf{x}} (\mathbf{x}^\top \mathbf{A} \mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top) \mathbf{x}$
- $\nabla_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = 2\mathbf{A}^\top (\mathbf{A}\mathbf{x} - \mathbf{b})$

Least Squares

- Normal equations: $\mathbf{A}^\top \mathbf{A} \hat{\mathbf{x}} = \mathbf{A}^\top \mathbf{b}$
- Solution: $\hat{\mathbf{x}} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$

2.10 Practice Problems

$[4 \times 2]$

1. **EVD by hand**: Find the eigenvalues and eigenvectors of $\mathbf{A} = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}$.
2. **Positive definiteness**: Is $\mathbf{B} = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$ positive definite? Prove using eigenvalues.
3. **Gradient derivation**: Starting from $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$, derive the gradient and set to zero to obtain the normal equations.
4. **SVD application**: Given data matrix $\mathbf{X} \in \mathbb{R}^{100 \times 50}$, how many singular values can be nonzero at most? If you compute a rank-10 approximation, how many parameters do you need to store?
5. **Cholesky sampling**: Describe the algorithm to sample from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ using Cholesky decomposition.
6. **Cosine similarity**:
   - (a) Compute the cosine similarity between $\mathbf{x} = [1, 2, 3]^\top$ and $\mathbf{y} = [4, 5, 6]^\top$.
   - (b) If both vectors are normalized to unit length, what is their Euclidean distance in terms of their cosine similarity?
   - (c) Why might cosine similarity be preferred over Euclidean distance when comparing documents of different lengths?
7. **Projection**: Find the projection of $\mathbf{y} = [3, 4]^\top$ onto the vector $\mathbf{x} = [1, 0]^\top$. Verify that the residual $\mathbf{y} - \text{proj}_{\mathbf{x}}(\mathbf{y})$ is orthogonal to $\mathbf{x}$.

These notes cover the essential linear algebra needed for machine learning. For deeper treatment, consult Murphy Ch. 7 and the references therein.