Mathematical Foundations of Machine Learning

Week 1: Linear Algebra Basics

Ayberk Zeytin

1 Fundamental definitions

Linear algebra is the language of modern machine learning. Data, models, and algorithms are expressed using vectors, matrices, and linear transformations. This lecture sets the mathematical backbone: vector spaces, inner products, projections, norms, and dual norms.

Definition 1.1 (vector space)**.** A **vector space** V over $\mathbb{R}$ (or a real vector space, in short) is a set with addition and scalar multiplication satisfying the standard axioms two operations (vector addition and scalar multiplication) such that for all $u, v, w \in V$ and $\alpha, \beta \in \mathbb{R}$:

(i) $u + v \in V, \quad (u + v) + w = u + (v + w)$ — closure & associativity

(ii) $u + v = v + u, \quad$ there is $0 \in V$ such that $u + 0 = u$ — commutativity & identity(neutral) element

(iii) For each $u \in V$, there exists $-u \in V$ with $u + (-u) = 0$ — inverse element

(iv) $\alpha u \in V, \quad (\alpha\beta)u = \alpha(\beta u)$ — closure & compatibility

(v) $1 \cdot u = u, \quad (\alpha + \beta)u = \alpha u + \beta u$ — distributivity

(vi) $\alpha(u + v) = \alpha u + \alpha v$ — distributivity

Example 1.1. The Euclidean space $\mathbb{R}^n$ with coordinate-wise operations is a vector space. Elements of this vector space are denoted usually by column vectors :

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}^\top$$

```python
# R^n as vectors with coordinate-wise operations (using NumPy)
import numpy as np

# Example with n = 4 (replace the numbers with any reals you like)
x = np.array([1.0, -2.0, 3.5, 0.0])
y = np.array([4.0,  5.0, -1.0, 2.0])

# View x as a column vector of shape (n,1)
x_col = x.reshape(-1, 1)

# Coordinate-wise vector addition
x_plus_y = x + y

# Scalar multiplication with alpha in R
alpha = 2.5
alpha_times_x = alpha * x

# Simple checks for vector space laws (numerical)
z = np.array([0.5, 1.5, -2.0, 4.0])
print(np.allclose(x + y, y + x))              # commutativity
print(np.allclose((x + y) + z, x + (y + z))) # associativity
print(np.allclose(alpha*(x + y), alpha*x + alpha*y)) # distributivity

```

Example 1.2. The set of polynomials of degree $\leq d$ is another example (with point-wise addition and scalar multiplication). Elements of this vector space are denoted by *finite* sums : $p(t) = a_0 + a_1 t + \ldots + a_d t^d$. These elements are usually identified with the column vector :

$$p = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & \dots & a_d \end{pmatrix}^\top.$$

Observe that we need $d+1$ real numbers to specify an element of this vector space - this is called the *dimension*.

```
import numpy as np
def poly_add(p, q):
"""
Add two polynomials p and q given by coefficient lists:
p = [a0, a1, ..., ad], q = [b0, b1, ..., bd]
Returns r = p + q (padded to the longer length if needed).
"""
p = np.array(p, dtype=float)
q = np.array(q, dtype=float)
m, n = len(p), len(q)
if m < n:
p = np.pad(p, (0, n - m))
elif n < m:
q = np.pad(q, (0, m - n))
return p + q

def scalar_mul(alpha, p):
"""Scalar multiplication: alpha p."""
return float(alpha) * np.array(p, dtype=float)

def poly_eval(p, t):
"""
Evaluate p(t) = a0 + a1 t + ... + ad t^d at a real number t.
"""
p = np.array(p, dtype=float)
powers = t ** np.arange(len(p))  # [1, t, t^2, ..., t^d]
return float(p @ powers)

# Example (degree less than or equal to 3, so 4 coefficients):
# p(t) = 1 - 2 t + 0.5 t^2 + 3 t^3
p = [1.0, -2.0, 0.5, 3.0]
# q(t) = 0 + 4 t - t^2 + 2 t^3
q = [0.0, 4.0, -1.0, 2.0]

# Vector space operations on coefficients
r = poly_add(p, q)      # p + q
alpha = 3.0
s = scalar_mul(alpha, p) # alpha p

# Evaluation at t = 2
val_p_at_2 = poly_eval(p, t=2.0)

# Dimension check for degree less than or equal to d
d = 3
dimension = d + 1       # should equal the number of coefficients
print(dimension == len(p))
```

Exercise 1.1. ▶ Define a *basis* for a vector space.

- ▶ Give two distinct examples of bases of $\mathbb{R}^2$, $\mathbb{R}^3$ and more generally $\mathbb{R}^n$.
- ▶ How many elements does a basis of $\mathbb{R}^2$ have? What about $\mathbb{R}^3$? $\mathbb{R}^n$?
- ▶ Show that if $\mathcal{B}_1$ and $\mathcal{B}_2$ are two bases of a vector space V, then they have the same cardinality[1], i.e. $|\mathcal{B}_1| = |\mathcal{B}_2|$.
- ▶ Define the dimension of a vector space V.

There are several extra structures that a vector space may admit. One of them is an inner (or dot) product :

Definition 1.2 (Inner Product Spaces)**.** An **inner product** $\langle\cdot,\cdot\rangle$ on a real vector space V is a map

$$\langle\cdot,\cdot\rangle : V \times V \to \mathbb{R}$$

satisfying for all $x, y, z \in V$ and $\alpha \in \mathbb{R}$:

(i) $\langle x, y\rangle = \langle y, x\rangle$ — symmetry,

(ii) $\langle \alpha x + y, z\rangle = \alpha\langle x, z\rangle + \langle y, z\rangle$ — linearity,

(iii) $\langle x, x\rangle \geq 0$ with equality if and only if $x = 0$ — positive-definiteness.

[1] Cardinality is defined as the number of elements if the set is finite. For infinite sets however, it is defined as the equivalence class of the set in question under the (equivalence) relation of *equinumerosity*. Under this relation, the equivalence class of a set is denoted usually by $|\cdot|$.

Example 1.3. Take our vector space as $\mathbb{R}^n$. For $x = (x_1, \dots, x_n)^\top$ and $y = (y_1, \dots, y_n)^\top$ vectors in $\mathbb{R}^n$ we may define their inner product as:

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n = x^\top \cdot y$$

This satisfies all the properties of an inner product:

(i) $\langle x, y \rangle = \sum x_i y_i = \sum y_i x_i = \langle y, x \rangle$ (symmetry),

(ii) $\langle \alpha x + y, z \rangle = \sum (\alpha x_i + y_i) z_i = \alpha \sum x_i z_i + \sum y_i z_i = \alpha \langle x, z \rangle + \langle y, z \rangle$ (linearity),

(iii) $\langle x, x \rangle = \sum x_i^2 \geq 0$. The sum is zero if and only if each $x_i = 0$, which means $x = 0$ (positive-definiteness).

```
# Inner product in R^n (using NumPy)
import numpy as np

# define two vectors x and y
x = np.array([1.0, -2.0, 3.0])
y = np.array([4.0, 0.5, -1.0])

# compute inner product <x,y>
inner_xy = np.dot(x, y)

# properties:

# (i) symmetry: <x,y> == <y,x>
print(np.dot(x, y) == np.dot(y, x))

# (ii) linearity: <a*x + y, z> == a<x,z> + <y,z>
a = 2.5
z = np.array([0.0, 1.0, 2.0])
lhs = np.dot(a*x + y, z)
rhs = a*np.dot(x, z) + np.dot(y, z)
print(np.allclose(lhs, rhs))

# (iii) positive-definiteness: <x,x> >= 0
print(np.dot(x, x) >= 0)

```

Example 1.4. Take our vector space to be the set of all continuous functions on the closed interval $C[a, b]$. For f and g in $C([a, b])$ we define their inner product by the integral:

$$\langle f, g \rangle = \int_a^b f(x) g(x) \, dx$$

This also satisfies the inner product properties:

(i) $\langle f, g \rangle = \int_a^b f(x) g(x) \, dx = \int_a^b g(x) f(x) \, dx = \langle g, f \rangle$ (symmetry),

(ii) $\langle \alpha f + g, h \rangle = \int_a^b (\alpha f(x) + g(x)) h(x) \, dx = \alpha \int_a^b f(x) h(x) \, dx + \int_a^b g(x) h(x) \, dx = \alpha \langle f, h \rangle + \langle g, h \rangle$ (linearity),

(iii) $\langle f, f \rangle = \int_a^b (f(x))^2 \, dx \geq 0$. Since f is continuous, the integral is zero if and only if $f(x) = 0$ for all $x \in [a, b]$ (positive-definiteness).

```python
# Inner product for continuous functions on [a,b]
# Approximated numerically with integration
import numpy as np
from scipy.integrate import quad

# define interval
a, b = 0.0, 1.0

# define functions f, g, h (continuous on [a,b])
f = lambda x: np.sin(np.pi * x)
g = lambda x: np.cos(np.pi * x)
h = lambda x: x

# inner product using numerical integration
def inner_product(u, v, a, b):
val, _ = quad(lambda t: u(t) * v(t), a, b)
return val

# compute examples
print(inner_product(f, g, a, b))   # <f,g>
print(inner_product(f, f, a, b))   # <f,f>

# (i) symmetry: <f,g> == <g,f>
lhs = inner_product(f, g, a, b)
rhs = inner_product(g, f, a, b)
print(np.allclose(lhs, rhs))

# (ii) linearity: <a*f + g, h> == a<f,h> + <g,h>
alpha = 2.0
lhs = inner_product(lambda x: alpha*f(x) + g(x), h, a, b)
rhs = alpha*inner_product(f, h, a, b) + inner_product(g, h, a, b)
print(np.allclose(lhs, rhs))

# (iii) positive-definiteness: <f,f> >= 0
val = inner_product(f, f, a, b)
print(val >= 0)
```

Exercise 1.2. ► Give other families of examples of vector spaces admitting an inner product.

► Can you express the inner product on $\mathbb{R}^n$ in terms of *matrix product*? What about $C([a,b])$?

When a real vector space V is endowed with an inner product, geometry comes into play as follows :

► **Length (or Norm):** The length of a vector v is defined as $\|v\| = \sqrt{\langle v, v\rangle}$.

Example 1.5. In $\mathbb{R}^4$: Consider the vector $v = (1, 2, 3, 4)^\top$.

$$\|v\| = \sqrt{\langle v, v\rangle} = \sqrt{1^2 + 2^2 + 3^2 + 4^2} = \sqrt{30}$$

```python
# Norm in R^4
import numpy as np

v = np.array([1, 2, 3, 4])
norm_v = np.linalg.norm(v)

print(norm_v)   # should be sqrt(30)
```

Example 1.6. In $C([0,1])$: Consider the function $f(x) = x$ on the interval $[0,1]$.

$$\|f\| = \sqrt{\int_0^1 (x)^2\, dx} = \sqrt{\left[\frac{x^3}{3}\right]_0^1} = \sqrt{\frac{1^3}{3} - \frac{0^3}{3}} = \sqrt{\frac{1}{3}}$$

```python
# Norm in C([0,1]) for f(x) = x
import numpy as np
from scipy.integrate import quad

f = lambda x: x

# compute inner product <f,f>
val, _ = quad(lambda t: f(t)**2, 0, 1)

norm_f = np.sqrt(val)

print(norm_f)   # should be sqrt(1/3)
```

- **Angle:** The angle θ between two non-zero vectors u and v is given by

$$\cos(\theta) = \frac{\langle u, v\rangle}{\|u\|\|v\|}$$

Example 1.7. In $\mathbb{R}^4$: Consider vectors $u = (1, 0, 1, 0)^\top$ and $v = (1, 1, 0, 1)^\top$.

$$\cos(\theta) = \frac{\langle u, v\rangle}{\|u\|\|v\|} = \frac{(1)(1) + (0)(1) + (1)(0) + (0)(1)}{\sqrt{1^2 + 0^2 + 0^2 + 0^2}\sqrt{1^2 + 1^2 + 0^2 + 0^2}} = \frac{1}{\sqrt{1}\sqrt{2}} = \frac{1}{\sqrt{2}}$$

Therefore, $\theta = \frac{\pi}{4}$.

```python
# Angle between two vectors in R^4
import numpy as np

u = np.array([1, 0, 1, 0])
v = np.array([1, 1, 0, 1])

# inner product <u,v>
inner_uv = np.dot(u, v)

# norms
norm_u = np.linalg.norm(u)
norm_v = np.linalg.norm(v)

# cosine of the angle
cos_theta = inner_uv / (norm_u * norm_v)

# angle in radians
theta = np.arccos(cos_theta)

print("cos(theta) =", cos_theta)
print("theta (radians) =", theta)
```

Example 1.8. In $C([0, 1])$: Consider functions $f(x) = x$ and $g(x) = 1$ on $[0, 1]$.

$$\langle f, g\rangle = \int_0^1 x(1)\,dx = \left[\frac{x^2}{2}\right]_0^1 = \frac{1}{2}$$

$$\|f\| = \sqrt{\int_0^1 x^2\,dx} = \sqrt{\frac{1}{3}} \quad \text{and} \quad \|g\| = \sqrt{\int_0^1 1^2\,dx} = \sqrt{1} = 1$$

$$\cos(\theta) = \frac{\langle f, g\rangle}{\|f\|\|g\|} = \frac{1/2}{\sqrt{1/3} \cdot 1} = \frac{1}{2} \cdot \sqrt{3} = \frac{\sqrt{3}}{2}$$

Therefore, $\theta = \frac{\pi}{6}$.

```python
# Angle between functions f(x)=x and g(x)=1 on [0,1]
import numpy as np
from scipy.integrate import quad

f = lambda x: x
g = lambda x: 1

# inner product <f,g> defined by the integral
inner_fg, _ = quad(lambda t: f(t)*g(t), 0, 1)

# norms
norm_f = np.sqrt(quad(lambda t: f(t)**2, 0, 1)[0])
norm_g = np.sqrt(quad(lambda t: g(t)**2, 0, 1)[0])

# cosine of angle
cos_theta = inner_fg / (norm_f * norm_g)

# angle in radians
theta = np.arccos(cos_theta)

print("cos(theta) =", cos_theta)
print("theta (radians) =", theta)
```

- **Orthogonality:** Two vectors u and v are orthogonal if their inner product is zero: $\langle u, v\rangle = 0$.

Example 1.9. In $\mathbb{R}^4$: Consider vectors $u = (1, 2, 3, 4)^\top$ and $v = (2, -1, -2, 1)^\top$.

$$\langle u, v\rangle = (1)(2) + (2)(-1) + (3)(-2) + (4)(1) = 2 - 2 - 6 + 4 = 0$$

Since the inner product is 0, the vectors are orthogonal.

```python
# Orthogonality in R^4
import numpy as np

u = np.array([1, 2, 3, 4])
v = np.array([2, -1, -2, 1])

inner_uv = np.dot(u, v)

print("Inner product <u,v> =", inner_uv)
print("Are u and v orthogonal?", np.isclose(inner_uv, 0.0))
```

Example 1.10. In $C([-\pi, \pi])$: Consider the functions $f(x) = \sin(x)$ and $g(x) = \cos(x)$ on $[-\pi, \pi]$.

$$\langle f, g\rangle = \int_{-\pi}^{\pi} \sin(x)\cos(x)\,dx = \int_{-\pi}^{\pi} \frac{1}{2}\sin(2x)\,dx = \left[-\frac{1}{4}\cos(2x)\right]_{-\pi}^{\pi} = 0$$

Since the inner product is 0, the functions are orthogonal on this interval.

```python
# Orthogonality of sin(x) and cos(x) on [-pi, pi]
import numpy as np
from scipy.integrate import quad

f = lambda x: np.sin(x)
g = lambda x: np.cos(x)

inner_fg, _ = quad(lambda t: f(t)*g(t), -np.pi, np.pi)

print("Inner product <f,g> =", inner_fg)
print("Are f and g orthogonal?", np.isclose(inner_fg, 0.0))
```

- **Projection:** The projection of vector v onto vector u is given by

$$\text{proj}_u v = \frac{\langle v, u\rangle}{\langle u, u\rangle} u$$

Example 1.11. In $\mathbb{R}^4$: Consider the projection of $v = (1, 2, 3, 4)^\top$ onto $u = (1, 0, 0, 0)^\top$.

$$\begin{aligned}\text{proj}_u v = \frac{\langle v, u\rangle}{\langle u, u\rangle} u &= \frac{(1)(1) + (2)(0) + (3)(0) + (4)(0)}{1^2 + 0^2 + 0^2 + 0^2}(1, 0, 0, 0)^\top \\ &= \frac{1}{1}(1, 0, 0, 0)^\top = (1, 0, 0, 0)^\top\end{aligned}$$

```python
# Projection of v onto u in R^4
import numpy as np

v = np.array([1, 2, 3, 4])
u = np.array([1, 0, 0, 0])

proj_u_v = (np.dot(v, u) / np.dot(u, u)) * u

print("Projection of v onto u =", proj_u_v)
```

Example 1.12. In $C([0, \pi])$: Consider the projection of $g(x) = x$ onto $f(x) = \sin(x)$ on $[0, \pi]$.

$$\text{proj}_f g = \frac{\langle g, f\rangle}{\langle f, f\rangle} f(x) = \frac{\int_0^\pi x\sin(x)\,dx}{\int_0^\pi \sin^2(x)\,dx}\sin(x) = \frac{\pi}{\pi/2}\sin(x) = 2\sin(x)$$

```python
# Projection of g(x)=x onto f(x)=sin(x) on [0, pi]
import numpy as np
from scipy.integrate import quad

f = lambda x: np.sin(x)
g = lambda x: x

# inner products
inner_gf, _ = quad(lambda t: g(t)*f(t), 0, np.pi)
inner_ff, _ = quad(lambda t: f(t)*f(t), 0, np.pi)

# projection coefficient
coef = inner_gf / inner_ff

print("Coefficient multiplying sin(x) =", coef)
# Projection is coef * sin(x), i.e. 2*sin(x)

```

- **Pythagorean Theorem:** If u and v are orthogonal vectors, then $\|u+v\|^2 = \|u\|^2 + \|v\|^2$.

  Example 1.13. In $\mathbb{R}^4$: Consider vectors $u = (1, 0, 0, 0)$ and $v = (0, 1, 0, 0)$.

$$\|u+v\| = \|(1,1,0,0)\| = \sqrt{1^2+1^2} = \sqrt{2}$$

$$\|u\| + \|v\| = \sqrt{1^2} + \sqrt{1^2} = 1 + 1 = 2$$

  Here, $\|u+v\| = \sqrt{2} \leq 2 = \|u\| + \|v\|$.

  Example 1.14. In $C([-\pi, \pi])$: Consider orthogonal functions $f(x) = \sin(x)$ and $g(x) = \cos(x)$ on $[-\pi, \pi]$.

$$\|f+g\|^2 = \int_{-\pi}^{\pi} (\sin(x) + \cos(x))^2 \, dx = \int_{-\pi}^{\pi} (1 + \sin(2x)) \, dx = 2\pi$$

$$\|f\|^2 + \|g\|^2 = \int_{-\pi}^{\pi} \sin^2(x) \, dx + \int_{-\pi}^{\pi} \cos^2(x) \, dx = \int_{-\pi}^{\pi} 1 \, dx = 2\pi$$

  Thus, $\|f+g\|^2 = \|f\|^2 + \|g\|^2$.

- **Triangle Inequality:** For any two vectors u and v, the length of their sum is no more than the sum of their individual lengths: $\|u+v\| \leq \|u\| + \|v\|$.

  Example 1.15. In $\mathbb{R}^n$: Consider vectors $u = (1, 0)$ and $v = (0, 1)$ in $\mathbb{R}^2$.

$$\|u+v\| = \|(1,1)\| = \sqrt{1^2+1^2} = \sqrt{2} \approx 1.414$$

$$\|u\| + \|v\| = \sqrt{1^2+0^2} + \sqrt{0^2+1^2} = 1 + 1 = 2$$

  Here, $\|u+v\| = \sqrt{2} \leq 2 = \|u\| + \|v\|$.

```python
# Triangle inequality in R^4
import numpy as np

u = np.array([1, 0, 0, 0])
v = np.array([0, 1, 0, 0])

norm_u = np.linalg.norm(u)
norm_v = np.linalg.norm(v)
norm_u_plus_v = np.linalg.norm(u + v)

print("||u+v|| =", norm_u_plus_v)
print("||u|| + ||v|| =", norm_u + norm_v)
print("Triangle inequality holds?", norm_u_plus_v <= norm_u + norm_v)

```

  Example 1.16. In $C([0, 1])$: Consider functions $f(x) = x$ and $g(x) = 1$ on $[0, 1]$.

$$\|f+g\| = \sqrt{\int_0^1 (x+1)^2 \, dx} = \sqrt{\int_0^1 (x^2+2x+1) \, dx} = \sqrt{\frac{7}{3}} \approx 1.528$$

$$\|f\| + \|g\| = \sqrt{\int_0^1 x^2 \, dx} + \sqrt{\int_0^1 1^2 \, dx} = \sqrt{\frac{1}{3}} + 1 \approx 1.577$$

  Here, $\|f+g\| = \sqrt{\frac{7}{3}} \leq \frac{1}{\sqrt{3}} + 1 = \|f\| + \|g\|$.

```python
# Triangle inequality for functions f(x)=x and g(x)=1 on [0,1]
import numpy as np
from scipy.integrate import quad

f = lambda x: x
g = lambda x: 1

# compute squared norms
norm_f_sq, _ = quad(lambda t: f(t)**2, 0, 1)
norm_g_sq, _ = quad(lambda t: g(t)**2, 0, 1)
norm_f_plus_g_sq, _ = quad(lambda t: (f(t)+g(t))**2, 0, 1)

# norms
norm_f = np.sqrt(norm_f_sq)
norm_g = np.sqrt(norm_g_sq)
norm_f_plus_g = np.sqrt(norm_f_plus_g_sq)

print("||f|| =", norm_f)
print("||g|| =", norm_g)
print("||f+g|| =", norm_f_plus_g)
print("Triangle inequality holds?", norm_f_plus_g <= norm_f + norm_g)
```