Bab 04 — Matematika AI yang Menyenangkan

Status: Draft lengkap v0.3 untuk review editorial Bagian buku: Level C — Matematika yang Ramah Target pembaca: Pembaca yang merasa matematika menakutkan, tetapi ingin memahami AI dengan fondasi kuat dan menyenangkan.

Cara membaca bab ini

Bab ini bukan ujian matematika. Bab ini adalah taman bermain geometri AI. Kita akan melihat bagaimana angka berubah menjadi titik, panah, jarak, arah, tabel, garis pemisah, dan ruang fitur.

Jika kamu pernah merasa “saya tidak berbakat matematika”, pegang kalimat ini:

> Matematika AI bukan bakat misterius. Ia adalah bahasa untuk menulis pola yang sebenarnya bisa dilihat, dimainkan, dan dicoba dengan kode kecil.

Catatan motivasi:
Setiap kali melihat rumus di bab ini, jangan bertanya “kenapa saya tidak paham?”. Bertanyalah: “cerita apa yang sedang diringkas rumus ini?”

Pendalaman awal — sejarah singkat matematika yang menjadi bahasa AI

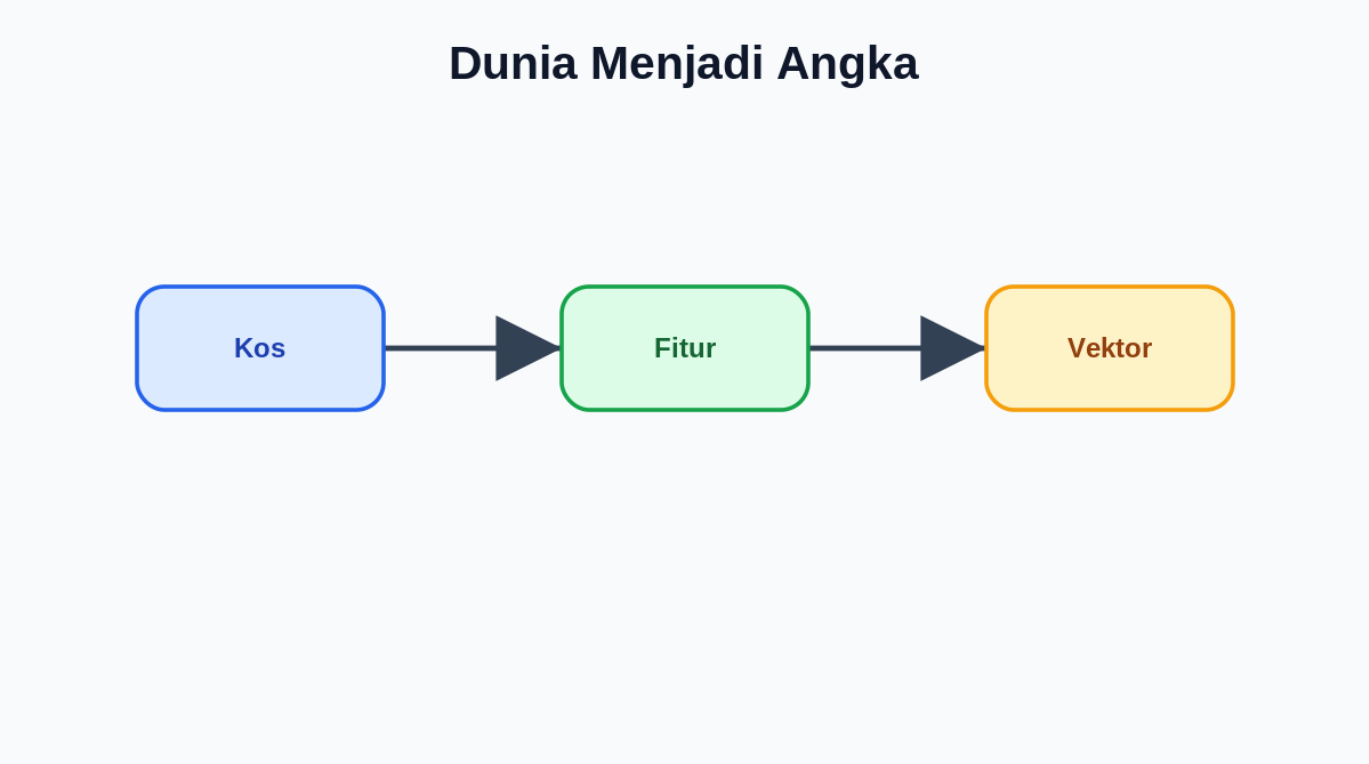
Sebelum kata *artificial intelligence* populer, matematika yang menopang AI sudah hidup lama. Geometri Euclid memperkenalkan cara berpikir tentang jarak dan bentuk. Aljabar linear tumbuh dari kebutuhan menyelesaikan banyak persamaan sekaligus. Kalkulus memberi bahasa perubahan. Statistik dan probabilitas memberi cara membaca ketidakpastian. Komputer modern kemudian membuat semua bahasa ini bisa dihitung cepat pada data besar.

AI modern memakai matematika bukan untuk membuat pembaca takut, tetapi karena model perlu cara yang konsisten untuk menjawab pertanyaan seperti: “seberapa dekat dua objek?”, “fitur mana yang paling berpengaruh?”, “ke arah mana parameter harus digeser?”, dan “bagaimana data berdimensi besar diringkas?”. Vektor, matriks, dot product, norma, dan proyeksi adalah alfabetnya.

Dalam bab ini, setiap konsep akan dibaca dengan empat lapis: makna visual, persamaan matematika, contoh hitung manual, lalu hubungan ke model AI. Tujuannya bukan sekadar mengenal istilah, tetapi bisa menghitung contoh kecil di kertas dan mengerti mengapa rumus itu berguna.

Subbab 1 — Mengapa AI butuh matematika?

AI tidak melihat dunia seperti manusia. Manusia melihat “rumah dekat stasiun”, “pesan mencurigakan”, “foto daun sakit”, atau “pelanggan sering beli kopi”. Komputer melihat angka. Maka langkah pertama AI adalah mengubah dunia menjadi angka yang tetap bermakna.



Dunia menjadi angka

Contoh: sebuah kos bisa ditulis sebagai data:

Fitur	Nilai
jarak ke kampus	1.2 km
harga per bulan	900000
luas kamar	12 m²
ada AC	1
rating	4.6

Ketika fitur-fitur itu disusun, kita mendapat vektor: daftar angka yang mewakili objek. Dari sini AI bisa menghitung jarak, kemiripan, prediksi, dan keputusan.

Catatan hijau — kata kunci: Fitur
Fitur adalah pertanyaan yang kita ubah menjadi angka. Model hanya bisa belajar dari fitur yang kita berikan atau fitur yang ia pelajari.

Latihan ketik manual 1A

```
kos = [1.2, 900000, 12, 1, 4.6]
print(kos)
print("rating kos:", kos[4])
```

Tes cepat subbab 1

1. Mengapa AI perlu mengubah objek menjadi angka?

2. Sebutkan 5 fitur untuk menilai warung makan.

3. Apa risiko jika fitur penting tidak dimasukkan?

Subbab 2 — Vektor: profil objek dalam bentuk angka

Vektor adalah daftar angka yang punya urutan. Dalam AI, vektor sering menjadi profil sebuah objek. Satu pelanggan, satu produk, satu rumah, satu kata, bahkan satu gambar bisa diubah menjadi vektor.

Vektor sebagai profil

Contoh pelanggan warung:

```
pelanggan_a = [5, 2, 1]  # kopi, es_teh, roti
pelanggan_b = [1, 5, 2]
```

Artinya pelanggan A lebih sering beli kopi, pelanggan B lebih sering beli es teh. Tanpa gambar pun, angka sudah mulai bercerita.

Rumus vektor ditulis seperti ini:

```
x = [x1, x2, x3, ..., xn]
```

Terjemahan manusia:

- x: nama vektor.
- `x1, x2, ...`: isi angka dalam vektor.
- n: jumlah fitur/dimensi.

Catatan kuning — kata kunci: Dimensi
Dimensi adalah jumlah arah/fitur yang dipakai. Vektor 2D mudah digambar; vektor 100D sulit digambar, tetapi idenya sama: daftar angka.

Latihan ketik manual 2A

```
produk = [25000, 4.8, 120]  # harga, rating, jumlah terjual
print("harga:", produk[0])
print("rating:", produk[1])
print("terjual:", produk[2])
```

Pendalaman matematika vektor

Secara formal, vektor berdimensi n ditulis:

$$x = [x_1, x_2, ..., x_n]$$

Jika sebuah produk punya fitur `[rating, diskon, klik] = [4,5, 0,2, 6]`, maka model tidak melihat produk sebagai cerita panjang, tetapi sebagai titik di ruang 3 dimensi. Setiap dimensi adalah satu fitur.

Contoh hitung terstruktur

Produk A: `[4,5, 0,2, 6]` Produk B: `[3,0, 0,1, 2]`

Selisih fitur:

$$A - B = [4,5-3,0, 0,2-0,1, 6-2]$$
$$= [1,5, 0,1, 4]$$

Makna: A lebih tinggi rating 1,5 poin, diskon lebih besar 0,1, dan klik lebih banyak 4. Operasi vektor membuat perbandingan banyak fitur bisa dilakukan sekaligus.

Tes cepat subbab 2

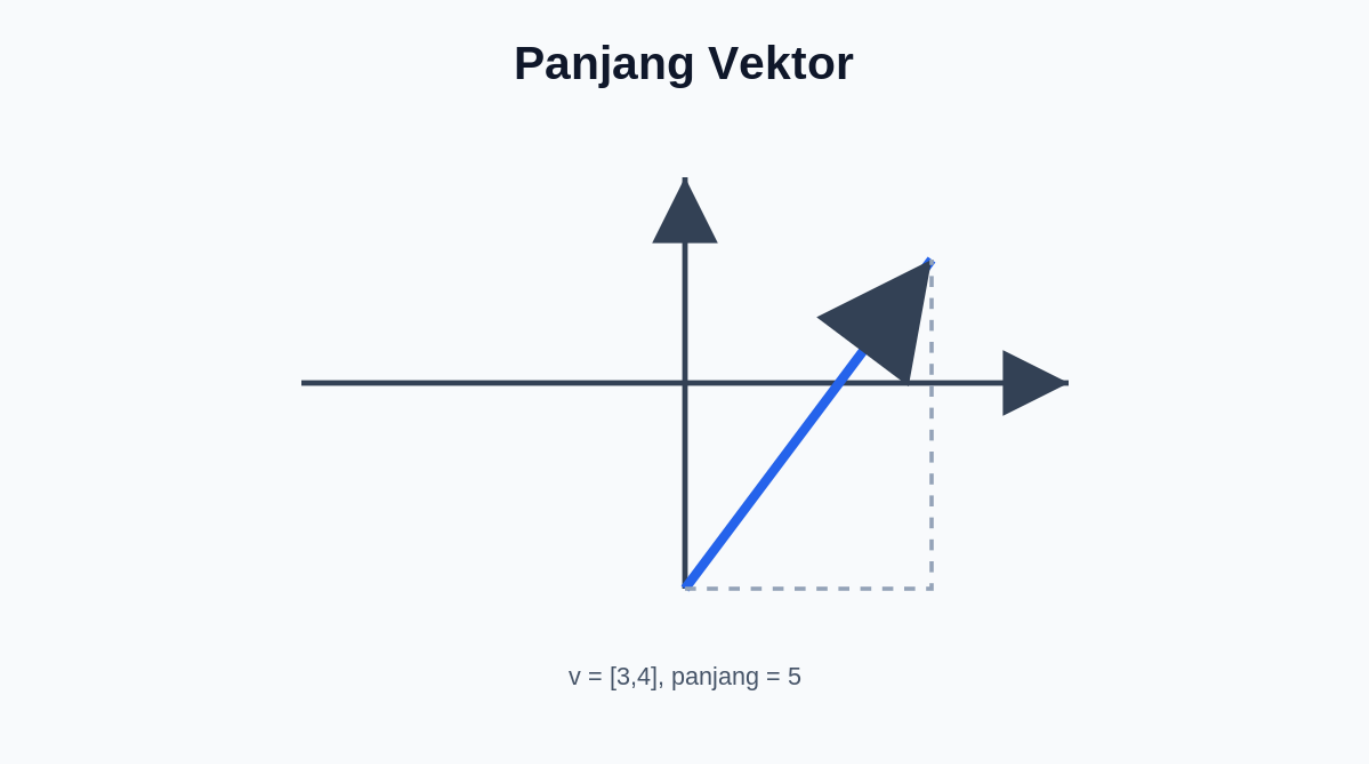
1. Apa itu vektor dalam bahasa sederhana?

2. Apa arti dimensi?

3. Buat vektor 4 fitur untuk sebuah produk marketplace.

Subbab 3 — Panjang vektor: seberapa besar langkahnya?

Bayangkan kamu berjalan 3 langkah ke timur dan 4 langkah ke utara. Walau total langkah horizontal+vertikal adalah 7, jarak lurus dari titik awal ke akhir adalah 5. Ini adalah ide panjang vektor.



Panjang vektor

Untuk vektor 2D `v = [x, y]`, panjangnya:

$$||v|| = \sqrt{x^2 + y^2}$$

Terjemahan manusia:

- `||v||`: panjang vektor.
- `sqrt`: akar kuadrat.
- $x^2 + y^2$: gabungan kontribusi arah horizontal dan vertikal.

Contoh:

```
import math
x = 3
y = 4
panjang = math.sqrt(x*x + y*y)
print(panjang)  # 5.0
```

Catatan ungu — intuisi:
Panjang vektor menjawab: “seberapa besar profil/langkah ini?” Dalam AI, panjang bisa terkait intensitas, skala, atau kekuatan sinyal.

Dari Teorema Pythagoras ke norma vektor

Untuk vektor 2D `[a,b]`, panjangnya mengikuti Pythagoras:

$$||x|| = \sqrt{a^2 + b^2}$$

Untuk n dimensi:

$$||x||_2 = \text{sqrt}(x_1^2 + x_2^2 + ... + x_n^2)$$

Contoh hitung terstruktur

Jika x=[3,4]:

$$||x|| = \text{sqrt}(3^2 + 4^2)$$
$$= \text{sqrt}(9 + 16)$$
$$= \text{sqrt}(25)$$
$$= 5$$

Jika x=[1,2,2]:

$$||x|| = \text{sqrt}(1^2 + 2^2 + 2^2)$$
$$= \text{sqrt}(1+4+4)$$
$$= 3$$

Dalam AI, panjang vektor dapat mewakili besar sinyal. Tetapi hati-hati: vektor panjang belum tentu “lebih baik”; bisa jadi hanya karena skala fitur lebih besar.

Tes cepat subbab 3

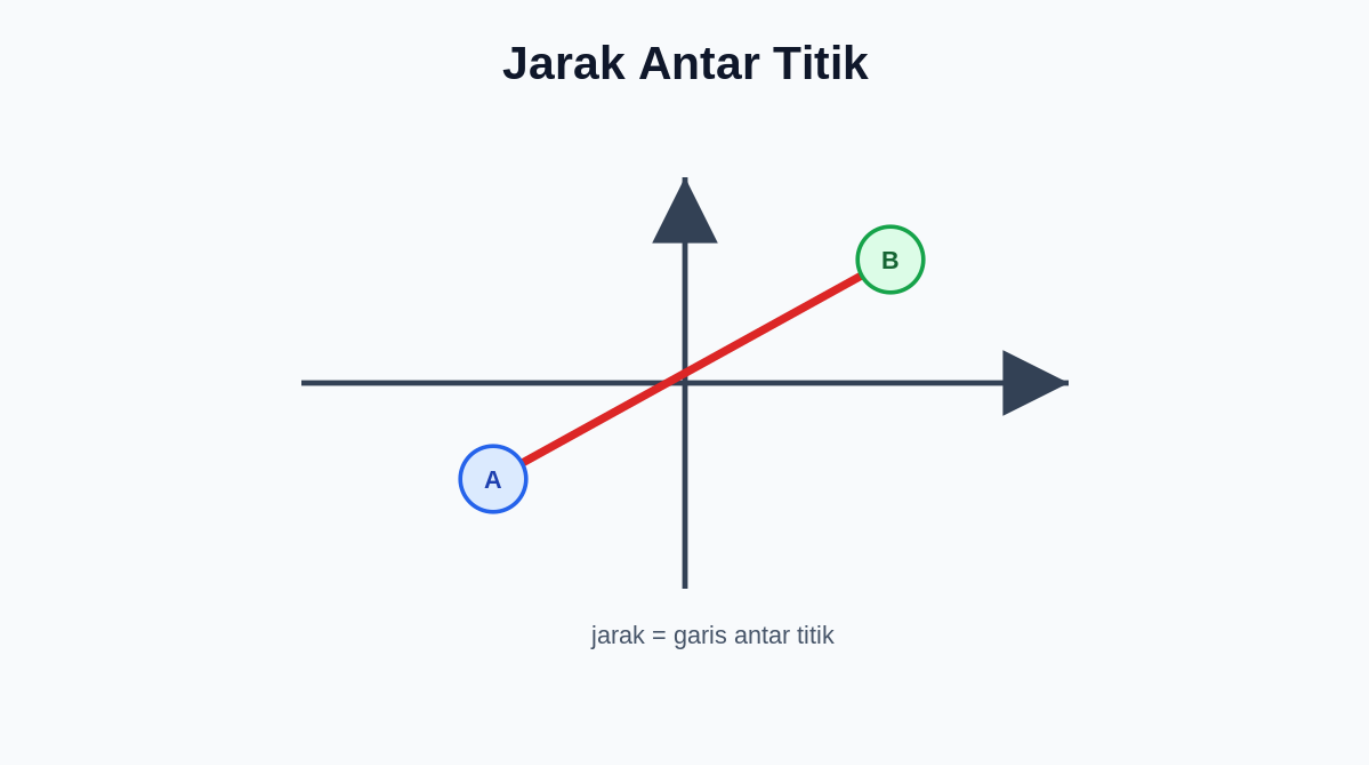
1. Hitung panjang vektor [6, 8].

2. Mengapa [3, 4] panjangnya 5, bukan 7?

3. Apa arti panjang vektor untuk profil pelanggan?

Subbab 4 — Jarak: seberapa mirip dua objek?

Jika dua kos punya fitur yang mirip, titiknya berdekatan di ruang fitur. Jika fiturnya sangat berbeda, titiknya berjauhan. Jarak membantu AI menjawab “mana yang mirip?”.



Jarak antar titik

Misal dua produk:

```
produk_a = [25000, 4.8]
produk_b = [27000, 4.7]
produk_c = [90000, 3.5]
```

Jika harga dan rating dinormalisasi, produk A dan B mungkin dekat, sedangkan C jauh. Rumus jarak Euclidean untuk dua vektor 2D:

```
jarak(a,b) = sqrt((a1-b1)² + (a2-b2)²)
```

Terjemahan manusia: kurangi fitur satu per satu, kuadratkan agar selisih negatif tidak membatalkan, jumlahkan, lalu akar.

Latihan ketik manual 4A

```
import math

a = [3, 2]
b = [1, 4]
jarak = math.sqrt((a[0]-b[0])**2 + (a[1]-b[1])**2)
print(jarak)
```

Catatan merah — hati-hati skala:
Jika satu fitur memakai rupiah jutaan dan fitur lain rating 1–5, fitur rupiah bisa mendominasi jarak. Karena itu nanti kita belajar scaling/normalization.

Euclidean distance sebagai panjang selisih

Jarak dua vektor adalah panjang dari selisihnya:

```
d(x,y) = ||x-y||₂ = sqrt(Σ(xᵢ-yᵢ)²)
```

Contoh hitung terstruktur

A `[3,2]`, B `[1,5]`:

A-B = [2, -3]
d(A,B) = sqrt(2^2 + $(-3)^2$)
= sqrt(4 + 9)
= sqrt(13)
≈ 3,61

Jika C `[3,3]`:

A-C = [0, -1]
d(A,C) = 1

Maka C lebih dekat ke A daripada B. Inilah dasar kNN dan pencarian kemiripan.

Tes cepat subbab 4

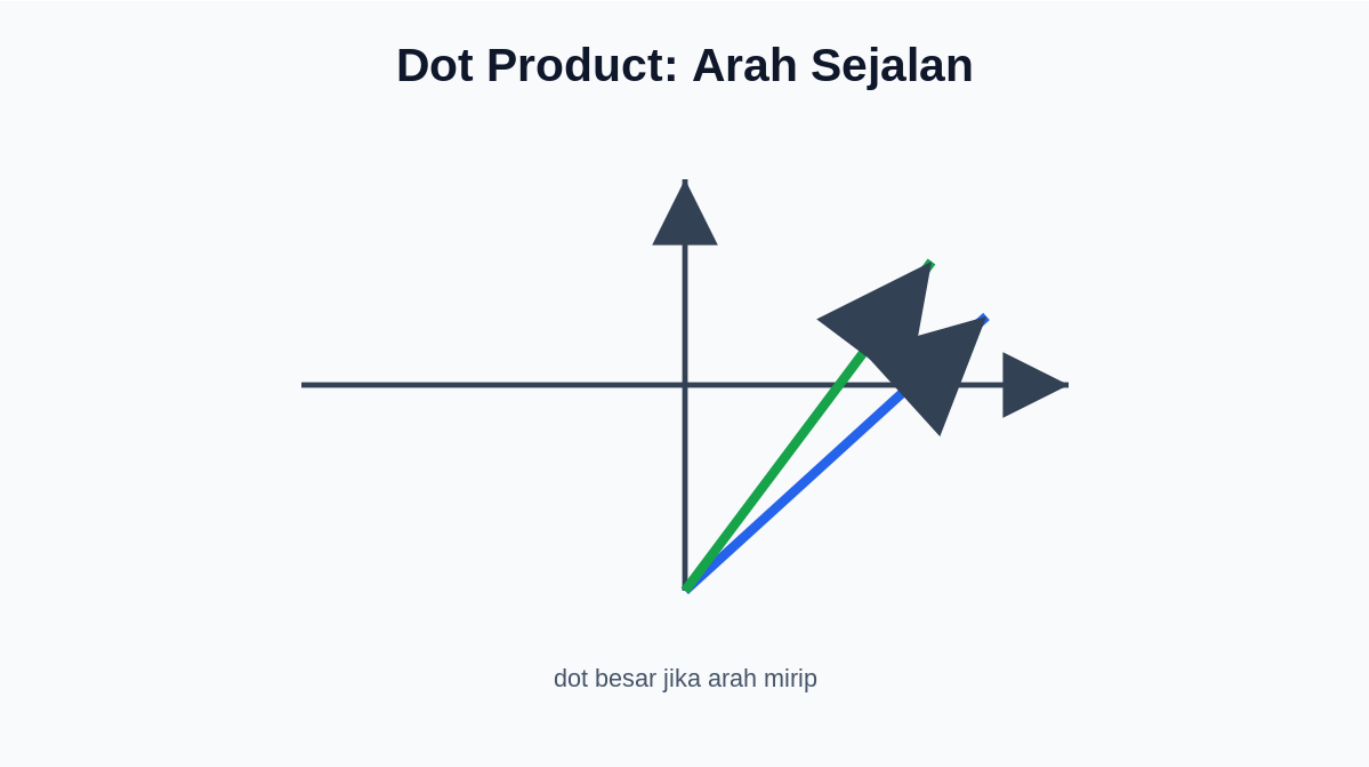
1. Apa arti jarak kecil antar dua vektor?

2. Mengapa skala fitur bisa berbahaya?

3. Hitung jarak antara `[0,0]` dan `[3,4]`.

Subbab 5 — Dot product: apakah dua arah sejalan?

Jarak menjawab “seberapa dekat?”. Dot product menjawab “seberapa searah?”. Dua vektor yang menunjuk arah mirip punya dot product besar. Jika berlawanan, dot product bisa negatif.



Dot product arah

Rumus:

$$a \cdot b = a1*b1 + a2*b2 + \ldots + an*bn$$

Terjemahan manusia: kalikan fitur yang posisinya sama, lalu jumlahkan.

Contoh:

```
a = [3, 2]
b = [1, 4]
dot = a[0]*b[0] + a[1]*b[1]
print(dot)  # 11
```

Dalam rekomendasi, dot product bisa dipakai sebagai skor kecocokan kasar. Misal profil pengguna dan profil produk sama-sama memakai fitur “murah”, “kopi”, “manis”. Jika arahnya sejalan, skornya tinggi.

Catatan biru — kata kunci: Kesearahan
Dot product bukan sekadar perkalian. Ia memberi sinyal apakah dua vektor “mendukung arah yang sama”.

Persamaan dan makna dot product

$$x \cdot y = x_1y_1 + x_2y_2 + \ldots + x_ny_n$$

Secara geometri:

$$x \cdot y = ||x||\ ||y||\ \cos(\theta)$$

Jika dot product positif besar, dua vektor cenderung searah. Jika mendekati nol, arahnya hampir tegak lurus. Jika negatif, arahnya berlawanan.

Contoh hitung terstruktur

A `[3,2]`, B `[1,4]`:

A·B = 3×1 + 2×4 = 3 + 8 = 11

Jika bobot model `w=[0,7, 0,3]` dan fitur pelanggan `x=[10, 2]`:

w·x = 0,7×10 + 0,3×2 = 7 + 0,6 = 7,6

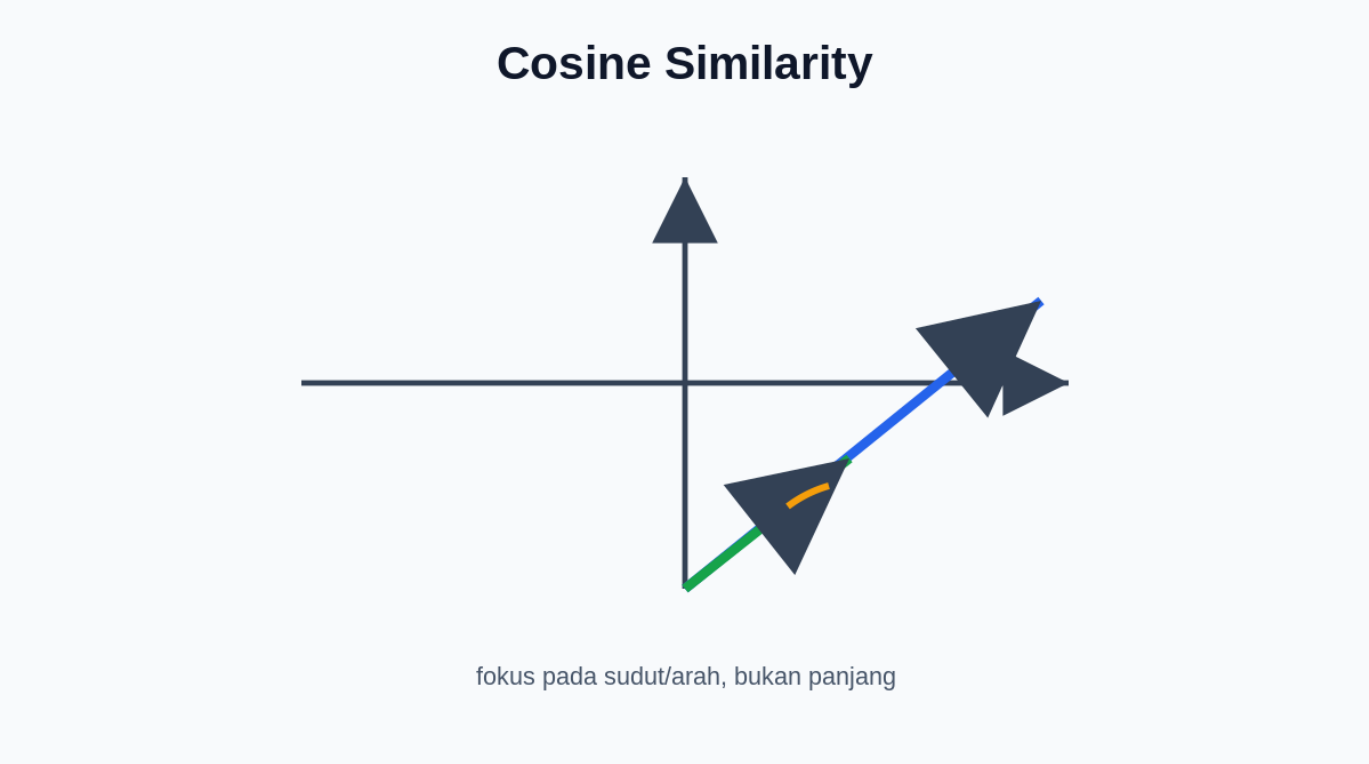
Dot product adalah mesin kecil di balik skor linear.

Tes cepat subbab 5

1. Hitung dot product `[2, 3]` dan `[4, 1]`.
2. Apa intuisi dot product besar?
3. Mengapa dot product berguna untuk rekomendasi?

Subbab 6 — Cosine similarity: mirip arah walau beda ukuran

Kadang kita peduli arah, bukan besar. Dua pelanggan bisa sama-sama suka kopi dan roti, tetapi satu belanja sedikit, satu belanja banyak. Cosine similarity membantu membandingkan arah setelah mengurangi efek panjang.



Cosine similarity

Rumus:

```
cosine(a,b) = (a · b) / (||a|| ||b||)
```

Terjemahan manusia:

- a · b: seberapa searah.
- ||a|| dan ||b||: panjang masing-masing vektor.
- pembagian membuat skor fokus pada arah.

Skor cosine biasanya antara -1 sampai 1. Untuk banyak data non-negatif seperti pembelian, sering berada antara 0 sampai 1.

Latihan ketik manual 6A

```
import math

def dot(a, b):
    return sum(x*y for x, y in zip(a, b))

def length(v):
    return math.sqrt(sum(x*x for x in v))

def cosine(a, b):
    return dot(a, b) / (length(a) * length(b))

print(cosine([5, 2, 1], [10, 4, 2]))
print(cosine([5, 2, 1], [1, 5, 2]))
```

Persamaan cosine similarity

```
cosine(x,y) = (x·y) / (||x|| ||y||)
```

Nilainya biasanya antara -1 dan 1. Untuk fitur non-negatif seperti klik atau frekuensi kata, nilainya sering 0 sampai 1.

Contoh hitung terstruktur

A `[1,2]`, B `[2,4]`:

A·B = 1×2 + 2×4 = 10
||A|| = sqrt(1²+2²) = sqrt(5)
||B|| = sqrt(2²+4²) = sqrt(20)
cosine = 10 / (sqrt(5)×sqrt(20)) = 10/10 = 1

A dan B arahnya sama meski B lebih besar. Ini berguna untuk rekomendasi dan teks: dua pengguna bisa punya intensitas aktivitas berbeda tetapi pola minat searah.

Tes cepat subbab 6

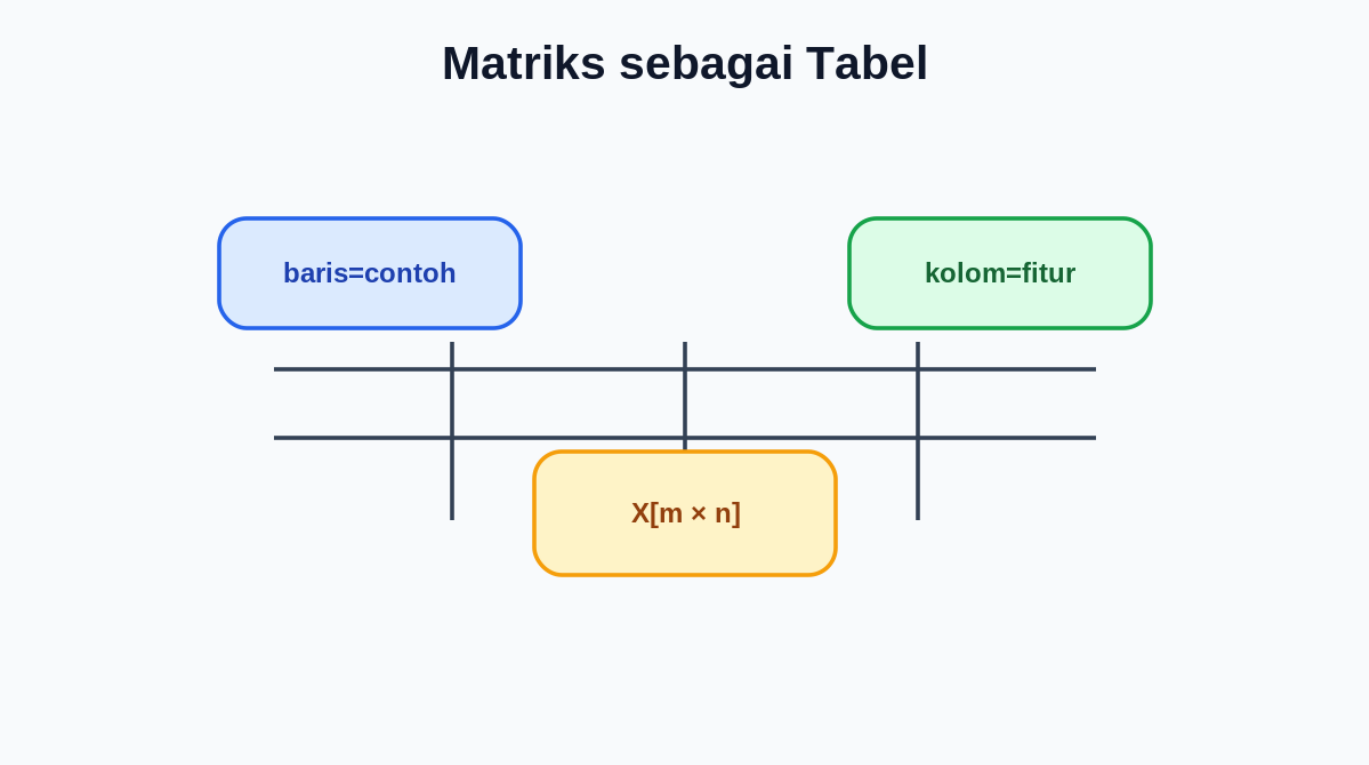
1. Apa bedanya dot product dan cosine similarity?

2. Kapan cosine lebih adil daripada dot product?

3. Mengapa profil `[5,2,1]` dan `[10,4,2]` sangat mirip arah?

Subbab 7 — Matriks: banyak vektor dalam satu tabel

Jika satu vektor adalah satu profil, matriks adalah banyak profil yang disusun menjadi tabel. Dataset tabular adalah matriks: baris adalah contoh, kolom adalah fitur.



Matriks sebagai tabel

Contoh:

```
X = [
  [25000, 4.8, 120],
  [27000, 4.7, 100],
  [90000, 3.5,  12],
]
```

Rumus umum:

```
X memiliki bentuk m × n
```

Terjemahan manusia:

- `m`: jumlah baris/contoh.
- `n`: jumlah kolom/fitur.
- `X[i][j]`: nilai pada baris ke-i, kolom ke-j.

Latihan ketik manual 7A

```
X = [
   [25000, 4.8, 120],
   [27000, 4.7, 100],
   [90000, 3.5, 12],
]
print("jumlah baris:", len(X))
print("jumlah kolom:", len(X[0]))
print("rating produk kedua:", X[1][1])
```

Catatan hijau:
Ketika kamu melihat dataset, bayangkan matriks. Ketika kamu melihat satu baris dataset, bayangkan vektor.

Matriks sebagai kumpulan baris dan kolom

Matriks X dengan m baris dan n kolom sering ditulis:

```
X ∈ R^(m×n)
```

m = jumlah contoh, n = jumlah fitur.

Contoh hitung struktur data

```
X = [[4,5, 0,2, 6],
     [3,0, 0,1, 2],
     [4,1, 0,3, 5]]
```

Matriks ini punya 3 baris dan 3 kolom. Baris pertama adalah produk pertama; kolom pertama adalah rating semua produk. Dalam machine learning, dataset tabular sering menjadi matriks fitur seperti ini.

Tes cepat subbab 7

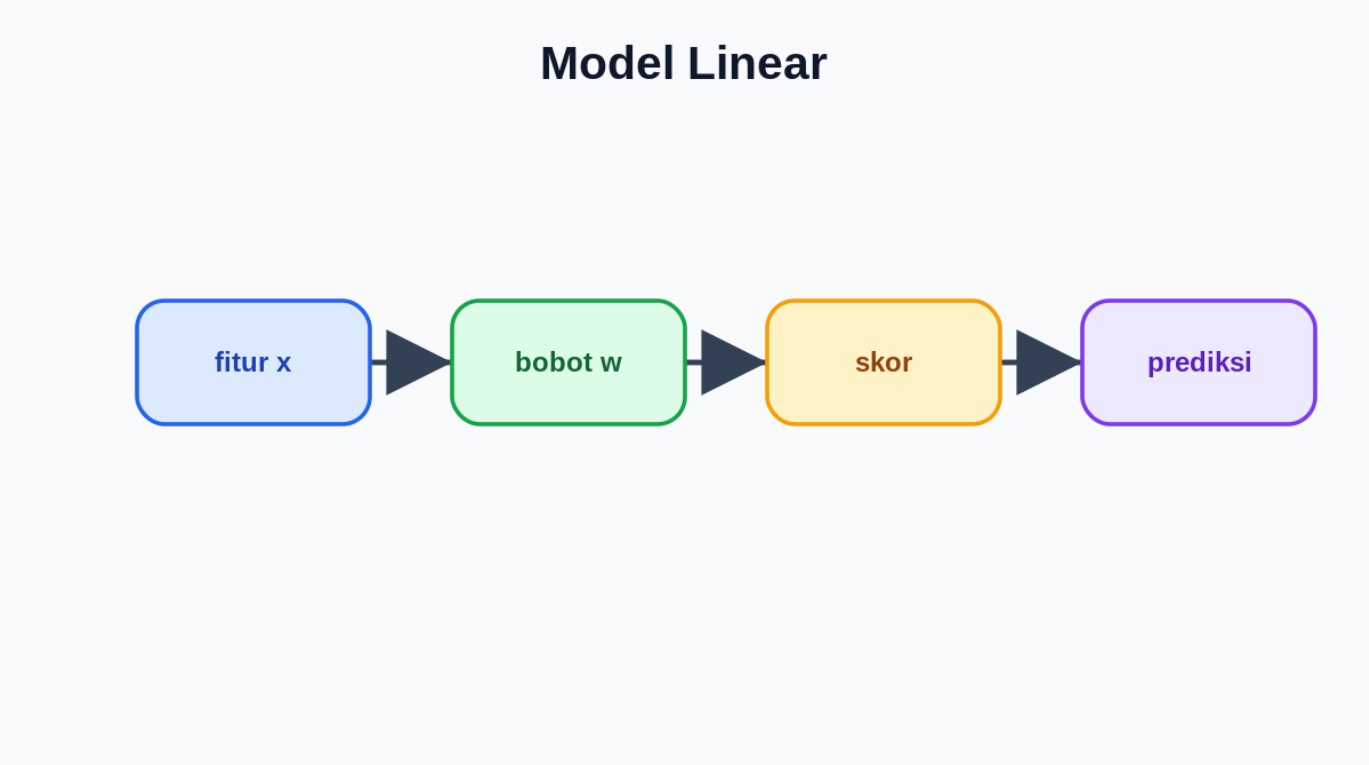
1. Apa perbedaan vektor dan matriks?

2. Dalam dataset 100 baris dan 5 fitur, bentuk matriksnya apa?

3. Apa arti baris dan kolom dalam dataset?

Subbab 8 — Matrix-vector multiplication: resep prediksi linear

Model linear sederhana dapat dilihat sebagai dot product antara fitur dan bobot. Bobot menunjukkan seberapa penting fitur untuk prediksi.



Model linear

Misal prediksi skor produk:

```
skor = harga_bobot*harga + rating_bobot*rating + terjual_bobot*terjual + bias
```

Dalam notasi singkat:

$$\hat{y} = w \cdot x + b$$

Terjemahan manusia:

- x: fitur produk.
- w: bobot model.
- b: bias/intercept, nilai dasar.
- ■: prediksi.

Latihan ketik manual 8A

```
x = [0.2, 0.9, 0.7]      # fitur yang sudah dinormalisasi
w = [-0.3, 0.8, 0.5]     # bobot
b = 0.1
skor = w[0]*x[0] + w[1]*x[1] + w[2]*x[2] + b
print(skor)
```

Catatan kuning:
Model linear bukan kuno. Ia adalah fondasi untuk memahami regression, classification, neural network, dan decision boundary.

Perkalian matriks-vektor

Jika X berisi banyak contoh dan w berisi bobot fitur, prediksi linear untuk semua baris dapat ditulis:

$$\hat{y} = Xw + b$$

Contoh hitung terstruktur

```
X = [[2, 1],
     [4, 3]]
w = [0,5, 2]
b = 1
```

Baris 1:

$$\hat{y}_1 = 2\times0{,}5 + 1\times2 + 1 = 1 + 2 + 1 = 4$$

Baris 2:

$$\hat{y}_2 = 4\times0{,}5 + 3\times2 + 1 = 2 + 6 + 1 = 9$$

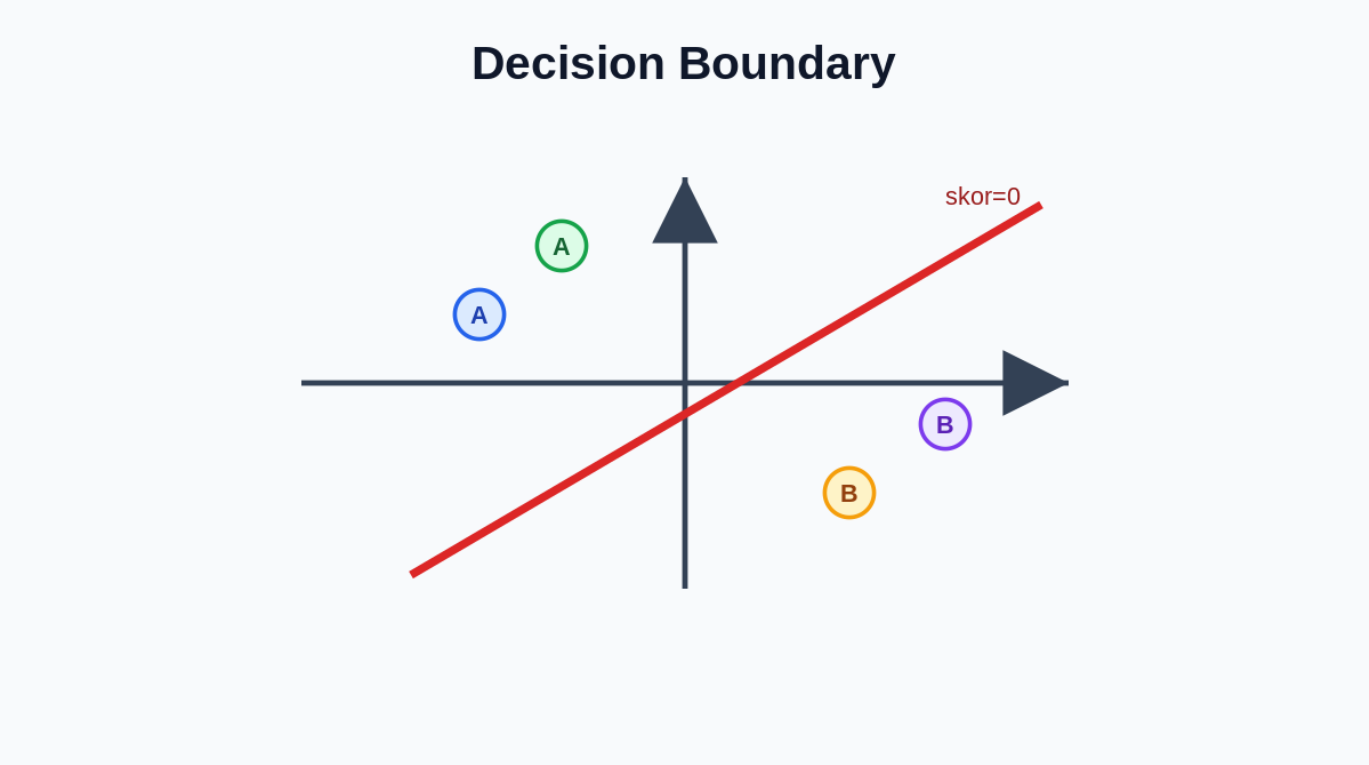
Dengan satu rumus, model dapat menghitung banyak prediksi.

Tes cepat subbab 8

1. Apa arti bobot `w`?
2. Mengapa fitur sering perlu dinormalisasi?
3. Hitung skor jika `x=[1,2]`, `w=[3,4]`, `b=5`.

Subbab 9 — Decision boundary: garis pemisah keputusan

Classification sering berarti memisahkan dua kelompok. Di ruang 2D, pemisah sederhana bisa berupa garis. Garis ini disebut decision boundary.



Decision boundary

Misal kita ingin memisahkan buah layak jual dan tidak layak jual berdasarkan dua fitur: warna dan tekstur. Model linear membuat skor:

```
skor = w1*x1 + w2*x2 + b
```

Jika skor ≥ 0, kelas A. Jika skor < 0, kelas B.

Terjemahan manusia: garis pemisah adalah tempat skor sama dengan nol.

Latihan ketik manual 9A

```
def predict(x1, x2):
    skor = 1.2*x1 + 0.8*x2 - 1.0
    return "layak" if skor >= 0 else "tidak layak"

print(predict(0.9, 0.8))
print(predict(0.2, 0.3))
```

Catatan merah:
Garis pemisah bisa terlihat rapi di gambar, tetapi dunia nyata sering berantakan. Karena itu evaluasi dan error analysis tetap wajib.

Persamaan boundary linear

Untuk dua fitur:

$$w_1x_1 + w_2x_2 + b = 0$$

Sisi positif diprediksi kelas 1, sisi negatif kelas 0.

Contoh hitung terstruktur

Misalkan:

$$\text{skor} = 2 \times x_1 + 1 \times x_2 - 5$$

Boundary:

$$2x_1 + x_2 - 5 = 0$$
$$x_2 = 5 - 2x_1$$

Titik A $(2, 2)$:

$\text{skor} = 2 \times 2 + 2 - 5 = 1 \rightarrow$ kelas positif

Titik B $(1, 1)$:

$\text{skor} = 2 \times 1 + 1 - 5 = -2 \rightarrow$ kelas negatif

Tes cepat subbab 9

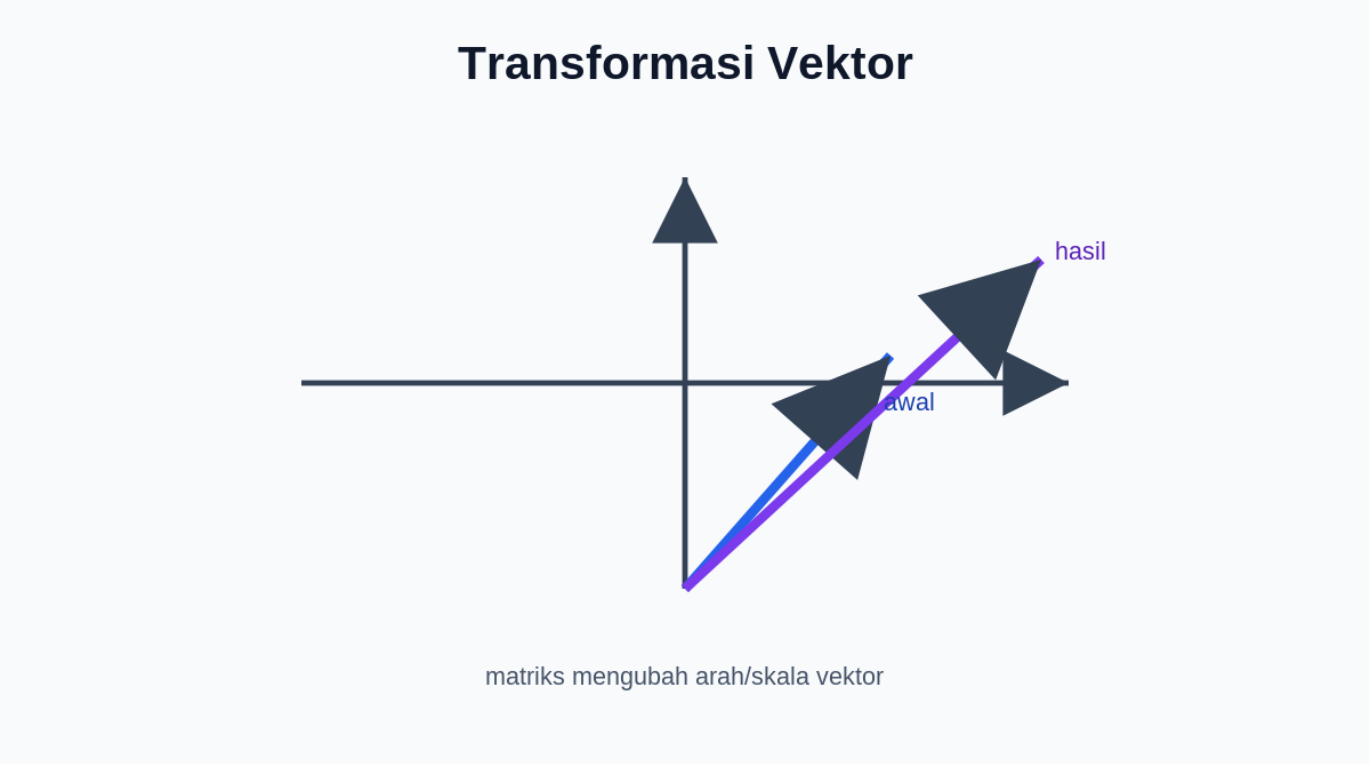
1. Apa itu decision boundary?

2. Apa arti skor ≥ 0 dalam contoh ini?

3. Mengapa garis pemisah sederhana bisa gagal?

Subbab 10 — Transformasi: menggeser, memutar, memperbesar

Matriks juga bisa dipakai sebagai transformasi: mengubah posisi titik atau vektor. Dalam AI, transformasi membantu memahami bagaimana data diubah dari ruang mentah menjadi representasi baru.



Transformasi vektor

Contoh memperbesar skala:

```
v = [2, 1]
scale = 3
hasil = [scale*v[0], scale*v[1]]
print(hasil)  # [6, 3]
```

Contoh transformasi matriks 2×2:

```
A = [[2, 0],
     [0, 1]]
v = [3, 4]
hasil = [A[0][0]*v[0] + A[0][1]*v[1],
         A[1][0]*v[0] + A[1][1]*v[1]]
print(hasil)  # [6, 4]
```

Terjemahan manusia: matriks A mengubah vektor v. Baris pertama menentukan koordinat baru pertama, baris kedua menentukan koordinat baru kedua.

Transformasi linear dengan matriks

Rotasi, scaling, shear, dan proyeksi dapat ditulis sebagai perkalian matriks.

```
x_baru = A x
```

Contoh scaling

```
A = [[2,0],
     [0,3]]
x = [4,5]
Ax = [2×4 + 0×5, 0×4 + 3×5] = [8,15]
```

Fitur pertama diperbesar 2 kali, fitur kedua 3 kali. Dalam neural network, setiap layer linear melakukan transformasi semacam ini sebelum aktivasi non-linear.

Tes cepat subbab 10

1. Apa itu transformasi dalam konteks vektor?

2. Jika $v=[2,3]$ dikali skala 2, hasilnya apa?

3. Mengapa transformasi penting untuk deep learning?

Subbab 11 — Proyeksi dan PCA: bayangan data yang lebih sederhana

Kadang data punya banyak fitur. Kita ingin melihat pola dalam 2D agar manusia bisa memahami. Proyeksi berarti membuat “bayangan” data ke arah tertentu.

Proyeksi data

Bayangkan memegang pulpen di bawah lampu. Bayangannya di meja lebih sederhana daripada bentuk 3D aslinya. PCA memakai ide yang mirip: mencari arah bayangan yang menyimpan variasi data paling besar.

Kita belum perlu rumus PCA lengkap di sini. Pegang intuisi:

- data banyak dimensi sulit dilihat,
- proyeksi membuat versi lebih sederhana,
- PCA mencari arah proyeksi yang informatif,
- informasi bisa hilang jika dimensi dikurangi terlalu banyak.

Catatan ungu:
Dimensionality reduction bukan sulap. Ia menukar detail dengan kemudahan melihat pola.

Proyeksi sebagai bayangan pada arah tertentu

Proyeksi vektor x ke arah unit u:

$$\text{proj}_u(x) = (x \cdot u)u$$

Jika u sudah panjang 1, $x \cdot u$ adalah koordinat x pada arah u.

Contoh hitung terstruktur

x `[3,4]`, u `[1,0]`:

$$x \cdot u = 3 \times 1 + 4 \times 0 = 3$$
$$\text{proj}_u(x) = 3 \times [1,0] = [3,0]$$

Bayangan x pada sumbu-x adalah `[3,0]`. PCA mencari arah u yang mempertahankan variasi data sebanyak mungkin. Secara teknis PCA berhubungan dengan covariance matrix dan eigenvector, tetapi intuisi awalnya adalah mencari “bayangan paling informatif”.

Tes cepat subbab 11

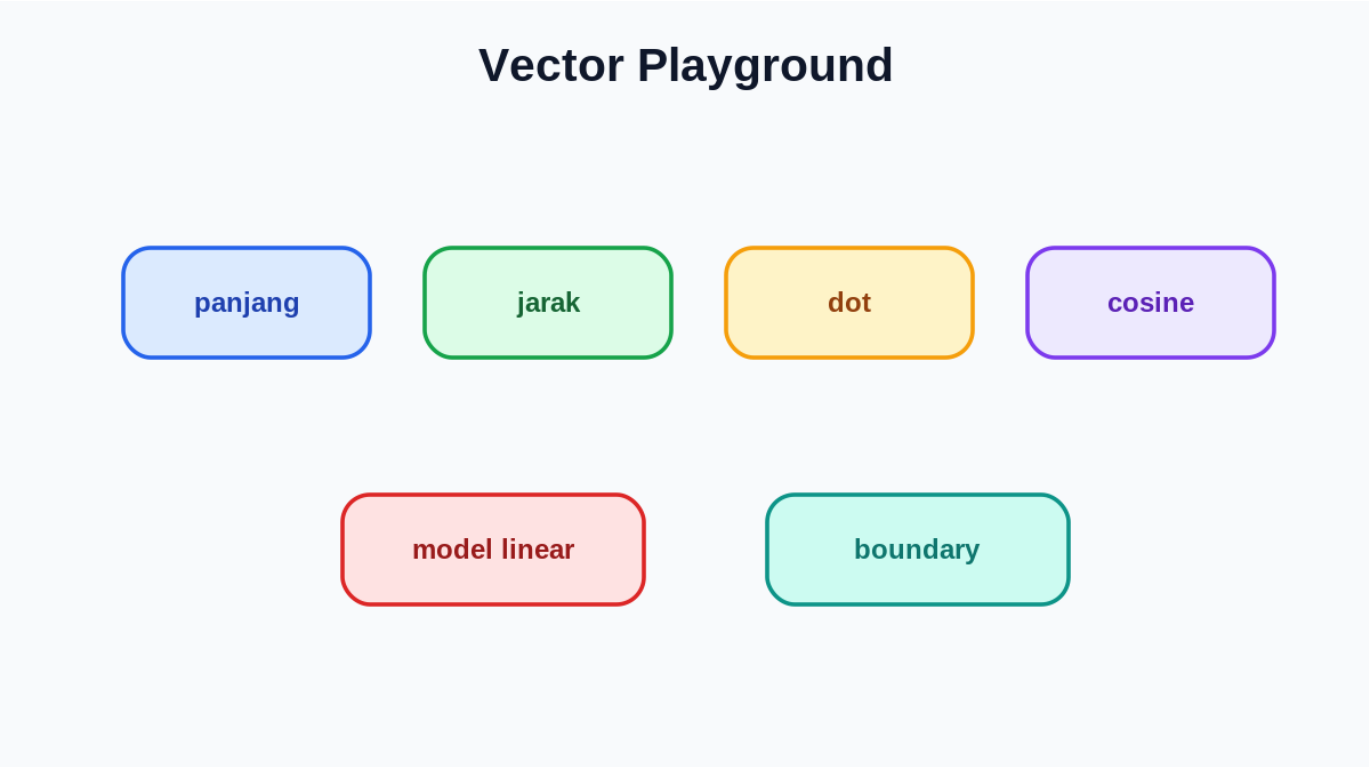
1. Apa itu proyeksi dengan analogi bayangan?

2. Mengapa data banyak dimensi sulit divisualisasikan?

3. Apa risiko mengurangi dimensi terlalu banyak?

Subbab 12 — Praktikum: vector playground

Sekarang kita gabungkan semua: vektor, panjang, jarak, dot product, cosine, matriks, model linear, dan decision boundary.



Vector playground

File praktikum:

- `code/vector_playground.py`
- `code/vector_playground.ipynb`

Jalankan script:

```
cd zero-to-hero-menaklukkan-ai/chapters/04-matematika-ai-visual/code
python3 vector_playground.py
```

Notebook bisa dibuka di Jupyter/VS Code/Colab/Kaggle. Praktikum ini sengaja memakai Python standard library agar pembaca fokus pada konsep, bukan instalasi paket.

Challenge:

1. Tambahkan vektor pelanggan baru.
2. Hitung pelanggan mana yang paling mirip dengan pelanggan A.
3. Ubah bobot model linear dan lihat keputusan berubah.
4. Jelaskan dengan bahasa manusia: fitur mana yang paling memengaruhi skor?

Motivasi penutup:
Jika kamu sampai subbab ini, kamu sudah melewati pintu yang sering membuat banyak orang menyerah: matematika AI. Jangan berhenti. Di bab berikutnya, probabilitas dan gradient akan terasa jauh lebih bersahabat karena kamu sudah punya peta geometri.

Tes cepat subbab 12

1. Jelaskan vektor, matriks, dot product, dan decision boundary dalam satu paragraf.

2. Buat contoh vektor untuk pelanggan toko online.

3. Hitung panjang `[5,12]`.

4. Jelaskan mengapa scaling fitur penting.

5. Jalankan notebook Bab 4 dan catat hasil cosine similarity.

Pendalaman tambahan — dari geometri ke model AI modern

Bab 4 sengaja tidak berhenti pada “vektor adalah daftar angka”. Dalam model AI modern, hampir semua objek akhirnya menjadi vektor: kata menjadi embedding, gambar menjadi tensor, pengguna menjadi profil fitur, dokumen menjadi representasi semantik, dan parameter model sendiri adalah vektor besar. Karena itu, memahami geometri vektor berarti memahami cara model “merasakan” kemiripan dan perbedaan.

Sejarah singkat aljabar linear dalam komputasi

Aljabar linear berkembang karena manusia perlu menyelesaikan banyak persamaan sekaligus. Ketika komputer muncul, operasi matriks menjadi sangat penting karena dapat dihitung cepat dan paralel. Deep learning modern sangat bergantung pada perkalian matriks besar. GPU menjadi berguna untuk AI karena GPU sangat kuat menghitung operasi vektor dan matriks secara paralel.

Persamaan umum model linear

$$\hat{y} = w \cdot x + b$$

x adalah fitur, w adalah bobot, b adalah bias. Jika x=[x■,x■,x■] dan w=[w■,w■,w■], maka:

$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Contoh hitung lengkap

Misalkan model rekomendasi sederhana memakai fitur:

```
x = [rating, diskon, klik] = [4,5, 0,2, 6]
w = [1,0, 2,0, 0,3]
b = -5
```

Hitung skor:

```
w·x = 1,0×4,5 + 2,0×0,2 + 0,3×6
   = 4,5 + 0,4 + 1,8
   = 6,7
ŷ = 6,7 - 5 = 1,7
```

Jika threshold 0, prediksi positif. Artinya produk cukup layak direkomendasikan menurut model ini.

Dari dot product ke gradient

Jika loss regresi satu data adalah:

$$L = (\hat{y} - y)^2$$
$$\hat{y} = w \cdot x + b$$

maka turunan terhadap bobot ke-j adalah:

$$\partial L/\partial w_j = 2(\hat{y}-y)x_j$$

Maknanya sangat indah: fitur yang besar dan error yang besar menghasilkan koreksi bobot yang besar. Ini menjembatani Bab 4 ke Bab 5. Vektor bukan hanya untuk menyimpan data, tetapi juga untuk mengubah parameter saat model belajar.

Contoh gradient kecil

Jika x=[2,3], w=[1,1], b=0, dan target y=10:

$\hat{y} = 1\times2 + 1\times3 = 5$
error = $\hat{y}$-y = -5
$\partial L/\partial w_1 = 2\times(-5)\times2 = -20$
$\partial L/\partial w_2 = 2\times(-5)\times3 = -30$

Gradient negatif berarti bobot perlu dinaikkan agar prediksi mendekati target. Ini adalah contoh konkret bagaimana aljabar linear bertemu kalkulus.

Tensor sebagai generalisasi matriks

Vektor adalah 1D, matriks adalah 2D, tensor adalah generalisasi ke dimensi lebih tinggi. Gambar warna misalnya dapat dianggap tensor `[tinggi, lebar, channel]`. Batch gambar menjadi `[batch, tinggi, lebar, channel]`. Buku ini belum masuk computer vision penuh, tetapi konsep tensor akan muncul lagi saat deep learning.

Latihan hitung terstruktur Bab 4

Kerjakan dengan kertas atau Markdown sebelum menjalankan kode.

1. Vektor A `[2,3,1]` dan B `[5,1,1]`. Hitung A-B, panjang A, panjang B, dan jarak A ke B.

2. Hitung dot product A·B. Jelaskan apakah nilainya menunjukkan arah yang cukup sejalan.

3. Hitung cosine similarity untuk A dan B memakai hasil panjang vektor.

4. Diberikan `X=[[1,2],[3,4]]`, `w=[2,-1]`, `b=0,5`. Hitung `Xw+b` untuk dua baris.

5. Untuk boundary `3x■ - 2x■ + 1 = 0`, tentukan apakah titik `(1,3)` berada di sisi positif atau negatif.

6. Proyeksikan vektor `[6,2]` ke arah unit `[1,0]` dan jelaskan maknanya.

Latihan ini sengaja manual. Tujuannya membuat pembaca merasakan bahwa operasi AI besar berasal dari operasi kecil yang bisa dihitung pelan-pelan.

Pembahasan contoh soal Bab 4 dengan langkah jawaban

Bagian ini sengaja seperti buku latihan. Pembaca dapat menutup jawaban terlebih dahulu, mengerjakan sendiri, lalu membandingkan.

Soal A — panjang dan jarak

Diberikan A `[2,3,1]` dan B `[5,1,1]`.

A-B = [2-5, 3-1, 1-1] = [-3, 2, 0]
$||A|| = \text{sqrt}(2^2+3^2+1^2) = \text{sqrt}(4+9+1) = \text{sqrt}(14) \approx 3{,}74$
$||B|| = \text{sqrt}(5^2+1^2+1^2) = \text{sqrt}(25+1+1) = \text{sqrt}(27) \approx 5{,}20$
$\text{jarak}(A,B) = \text{sqrt}((-3)^2 + 2^2 + 0^2) = \text{sqrt}(13) \approx 3{,}61$

Interpretasi: B punya magnitude lebih besar, tetapi jarak A-B tidak hanya bergantung pada magnitude; ia bergantung pada selisih per fitur.

Soal B — dot product dan cosine

$A\cdot B = 2\times5 + 3\times1 + 1\times1 = 10 + 3 + 1 = 14$
$\text{cosine}(A,B) = 14 / (\text{sqrt}(14)\times\text{sqrt}(27))$
$\approx 14 / (3{,}74\times5{,}20)$
$\approx 14 / 19{,}45$
$\approx 0{,}72$

Cosine 0,72 berarti arah A dan B cukup mirip, tetapi tidak identik.

Soal C — prediksi matriks-vektor

X = [[1,2],

```
    [3,4]]
w = [2,-1]
b = 0,5
```

Baris pertama:

```
1×2 + 2×(-1) + 0,5 = 2 - 2 + 0,5 = 0,5
```

Baris kedua:

```
3×2 + 4×(-1) + 0,5 = 6 - 4 + 0,5 = 2,5
```

Jadi `Xw+b = [0,5, 2,5]`.

Soal D — boundary

Boundary `3x■ - 2x■ + 1 = 0`. Titik `(1,3)`:

```
skor = 3×1 - 2×3 + 1 = 3 - 6 + 1 = -2
```

Karena skor negatif, titik berada di sisi negatif boundary. Jika model memakai threshold 0, prediksi kelas negatif.

Ringkasan Bab 4

- AI perlu mengubah dunia menjadi angka.
- Vektor adalah profil objek dalam bentuk daftar angka.
- Panjang vektor mengukur besar/langkah.
- Jarak mengukur kedekatan antar objek.
- Dot product mengukur kesearahan dan bisa menjadi skor kecocokan.
- Cosine similarity membandingkan arah dengan mengurangi efek panjang.
- Matriks adalah banyak vektor dalam tabel.
- Model linear adalah dot product fitur dan bobot plus bias.
- Decision boundary adalah garis/bidang pemisah keputusan.
- Transformasi mengubah vektor dari satu ruang ke ruang lain.
- Proyeksi/PCA membantu melihat data berdimensi tinggi dengan lebih sederhana.

Referensi utama bab

[R1] Deisenroth, Faisal, & Ong. *Mathematics for Machine Learning*. Digunakan sebagai rujukan aljabar linear untuk ML; tidak disalin. [R2] Goodfellow, Bengio, & Courville. *Deep Learning*. Digunakan untuk konsep representasi dan transformasi; tidak disalin. [R3] Géron. *Hands-On Machine Learning*. Digunakan sebagai inspirasi pipeline praktis dan scaling; tidak disalin. [R4] Strang. *Introduction to Linear Algebra*. Digunakan sebagai rujukan aljabar linear dasar; tidak disalin. [R5] Dokumentasi NumPy/scikit-learn. Digunakan sebagai arah lanjutan setelah konsep manual dipahami.

Catatan validasi internal v0.3

Aspek	Status	Catatan
Pedagogi	Sesuai playbook	Cerita → visual → kode → rumus.

Aspek	Status	Catatan
Matematika	Tervalidasi awal	Rumus dibatasi pada vektor/matriks dasar.
Praktikum	Siap v0.3	Script dan notebook standard library.
Motivasi	Ditambahkan	Bab dirancang agar pembaca percaya diri lanjut ke Bab 5.
Risiko	Terbuka	Perlu review ahli matematika sebelum v1.0.