Kitten AI Academy

From Purrs to Production



From Purrs to Production

Roy Philip

Kitten AI Academy

Belajar AI dan Machine Learning dari Dasar · Dari Nol hingga Produksi AI

SAMPEL GRATIS BAB 1 · EDISI 2026

Penulis: Roy Philip (royphilip.xyz · @royphilip_)
Edisi Web Resmi (Gratis): kittenai.royphilip.xyz/id
Notebook Pendamping (Lisensi MIT): github.com/royphilip/KittenAI-Notebooks
Versi & Pembaruan: Edisi 2026 (Terakhir diverifikasi: 26 November 2026)

Hak Cipta:
© 2026 Roy Philip. Hak Cipta Dilindungi Undang-Undang. Teks buku, terjemahan/transkreasi, ilustrasi, serta edisi PDF dan EPUB ini dilindungi hak cipta. Dilarang memperbanyak, mencetak ulang secara komersial, mendistribusikan ulang, atau membuat karya turunan tanpa izin tertulis dari penulis.

Kode & Notebook:
Seluruh potongan kode Python dan notebook pendamping di repositori `KittenAI-Notebooks` dirilis di bawah lisensi MIT.

Izin Atribusi Akademik (Non-Komersial):
Dosen perguruan tinggi dan guru SMK di Indonesia dipersilakan menautkan serta mengutip bab edisi web dalam RPS, silabus, atau modul ajar non-komersial dengan mencantumkan atribusi penulis dan tautan ke `kittenai.royphilip.xyz/id`.

Disusun dengan rasa ingin tahu, ketelitian, dan banyak snack.

Daftar Isi Sampel

Bab 1: Aljabar Linear untuk Machine Learning: Vektor, Matriks, SVD & PCA **6**

Cara Menggunakan Buku Ini

Prasyarat, dua jalur belajar, metode tiga langkah, dan 14 notebook praktik

Selamat datang di **Kitten AI Academy (Edisi 2026)**. Buku ini dirancang untuk membawamu dari konsep vektor paling dasar hingga arsitektur Large Language Models (LLM), Reinforcement Learning, AI Safety, dan deployment produksi—tanpa menyembunyikan matematika maupun kode aslinya.

Metode Belajar Tiga Langkah

Setiap bab mengikuti satu ritme yang konsisten:

- **Analogi Kucing:** Membangun intuisi visual dan fisik sebelum simbol matematika muncul.
- **Matematika Formal:** Menjelaskan persamaan inti dan maknanya secara terang—bukan hiasan opsional, melainkan peta kerja algoritma.
- **Kode Python & Tabel Rosetta Stone:** Memetakan langsung intuisi dan rumus ke implementasi `NumPy`, `scikit-learn`, dan `PyTorch`, lalu ditutup oleh **Lepas Roda Bantu**, **Kuis Refleksi**, dan **Benang Merah AI Safety**.

Prasyarat & Para Pemandu

Kamu cukup menguasai **dasar Python** (fungsi, loop, array) dan **aljabar SMA**. Kedalaman aljabar linear, probabilitas, dan kalkulus dibangun bertahap dari Bab 1 hingga Bab 3.
Sepanjang perjalanan, kamu akan ditemani dua pemandu:

- **Pixel (kucing tabby):** Melambangkan rasa ingin tahu, eksplorasi, dan kapabilitas model dari Bab 1 sampai Bab 29.
- **Luna (kucing hitam penjaga keamanan):** Bergabung mulai Fase V (Bab 17) untuk melambangkan pengawasan, alignment, dan pengamanan sistem.

Dua Jalur Belajar: Jalur Inti vs. Jalur Lanjutan

- **Jalur Inti (Fase I–IV · Bab 1–16):** Disarankan untuk mahasiswa, pemula, dan pembaca yang ingin fondasi utuh dari matematika dasar, Classical ML, Deep Learning, LLM, hingga Reinforcement Learning. Subbab bertanda 🧭 bersifat opsional pada bacaan pertama.
- **Jalur Lanjutan (Fase V–VII · Bab 17–29):** Membahas AI Alignment (RLHF/DPO), Robustness, Mechanistic Interpretability, Multi-Agent, Scaling Laws, Distributed Training, Optimasi Inference (vLLM/Quantization), dan MLOps. Software engineer yang sudah akrab dengan dasar ML dapat langsung memprioritaskan Bab 8, 10–13, 24, dan 26.

Menjalankan 14 Notebook Praktik (Google Colab & Lokal)

Seluruh **14 notebook pendamping** (8 **micro-notebook** opsional di `micro-notebooks/` dan 6 **checkpoint notebook** per fase di `appendix-repo/`) tersedia gratis berlisensi MIT di:
https://github.com/royphilip/KittenAI-Notebooks

- **Google Colab (tanpa instalasi):** Buka
  `https://colab.research.google.com/github/royphilip/KittenAI-Notebooks/blob/main/<path-to-notebook>.ipynb`
  atau klik tautannya di **Lampiran E**.
- **Di komputer lokal (ramah CPU, offline setelah instalasi):** Klon repositori lalu jalankan `pip install numpy matplotlib scikit-learn jupyter && jupyter notebook`. Lihat **Lampiran E** untuk daftar lengkap seluruh notebook beserta fungsi validasinya.

FASE I

Taman Bermain Matematika Kucing

BAB 1

Aljabar Linear untuk Machine Learning: Vektor, Matriks, SVD & PCA

Mengejar Bola Benang — Nullspace, Rank & Jacobian

Bab Evergreen: Matematika inti dan teori dasarnya tidak pernah kedaluwarsa. *Terakhir diverifikasi: 2026-11-26*

Prasyarat: Pendahuluan & Dasar; dasar Python; aljabar SMA.

Kata Kunci Teknis: Vektor, Matriks, Aljabar Linear, Nullspace, Rank, Column Space, SVD, PCA, Jacobian

Bayangkan seekor anak kucing bernama Pixel sedang duduk di tengah ruang tamu. Tiba-tiba, seberkas sinar matahari jatuh di atas karpet. Pixel ingin menerkamnya. Dari mana Pixel tahu ke mana harus melompat?

Dengan nalurinya, otak Pixel sedang menjalankan **Aljabar Linear** — bahasa matematika untuk ruang, pergerakan, dan transformasi. Setiap potongan data dalam Machine Learning — entah itu foto tikus, harga makanan kucing, atau satu kata dalam kalimat — pada akhirnya diterjemahkan ke bahasa ini. Di mata AI, seluruh alam semesta hanyalah kumpulan panah, tarikan, dan putaran.

Kotak Notasi

- **Konvensi bentuk data:** $X \in \mathbb{R}^{n \times d}$ dengan baris sebagai sampel dan kolom sebagai fitur.
- $\boldsymbol{A}$: Matriks atau transformasi umum.
- $\boldsymbol{X}$: Matriks data.
- $\vec{\boldsymbol{v}}$: Satu vektor atau sampel.
- $\boldsymbol{w}$: Vektor bobot (weights) pada model linear.
- $\boldsymbol{W}$: Matriks bobot (weights) pada neural network.
- $\boldsymbol{\theta}$: Seluruh parameter yang bisa dilatih dalam satu paket.
- **Notasi Learning Rate:** Kita memakai η; sebagian paper memakai α.

Panduan Bertahan Hidup: Aljabar SMA

Sepanjang buku ini, kamu akan sering bertemu huruf-huruf Yunani yang terlihat sangar. Jangan panik. Semuanya cuma singkatan untuk konsep yang sederhana:

- Σ **(Sigma):** Cuma huruf "S" raksasa untuk "Sum" (penjumlahan). Artinya "jumlahkan semua ini."
- Δ **(Delta):** Perubahan. Kalau suhu ruangan naik dari 27°C ke 32°C, nilai Δ adalah 5.
- d atau ∂ **(Turunan / Derivative):** Laju perubahan suatu fungsi terhadap variabelnya pada suatu titik.
- $\int$ **(Integral):** Versi mulus dan kontinu dari Sigma (Σ). Artinya "hitung total luas area."

Kalau suatu rumus terasa membingungkan pada bacaan pertama, pahami dulu Analogi Kucing dan kode Python-nya—lalu kembali ke persamaannya setelah intuisinya nempel. Matematika bukan hiasan opsional, tetapi jauh lebih mudah dicerna begitu kamu paham maknanya.

Vektor: Kaki Mungil di Dalam Ruang

Untuk menjangkau sinar matahari, Pixel perlu tahu dua hal: *seberapa jauh* harus melompat (magnitudo) dan *ke arah mana* harus melompat (arah). Panah yang punya arah dan panjang disebut **vektor**.

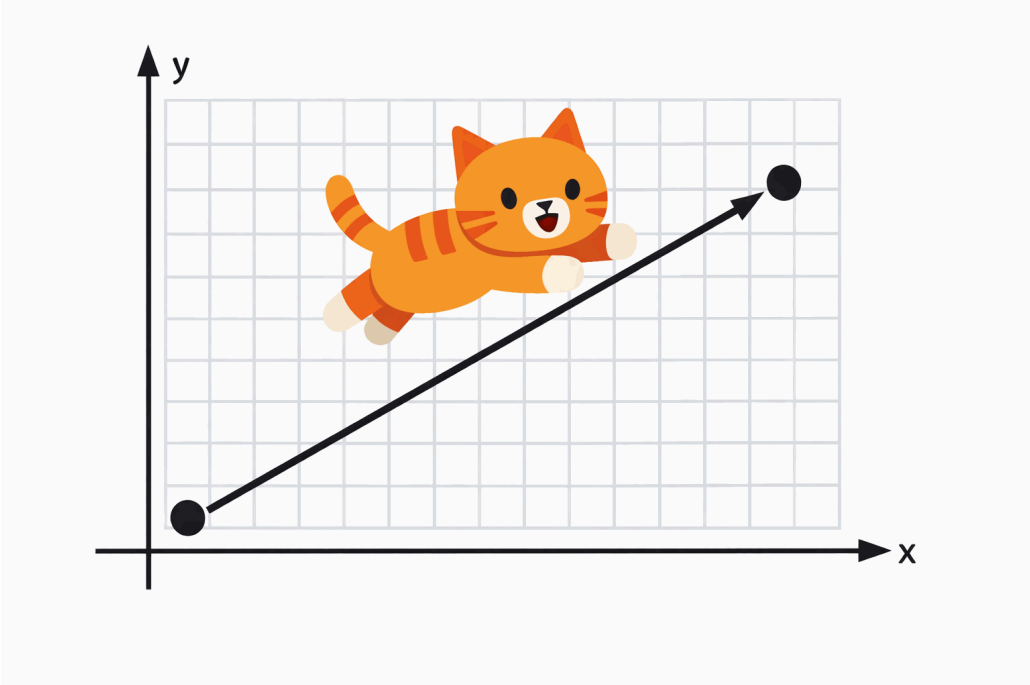
Kalau sinar mataharinya berada 3 langkah kaki mungil ke depan dan 4 langkah ke samping, lompatan Pixel adalah:

$$\vec{v} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Kotak Matematika: Transpose (T)
Kamu mungkin melihat huruf T kecil di ujung $[3, 4]^T$. Kalau kamu ingat matematika SMA, kamu mungkin mengira itu artinya "pangkat T". Bukan. Dalam aljabar linear, T berarti **Transpose**. Artinya menukar baris dan kolom (mencerminkan elemen melewati diagonal utama, sehingga $(A^T)_{ij} = A_{ji}$): untuk sebuah vektor, kolom vertikal berubah menjadi baris horizontal dan sebaliknya; untuk matriks berukuran $m \times n$, baris ke-i menjadi kolom ke-i, menghasilkan matriks $n \times m$. Kita melakukan ini supaya dimensi matriks cocok saat dikalikan, persis seperti menyambungkan balok Lego.

Norm (panjang) vektor ini adalah $\| \vec{v} \| = \sqrt{3^2 + 4^2} = 5$ langkah kaki mungil.

Analogi Kucing
Vektor adalah lompatannya. Norm adalah seberapa jauh jaraknya. Arah adalah tujuannya.



Pixel melompat ke arah sinar matahari — panah menunjukkan arah dan panjang lompatan.

Operasi inti yang akan kita pakai terus di akademi ini:

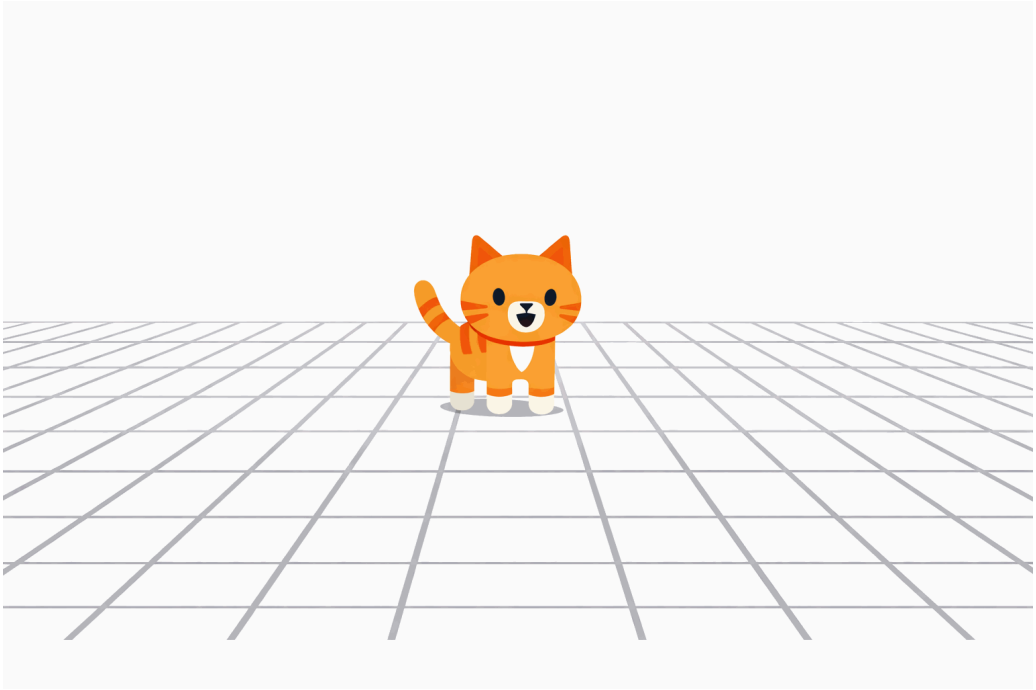
- **Penjumlahan:** Menggabungkan dua lompatan menjadi satu.
- **Perkalian skalar:** Memanjangkan atau memendekkan lompatan.
- **Dot product:** Seberapa searah dua lompatan? ($\vec{u} \cdot \vec{v} = \| \vec{u} \| \| \vec{v} \| \cos\theta$)
- **Norm:** Seberapa panjang lompatannya? ($\| \vec{v} \| = \sqrt{\sum_i v_i^2}$)

Rosetta Stone

Dunia Kucing	Matematika	Python
"Lompatan 3 ke depan, 4 ke samping"	$\vec{v} = [3, 4]^T$	`np.array([3, 4])`
"Seberapa jauh lompatannya?"	$\| \vec{v} \| = \sqrt{v_1^2 + v_2^2}$	`np.linalg.norm(v)`
"Apakah dua lompatan mengarah ke tempat yang sama?"	$\vec{u} \cdot \vec{v} = \sum_i u_i v_i$	`np.dot(u, v)`

Matriks: Stempel Kaki Ajaib

Sekarang bayangkan lantai ruang tamu terbuat dari lembaran karet elastis. Tiba-tiba, ada yang menarik lembaran karet itu — merentangkan segalanya dua kali lebih jauh ke kanan, sementara arah vertikalnya tetap diam. Setiap lompatan Pixel otomatis berubah bentuk mengikuti tarikan ini.



Matriks sebagai transformasi lembaran karet yang direntangkan.

Matriks adalah tarikan tersebut. Matriks adalah sebuah operasi — sebuah fungsi — yang mengambil satu vektor lalu mengubahnya menjadi vektor baru dengan cara merentangkan, memutar, atau memipihkan ruang.

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Terapkan matriks ini pada lompatan Pixel $[3, 4]^T$:

$$A\vec{v} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$$

Komponen x berlipat ganda, sedangkan komponen y tetap. Pada intinya, deep learning mengalirkan data melewati jutaan transformasi seperti ini — masing-masing membentuk ulang informasi sampai wujudnya sesuai dengan yang kita inginkan.

Kegagalan Fatal: Perkalian Matriks TIDAK Komutatif
Kalau kamu memakai kaus kaki dulu baru sepatu, hasilnya rapi. Balik urutannya, hasilnya kacau. Perkalian matriks bersifat tidak komutatif *secara umum*: $AB \neq BA$ (di luar pengecualian khusus seperti perkalian dengan matriks identitas, $AI = IA$, atau matriks diagonal tertentu). Mengubah bentuk ruang dulu lalu memutar Pixel **tidak sama** dengan memutar Pixel dulu lalu mengubah bentuk ruang. Membalik urutan perkalian bisa langsung gagal saat dimensi dalam tidak cocok, atau diam-diam menghasilkan transformasi yang keliru ketika kedua matriks berbentuk persegi. Urutan perkalian perlu diperhatikan karena menukarnya dapat menghasilkan transformasi yang berbeda.

Rosetta Stone

Dunia Kucing	Matematika	Python
"Stempel rentang-dan-pipihkan"	Matriks $A \in \mathbb{R}^{m \times n}$	`np.array([[2,0],[0,1]])`
"Terapkan stempel ke satu lompatan"	$A\vec{v}$	`A @ v`
"Dua stempel berurutan"	AB (urutan penting!)	`A @ B`
"Stempel yang tidak mengubah apa-apa"	I (matriks identitas)	`np.eye(n)`

Rank dan Column Space: Sejauh Mana Pixel Bisa Menjangkau?

Misalnya Pixel ingin menjelajahi ruang tamu. Ia bisa menggabungkan dua gerakan dasar: *maju* dan *ke kanan*. Dengan mencampur keduanya bebas, ia bisa menjangkau titik mana pun di lantai datar 2D tersebut. Seluruh himpunan posisi yang bisa dijangkau ini disebut **column space** (disebut juga *range* atau *image*) dari gerakannya.

Secara formal: **rank** = dimensi dari column space = jumlah kolom yang bebas linear.

$$\text{col}\ (A) = \{A\vec{x} \mid \vec{x} \in \mathbb{R}^n\}$$

Analogi Kucing
Rank 2 berarti Pixel bisa menjangkau seluruh permukaan lantai. Rank 1 berarti ia cuma bisa mondar-mandir di sepanjang satu garis sinar matahari saja.

Catatan Jalur Lanjutan (Bagian 1.4 – 1.7)
Beberapa bagian berikut mengenalkan konsep yang banyak dipakai di riset lanjutan (Nullspace, SVD, Jacobian). Kalau matematikanya terasa berat pada bacaan pertama, fokus saja ke analogi kucingnya lalu langsung lompat ke ringkasan bab. Untuk **Jalur Inti**, kamu cukup menguasai bagian 1.1–1.3.

Nullspace: Jalur Kaki Tak Kasatmata

Nullspace (disebut juga *kernel*) adalah himpunan semua vektor yang diubah matriks menjadi nol:

$$\text{Null}\ (A) = \left\{\vec{x} \mid A\vec{x} = \vec{0}\right\}$$

Teorema Rank-Nullity adalah salah satu hasil terpenting dalam aljabar linear:

$$\text{rank}\ (A) + \text{nullity}\ (A) = n \quad (\text{jumlah kolom})$$

Analogi Kucing
Pixel melompat di sepanjang garis-garis tak kasatmata. Stempel matriks menghapus lompatan ini mentah-mentah — semuanya lenyap jadi nol. Kalau datamu jatuh ke ruang ini, modelmu tidak akan bisa melihatnya sama sekali.

Rosetta Stone

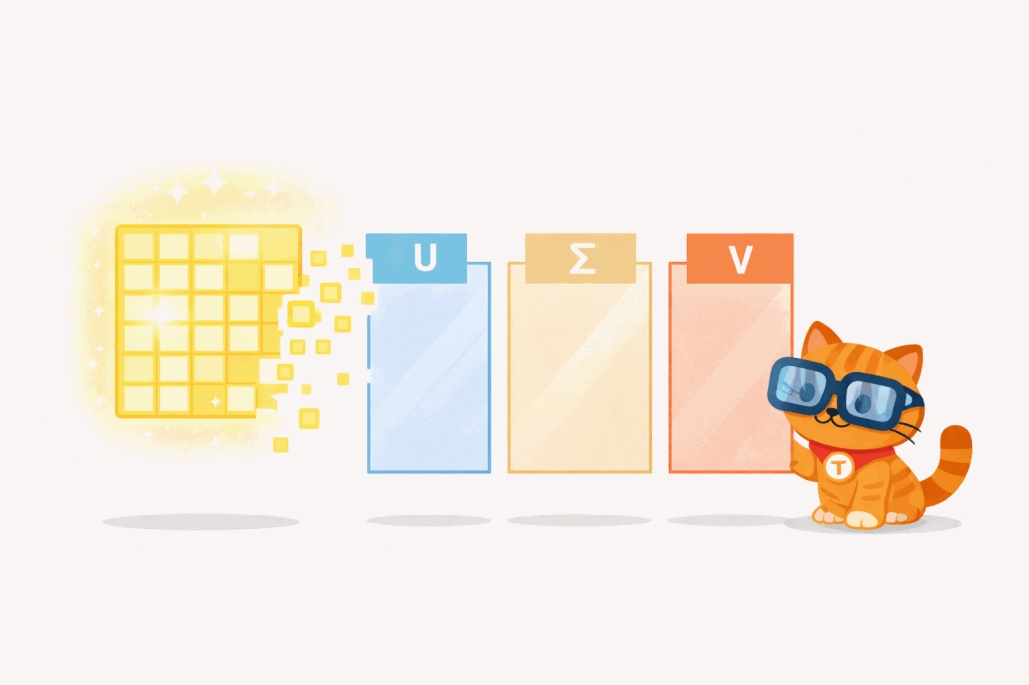
Dunia Kucing	Matematika	Python
"Arah yang menghilang"	$\text{Null}\ (A)$	`Vt[rank:].T` (via SVD)
"Berapa dimensi yang bertahan?"	$\text{rank}\ (A)$	`np.linalg.matrix_rank(A)`
"Berapa yang lenyap?"	$\text{nullity}\ (A) = n - \text{rank}\ (A)$	`A.shape[1] - np.linalg.matrix_rank(A)`

SVD: Penglihatan Sinar-X untuk Matriks

Singular Value Decomposition (SVD) memecah transformasi matriks apa pun menjadi tiga langkah bersih:

$$A = U\Sigma V^T$$

1. V^T: **Putar** Pixel agar menghadap sumbu baru (arah input).
2. Σ: **Rentangkan** atau pipihkan lompatan Pixel sepanjang sumbu baru tersebut sebesar $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$.
3. U: **Putar** lagi agar mendarat di ruang output.



Dekomposisi SVD: memecah matriks rumit apa pun menjadi tiga langkah bersih.

Rosetta Stone

Dunia Kucing	Matematika	Python
"Sinar-X matriksnya"	$A = U\Sigma V^T$	`U, s, Vt = np.linalg.svd(A)`
"Seberapa jauh arah ke-i direntangkan?"	Nilai singular σ_i	`s[i]`
"Arah-arah paling penting"	Kolom U (σ besar)	`U[:, :k]`
"Nullspace (lewat SVD)"	Kolom V untuk $\sigma \approx 0$	`Vt[rank:].T`

PCA: Jalur Sinar Matahari Favorit

Principal Component Analysis (PCA) mencari arah dengan varians maksimum dalam sebuah dataset lalu memproyeksikan data ke arah tersebut. Meski membuang arah bervarians rendah sering membantu meredam noise pengukuran, varians rendah tidak otomatis berarti noise tak berguna—fitur halus bervarians kecil terkadang tetap menyimpan sinyal prediktif yang penting.

Misalkan kita punya matriks data X. Berikut langkah kerja PCA:

1. Pusatkan data ke nol: $X \leftarrow X - \bar{X}$.
2. Hitung matriks kovarians: $C = \frac{1}{n-1} X^T X$.
3. Cari eigenvector dari C (sama dengan right singular vectors dari X lewat SVD).
4. Proyeksikan data ke k eigenvector teratas.

Rosetta Stone

Dunia Kucing	Matematika	Python
"Cari arah pergerakan utama"	Eigenvector dari $C = \frac{1}{n-1} X_c^T X_c$ ($X_c = X - \bar{X}$)	`PCA(n_components=k)`
"Berapa varians yang dijelaskan komponen ke-i?"	$\frac{\lambda_i}{\sum_j \lambda_j}$	`pca.explained_variance_ratio_[i]`
"Proyeksikan ke k arah teratas"	$\tilde{X} = X_c V_k$ (latih di data latih dulu pada alur prediktif)	`pca.fit_transform(X_train); pca.transform(X_test)`

> 🔗 ***Koneksi PCA ↔ SVD:*** *PCA sejatinya hanyalah SVD yang diterapkan pada matriks data yang sudah dipusatkan. Principal components adalah right singular vectors V.*

Jacobian: Seberapa Sensitif Kaki Mungil Saat Dicolek?

Kalau kamu mencolek Pixel yang sedang tidur pelan-pelan, telinganya mungkin cuma berkedut. Kalau colekanmu lebih keras, ia bisa langsung melompat. Pertanyaannya: *seberapa sensitif anak kucing ini terhadap colekanmu?*

Di aljabar SMA, kamu belajar bahwa "kemiringan" (slope) adalah *perubahan vertikal dibagi perubahan horizontal*. Kalau colekan 1 cm menyebabkan kedutan telinga 2 cm, kemiringannya adalah 2. Di kalkulus, **turunan** ($\frac{df}{dx}$) adalah kemiringan yang persis sama, hanya saja diukur pada satu titik yang sangat kecil.

Masalahnya, Pixel hidup di dunia multidimensi. Inputnya banyak (colekan, suara, bau) dan outputnya juga banyak (kedutan telinga, kibasan ekor, lompatan penuh). Kita tidak bisa cuma menghitung satu kemiringan. Kita butuh satu kisi-kisi lengkap berisi kemiringan untuk setiap pasangan input-output.

Kisi-kisi dari semua turunan parsial itulah yang disebut **matriks Jacobian**:

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

(Jangan takut melihat simbol ∂ yang melengkung. Itu cuma berarti "turunan parsial" — kemiringan dari satu variabel saja, dengan menganggap semua hal lain di ruangan diam tidak bergerak).

Jacobian menjawab pertanyaan ini: "Kalau kita menggeser input j sedikit saja, seberapa besar output i ikut berubah?"

Di deep learning, Jacobian ada di mana-mana. Backpropagation pada dasarnya adalah rantai perkalian Jacobian. Memahami Jacobian berarti memahami bagaimana gradien mengalir melewati sebuah jaringan saraf.

$$\vec{f} : \mathbb{R}^n \to \mathbb{R}^m \quad \Rightarrow \quad J \in \mathbb{R}^{m \times n}$$

Analogi Kucing
Jacobian adalah peta sensitivitas. Colek Pixel di sini → telinganya bergerak sekian. Colek di sana → ekornya mengibas sekian. Setiap elemen dalam Jacobian adalah satu pasangan colekan-dan-respons seperti itu.

Rosetta Stone

Dunia Kucing	Matematika	Python (PyTorch)
"Peta sensitivitas"	$J_{ij} = \partial f_i / \partial x_j$	`torch.autograd.functional.jacobian(f, x)`
"Bagaimana gradien mengalir mundur"	Aturan rantai lewat Jacobian	`loss.backward()`
"Satu output, banyak input"	Gradien $\nabla f \in \mathbb{R}^n$	`torch.autograd.grad(output, params)`

Bagaimana Semuanya Saling Terhubung

Konsep	Makna & Fungsi	Di Mana Dipakai Lagi
Vektor	Arah + magnitudo data	Di mana-mana
Matriks	Transformasi ruang	Setiap layer pada neural network
Column Space	Output apa saja yang bisa dijangkau	Ekspresivitas model
Nullspace	Input apa saja yang tidak terlihat	Adversarial examples, titik buta model
Rank	Kekayaan informasi	Aproksimasi low-rank, Attention
SVD	Dekomposisi sinar-X untuk matriks apa pun	PCA, analisis Embedding
PCA	Reduksi dimensi yang menjaga varians	Seleksi fitur, visualisasi
Jacobian	Peta sensitivitas dari fungsi apa pun	Backpropagation, adversarial robustness

Micro-notebook (opsional): `micro-notebooks/phase01_ch01_svd_pca_intuition.ipynb`
Durasi: ~20 menit. Tujuan: memvisualisasikan arah utama dengan SVD/PCA pada data sintetis 2D sederhana.

Lepas Roda Bantu

Realitas Engineering
Di dunia produksi, sebuah gambar diratakan menjadi vektor raksasa berisi intensitas piksel mentah, dan satu layer neural network adalah perkalian matriks besar `A @ x`. Nullspace bukan jalur tak kasatmata; ia adalah subruang matematis tempat informasi fitur dipetakan menjadi nol saat perkalian berlangsung. Kamu juga harus membedakan **ketidakcocokan dimensi** dari **non-komutativitas**: mengalikan matriks $m \times n$ dengan matriks $p \times q$ saat $n \neq p$ akan langsung crash karena *shape mismatch*, sedangkan pada matriks persegi (ketika kedua urutan punya dimensi yang cocok) `A @ B != B @ A` secara umum tetap berjalan tanpa error namun dapat diam-diam memberi hasil matematika yang salah.

Jeda & Refleksi (Kuis)

Coba pikirkan jawabannya dulu sebelum membuka kunci jawaban.

1. Foto tikus Pixel punya 10.000 piksel, tetapi sebagian besar isinya cuma tembok kosong. Teknik apa yang memungkinkan kita mengompres gambar ini sambil tetap mempertahankan bagian yang paling banyak bervariasi?
2. Satu layer neural network menerapkan sebuah matriks ke datamu. Tiga fitur dari inputmu lenyap sepenuhnya — output memberinya bobot nol dalam kondisi apa pun. Masuk ke "ruang" manakah ketiga fitur tersebut?
3. Kamu punya matriks 4×4 dengan rank 2. Berapa nilai nullity-nya? Apa artinya secara geometris?
4. Benar atau Salah: Matriks Jacobian memberi tahu seberapa besar peluang kucing mendapatkan snack.
5. Kenapa matriks ber-rank 1 punya column space berdimensi lebih rendah, dan kenapa hal ini bisa menjadi masalah bagi model Machine Learning?

Kunci Jawaban

Lihat Jawaban

1. **PCA** — teknik ini mencari arah dengan varians maksimum.
2. **Nullspace** dari matriks bobot (weights) layer tersebut.
3. Nullity = 4 − 2 = 2; dua arah input terhapus sepenuhnya oleh transformasi tersebut.
4. **Salah** — Jacobian mengukur laju perubahan output terhadap input.
5. Matriks ber-rank 1 memproyeksikan semua input ke satu garis tunggal — model hanya bisa menghasilkan output di satu arah saja, sehingga ekspresivitasnya sangat terbatas.

Ringkasan Bab 1

Kamu baru saja menguasai bahasa geometris dalam Machine Learning:

- **Vektor** adalah data yang punya arah dan panjang. Semuanya adalah vektor.
- **Matriks** mentransformasi ruang. Setiap layer neural network adalah matriks (ditambah fungsi nonlinear).
- **Rank dan column space** menunjukkan apa yang bisa diekspresikan oleh sebuah matriks.
- **Nullspace** menunjukkan hal-hal yang tidak bisa dilihat oleh matriks.
- **SVD** membedah matriks apa pun seperti sinar-X ke struktur dasarnya.
- **PCA** menemukan arah varians terbesar dan mengompres data dengan cerdas.
- **Jacobian** mengukur sensitivitas — fondasi dari Backpropagation.

Di Bab 2, kita tambahkan probabilitas — dan dunia pun mulai penuh ketidakpastian.

Benang Merah AI Safety: Bab 1

Nullspace matriks ($\text{Null}\ (A) = \left\{\vec{x} \mid A\vec{x} = \vec{0}\right\}$) adalah sifat struktural dari transformasi atau layer linear A: setiap gangguan input $\vec{\delta} \in \text{Null}\ (A)$ membuat $A\left(\vec{x} + \vec{\delta}\right) = A\vec{x}$ tetap tidak berubah, sehingga menciptakan titik buta tempat layer tidak bisa mendeteksi perubahan di arah tersebut. Sementara itu, pergeseran distribusi data latih (*distribution shift*) adalah masalah statistik yang terpisah (kita bahas di Bab 18 dan 26) ketika input produksi berbeda dari data latih—meski pada model linear, arah yang tidak pernah muncul di data latih jatuh ke nullspace matriks data X, sehingga bobot di arah itu tidak terkendali oleh training loss. Selalu tanyakan: *apa yang secara struktural tidak bisa dilihat model ini, dan bagaimana input di dunia nyata bisa jatuh di luar wilayah yang pernah dipelajarinya?*

Suka dengan Bab 1? Lanjutkan Perjalanan Pixel & Luna!

Seluruh **29 bab dan 8 lampiran** Kitten AI Academy **100% gratis dibaca online** di:
https://kittenai.royphilip.xyz/id

Jika kamu ingin menyimpan edisi lengkap untuk **baca offline tanpa kuota internet** di tablet, e-reader, atau dicetak mandiri, dapatkan **Edisi Ebook Lengkap (PDF + EPUB)** seharga **Rp 99.000** (QRIS / GoPay / OVO / VA · sekali beli, tanpa langganan · termasuk pembaruan errata Edisi 2026):

https://royphilip.myr.id/ebook/kitten-ai-academy-bahasa-indonesia-ebook-lengkap-pdf-epub