Pertemuan 7 — SVM dan *Hidden Markov Model*

Margin terlebar dan keadaan tersembunyi, ditutup rekap Bagian I

Hendri Karisma

Semester Ganjil 2026/2027

Program Studi Teknik Informatika, STMIK Tazkia

Alur pertemuan hari ini

Bagian A — *Support Vector Machine*

1. *Margin* maksimum dan *support vector*
2. Bentuk dual dan contoh enam titik
3. *Soft margin* dan parameter C
4. *Kernel trick*: polinomial dan RBF
5. Menyetel C dan γ; SVM dibanding NB dan kNN
6. Studi kasus penyaring SMS penipuan

Bagian B — *Hidden Markov Model*

7. Data berurutan dan rantai Markov
8. HMM $\lambda = (A, B, \pi)$ dan tiga masalah Rabiner
9. *Forward*, *backward*, Viterbi
10. Baum–Welch sebagai EM; *underflow*
11. Kode NumPy dan POS *tagging*
12. Rekap Bagian I dan persiapan UTS

Bahan ini mengikuti buku ajar Bab 9 (SVM) dan Bab 7 (HMM). Semua angka contoh di slide sama dengan angka di buku, sehingga hitungan tangan dapat dicocokkan baris demi baris.

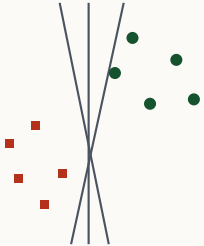
Tujuan pembelajaran

Setelah pertemuan ini mahasiswa mampu:

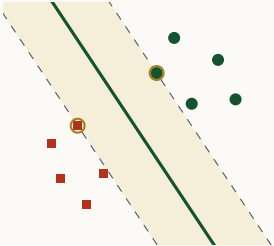
1. menjelaskan *hyperplane*, *margin*, dan *support vector*, lalu menghitung solusi SVM enam titik termasuk nilai α_i;
2. menjelaskan *soft margin*, *hinge loss*, arti C, dan cara kerja *kernel trick*;
3. menyetel C dan γ dengan *grid search* di dalam `Pipeline` dan memilih antara SVM, naive Bayes, dan kNN;
4. mendefinisikan HMM dengan $\lambda = (A, B, \pi)$ dan menyebutkan tiga masalah klasiknya;
5. menghitung *forward* ($P(O \mid \lambda) = 0{,}0244$) dan Viterbi ($P^* = 0{,}0126$) dengan tangan pada contoh regime pasar;
6. menjelaskan Baum–Welch sebagai EM dan mencegah *underflow* dengan faktor skala atau ruang log;
7. memilih model klasik Bagian I yang cocok untuk masalah baru.

Support Vector Machine: *margin* terlebar

Banyak garis bisa memisahkan; mana yang dipilih?



semuanya benar di *training data*



margin terlebar; *support vector* dilingkari

Boser, Guyon, dan Vapnik (1992) memilih pemisah dari jaraknya ke titik terdekat; Cortes dan Vapnik (1995) menambahkan *soft margin*. Ketiga garis di kiri sama-sama punya *training error* nol.

Ringkasan

SVM memilih garis yang **jaraknya ke titik terdekat dari kedua kelas paling besar**. Garis yang menempel pada data berpindah sisi begitu ada sedikit *noise*; garis yang berjarak lega masih memberi keputusan yang sama.

Menulis *hyperplane*, jarak, dan *margin*

Sebuah ***hyperplane*** dinyatakan oleh vektor *weight* w dan bias b, yaitu $w^\top x + b = 0$. Vektor w tegak lurus permukaan itu, b menggesernya dari titik asal, dan keputusannya $\hat{y} = \operatorname{sign}(w^\top x + b)$ dengan label $y \in \{-1, +1\}$.

- **Jarak satu titik:** $\operatorname{jarak}(x) = |w^\top x + b| \,/\, \|w\|$. Pembagian oleh $\|w\|$ mengubah nilai fungsi menjadi jarak sesungguhnya.
- ***Scaling* kanonik.** (w, b) dan $(2w, 2b)$ menggambarkan *hyperplane* yang sama. Kebebasan itu dipakai untuk menetapkan $|w^\top x + b| = 1$ pada titik terdekat, sehingga jaraknya $1/\|w\|$.
- **Lebar *margin*:** jarak antara $w^\top x + b = +1$ dan $w^\top x + b = -1$, yaitu $2/\|w\|$.

Hard-margin SVM

Memperlebar *margin* sama dengan memperkecil $\|w\|$:

$$\min_{w,\,b} \ \tfrac{1}{2}\|w\|^2 \qquad \text{dengan kendala} \qquad y_i\,(w^\top x_i + b) \geq 1, \quad i = 1, \ldots, n$$

Quadratic programming cembung dengan minimum tunggal.

Bentuk dual dan peran α_i

Setiap kendala mendapat pengali Lagrange $\alpha_i \geq 0$. *Derivative* fungsi Lagrange terhadap w dan b yang disamakan dengan nol memberi

$$w = \sum_i \alpha_i y_i x_i, \qquad \sum_i \alpha_i y_i = 0,$$

dan setelah disubstitusikan kembali diperoleh **bentuk dual**

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j \quad \text{dengan} \quad \alpha_i \geq 0,\ \sum_i \alpha_i y_i = 0.$$

Complementary slackness (KKT)

$\alpha_i [y_i (w^\top x_i + b) - 1] = 0$. Jadi $\alpha_i > 0$ hanya pada titik yang kendalanya tepat terpenuhi, yaitu ***support vector***. Titik lain punya $\alpha_i = 0$ dan boleh dibuang tanpa mengubah solusi.

Data hanya muncul sebagai $x_i^\top x_j$

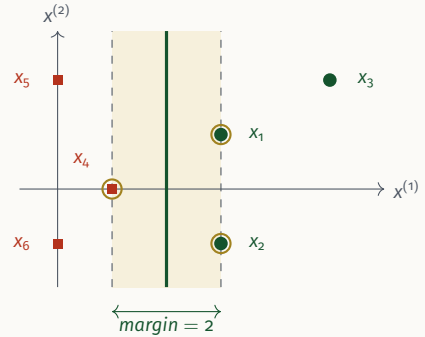
Fungsi keputusan $f(x) = \sum_{i \in SV} \alpha_i y_i x_i^\top x + b$ cukup menyimpan *support vector*. Hasil kali dalam $x_i^\top x_j$ itu nanti diganti oleh *kernel*.

Contoh terhitung: enam titik, dua dimensi

Titik	y	$y(w^\top x + b)$	α_i
$x_1 = (3, 1)$	$+1$	1 (SV)	0,25
$x_2 = (3, -1)$	$+1$	1 (SV)	0,25
$x_3 = (5, 2)$	$+1$	3	0
$x_4 = (1, 0)$	-1	1 (SV)	0,5
$x_5 = (0, 2)$	-1	2	0
$x_6 = (0, -1)$	-1	2	0

Solusinya $w = (1, 0)$, $b = -2$: *hyperplane* $x^{(1)} = 2$ dan lebar *margin* $2/\|w\| = 2$.

Memeriksa lewat dual. Dari $w = \sum \alpha_i y_i x_i$: $(1, 0) = \alpha_1(3, 1) + \alpha_2(3, -1) - \alpha_4(1, 0)$. Komponen kedua memberi $\alpha_1 = \alpha_2 = a$; $\sum \alpha_i y_i = 0$ memberi $\alpha_4 = 2a$; komponen pertama memberi $6a - 2a = 1$, jadi $a = 0{,}25$ dan $\alpha_4 = 0{,}5$. Bias: $b = y_4 - w^\top x_4 = -2$. Pemeriksaan: $\sum_i \alpha_i = 1 = \|w\|^2$.



Pilihan $w = (2, 0)$, $b = -4$ memberi garis yang sama dan tetap memenuhi kendala, tetapi $\frac{1}{2}\|w\|^2 = 2$, lebih besar dari 0,5. Pilihan $w = (0{,}5;\ 0)$, $b = -1$ melanggar kendala di x_1 ($0{,}5 < 1$). Membuang x_3, x_5, atau x_6 tidak mengubah jawaban; membuang x_4 menggeser *hyperplane*.

Memeriksa dengan scikit-learn

Nilai *C* yang sangat besar membuat SVC berperilaku seperti *hard margin*.

```
import numpy as np
from sklearn.svm import SVC

X = np.array([[3, 1], [3, -1], [5, 2], [1, 0], [0, 2], [0, -1]], float)
y = np.array([1, 1, 1, -1, -1, -1])

svm = SVC(kernel="linear", C=1e6).fit(X, y)    # C besar ~ hard margin
w, b = svm.coef_[0], svm.intercept_[0]
print(w.round(3), round(b, 3))                 # [1. 0.] -2.0
print(2 / np.linalg.norm(w))                   # lebar margin 2.0
print(svm.support_vectors_)                    # (1,0), (3,1), (3,-1)
print(svm.dual_coef_)                          # alpha_i*y_i: -0.5 0.25 0.25
```

Atribut `dual_coef_` menyimpan $\alpha_i y_i$, hanya untuk *support vector*. Tanda minus pada $-0{,}5$ berasal dari $y_4 = -1$; besarnya sama dengan $\alpha_4 = 0{,}5$ hasil hitungan tangan. Kode lengkap: `bab09-svm/svm_enam_titik.py`.

Variabel *slack* dan *hinge loss*

Satu titik yang labelnya keliru sudah membuat kendala *hard margin* mustahil dipenuhi. Cortes dan Vapnik mengizinkan pelanggaran dengan *cost*:

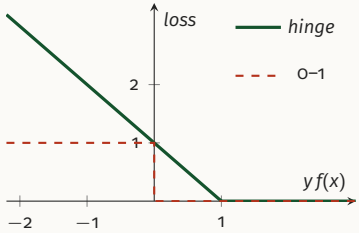
$$\min_{w,\,b,\,\xi} \ \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{dengan} \quad y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

Pada titik optimum ξ_i (dibaca *xi*) $= \max\left(0,\, 1 - y_i(w^\top x_i + b)\right)$, yaitu ***hinge loss***.

- $\xi_i = 0$: aman di luar *margin*
- $0 < \xi_i \leq 1$: masuk *margin*, masih di sisi yang benar
- $\xi_i > 1$: salah *classification*

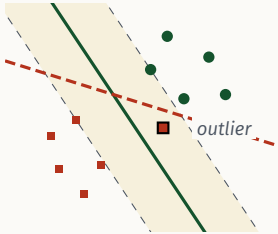
Pada bentuk dual, perubahannya hanya batas atas: $0 \leq \alpha_i \leq C$.



Contoh (solusi enam titik)

$x_7 = (1{,}2;\ 0)$ berlabel $+1$: $y_7(w^\top x_7 + b) = -0{,}8$, jadi $\xi_7 = 1{,}8$ dan dendanya $C \times 1{,}8$. Titik $(2{,}5;\ 1{,}7)$ berlabel $+1$ memberi $0{,}5$, jadi $\xi = 0{,}5$: benar, tetapi di dalam *margin*.

Apa yang terjadi bila C besar atau kecil



Garis merah putus-putus: C besar memaksa *outlier* benar. Garis hijau dengan pita: C kecil membiarkannya salah.

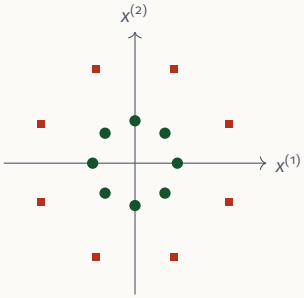
	C **besar**	C **kecil**
Prioritas	Jangan ada pelanggaran	*Margin* harus lebar
Bias–*variance*	*Variance* tinggi, bias rendah	Bias tinggi, *variance* rendah
Risiko khas	*Overfitting*	*Underfitting*
Ekstrem	$C \to \infty$: kembali ke *hard margin*	$C \to 0$: *margin* melebar tanpa peduli label

Ringkasan

C adalah pengendali ***regularization*** pada SVM dengan arah baca terbalik: C besar berarti *regularization* **lemah**. Nilainya dicari dengan *cross-validation* pada data Anda sendiri.

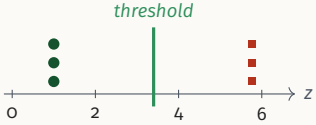
Kernel trick dan menyetel SVM

Data yang tidak punya garis pemisah



ruang asli: tak terpisah garis

Tambahkan satu *feature*, jarak kuadrat dari pusat $z = (x^{(1)})^2 + (x^{(2)})^2$, sehingga $\phi(x) = \left(x^{(1)}, x^{(2)}, z\right)$, dengan ϕ dibaca *phi*.



Cincin dalam (jari-jari 1) berkumpul di $z = 1$, cincin luar (jari-jari 2,4) di $z = 5{,}76$; satu *threshold* pada z sudah memisahkan keduanya.

XOR dengan *feature* hasil kali

$(+1, +1)$ dan $(-1, -1)$ berlabel $+1$; $(+1, -1)$ dan $(-1, +1)$ berlabel -1: tidak ada garis pemisahnya. Dengan $\phi(x) = (x^{(1)},\ x^{(2)},\ x^{(1)}x^{(2)})$, kolom terakhir persis setanda dengan y, sehingga $w = (0, 0, 1)$ dan $b = 0$ memisahkannya sempurna.

Mengapa disebut *trick*

Polinomial derajat 3 atas 100 *feature* sudah menghasilkan ratusan ribu *feature* baru, dan *feature space* RBF berdimensi tak hingga. Bentuk dual hanya memerlukan $\phi(x_i)^\top \phi(x_j)$, sehingga cukup ada fungsi ***kernel*** $K(x, z) = \phi(x)^\top \phi(z)$ yang dihitung di ruang asli:

$$f(x) = \sum_{i \in SV} \alpha_i y_i K(x_i, x) + b.$$

Periksa sendiri. Untuk $K(x, z) = (x^\top z)^2$ di $\mathbb{R}^2$,

$$\phi(x) = (x_1^2,\ \sqrt{2}\, x_1 x_2,\ x_2^2).$$

Dengan $x = (1, 2)$, $z = (3, 1)$: $x^\top z = 5$, jadi $K = 25$.
Secara eksplisit $\phi(x) = (1,\ 2\sqrt{2},\ 4)$,
$\phi(z) = (9,\ 3\sqrt{2},\ 1)$, dan
$\phi(x)^\top \phi(z) = 9 + 12 + 4 = 25$.

```python
import numpy as np

def phi(v):            # pemetaan eksplisit untuk (x.z)^2 di R^2
    return np.array([v[0]**2, np.sqrt(2) * v[0] * v[1], v[1]**2])

def rbf(x, z, gamma):
    return np.exp(-gamma * np.sum((x - z) ** 2))

x, z = np.array([1.0, 2.0]), np.array([3.0, 1.0])
print((x @ z) ** 2, phi(x) @ phi(z))     # 25.0 25.0
for g in (0.01, 0.1, 1, 10):             # jarak kuadrat 4
    print(g, rbf(np.zeros(2), np.array([2.0, 0.0]), g))
```

Tidak semua fungsi boleh menjadi *kernel*: matriks $K_{ij} = K(x_i, x_j)$ harus simetris dan semi-definit positif (syarat Mercer).

Tiga *kernel* yang dipakai sehari-hari

Kernel	**Rumus**	**Kapan dipakai**
Linear	$K(x, z) = x^\top z$	*Feature* sudah banyak dan informatif, misalnya teks TF-IDF. Paling cepat dan paling mudah dibaca.
Polinomial	$K(x, z) = (\gamma\, x^\top z + r)^d$	Interaksi antarfitur penting sampai derajat tertentu. Derajat di atas 3 mudah tidak stabil secara numerik.
RBF (Gaussian)	$K(x, z) = \exp\big(-\gamma \lVert x - z \rVert^2\big), \gamma > 0$	Pilihan bawaan bila bentuk batas tidak diketahui; padanan *feature space*-nya berdimensi tak hingga.

Arti γ pada RBF. Nilai *kernel* adalah kemiripan: 1 untuk dua titik yang sama, mendekati 0 untuk dua titik yang berjauhan. Untuk $\lVert x - z \rVert^2 = 4$ (*output* kode sebelumnya):

γ	0,01	0,1	1	10
$K(x, z)$	0,961	0,670	0,018	$\approx 4 \times 10^{-18}$

γ kecil membuat setiap titik berpengaruh luas sehingga batasnya *smooth*. γ besar membatasi pengaruh di sekitar titik itu sendiri, dan batasnya pecah menjadi pulau-pulau kecil.

Menyetel C dan γ

Param.	Terlalu kecil: gejala *underfit*	Terlalu besar: gejala *overfit*
C	*Training error* dan *validation error* sama-sama tinggi; batas terlalu lurus.	*Training error* mendekati nol tetapi *validation error* naik; batas berliku mengikuti titik tunggal.
γ	Model berperilaku seperti *kernel* linear; kantong kecil tidak tertangkap.	Hampir semua *training data* menjadi *support vector*; *training accuracy* 100% sedangkan validasi runtuh.

Panduan LIBSVM (Hsu, Chang, Lin)

1. Skalakan *feature*, lalu mulai dengan *kernel* RBF.
2. Cari pasangan (C, γ) pada skala log:
   $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$, $\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$.
3. *Grid search* dengan *cross-validation* bertingkat; kasar dulu, lalu perhalus di sekitar titik terbaik.
4. Latih ulang SVM(C^*, γ^*) pada seluruh *training data*.

Scaling feature wajib

RBF bergantung pada $\|x - z\|^2$, yang dikuasai *feature* bersatuan besar. Bila penghasilan (puluhan juta rupiah) dicampur dengan usia (puluhan tahun), suku usia praktis hilang. Parameter *scaling* dihitung hanya dari *training data*, di dalam `Pipeline`, agar informasi *test data* tidak bocor.

Grid search dengan dan tanpa *scaling*

```
import numpy as np
from sklearn.datasets import make_circles
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

X, y = make_circles(n_samples=400, noise=0.1, factor=0.4, random_state=0)
X[:, 1] *= 1000                          # satu fitur bersatuan besar
grid = {"svc__C": 2.0 ** np.arange(-5, 16, 2),
        "svc__gamma": 2.0 ** np.arange(-15, 4, 2)}
cv = StratifiedKFold(5, shuffle=True, random_state=0)
for nama, pipa in {"scaling": make_pipeline(StandardScaler(), SVC()),
                   "tanpa": make_pipeline(SVC())}.items():
    gs = GridSearchCV(pipa, grid, cv=cv, n_jobs=-1).fit(X, y)
    print(nama, round(gs.best_score_, 3), gs.best_params_)
```

Data dua cincin, satu *feature*-nya sengaja dibesarkan seribu kali. Pada satu kali percobaan, *accuracy* CV terbaik dengan *scaling* mencapai **1,00**; tanpa *scaling* hanya sekitar **0,81**. *Scaler* berada di dalam `Pipeline`, sehingga ia dipasang ulang pada setiap lipatan.

Multikelas, *regression*, dan data besar

Multikelas

SVM pada dasarnya biner.

- ***One-vs-Rest***: K model, "kelas ini melawan semua". Data menjadi timpang.
- ***One-vs-One***: $K(K-1)/2$ model, suara terbanyak; bawaan `SVC`. Pada 10 kelas ada 45 model kecil, sering lebih cepat daripada 10 model besar.

SVR

Support vector regression mencari pita selebar ϵ di sekitar fungsi. *Error* di dalam pita tidak didenda sama sekali; hanya titik di luar pita yang menjadi *support vector*. Bersama RBF, SVR mencocokkan hubungan tak linier dan cukup tahan terhadap *noise* kecil.

Data besar

`SVC` memakai LIBSVM (turunan SMO). *Training cost* kira-kira $O(n^2)$ sampai $O(n^3)$ dan matriks *kernel* $n \times n$, sehingga berat begitu n mencapai ratusan ribu. Untuk teks besar pakai `LinearSVC` (LIBLINEAR) atau `SGDClassifier` dengan *hinge loss*.

SVM dibanding naive Bayes dan kNN

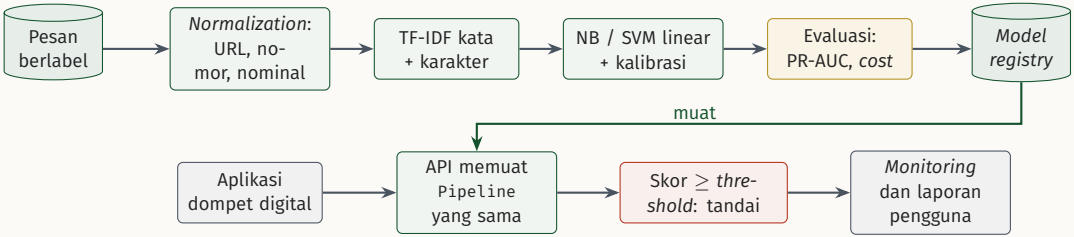
	Naive Bayes (P5)	**kNN (P6)**	**SVM**
Jenis model	Generatif, probabilistik	Berbasis contoh, *lazy*	Diskriminatif, geometris
Asumsi utama	*Feature* bebas bersyarat diberikan kelas	Titik berdekatan berlabel sama	*Margin* lebar menggeneralisasi baik
Training cost	$O(nd)$, satu lintasan	$O(1)$, hanya menyimpan	$O(n^2)$–$O(n^3)$ ber-*kernel*
Inference cost	$O(dc)$	$O(nd)$ tiap pertanyaan	$O(\lvert SV\rvert \cdot d)$; $O(d)$ bila linear
Output peluang	Ada, kalibrasinya buruk	Proporsi tetangga	Perlu *Platt scaling*
Scaling feature	Tidak perlu	Wajib	Wajib
Hyperparameter	Praktis hanya α	k, *distance metric*	C, *kernel*, γ atau d

Urutan yang lazim untuk teks

Naive Bayes sebagai *baseline* hari itu juga, lalu SVM linear pada TF-IDF, baru model yang lebih berat. SVM ber-*kernel* cocok untuk data numerik berukuran menengah dengan batas rumit. kNN dihindari bila $d \gg n$ atau jawaban harus keluar dalam milidetik.

Studi kasus: penyaring SMS penipuan dompet digital

"Selamat! Nomor Anda memenangkan undian Rp100 juta, klik *link* berikut" harus ditandai, sedangkan kode OTP dan pemberitahuan transaksi tidak boleh terganggu. Untuk latihan dipakai SMS Spam Collection (Almeida dkk., 2011) sebelum data berbahasa Indonesia tersedia.



- *n-gram* karakter 3–5 huruf menangkap tulisan samaran seperti "h4d1ah" dan "gr4tis"; URL, nomor telepon, dan nominal diganti token `<URL>`, `<TELP>`, `<RUPIAH>`.
- *Threshold* dipilih dari *cost*: menahan satu OTP sah jauh lebih merugikan daripada meloloskan satu promosi. Pola penipuan cepat berubah, dan murahnya NB serta SVM linear membuat *training* ulang berkala terjangkau.

Data berurutan dan *hidden Markov model*

Ketika urutan baris ikut membawa informasi

Model sejauh ini menganggap data **i.i.d.**, sehingga $p(x_1, \ldots, x_T) = \prod_t p(x_t)$ dan urutan baris tidak berpengaruh. Banyak data melanggarnya:

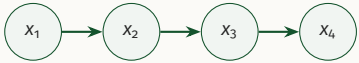
- **Teks.** “bisa” berarti dapat dalam “saya bisa datang” dan racun dalam “bisa ular kobra”.
- **Suara.** Satu fonem terentang beberapa *frame* 10–25 ms berturut-turut.
- **Keuangan.** Hari bergejolak cenderung diikuti hari bergejolak (*volatility clustering*).
- **DNA.** *CpG island* terentang ratusan basa dengan komposisi berbeda.

Aturan rantai $p(x_1) \prod_t p(x_t \mid x_1, \ldots, x_{t-1})$ selalu benar, tetapi untuk kosakata 10.000 kata tabel $p(x_3 \mid x_1, x_2)$ saja memuat 10^{12} entri.

i.i.d.



Markov orde pertama



Data Markov, 1913

20.000 huruf pertama *Eugene Onegin*: 43,2% vokal. Setelah vokal, peluang vokal lagi hanya 12,8%; setelah konsonan 66,3%. Mengingat satu huruf ke belakang sudah menangkap sebagian besar ketergantungan.

Rantai Markov dan matriks transisi

Sifat Markov: masa depan hanya bergantung pada keadaan sekarang,

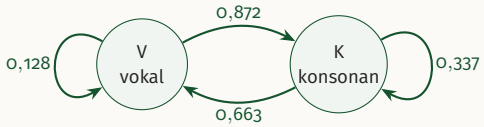
$$P(q_{t+1} = s_j \mid q_t = s_i, q_{t-1}, \ldots, q_1) = a_{ij}.$$

Matriks transisi $A = [a_{ij}]$ stokastik baris ($\sum_j a_{ij} = 1$). Bersama $\pi_i = P(q_1 = s_i)$:

$$P(q_1, \ldots, q_T) = \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1} q_t}.$$

Taksiran *maximum likelihood*: jumlah pasangan $i \to j$ dibagi jumlah kemunculan i. Dari 8.638 vokal, 11.362 konsonan, 1.104 pasangan V–V, dan 3.827 pasangan K–K: $a_{VV} = 1104/8638$, $a_{VK} = 7534/8638$, $a_{KV} = 7535/11362$, $a_{KK} = 3827/11362$:

$$A \approx \begin{pmatrix} 0{,}128 & 0{,}872 \\ 0{,}663 & 0{,}337 \end{pmatrix}.$$



Setiap lingkaran adalah keadaan, setiap panah satu entri a_{ij}. Model ini memerlukan $N(N-1)$ parameter untuk A dan $N-1$ untuk π, berapa pun panjang urutannya.

Peluang n langkah dan distribusi stasioner

Peluang huruf kedua sesudah vokal juga vokal:
$a_{VV}a_{VV} + a_{VK}a_{KV} = 0{,}128 \cdot 0{,}128 + 0{,}872 \cdot 0{,}663 \approx 0{,}595$, yaitu entri (V, V) dari A^2. Secara umum peluang n langkah adalah A^n dan $p_t = p_1 A^{t-1}$.

n	1	2	3	5	10	stasioner
vokal	0,128	0,595	0,345	0,407	0,433	0,432
konsonan	0,872	0,405	0,655	0,593	0,567	0,568

Distribusi stasioner memenuhi $sA = s$, $\sum_i s_i = 1$. Untuk dua keadaan:

$$s_V = \frac{a_{KV}}{a_{VK} + a_{KV}} = \frac{0{,}663}{0{,}872 + 0{,}663} \approx 0{,}432,$$

sama dengan proporsi vokal di teks, 8.638/20.000.

Rantai yang *irreducible* dan *aperiodic* punya distribusi stasioner tunggal. Nilai eigen kedua A adalah $-0{,}535$: selisih dengan s menyusut kira-kira $0{,}535$ kali per langkah sambil berganti tanda, sehingga baris tabel berayun. PageRank adalah distribusi stasioner pengguna yang berpindah acak antarhalaman web.

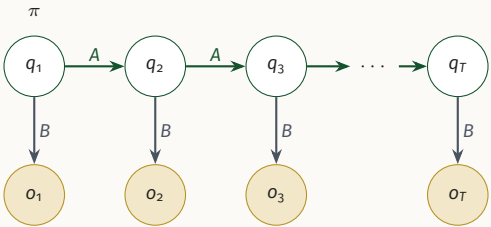
Keadaan yang tidak teramati: $\lambda = (A, B, \pi)$

Pada POS *tagging* yang tertulis kata, kelas katanya harus ditebak. Di pasar harga terlihat, "kondisi pasar" tidak diumumkan. HMM diskret dengan N keadaan dan M simbol:

- transisi $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$
- emisi $b_j(k) = P(o_t = v_k \mid q_t = s_j)$
- distribusi awal $\pi_i = P(q_1 = s_i)$

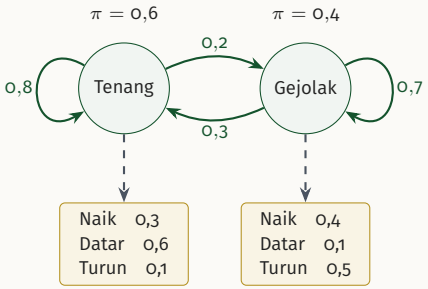
Dua asumsi: keadaan membentuk rantai Markov orde pertama, dan observasi o_t hanya bergantung pada q_t (**independensi output**). HMM generatif, sehingga

$$P(O, Q \mid \lambda) = \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1} q_t} \, b_{q_t}(o_t).$$



Lingkaran putih tersembunyi, lingkaran kuning teramati. Rabiner dan Jurafsky–Martin menulis q_t, o_t; Bishop memakai $\mathbf{z}_n$, $\mathbf{x}_n$ seperti GMM di Pertemuan 5. Besaran EM (γ, ξ) mengikuti Bishop.

Contoh berjalan: regime pasar



Dua regime tersembunyi (Tenang, Gejolak); setiap hari dicatat satu simbol (Naik, Datar, Turun):

$$A = \begin{pmatrix} 0{,}8 & 0{,}2 \\ 0{,}3 & 0{,}7 \end{pmatrix}, \quad B = \begin{pmatrix} 0{,}3 & 0{,}6 & 0{,}1 \\ 0{,}4 & 0{,}1 & 0{,}5 \end{pmatrix},$$

dan $\pi = (0{,}6;\ 0{,}4)$.

Angka dipilih agar hitungan tangan mudah diikuti, tidak ditaksir dari data pasar mana pun.

Untuk $O = (\text{Datar}, \text{Turun}, \text{Turun})$ dan $Q = (\text{T}, \text{G}, \text{G})$:

$$\begin{aligned} P(O, Q \mid \lambda) &= \pi_T b_T(\text{D}) \cdot a_{TG} b_G(\text{Tu}) \cdot a_{GG} b_G(\text{Tu}) \\ &= 0{,}6 \cdot 0{,}6 \cdot 0{,}2 \cdot 0{,}5 \cdot 0{,}7 \cdot 0{,}5 = 0{,}0126. \end{aligned}$$

Urutan O ini dipakai di semua hitungan berikutnya.

Tiga masalah klasik (Rabiner, 1989)

Masalah	Pertanyaan	Algoritma dan contoh pemakaian
Evaluasi	Diberikan λ dan O, berapa $P(O \mid \lambda)$?	*Forward* (atau *backward*); memilih model kata yang paling cocok dengan rekaman suara
Decoding	Diberikan λ dan O, urutan keadaan Q mana yang paling mungkin?	Viterbi; menebak kelas kata setiap kata, menebak regime setiap hari
Learning	Diberikan O (satu atau banyak urutan), λ mana yang memaksimalkan $P(O \mid \lambda)$?	Baum–Welch (EM); melatih dari rekaman tanpa label per *frame*

Sejarah singkat: Baum dan rekan di Institute for Defense Analyses menurunkan cara melatih rantai yang keadaannya tersembunyi (1970); Viterbi merancang algoritmanya untuk sandi konvolusi (1967); kelompok suara di IBM dan Bell Labs merangkai keduanya menjadi HMM pada 1970–1980-an.

Forward, *backward*, dan Viterbi

Cara naif: jumlahkan semua lintasan

$$P(O \mid \lambda) = \sum_{Q} P(O, Q \mid \lambda)$$

Untuk contoh berjalan ada $2^3 = 8$ lintasan. Lintasan terbesar T–G–G (0,0126) menyumbang sekitar separuh peluang total.

Masalahnya, ada N^T lintasan dan masing-masing memerlukan sekitar $2T$ perkalian. Untuk $N = 5$ dan $T = 100$:

$$2 \cdot 100 \cdot 5^{100} \approx 1{,}6 \times 10^{72} \text{ operasi.}$$

Lintasan	$P(O, Q)$	**Lintasan**	$P(O, Q)$
T–T–T	0,002304	G–T–T	0,000096
T–T–G	0,002880	G–T–G	0,000120
T–G–T	0,001080	G–G–T	0,000420
T–G–G	**0,012600**	G–G–G	0,004900
Jumlah $= P(O \mid \lambda)$			**0,024400**

T = Tenang, G = Gejolak, O = (Datar, Turun, Turun). Perhatikan T–T–T dan T–T–G berbagi awalan T–T yang dihitung dua kali. Algoritma *forward* menghapus pemborosan itu.

Algoritma *forward*

Variabel *forward* menyimpan hasil antara:

$$\alpha_t(i) = P(o_1, \ldots, o_t, q_t = s_i \mid \lambda),$$

peluang melihat t observasi pertama *dan* berada di s_i pada waktu t. Karena sifat Markov,

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}\right] b_j(o_{t+1}).$$

Ini pemrograman dinamis: setiap α dihitung sekali lalu dipakai ulang. *Cost*-nya $O(N^2 T)$; untuk $N = 5$, $T = 100$ hanya sekitar 2.500 perkalian.

Algoritma *Forward*

Input: $\lambda = (A, B, \pi)$, $O = (o_1, \ldots, o_T)$

1. ***Initialization.*** Untuk $i = 1..N$: $\alpha_1(i) \leftarrow \pi_i\, b_i(o_1)$
2. **Induksi.** Untuk $t = 1..T-1$ dan $j = 1..N$: $\alpha_{t+1}(j) \leftarrow \left[\sum_i \alpha_t(i)\, a_{ij}\right] b_j(o_{t+1})$
3. **Terminasi.** $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Output: $P(O \mid \lambda)$ dan tabel α berukuran $T \times N$

Forward dengan tangan untuk O = (Datar, Turun, Turun)

$t = 1$**, Datar:**

$$\alpha_1(T) = 0{,}6 \cdot 0{,}6 = 0{,}36$$
$$\alpha_1(G) = 0{,}4 \cdot 0{,}1 = 0{,}04$$

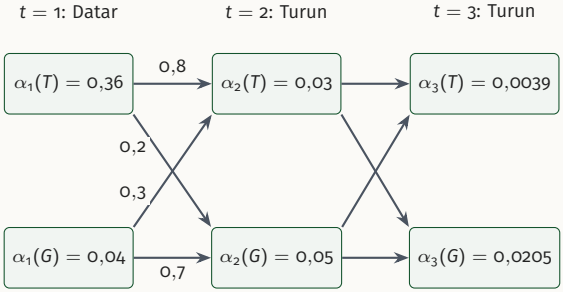
$t = 2$**, Turun:**

$$\alpha_2(T) = (0{,}36 \cdot 0{,}8 + 0{,}04 \cdot 0{,}3) \cdot 0{,}1 = 0{,}03$$
$$\alpha_2(G) = (0{,}36 \cdot 0{,}2 + 0{,}04 \cdot 0{,}7) \cdot 0{,}5 = 0{,}05$$

$t = 3$**, Turun:**

$$\alpha_3(T) = (0{,}03 \cdot 0{,}8 + 0{,}05 \cdot 0{,}3) \cdot 0{,}1 = 0{,}0039$$
$$\alpha_3(G) = (0{,}03 \cdot 0{,}2 + 0{,}05 \cdot 0{,}7) \cdot 0{,}5 = 0{,}0205$$

Terminasi: $P(O \mid \lambda) = 0{,}0039 + 0{,}0205 = \mathbf{0{,}0244}$**,** sama dengan jumlah delapan lintasan, dengan 14 perkalian (12 di langkah induksi) alih-alih 40.



Trellis: baris adalah keadaan, kolom adalah waktu. Setiap $\alpha_{t+1}(j)$ adalah jumlah dua panah masuk $\alpha_t(i)\, a_{ij}$, dikali peluang emisi o_{t+1} di keadaan j. Angka pada panah pertama adalah a_{ij}.

Backward dan peluang *posterior* keadaan

Variabel *backward* merangkum masa depan:

$$\beta_t(i) = P(o_{t+1}, \ldots, o_T \mid q_t = s_i, \lambda),$$

dihitung mundur dari $\beta_T(i) = 1$:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j).$$

Karena $P(O, q_t = s_i \mid \lambda) = \alpha_t(i)\,\beta_t(i)$,

$$\gamma_t(i) = P(q_t = s_i \mid O, \lambda) = \frac{\alpha_t(i)\,\beta_t(i)}{P(O \mid \lambda)}.$$

Angka contoh berjalan

$\beta_3 = (1;\ 1)$

$\beta_2(T) = 0{,}8 \cdot 0{,}1 + 0{,}2 \cdot 0{,}5 = 0{,}18$
$\beta_2(G) = 0{,}3 \cdot 0{,}1 + 0{,}7 \cdot 0{,}5 = 0{,}38$

$\beta_1(T) = 0{,}8 \cdot 0{,}1 \cdot 0{,}18 + 0{,}2 \cdot 0{,}5 \cdot 0{,}38 = 0{,}0524$
$\beta_1(G) = 0{,}3 \cdot 0{,}1 \cdot 0{,}18 + 0{,}7 \cdot 0{,}5 \cdot 0{,}38 = 0{,}1384$

Periksa: $0{,}36 \cdot 0{,}0524 + 0{,}04 \cdot 0{,}1384 = 0{,}0244$.

$\gamma_1 = (0{,}773;\ 0{,}227)$, $\gamma_2 = (0{,}221;\ 0{,}779)$, $\gamma_3 = (0{,}160;\ 0{,}840)$. Hari pertama (Datar) kemungkinan besar Tenang; dua hari Turun berikutnya kemungkinan besar Gejolak. Gabungan ini disebut *forward–backward* (Bishop: *alpha–beta*).

Viterbi: jumlah diganti maksimum

Memilih keadaan terbaik per waktu dari γ_t (*posterior decoding*) bisa menghasilkan urutan mustahil bila suatu $a_{ij} = 0$. Untuk POS *tagging* kita perlu satu urutan konsisten $Q^* = \arg\max_Q P(O, Q \mid \lambda)$.

$$\delta_t(i) = \max_{q_1,\ldots,q_{t-1}} P(q_1, \ldots, q_{t-1}, q_t = s_i, o_1, \ldots, o_t \mid \lambda)$$

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i)\, a_{ij}\right] b_j(o_{t+1}), \quad \psi_{t+1}(j) = \arg\max_i \delta_t(i)\, a_{ij}$$

Penunjuk ψ (*psi*) mencatat asal lintasan terbaik; δ dibaca *delta*. Di $t = T$ pilih δ_T terbesar, lalu telusuri ψ mundur (*backtracking*): $q_t^* = \psi_{t+1}(q_{t+1}^*)$. Waktu $O(N^2T)$, memori $O(NT)$; dalam bahasa Bishop ini algoritma *max-sum*.

Dengan tangan, O = (Datar, Turun, Turun)

$\delta_1 = (0{,}36;\ 0{,}04)$, sama dengan *forward*.

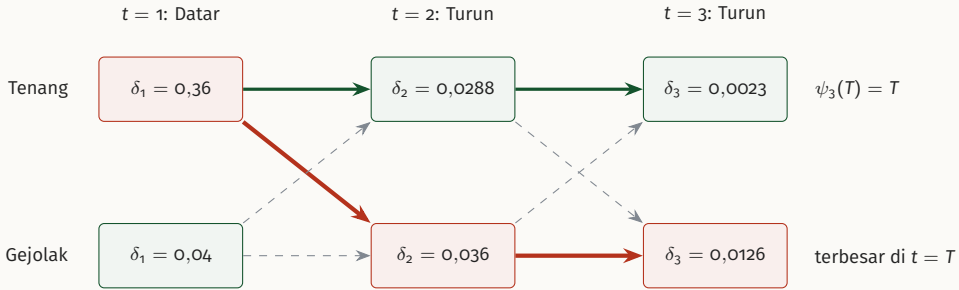
$t = 2$: $\delta_2(T) = \max(0{,}36 \cdot 0{,}8;\ 0{,}04 \cdot 0{,}3) \cdot 0{,}1 = 0{,}0288$
$\delta_2(G) = \max(0{,}36 \cdot 0{,}2;\ 0{,}04 \cdot 0{,}7) \cdot 0{,}5 = 0{,}036$
Keduanya datang dari Tenang: $\psi_2(T) = \psi_2(G) = T$.

$t = 3$: $\delta_3(T) = \max(0{,}0288 \cdot 0{,}8;\ 0{,}036 \cdot 0{,}3) \cdot 0{,}1 = 0{,}002304$
$\delta_3(G) = \max(0{,}0288 \cdot 0{,}2;\ 0{,}036 \cdot 0{,}7) \cdot 0{,}5 = 0{,}0126$
$\psi_3(T) = T$ ($0{,}02304 > 0{,}0108$), $\psi_3(G) = G$
($0{,}0252 > 0{,}00576$).

***Backtracking*:** $q_3^* = G$, $q_2^* = \psi_3(G) = G$, $q_1^* = \psi_2(G) = T$.

Lintasan terbaik Tenang–Gejolak–Gejolak dengan $P^* = \mathbf{0{,}0126}$, sama dengan baris tebal tabel delapan lintasan. Di contoh ini hasilnya kebetulan sama dengan *posterior decoding*.

Trellis Viterbi



Panah hijau tebal adalah *survivor* (ψ) setiap *node*; panah abu-abu putus-putus adalah kandidat yang kalah. Panah merah adalah hasil *backtracking* dari $\delta_3(G) = 0{,}0126$: Tenang, Gejolak, Gejolak.

Viterbi di luar HMM

Viterbi aslinya men-*decode* sandi konvolusi pada kanal berderau, dan masih dipakai di modem, telepon seluler, dan komunikasi satelit. Bentuk yang sama muncul di lintasan terpendek pada graf berlapis, *decoding* CRF berantai, dan penyelarasan urutan DNA.

Baum–Welch, *underflow*, dan kode

Dari menghitung ke EM

Keadaan dilabeli (korpus POS beranotasi): cukup menghitung,

$$\hat{a}_{ij} = \frac{\#(s_i \to s_j)}{\#(s_i \to \cdot)}, \quad \hat{b}_j(k) = \frac{\#(q_t = s_j,\ o_t = v_k)}{\#(q_t = s_j)}.$$

Keadaan tersembunyi: situasinya sama dengan GMM di Pertemuan 5. Di sana komponen setiap titik tidak diketahui, di sini keadaan setiap waktu. Jalan keluarnya juga EM, versi HMM-nya disebut **Baum–Welch**.

Langkah E: $\gamma_t(i)$ dan

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)}{P(O \mid \lambda)}.$$

Langkah M: hitungan pasti diganti *expected counts*,

$$\hat{\pi}_i = \gamma_1(i), \qquad \hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)},$$
$$\hat{b}_j(k) = \frac{\sum_{t:\, o_t = v_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.$$

Kaitan dengan GMM (Pertemuan 5)

Pada GMM *weight* π_k diperbarui dengan rata-rata *responsibility* $\gamma(z_{nk})$. Pada HMM peran itu dipecah menjadi $\hat{\pi}$ untuk waktu pertama dan $\hat{a}_{ij}$ untuk setiap perpindahan. Bila semua baris A dipaksa sama, keadaan tidak lagi bergantung pada waktu sebelumnya dan HMM berubah menjadi model campuran. HMM emisi Gaussian memakai rumus $\hat{\mu}_j$, $\hat{\Sigma}_j$ dengan *weight* γ yang persis sama dengan langkah M GMM.

Satu iterasi Baum–Welch, dan sifat-sifatnya

Dari $\gamma_1, \gamma_2, \gamma_3$ di atas:

$$\xi_1 = \begin{pmatrix} 0{,}2125 & 0{,}5607 \\ 0{,}0089 & 0{,}2180 \end{pmatrix}, \quad \xi_2 = \begin{pmatrix} 0{,}0984 & 0{,}1230 \\ 0{,}0615 & 0{,}7172 \end{pmatrix}$$

misalnya $\xi_1(T, G) = 0{,}36 \cdot 0{,}2 \cdot 0{,}5 \cdot 0{,}38 / 0{,}0244 = 0{,}5607$.

$$\hat{a}_{TT} = \frac{0{,}2125 + 0{,}0984}{0{,}773 + 0{,}221} = 0{,}313, \quad \hat{b}_T(\text{Datar}) = \frac{0{,}773}{1{,}154} = 0{,}670$$

$$\hat{A} = \begin{pmatrix} 0{,}313 & 0{,}687 \\ 0{,}070 & 0{,}930 \end{pmatrix}, \quad \hat{B} = \begin{pmatrix} 0 & 0{,}670 & 0{,}330 \\ 0 & 0{,}123 & 0{,}877 \end{pmatrix}$$

$P(O \mid \lambda)$ naik dari 0,0244 menjadi 0,3192. Kolom Naik menjadi nol karena simbol itu tidak pernah muncul: dengan tiga observasi EM menyesuaikan diri berlebihan.

Baum–Welch menjamin $P(O \mid \hat{\lambda}) \geq P(O \mid \lambda)$ di setiap iterasi, tetapi hanya sampai **optimum lokal**.

- ***Initialization.*** Keadaan yang diinisialisasi identik tidak pernah terpisah. Jalankan dari beberapa titik awal acak.
- **Nol yang terkunci.** a_{ij} atau $b_j(k)$ yang pernah nol tetap nol. Tambahkan *pseudocount*, sejalan dengan *Laplace smoothing* di Pertemuan 5.
- **Label tertukar.** Keadaan pertama hasil latih bisa saja yang kita sebut Gejolak.

Lima titik awal, 20 urutan × 100 hari

Model sebenarnya $-2136{,}95$. Seed 1–4 berakhir di sekitar $-2133{,}7$; seed 0 tertahan di $-2175{,}85$, hampir sama dengan model satu keadaan. Seed 4 berada di dataran yang sama sekitar 280 iterasi sebelum lepas. Kurva datar belum tentu konvergen.

Underflow: faktor skala dan ruang log

T	*forward* polos	berskala: $\log_{10} P$
10	$6{,}28 \times 10^{-6}$	$-5{,}20$
100	$1{,}25 \times 10^{-45}$	$-44{,}90$
500	$4{,}41 \times 10^{-231}$	$-230{,}36$
1000	0	$-464{,}20$
2000	0	$-929{,}67$

Float 64-bit menyimpan bilangan normal sampai sekitar $2{,}2 \times 10^{-308}$. Pada urutan sintetis ini $\sum_i \alpha_t(i)$ tepat nol mulai $t = 698$. Satu detik ucapan dengan *frame* 10 ms sudah 100 observasi.

Faktor skala (Rabiner; Bishop 13.2.4). Normalkan setiap langkah:

$$c_t = \frac{1}{\sum_i \alpha_t(i)}, \qquad \hat{\alpha}_t(i) = c_t\, \alpha_t(i),$$

$$\log P(O \mid \lambda) = -\sum_t \log c_t.$$

Contoh berjalan: jumlah per langkah sebelum *normalization* ($1/c_t$) 0,40, 0,20, 0,305, dan $0{,}40 \cdot 0{,}20 \cdot 0{,}305 = 0{,}0244$; $\hat{\alpha}_1 = (0{,}9;\ 0{,}1)$.

Ruang log. Simpan $\log \alpha$; penjumlahan memerlukan ***log-sum-exp***:

$$\log \sum_i e^{x_i} = m + \log \sum_i e^{x_i - m}, \qquad m = \max_i x_i.$$

Viterbi lebih sederhana lagi karena hanya ada maksimum dan perkalian: $\log \delta_{t+1}(j) = \max_i [\log \delta_t(i) + \log a_{ij}] + \log b_j(o_{t+1})$. Faktor skala dirancang untuk *forward*; untuk Viterbi pakai ruang log.

Kode NumPy (1): *forward* berskala

```python
import numpy as np

def forward_scaled(A, B, pi, O):
    T, N = len(O), A.shape[0]
    alpha, c = np.zeros((T, N)), np.zeros(T)
    a = pi * B[:, O[0]]                    # inisialisasi
    c[0] = 1.0 / a.sum(); alpha[0] = a * c[0]
    for t in range(1, T):                  # induksi, lalu normalkan
        a = (alpha[t - 1] @ A) * B[:, O[t]]
        c[t] = 1.0 / a.sum(); alpha[t] = a * c[t]
    return alpha, c, -np.log(c).sum()      # log P(O | lambda)

A  = np.array([[0.8, 0.2], [0.3, 0.7]])              # (Tenang, Gejolak)
B  = np.array([[0.3, 0.6, 0.1], [0.4, 0.1, 0.5]])    # (Naik, Datar, Turun)
pi = np.array([0.6, 0.4])
O  = np.array([1, 2, 2])                             # Datar, Turun, Turun
alpha, c, ll = forward_scaled(A, B, pi, O)
print(1 / c, np.exp(ll))                             # [0.4 0.2 0.305] 0.0244
```

Baris `1 / c` adalah jumlah α per langkah dari slide *underflow*, dan hasil kalinya $P(O \mid \lambda) = 0{,}0244$. Fungsi `backward_scaled`, `e_step`, dan `m_step` (Baum–Welch) ada di `bab07-hmm/hmm_numpy.py`.

Kode NumPy (2): Viterbi di ruang log

```
def viterbi_log(A, B, pi, O):
    T, N = len(O), A.shape[0]
    with np.errstate(divide="ignore"):       # log(0) = -inf, aman
        lA, lB, lpi = np.log(A), np.log(B), np.log(pi)
    delta = np.zeros((T, N))
    psi = np.zeros((T, N), dtype=int)
    delta[0] = lpi + lB[:, O[0]]
    for t in range(1, T):
        skor = delta[t - 1][:, None] + lA    # skor[i, j]: dari i ke j
        psi[t] = skor.argmax(axis=0)
        delta[t] = skor.max(axis=0) + lB[:, O[t]]
    q = np.zeros(T, dtype=int)
    q[-1] = delta[-1].argmax()
    for t in range(T - 2, -1, -1):           # backtracking
        q[t] = psi[t + 1, q[t + 1]]
    return q, delta[-1].max()

q, lp = viterbi_log(A, B, pi, O)
print(q, np.exp(lp))                         # [0 1 1] 0.0126  -> T, G, G
```

Transisi mustahil ($a_{ij} = 0$) menjadi $-\infty$ dan otomatis tidak pernah terpilih. Jangan menghitung log dari *forward* polos yang sudah nol: hasilnya `-inf` tanpa pesan *error*.

Aplikasi dan praktikum

POS *tagging* bahasa Indonesia: kasus “bisa”

Tag adalah keadaan tersembunyi, kata adalah observasi. Untuk bahasa Indonesia, Wicaksono dan Purwarianti (2010) membangun tagger HMM, dan Dinakaramani dkk. (2014) merancang tagset 23 tag beserta korpus beranotasi.

Korpus mini 12 kalimat (58 token, 33 kata), dilatih dengan menghitung dan *add-k smoothing* ($k = 0{,}1$), lalu Viterbi. Emisinya sudah condong: $b_{MD}(\text{bisa}) = 0{,}554$, $b_{NN}(\text{bisa}) = 0{,}098$. Yang menentukan adalah transisi:

- setelah pronomina: $a_{PRP,MD} = 0{,}404$, $a_{PRP,NN} = 0{,}019$
- setelah verba: $a_{VB,NN} = 0{,}696$, $a_{VB,MD} = 0{,}010$

Hasil Viterbi pada tiga kalimat baru

```
adik/PRP bisa/MD membaca/VB buku/NN itu/PR
  (log P = -12.52)

racun/NN ular/NN itu/PR punya/VB bisa/NN
  (log P = -14.73)

kobra/NNP punya/VB bisa/NN yang/SC berbahaya/JJ
  (log P = -15.09)
```

“Bisa” benar di ketiga kalimat. Dua kesalahan: “adik” tidak ada di korpus, sehingga semua tag memancarkan `<unk>` hampir sama dan posisi awal kalimat memilih PRP (seharusnya NN). “Kobra” diberi NNP karena NN memancarkan banyak kata sehingga peluang per katanya kecil. Tagger sungguhan memakai korpus besar, *feature* akhiran, dan penanganan kata tak dikenal.

Batas HMM dan penerusnya

- **Ingatan satu langkah.** Ketergantungan jarak jauh butuh orde lebih tinggi, dengan *cost* N^2 yang tumbuh cepat.
- **Durasi geometris.** Peluang bertahan d langkah $a_{ii}^{d-1}(1 - a_{ii})$ paling besar di $d = 1$; *hidden semi-Markov model* menambah distribusi durasi.
- ***Feature* tumpang tindih** (kata, akhiran, huruf kapital) sulit masuk ke $b_j(o_t)$.
- **Generatif.** HMM memaksimalkan $P(O, Q)$, padahal *tagging* hanya butuh $P(Q \mid O)$.

	HMM	**CRF berantai**	**RNN / *transformer***
Dimodelkan	$P(O, Q)$	$P(Q \mid O)$	$P(Q \mid O)$
Keadaan	diskret, N nilai	label diskret	vektor kontinu
Feature	satu observasi per waktu	bebas atas seluruh urutan	dipelajari
Training	hitung atau Baum–Welch	konveks, perlu label	*backprop*, data besar

HMM tetap layak bila data sedikit, keadaannya perlu ditafsirkan (regime, fase penyakit, tahap proses), atau *training* harus tanpa label. GMM-HMM menjadi tulang punggung pengenalan suara sampai sekitar 2012, dan *layer* CRF di atas BiLSTM atau *transformer* masih menjalankan Viterbi.

Lab: SVM dan HMM dengan tangan, lalu dengan kode

Bagian SVM (45 menit)

1. Jalankan `svm_enam_titik.py`; cocokkan w, b, *support vector*, dan `dual_coef_` dengan hitungan tangan.
2. Tambahkan $x_8 = (2{,}5;\ 0)$ berlabel -1. Latih ulang dengan $C = 10^6$ dan $C = 1$; catat perubahan w, b, dan jumlah *support vector*.
3. Jalankan `svm_grid_search.py`; buat *heatmap accuracy* CV atas (C, γ) untuk versi dengan dan tanpa *scaling*.
4. SMS Spam Collection: `MultinomialNB` lawan `LinearSVC` pada TF-IDF; laporkan *precision*, *recall*, PR-AUC, dan *training time*.

Bagian HMM (45 menit)

5. Ketik `forward_scaled` dan `viterbi_log`; pastikan *output*-nya 0,0244 dan T–G–G dengan $P^* = 0{,}0126$.
6. Bangkitkan urutan $T = 2000$ dari model regime; bandingkan *forward* polos dan berskala, lalu temukan t saat *forward* polos menjadi nol.
7. Latih Baum–Welch dari lima seed pada 20 urutan $\times$ 100 hari; gambar riwayat *log-likelihood* dan tunjukkan seed yang tertahan.
8. Jalankan `pos_tagging.py`; tambahkan tiga kalimat dengan “jalan” sebagai nomina dan verba, lalu periksa tag yang dihasilkan.

Kumpulkan satu *notebook* berisi kode, *output*, dan dua sampai tiga kalimat tafsiran untuk setiap langkah.

Rekap Bagian I menjelang UTS

Model klasik Bagian I dalam satu tabel

Model (pertemuan)	Asumsi / *inductive bias*	*Training cost*	Inferensi	Keterbacaan
Belajar dari data berlabel				
Find-S, *version space* (P2)	Konsep konjungtif; data bebas *noise*	$O(nd)$	$O(d)$	sangat tinggi
Decision tree (P3)	Pohon pendek lebih disukai; batas sejajar sumbu	$O(dn \log n)$	$O(h)$	tinggi: aturan
Random forest, XGBoost (P4)	Banyak pohon saling menutup *error*	$B\times$ satu pohon	$O(Bh)$	rendah; SHAP
Naive Bayes (P5)	*Feature* bebas bersyarat diberikan kelas	$O(nd)$	$O(dc)$	menengah
kNN (P6)	Titik berdekatan berlabel sama; skala seragam	$O(1)$	$O(nd)$	tunjukkan tetangga
SVM (P7)	*Margin* terlebar; *kernel* menentukan batas	$O(n^2)$–$O(n^3)$	$O(\lvert SV\rvert \cdot d)$	rendah bila *kernel*
Mengelompokkan tanpa label				
K-Means (P6)	*Cluster* bulat, berukuran mirip	$O(tnKd)$	$O(Kd)$	tinggi: *centroid*
GMM + EM (P5)	Campuran K Gaussian; *cluster* boleh lonjong	$O(tnKd^2)$	$O(Kd^2)$	menengah
Memodelkan urutan				
HMM (P7)	Markov pada keadaan; independensi output	$O(tN^2T)$	$O(N^2T)$	menengah: A

h kedalaman pohon, B jumlah pohon, K *cluster*, N keadaan, T panjang urutan, t iterasi; kompleksitas hanya gambaran kasar.

Peta pertanyaan untuk memilih model

Training data-nya berlabel?	Tidak: *K-Means* bila *cluster* bulat dan perlu cepat; GMM bila lonjong atau perlu peluang keanggotaan.
Urutan dengan keadaan tersembunyi?	Ya: HMM. Viterbi menebak urutan keadaan, Baum–Welch melatih tanpa label keadaan.
Output-nya bilangan?	Ya: *linear regression* (Bagian II) sebagai patokan, lalu *regression tree*, kNN, atau SVR.
Keputusan harus dapat dijelaskan?	Wajib: *decision tree* atau naive Bayes. Cukup akurat: *ensemble* atau SVM ber-*kernel*.
Anggaran waktu satu jawaban?	Milidetik: hindari kNN tepat dan SVM dengan banyak *support vector*. Longgar: kNN boleh.
Berapa banyak data dan *feature*?	Sedikit: naive Bayes, kNN, pohon pendek. $d \gg n$ (teks): naive Bayes atau SVM linear. Ratusan ribu baris tabel: *ensemble* pohon.
Kelasnya sangat timpang?	Ya: jangan pakai *accuracy*; lihat *precision*, *recall*, PR-AUC; timbang *weight* kelas.

Aturan kerja: mulai dari patokan sederhana (naive Bayes, pohon tunggal, SVM linear), naikkan kerumitan hanya bila angka pada *validation data* benar-benar membaik.

Tugas besar UTS — *weight* 35%

Bentuk. Berkelompok 3–4 orang. *Output*-nya tiga: repositori kode (*notebook*), laporan 8–12 halaman, dan presentasi 12 menit ditambah tanya jawab.

Isi yang wajib ada.

1. **Perumusan masalah** dengan *T*, *P*, dan *E* ala Mitchell.
2. **Data** publik atau data mitra berizin; asal, ukuran, dan keterbatasannya.
3. **Praproses** di dalam *pipeline*; *transformation* dipasang hanya pada *training data*.
4. **Minimal tiga model pembanding** dari Pertemuan 2–7, salah satunya model sederhana sebagai patokan.
5. **Protokol evaluasi**: *split* latih/validasi/uji atau *cross-validation*, *metric* yang cocok, dan selang ketidakpastian. Rinciannya di Pertemuan 8.
6. **Pembahasan**: mengapa model terbaik menang, di mana ia salah, apa risikonya.

Rubrik ringkas

Komponen	*Weight*
Perumusan masalah dan pemahaman data	15%
Praproses dan *feature engineering*	15%
Ragam model dan ketepatan penerapannya	20%
Kebenaran protokol evaluasi	25%
Pembahasan, keterbatasan, dan etika	15%
Laporan dan presentasi	10%

Weight terbesar ada pada protokol. *Accuracy* 84% dengan protokol benar bernilai lebih tinggi daripada *accuracy* 97% yang ternyata membocorkan *test data* ke praproses.

Alat bantu AI boleh dipakai bila bagian yang dibantu dituliskan dan setiap anggota mampu menjelaskan kodenya. Judul dan data disetujui satu pekan sebelum pengumpulan.

Rangkuman

SVM

SVM memilih *hyperplane* dengan *margin* $2/\|w\|$ terlebar, sebuah *quadratic programming* cembung. Hanya *support vector* yang punya $\alpha_i > 0$; pada contoh enam titik, $\alpha = (0{,}25;\ 0{,}25;\ 0{,}5)$ memberi $w = (1, 0)$, $b = -2$, dan *margin* 2.
Soft margin mendenda $C \sum \xi_i$, dengan ξ_i sama dengan *hinge loss*; C besar berarti *regularization* lemah. Bentuk dual hanya memakai $x_i^\top x_j$, sehingga *kernel* dapat menggantinya; γ pada RBF mengatur seberapa lokal pengaruh titik.
Feature wajib diskalakan di dalam `Pipeline`, dan (C, γ) dicari bersama pada kisi berskala log. Untuk teks besar pakai `LinearSVC`.

HMM

- Rantai Markov meringkas urutan dengan A dan π; distribusi stasioner s$A = s$ memberi 0,432 pada data *Eugene Onegin*.
- HMM $\lambda = (A, B, \pi)$ dengan asumsi Markov dan independensi output.
- *Forward* $O(N^2 T)$ memberi $P(O \mid \lambda) = 0{,}0244$; enumerasi $O(TN^T)$.
- Viterbi mengganti jumlah dengan maksimum dan menelusuri ψ: T–G–G, $P^* = 0{,}0126$.
- Baum–Welch adalah EM: langkah E menghitung γ dan ξ, langkah M menormalkan hitungan harapan; hanya optimum lokal.
- Urutan panjang: faktor skala untuk *forward–backward*, ruang log untuk Viterbi.

Kuis singkat dan tugas

Kuis (10 menit)

1. Mengapa membuang titik yang bukan *support vector* tidak mengubah solusi SVM?
2. Pada solusi enam titik, berapa ξ untuk titik (1,5; 1) berlabel $+1$, dan termasuk keadaan mana?
3. Apa gejala γ RBF yang terlalu besar?
4. Dengan parameter regime pasar, berapa α_1 untuk o_1 = Naik?
5. Apa yang berubah dari *forward* ke Viterbi, selain penunjuk ψ?
6. Mengapa *forward* polos gagal pada $T = 1000$?

Tugas

1. Baca buku ajar Bab 9 dan Bab 7; pendalaman: Bishop Bab 7 dan 13, Rabiner (1989).
2. Dua titik $x_1 = (2, 2)$ berlabel $+1$ dan $x_2 = (0, 0)$ berlabel -1: tentukan w, b, lebar *margin*, α_1, α_2; periksa $\sum_i \alpha_i = \|w\|^2$.
3. Tunjukkan bahwa $K(x, z) = (x^\top z + 1)^2$ di $\mathbb{R}^2$ adalah hasil kali dalam pemetaan ϕ berdimensi 6.
4. Dengan parameter regime pasar, hitung dengan tangan α_t, β_t, $P(O \mid \lambda)$, dan Viterbi untuk O = (Naik, Naik, Datar). Bandingkan dengan *posterior decoding*.
5. Pilih model Bagian I beserta dua alasan: (a) status mesin dari urutan sensor per jam; (b) persetujuan pembiayaan yang penolakannya wajib dijelaskan; (c) 200 ribu pengaduan ke 12 kategori.
6. Bentuk kelompok tugas besar dan kirim calon masalah serta calon data.

Pertemuan depan: UTS dalam bentuk tugas besar; syarat *dataset* dan rubrik lengkap dirinci di sana.

Terima kasih

Pertanyaan?