Pertemuan 2 — *Concept Learning* dan *Version Space*

Belajar sebagai pencarian, dan mengapa tidak ada belajar tanpa bias

Hendri Karisma

Semester Ganjil 2026/2027

Program Studi Teknik Informatika, STMIK Tazkia

Alur pertemuan hari ini

1. Belajar konsep sebagai pencarian di *hypothesis space*
2. Data *EnjoySport* milik Mitchell
3. Representasi hipotesis dan ukuran ruangnya
4. Urutan umum-ke-khusus
5. Algoritma Find-S dan keterbatasannya
6. *Version space* dan batas *S* serta *G*
7. Algoritma Candidate-Elimination
8. Mengklasifikasi *instance* baru
9. *Inductive bias* dan "*no free lunch*"
10. Jejaknya pada model modern

Pertemuan 1 menutup dengan satu kesimpulan: ML adalah ***inductive learning*** — menyimpulkan aturan umum dari contoh yang terbatas, dengan jawaban yang bersifat kemungkinan. Hari ini kita membedah mesin di balik kalimat itu pada bentuknya yang paling jernih, sebelum semuanya menjadi numerik.

Sasaran pertemuan ini

Anda dapat merumuskan tugas belajar sebagai **pencarian di *hypothesis space***, menjalankan Find-S dan Candidate-Elimination dengan tangan, dan menjelaskan mengapa setiap pembelajar wajib punya ***inductive bias***.

Belajar konsep sebagai pencarian

Konsep, dan empat bahan yang harus selalu disebut

Sebuah **konsep** adalah himpunan objek yang memenuhi suatu sifat — burung, transaksi janggal, hari yang cocok untuk berolahraga. Secara formal ia sama dengan sebuah fungsi boolean atas semesta objek. ***Concept learning*** berarti menyimpulkan fungsi itu dari contoh-contoh yang sudah ditandai anggota atau bukan anggota.

- **Ruang *instance*** X — semua objek yang mungkin muncul; satu anggotanya x.
- **Konsep target** c — fungsi sebenarnya, $c : X \to \{0, 1\}$; inilah yang tidak kita ketahui.
- ***Training data*** D — pasangan $\langle x, c(x) \rangle$; **positif** bila $c(x) = 1$, **negatif** bila $c(x) = 0$.
- ***Hypothesis space*** H — semua dugaan yang boleh diajukan, $h : X \to \{0, 1\}$.

Pergeseran cara pandang

Tujuannya: temukan $h \in H$ dengan $h(x) = c(x)$ untuk setiap $x \in X$, bukan hanya untuk x di dalam D.

Begitu H ditetapkan, belajar menjadi urusan **pencarian**. *Decision tree*, SVM, dan *neural network* hanyalah cara lain memilih H dan menyisirnya.

Sumber: Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997, Bab 2.

Tugas belajar kita hari ini: EnjoySport

Aldo menyukai olahraga air tertentu. Kita ingin memprediksi, dari keadaan cuaca suatu hari, apakah ia akan menikmati olahraga itu. Setiap hari dinyatakan dengan enam atribut.

Atribut	**Nilai yang mungkin**	**Atribut**	**Nilai yang mungkin**
Sky	Sunny, Cloudy, Rainy	Wind	Strong, Weak
AirTemp	Warm, Cold	Water	Warm, Cool
Humidity	Normal, High	Forecast	Same, Change

Maka ukuran ruang instance adalah

$$|X| = 3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96 \text{ hari yang berbeda.}$$

Konsep target c adalah atribut ketujuh, `EnjoySport`, yang bernilai Yes atau No.

Empat *training example* — kita pakai ini sampai akhir

No	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	**Yes**
2	Sunny	Warm	High	Strong	Warm	Same	**Yes**
3	Rainy	Cold	High	Strong	Warm	Change	**No**
4	Sunny	Warm	High	Strong	Cool	Change	**Yes**

Empat baris ini adalah seluruh D yang kita miliki: tiga positif dan satu negatif, dari 96 hari yang mungkin. Perhatikan betapa sedikitnya bukti yang ada — dan tetap kita akan memberanikan diri menyimpulkan sesuatu tentang 92 hari sisanya. Itulah induksi.

Sumber: Mitchell (1997), Tabel 2.1.

Hypothesis space dan urutan umum-ke-khusus

Hipotesis sebagai konjungsi batasan atribut

Kita batasi diri pada bentuk yang sangat sederhana: sebuah hipotesis adalah **vektor enam batasan**, satu untuk setiap atribut, yang harus dipenuhi semuanya sekaligus. Setiap batasan boleh berupa satu nilai tertentu (misalnya `Warm`), tanda ? yang berarti **apa saja boleh**, atau tanda $\emptyset$ yang berarti **tidak ada yang cocok**.

$\langle$ Sunny, ?, ?, Strong, ?, ? $\rangle$
"Aldo senang pada hari cerah dan berangin kencang, apa pun sisanya."

$\langle$?, ?, ?, ?, ?, ? $\rangle$
Paling **umum**: setiap hari positif.

$\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$
Paling **khusus**: tidak ada hari yang positif.

Harga dari kesederhanaan

Bahasa ini tidak bisa menyatakan "Sunny **atau** Cloudy". Disjungsi, negasi, dan gabungan atribut di luar jangkauannya. Jika konsep sebenarnya menuntut disjungsi, maka **tidak ada satu pun** anggota H yang benar.

Berapa besar *hypothesis space*-nya

Hitungan sintaktis

Setiap atribut boleh diisi salah satu nilainya, atau ?, atau $\emptyset$. Sky punya $3 + 2 = 5$ pilihan, lima atribut lain masing-masing $2 + 2 = 4$:

$$5 \times 4 \times 4 \times 4 \times 4 \times 4 = \mathbf{5.120}$$

Ini jumlah cara *menuliskan* hipotesis.

Hitungan semantis

Tetapi begitu *satu saja* batasan bernilai $\emptyset$, hipotesisnya menolak semua hari. Semuanya bermakna sama — himpunan kosong — jadi cukup dihitung sekali:

$$1 + (4 \times 3 \times 3 \times 3 \times 3 \times 3) = \mathbf{973}$$

Ini jumlah konsep yang benar-benar *berbeda*.

Ringkasan

Bandingkan 973 dengan jumlah semua konsep yang mungkin atas 96 hari, yaitu $2^{96} \approx 7{,}9 \times 10^{28}$. Dengan memilih bahasa konjungsi, kita baru saja membuang hampir seluruh kemungkinan — dan justru karena itu kita bisa belajar dari empat contoh saja.

Relasi yang menata seluruh *hypothesis space*

Ruang H bukan tumpukan tak berbentuk. Ada satu relasi alami yang menatanya.

more general than or equal to

$h_j \succeq_g h_k$ jika dan hanya jika setiap *instance* yang diterima h_k juga diterima h_j:

$$(\forall x \in X) \left[h_k(x) = 1 \Rightarrow h_j(x) = 1 \right]$$

Notasi: $\forall$ (A terbalik, *for all*): untuk setiap atau semua x, tanpa kecuali; $\in$: anggota dari; $\Rightarrow$: mengakibatkan.

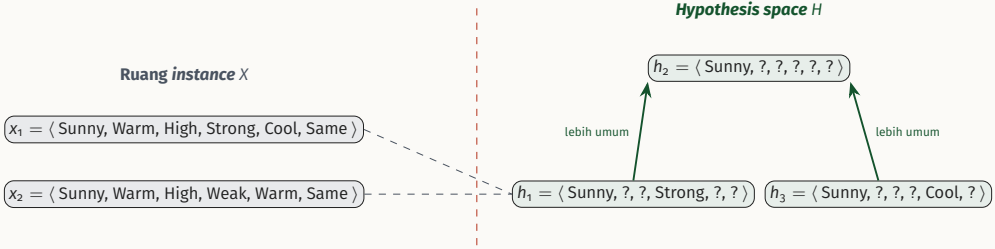
Bila selain itu ada *instance* yang diterima h_j tetapi ditolak h_k, relasinya **lebih umum secara tegas**, ditulis $h_j \succ_g h_k$. Mengganti sebuah nilai menjadi ? selalu membuat hipotesis lebih umum.

Dua sifatnya penting. Relasi ini **tidak bergantung pada data**, hanya pada bentuk hipotesisnya. Dan ia **parsial**, bukan total: ada pasangan hipotesis yang sama sekali tidak dapat dibandingkan.

Ringkasan

Karena relasi ini ada, pencarian di H dapat dilakukan **terarah**: kita tahu arah “ke atas” (menggeneralisasi) dan “ke bawah” (mengkhususkan).

Melihat urutannya



$h_2 \succ_g h_1$ dan $h_2 \succ_g h_3$: keduanya menuntut Sunny, tetapi h_2 tidak menuntut apa pun lagi. Sementara h_1 dan h_3 **tidak dapat dibandingkan**: h_1 menerima x_1 tetapi menolak x_2 karena anginnya lemah, dan h_3 menolak keduanya untuk alasan yang berbeda lagi. Inilah maksud "urutan parsial".

Sumber: Diadaptasi dari Mitchell (1997), Gambar 2.1.

Algoritma Find-S

Find-S: mendaki dari yang paling khusus

Gagasannya sesederhana mungkin: mulai dari hipotesis paling khusus, lalu setiap kali ia gagal menerima sebuah contoh positif, longgarkan seminimal mungkin agar contoh itu tertampung.

Find-S

1. *Initialization* h sebagai hipotesis paling khusus di H: $\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$
2. Untuk setiap **contoh positif** x di dalam D, periksa setiap batasan atribut a_i pada h: jika a_i sudah dipenuhi oleh x, biarkan; jika tidak dipenuhi, ganti a_i dengan batasan paling khusus berikutnya yang dipenuhi x.
3. Keluarkan h.

Perhatikan langkah 2: **contoh negatif tidak pernah dibaca**. Ingat baik-baik, kita akan kembali ke hal ini.

Menjalankan Find-S — dua contoh pertama

Awal. $h_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$
Belum ada apa pun yang diterima.

Contoh 1 ⟨ Sunny, Warm, Normal, Strong, Warm, Same ⟩, positif.
Semua batasan $\emptyset$ gagal, jadi semuanya diganti dengan nilai yang ada pada contoh itu:

$$h_1 = \langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle$$

Pada tahap ini h menerima *tepat satu* hari, yaitu contoh itu sendiri.

Contoh 2 ⟨ Sunny, Warm, High, Strong, Warm, Same ⟩, positif.
Hanya Humidity yang berbeda: h_1 menuntut `Normal`, contohnya `High`. Batasan paling khusus yang menampung keduanya adalah ?:

$$h_2 = \langle \text{Sunny, Warm, ?, Strong, Warm, Same} \rangle$$

Hipotesis melonggar satu langkah, tidak lebih. Inilah arti "*generalization* minimal".

Menjalankan Find-S — dua contoh terakhir

Contoh 3 ⟨ Rainy, Cold, High, Strong, Warm, Change ⟩, negatif. Find-S **melewatinya tanpa melakukan apa pun**, sehingga $h_3 = h_2 = \langle$ Sunny, Warm, ?, Strong, Warm, Same $\rangle$.
Secara kebetulan ini aman: h_2 memang sudah menolak contoh itu, karena `Rainy` $\neq$ `Sunny`. Selama H memuat konsep target dan data tidak mengandung *error*, hal ini dijamin selalu terjadi.

Contoh 4 ⟨ Sunny, Warm, High, Strong, Cool, Change ⟩, positif. Dua atribut tidak dipenuhi — Water dan Forecast — sehingga keduanya dilonggarkan menjadi ?:

$$h_4 = \langle \text{Sunny, Warm, ?, Strong, ?, ?} \rangle$$

Hasil akhir Find-S
⟨ Sunny, Warm, ?, Strong, ?, ? ⟩ — "Aldo menikmati olahraganya pada hari yang cerah, hangat, dan berangin kencang; kelembapan, suhu air, dan ramalan tidak berpengaruh."

h hanya bergerak satu arah, dari khusus ke umum, dan hanya digerakkan oleh contoh positif. Ia mendaki *lattice* tadi, selangkah sekecil mungkin setiap kali.

Tiga hal yang tidak diketahui Find-S

- **Ia tidak tahu apakah sudah selesai.** Find-S memberi satu hipotesis yang konsisten, tetapi tidak ada cara mengetahui apakah itu *satu-satunya*, atau hanya satu dari puluhan. Untuk EnjoySport, sebentar lagi kita lihat jawabannya ada **enam**.
- **Ia tidak tahu kalau datanya bermasalah.** Karena contoh negatif tidak pernah dibaca, satu *label* yang salah atau data yang saling bertentangan lewat tanpa terdeteksi. Find-S tetap mengeluarkan sebuah hipotesis, dengan percaya diri yang sama.
- **Ia memilih tanpa alasan yang diuji.** Mengapa yang paling khusus, bukan yang paling umum? Pilihan itu keputusan desain yang tidak diturunkan dari data, dan keputusan semacam inilah yang nanti kita namai *bias*.

Ringkasan

Kelemahannya semua bermuara pada satu hal: Find-S memberi **satu jawaban** padahal bukti yang ada hanya cukup untuk menyempitkan ke **sehimpunan jawaban**. Perbaikannya: jangan pilih satu, simpan semuanya.

Version space dan Candidate-Elimination

Konsisten, dan *version space*

Konsisten

Hipotesis h **konsisten** dengan *training data* D jika $h(x) = c(x)$ untuk **setiap** contoh $\langle x, c(x) \rangle$ di dalam D.

Version space

Version space $VS_{H,D}$ adalah himpunan **semua** hipotesis di dalam H yang konsisten dengan D:

$$VS_{H,D} = \{ h \in H \mid h \text{ konsisten dengan } D \}$$

Namanya berasal dari gagasan bahwa himpunan ini memuat semua "versi" konsep target yang masih mungkin, mengingat bukti yang sudah terlihat. Setiap contoh baru dapat menyusutkannya, tetapi tidak pernah memperbesarnya.

Cara paling naif untuk mendapatkannya — *List-Then-Eliminate* — adalah menuliskan seluruh H lalu membuang setiap hipotesis yang salah pada suatu contoh. Untuk EnjoySport itu berarti memeriksa 973 hipotesis; untuk masalah nyata, angkanya langsung mustahil.

Menyimpan himpunan besar dengan dua batas saja

Kuncinya: *version space* tertata oleh relasi $\succeq_g$ tadi. Himpunan yang tertata seperti itu dapat dijelaskan hanya dengan **tepi atas** dan **tepi bawahnya**.

Batas khusus S

Anggota *version space* yang **paling khusus**: tidak ada anggota lain yang lebih khusus darinya.

Batas umum G

Anggota *version space* yang **paling umum**: tidak ada anggota lain yang lebih umum darinya.

Teorema representasi *version space*

Sebuah hipotesis h ada di dalam $VS_{H,D}$ jika dan hanya jika ia terletak **di antara** kedua batas itu:

$$VS_{H,D} = \{\, h \in H \mid \exists s \in S,\ \exists g \in G :\ g \succeq_g h \succeq_g s \,\}$$

Notasi: $\{h \in H \mid \ldots\}$: semua h di H yang memenuhi syarat; $\exists$ (*there exists*): ada paling sedikit satu.

Artinya kita tidak perlu menyimpan daftar hipotesisnya. Cukup simpan S dan G, dan seluruh isi *version space* dapat dibangkitkan kapan pun dibutuhkan.

Candidate-Elimination

Initialization

$G \leftarrow \{\langle ?, ?, ?, ?, ?, ? \rangle\}$ $\quad$ $S \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$

Jika contoh d positif

- Buang dari G setiap g yang tidak konsisten dengan d.
- Untuk setiap $s \in S$ yang tidak konsisten dengan d: buang s, lalu masukkan semua ***generalization* minimal** h dari s yang konsisten dengan d dan masih lebih khusus dari suatu anggota G.
- Buang dari S setiap hipotesis yang lebih umum daripada anggota S yang lain.

Jika contoh d negatif

- Buang dari S setiap s yang tidak konsisten dengan d.
- Untuk setiap $g \in G$ yang tidak konsisten dengan d: buang g, lalu masukkan semua **spesialisasi minimal** h dari g yang konsisten dengan d dan masih lebih umum dari suatu anggota S.
- Buang dari G setiap hipotesis yang lebih khusus daripada anggota G yang lain.

Perhatikan simetrinya: contoh positif **mendorong** S **naik**, contoh negatif **menarik** G **turun**. Keduanya bergerak saling mendekat.

Jalankan — contoh 1 dan 2, keduanya positif

Awal. $S_0 = \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$ $\quad G_0 = \{\langle ?, ?, ?, ?, ?, ? \rangle\}$

Contoh 1 $\langle$ Sunny, Warm, Normal, Strong, Warm, Same $\rangle$ +
G_0 sudah menerima contoh itu, jadi G tidak berubah. S_0 menolaknya, jadi ia digeneralisasi minimal:

$$S_1 = \{\langle \text{Sunny, Warm, Normal, Strong, Warm, Same} \rangle\} \qquad G_1 = G_0$$

Contoh 2 $\langle$ Sunny, Warm, High, Strong, Warm, Same $\rangle$ +
Kembali hanya Humidity yang bentrok, sehingga dilonggarkan menjadi ?:

$$S_2 = \{\langle \text{Sunny, Warm, ?, Strong, Warm, Same} \rangle\} \qquad G_2 = G_0$$

Sampai di sini Candidate-Elimination mengerjakan hal yang sama dengan Find-S pada sisi S, sementara G belum tersentuh — belum ada contoh negatif yang memaksanya menyempit.

Jalankan — contoh 3, negatif: di sini G mulai bekerja

Contoh 3 $\langle$ Rainy, Cold, High, Strong, Warm, Change $\rangle$ —

S_2 sudah menolak contoh ini, jadi $S_3 = S_2$. Tetapi G_2 *menerimanya*, padahal seharusnya menolak. Maka G dikhususkan seminimal mungkin. Ada enam cara — satu untuk setiap atribut — tetapi hanya yang **masih lebih umum daripada** S_3 yang boleh bertahan:

Kandidat spesialisasi	**Menolak contoh 3?**	**Lebih umum dari S_3?**
$\langle$ Sunny, ?, ?, ?, ?, ? $\rangle$	ya	ya $\Rightarrow$ **disimpan**
$\langle$?, Warm, ?, ?, ?, ? $\rangle$	ya	ya $\Rightarrow$ **disimpan**
$\langle$?, ?, ?, ?, ?, Same $\rangle$	ya	ya $\Rightarrow$ **disimpan**
$\langle$?, ?, Normal, ?, ?, ? $\rangle$	ya	tidak (S_3 memakai ?) $\Rightarrow$ dibuang
$\langle$?, ?, ?, Weak, ?, ? $\rangle$	ya	tidak (S_3 menuntut Strong) $\Rightarrow$ dibuang
$\langle$?, ?, ?, ?, Cool, ? $\rangle$	ya	tidak (S_3 menuntut Warm) $\Rightarrow$ dibuang

$$G_3 = \{\langle \text{Sunny}, ?, ?, ?, ?, ? \rangle,\ \langle ?, \text{Warm}, ?, ?, ?, ? \rangle,\ \langle ?, ?, ?, ?, ?, \text{Same} \rangle\}$$

Jalankan — contoh 4, positif: S naik, G menyusut

Contoh 4 $\langle$ Sunny, Warm, High, Strong, Cool, Change $\rangle$ +

Sisi S**.** S_3 menuntut Water = Warm dan Forecast = Same, keduanya bentrok, jadi dilonggarkan menjadi ?:

$$S_4 = \{\langle \text{Sunny, Warm, ?, Strong, ?, ?} \rangle\}$$

Sama persis dengan hasil akhir Find-S. Itu bukan kebetulan: sisi S dari Candidate-Elimination *adalah* Find-S.

Sisi G**.** Anggota G yang menolak contoh positif harus dibuang.

- $\langle$ Sunny, ?, ?, ?, ?, ? $\rangle$ — menerima contoh 4, **disimpan**.
- $\langle$?, Warm, ?, ?, ?, ? $\rangle$ — menerima contoh 4, **disimpan**.
- $\langle$?, ?, ?, ?, ?, Same $\rangle$ — contoh 4 punya Forecast = Change, jadi hipotesis ini *menolak* sebuah contoh positif. Dibuang.

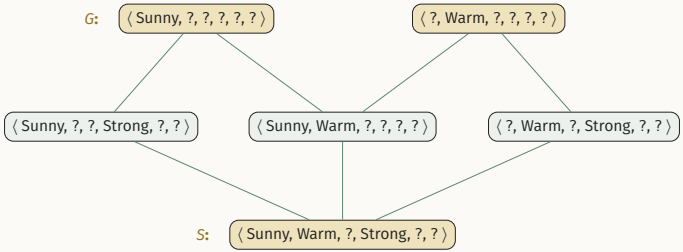
$$G_4 = \{\langle \text{Sunny, ?, ?, ?, ?, ?} \rangle,\ \langle \text{?, Warm, ?, ?, ?, ?} \rangle\}$$

Ringkasan seluruh jalannya

Setelah	**Batas khusus** S	**Batas umum** G
—	$\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$	$\langle ?, ?, ?, ?, ?, ? \rangle$
contoh 1 +	$\langle$ Sunny, Warm, Normal, Strong, Warm, Same $\rangle$	$\langle ?, ?, ?, ?, ?, ? \rangle$
contoh 2 +	$\langle$ Sunny, Warm, ?, Strong, Warm, Same $\rangle$	$\langle ?, ?, ?, ?, ?, ? \rangle$
contoh 3 —	$\langle$ Sunny, Warm, ?, Strong, Warm, Same $\rangle$	$\langle$ Sunny, ?, ?, ?, ?, ? $\rangle$ $\langle ?,$ Warm, ?, ?, ?, ? $\rangle$ $\langle ?, ?, ?, ?, ?,$ Same $\rangle$
contoh 4 +	$\langle$ Sunny, Warm, ?, Strong, ?, ? $\rangle$	$\langle$ Sunny, ?, ?, ?, ?, ? $\rangle$ $\langle ?,$ Warm, ?, ?, ?, ? $\rangle$

Kedua batas bergerak saling mendekat. Bila S dan G menjadi sama dan berisi satu hipotesis, konsep target sudah teridentifikasi sepenuhnya. Bila salah satu menjadi kosong, itu tanda tidak ada hipotesis di H yang konsisten — entah karena data mengandung *error*, atau karena H terlalu sempit.

Version space akhir: enam hipotesis



Garis ke atas berarti “lebih umum”. Keenam hipotesis ini **semuanya** konsisten dengan keempat *training example* — tidak ada satu pun cara, dari data yang ada, untuk memilih di antara mereka. Find-S hanya melaporkan yang paling bawah dan diam soal lima lainnya.

Sumber: Diadaptasi dari Mitchell (1997), Gambar 2.3.

Mengklasifikasi *instance* baru: bertanya kepada enam hipotesis

Ajukan *instance* baru kepada **setiap** anggota *version space*, lalu lihat apakah jawabannya bulat.

	***Instance* baru**	**Positif**	**Negatif**	**Kesimpulan**
A	⟨ Sunny, Warm, Normal, Strong, Cool, Change ⟩	6	0	**Yes**, suara bulat
B	⟨ Rainy, Cold, Normal, Weak, Warm, Same ⟩	0	6	**No**, suara bulat
C	⟨ Sunny, Warm, Normal, Weak, Warm, Same ⟩	3	3	terbagi rata, tidak dapat disimpulkan
D	⟨ Sunny, Cold, Normal, Strong, Warm, Same ⟩	2	4	terbagi, cenderung No

- A diterima oleh anggota *S*; karena semua anggota lain lebih umum, mereka pasti juga menerimanya — **cukup periksa** *S*.
- B ditolak oleh setiap anggota *G*; karena semua anggota lain lebih khusus, mereka pasti juga menolaknya — **cukup periksa** *G*.
- C dan D jatuh di antara kedua batas. Di sinilah *version space* berkata jujur: bukti yang ada belum cukup. *Instance* seperti C, yang membelah *version space* tepat di tengah, adalah contoh yang paling berharga untuk dilabeli berikutnya — inilah dasar *active learning*.

Sumber: Mitchell (1997), Tabel 2.6. Nilai Wind ditulis `Weak` agar selaras dengan tabel atribut kita.

Inductive bias dan jejaknya pada model modern

Apa yang sebenarnya kita lakukan ketika menggeneralisasi

Perhatikan bahwa dari empat contoh, kita menyimpulkan sesuatu tentang 96 hari. Dari mana keberanian itu datang? Bukan dari data — data hanya berbicara tentang empat hari.

Inductive bias

Himpunan asumsi tambahan yang, bila digabungkan dengan *training data*, membuat kesimpulan sebuah pembelajar menjadi **deduktif** — yaitu mengikuti secara logis.

Untuk Candidate-Elimination pada EnjoySport, asumsi itu dapat dituliskan dalam satu kalimat:

"Konsep target c berada di dalam H, yaitu dapat dinyatakan sebagai konjungsi batasan atribut."

Tambahkan kalimat itu ke dalam data, dan setiap *classification* yang dihasilkan Candidate-Elimination mengikuti secara logis. Cabut kalimat itu, dan tidak ada apa pun yang mengikuti.

Pembelajar tanpa bias: kedengarannya ideal, nyatanya lumpuh

Sekarang biasnya dihapus. Ambil H sebagai **seluruh** himpunan konsep yang mungkin atas 96 hari — semua 2^{96} himpunan bagian, dengan disjungsi dan negasi diizinkan. Sekarang H pasti memuat c, dan tidak ada asumsi yang bisa salah.

Jalankan Candidate-Elimination pada H ini dengan data kita:

$$S = \{ x_1 \vee x_2 \vee x_4 \} \qquad G = \{ \neg x_3 \}$$

S menjadi disjungsi tepat dari contoh positif yang terlihat; G menjadi negasi dari contoh negatif yang terlihat.

Sekarang klasifikasikan sebuah hari baru x yang belum pernah terlihat. *Version space* ini memuat hipotesis yang mengatakan x positif *dan* hipotesis yang mengatakan x negatif, dalam jumlah yang persis sama. Suaranya selalu terbagi rata, selamanya.

Ringkasan

Pembelajar tanpa bias tidak pernah salah, dan justru karena itu tidak pernah berguna: ia hanya bisa mengulang *label* yang sudah ia lihat. **Tanpa bias tidak ada *generalization*.**

Tiga pembelajar, tiga kekuatan bias

Pembelajar	*Inductive bias*-nya	Akibatnya
Rote learner simpan data, jawab hanya bila pernah dilihat	Tidak ada sama sekali.	Tidak pernah salah, tidak pernah menggeneralisasi.
Candidate-Elimination	Konsep target dapat dinyatakan sebagai konjungsi batasan atribut ($c \in H$).	Menggeneralisasi, tetapi hanya menjawab bila seluruh *version space* sepakat.
Find-S	Bias Candidate-Elimination *ditambah*: jika ragu pilih hipotesis paling khusus, yaitu anggap negatif kecuali terbukti sebaliknya.	Selalu memberi satu jawaban, termasuk pada kasus yang belum ditentukan data.

Urutannya jelas: **rote learner** $<$ **Candidate-Elimination** $<$ **Find-S**, dari yang tak berbias sampai yang paling berbias. Bias yang lebih kuat berarti kesediaan menyimpulkan lebih jauh dari bukti yang ada — itulah harga masuk untuk bisa menggeneralisasi, sekaligus lebih banyak kesempatan untuk salah.

No free lunch, secara intuitif

Ada teorema yang menyatakan hal ini secara formal, dikenal sebagai *no free lunch theorem*. Intuisinya mudah ditangkap dari apa yang baru saja kita lihat.

Sebuah algoritma pembelajar bekerja baik pada suatu masalah karena biasnya **cocok** dengan struktur masalah itu. Bias yang sama akan menyesatkan pada masalah lain yang strukturnya berbeda. Dirata-ratakan atas *semua* masalah yang mungkin — termasuk yang polanya acak dan tidak ada pola untuk ditemukan — setiap pembelajar sama baiknya, yaitu sama-sama tidak lebih baik daripada menebak.

Yang menyelamatkan kita: masalah nyata bukan diambil acak dari semua kemungkinan. Dunia punya keteraturan, dan bias yang berguna adalah bias yang menebaknya dengan benar.

Ringkasan

Tidak ada algoritma yang terbaik untuk segala hal. Karena itu pertanyaan "model apa yang paling bagus?" selalu salah bentuk. Yang benar: **asumsi apa yang masuk akal untuk masalah ini**, lalu model mana yang asumsinya cocok.

Setiap model punya *inductive bias* — kenali dulu, baru pakai

Model	*Inductive bias*-nya	Gagal ketika
Decision tree (ID3, C4.5)	Batas keputusan berupa potongan sejajar sumbu; pohon lebih pendek lebih disukai	Batas sebenarnya berupa kombinasi miring antar *feature*
kNN	Titik yang berdekatan punya *label* sama; semua *feature* sama pentingnya	Jarak tak bermakna pada dimensi tinggi, atau skala *feature* berbeda-beda
Linear regression	Hubungan *feature* dan target bersifat aditif dan linier	Ada interaksi atau kelengkungan yang penting
CNN	Pola bersifat lokal dan berarti sama di mana pun letaknya	Posisi absolut justru penting
Transformer	Bias arsitekturnya sangat lemah; keterkaitan dipelajari dari data	*Training data* sedikit

Perhatikan baris terakhir. Alasan *transformer* membutuhkan data raksasa adalah justru karena biasnya lemah: apa yang tidak diasumsikan arsitektur harus dipelajari dari contoh.

Apa yang perlu dibawa dari pertemuan ini

- **Belajar itu mencari.** Tetapkan X, H, dan D, maka tugas belajar berubah menjadi pencarian yang dapat dianalisis. Rumusan ini tidak berubah sampai akhir semester.
- **Struktur pada H membuat pencarian mungkin.** Urutan umum-ke-khusus adalah contoh pertamanya; pada model numerik nanti penggantinya adalah *smoothness* dan *gradient*.
- **Satu jawaban menyembunyikan ketidakpastian.** *Version space* memperlihatkan seluruh kemungkinan yang masih hidup — pelajaran yang terus dipakai oleh *ensemble* dan metode Bayesian.
- **Tidak ada belajar tanpa bias.** Setiap kali Anda memilih model, Anda sedang memilih asumsi tentang dunia. Sebaiknya itu pilihan yang sadar.

Pertemuan depan

Kita naik ke H yang jauh lebih kaya: ***decision tree***. Pencariannya *greedy* dengan *entropy* sebagai penuntun, dan biasnya bergeser menjadi “pohon yang lebih pendek lebih baik”.

Tugas untuk pertemuan berikutnya

1. **Baca** Mitchell Bab 2 secara utuh, lalu Bab 3 bagian awal sebagai persiapan.
2. **Latihan tangan.** Ambil keempat contoh EnjoySport di kelas, ubah *label* contoh ke-3 menjadi **positif**, lalu
   - jalankan Find-S langkah demi langkah dan tuliskan h setelah setiap contoh;
   - jalankan Candidate-Elimination dan tuliskan S serta G setelah setiap contoh;
   - laporkan apa yang terjadi pada *version space*, dan jelaskan mengapa.
3. **Implementasi.** Tulis program Python sederhana — cukup `list` dan `for`, tanpa *library* ML — yang
   - membaca *training data* dari sebuah *file* CSV berisi tabel EnjoySport;
   - menjalankan Find-S dan mencetak h setelah setiap contoh positif;
   - menerima satu hari baru dan mencetak Yes atau No menurut h akhir.

   Kumpulkan kode beserta *output*-nya. Nilai tambah bagi yang juga mengerjakan Candidate-Elimination.

Terima kasih

Pertanyaan?