THE FIND-S ALGORITHM

Finding a Maximally Specific Hypothesis

Source: Tom M. Mitchell, Machine Learning (1997), Chapter 2, Section 2.4 — FIND-S: Finding a Maximally Specific Hypothesis

Prepared by Dr. S. S. Roy | Date: 15 July 2026

1. The Big Idea in One Minute

We know from the previous lecture that learning = searching the hypothesis space H. But H can be huge. How do we search it **without checking every hypothesis one by one?**

Mitchell's answer: use the *more_general_than* partial ordering as a ladder. **FIND-S** starts at the very bottom of the ladder (the most specific hypothesis possible) and climbs upward — generalizing — only when a positive training example forces it to. It never generalizes more than necessary, and it completely ignores negative examples.

> **FIND-S in one sentence:** Begin with the most specific hypothesis in H, and each time it fails to cover an observed positive example, generalize it just enough ("minimally") to cover that example. (We say a hypothesis "covers" a positive example if it correctly classifies the example as positive.)

2. The Algorithm (Mitchell, Table 2.3)

The algorithm exactly as given in the book (Table 2.3), written as formal pseudocode:

```
FIND-S Algorithm
1.  Initialize h to the most specific hypothesis in H
2.  For each positive training instance x
        For each attribute constraint a_i in h
            If the constraint a_i is satisfied by x
            Then do nothing
            Else replace a_i in h by the next more general
                 constraint that is satisfied by x
3.  Output hypothesis h
```

The same three steps explained in a table:

Step	What FIND-S does
1	Initialize h to the **most specific** hypothesis in H: $h \leftarrow \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$
2	For each **positive** training instance x: for each attribute constraint a_i in h — if a_i is satisfied by x, do nothing; else replace a_i in h by the **next more general constraint** that is satisfied by x.
3	Output hypothesis h.

"Next more general constraint" in this representation means exactly two possible moves per slot:

- **Ø → the specific value seen in the example** (e.g., Ø becomes Sunny). This is the smallest possible generalization from Ø.
- **a specific value → ?** (e.g., Normal becomes ?). If the stored value disagrees with the example's value, the only constraint general enough to accept both is "?".

Note two things students always miss: (1) FIND-S looks only at POSITIVE examples — negative examples are simply skipped. (2) Once a slot becomes "?", it can never go back; generalization is one-way, upward.

3. The Training Data (Mitchell, Table 2.1)

Example	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**

4. Full Trace of FIND-S on Table 2.1 (Every Step Explained)

Step 0 — Initialization

$$h_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$$

This hypothesis says "no day is positive." It is the most specific member of H — the bottom of the ladder.

Step 1 — Example 1: ⟨Sunny, Warm, Normal, Strong, Warm, Same⟩, Yes (positive)

Is Example 1 covered by h_0? No — none of the Ø constraints is satisfied by this example. So each Ø is replaced by the next more general constraint that fits the example: the attribute value itself.

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

As Mitchell says, h is still very specific: it asserts that ALL days are negative except the one exact day we observed.

Step 2 — Example 2: ⟨Sunny, Warm, High, Strong, Warm, Same⟩, Yes (positive)

Compare slot by slot with h_1. Sky: Sunny = Sunny ✓keep. AirTemp: Warm = Warm ✓keep. **Humidity: h says Normal, example says High ✗— disagreement → replace with "?".** Wind, Water, Forecast: all agree ✓keep.

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

Step 3 — Example 3: ⟨Rainy, Cold, High, Strong, Warm, Change⟩, No (negative)

FIND-S IGNORES it. $h_3 = h_2$, unchanged.

Why is ignoring negative examples justified? Mitchell's argument: we assume the target concept c is in H and is consistent with the positive training examples. Then c must be more_general_than_or_equal_to the current h (h is the most specific hypothesis fitting the positives). Since c never covers a negative example, and h is contained within c, **h can never cover a negative example either** — so no revision is ever needed in response to a negative example. Verify it here: h_2 requires Sky = Sunny, but Example 3 has Sky = Rainy, so h_2 already correctly classifies Example 3 as negative.

Step 4 — Example 4: ⟨Sunny, Warm, High, Strong, Cool, Change⟩, Yes (positive)

Compare with h_2 slot by slot. Sky: Sunny ✓. AirTemp: Warm ✓. Humidity: already "?" ✓. Wind: Strong ✓. **Water: h says Warm, example says Cool ✗→ "?". Forecast: h says Same, example says Change ✗→ "?".**

h_4 = ⟨Sunny, Warm, ?, Strong, ?, ?⟩ ← FINAL OUTPUT

Read it as a sentence: "Aldo enjoys his sport on sunny, warm, strong-wind days — humidity, water temperature and forecast do not matter." This is exactly the final hypothesis in Mitchell's trace.

The whole trace in one summary table

After seeing…	Type	Hypothesis h	What changed
(start)	—	⟨∅, ∅, ∅, ∅, ∅, ∅⟩	Initialized to most specific
Example 1	+	⟨Sunny, Warm, Normal, Strong, Warm, Same⟩	All six ∅ → the observed values
Example 2	+	⟨Sunny, Warm, ?, Strong, Warm, Same⟩	Humidity → ?
Example 3	−	⟨Sunny, Warm, ?, Strong, Warm, Same⟩	Nothing — negative examples ignored
Example 4	+	⟨Sunny, Warm, ?, Strong, ?, ?⟩	Water → ?, Forecast → ?

Picture in your head (Mitchell's Figure 2.2): the search moves from hypothesis to hypothesis, from the most specific end toward more general hypotheses, along ONE chain of the partial ordering. At each step h is generalized only as far as necessary to cover the new positive example. Hence, at every stage, h is the MOST SPECIFIC hypothesis consistent with the training examples seen so far — that is why the algorithm is called FIND-S (S = specific).

5. What FIND-S Guarantees — and Its Four Honest Problems

Guarantee (Mitchell): for hypothesis spaces described by conjunctions of attribute constraints (like EnjoySport's H), FIND-S is guaranteed to output the most specific hypothesis in H consistent with the positive training examples. Its final hypothesis will also be consistent with the negative examples

PROVIDED (i) the correct target concept is contained in H, and (ii) the training examples are correct (no noise). Both provisos matter — drop either one and the guarantee collapses.

Mitchell then lists the questions FIND-S leaves unanswered. These are the standard exam questions on this topic:

- **1. Has the learner converged to the correct target concept?** FIND-S finds A hypothesis consistent with the data, but it cannot tell whether it is the ONLY consistent one in H, or one among many. It cannot even characterize its own uncertainty.
- **2. Why prefer the most specific hypothesis?** If several hypotheses are consistent with the data, it is unclear why we should prefer the most specific one over the most general one, or something in between.
- **3. Are the training examples consistent (noise-free)?** Real data contains errors. Noisy examples can severely mislead FIND-S, precisely because it ignores negative examples — it has no way to even detect an inconsistency.
- **4. What if there are several maximally specific consistent hypotheses?** In EnjoySport's H there is always a unique most specific consistent hypothesis, but for other hypothesis spaces there can be several — and FIND-S has no policy for that situation.

These four weaknesses are exactly the motivation for the Candidate-Elimination algorithm, which comes next in the chapter.

6. Worked Numerical Exercise (Fully Solved, Step by Step)

Problem. A college wants to learn the concept "Student gets placed" (Yes/No). Attributes: CGPA (High, Low), Communication (Good, Poor), Internship (Yes, No), Backlogs (None, Some). Run FIND-S on the following four training examples, showing the hypothesis after every example.

Example	CGPA	Communication	Internship	Backlogs	Placed
1	High	Good	Yes	None	**Yes**
2	High	Good	No	None	**Yes**
3	Low	Poor	No	Some	**No**
4	High	Good	No	Some	**Yes**

Solution

Step 0. $h_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle$ (most specific; classifies every student as not placed).

Step 1 — Example 1 (positive). h_0 covers nothing, so every $\emptyset$ is replaced by the example's value: $h_1 = \langle$ High, Good, Yes, None$\rangle$.

Step 2 — Example 2 (positive). Compare slot by slot: CGPA High = High ✓; Communication Good = Good ✓; Internship: h says Yes, example says No ✗→ "?"; Backlogs None = None ✓. So $h_2 = \langle$ High, Good, ?, None$\rangle$.

Step 3 — Example 3 (negative). Ignored. $h_3 = h_2 = \langle$ High, Good, ?, None$\rangle$. (Check: h_2 requires CGPA = High; Example 3 has Low, so it is already correctly classified negative.)

Step 4 — Example 4 (positive). CGPA High ✓, Communication Good ✓, Internship already "?" ✓, Backlogs: h says None, example says Some ✗→ "?".

FINAL: $h_4 = \langle$ High, Good, ?, ?$\rangle$

In words: "A student is placed if CGPA is High and Communication is Good — internship and backlogs do not matter (according to this data)." Note honestly: this rule is only the most specific hypothesis consistent with these four examples; it is not guaranteed to be the true concept.

7. Practice Exercises (Keys at the End)

Exercise 1

Run FIND-S on the following data for the concept "Go for a picnic" (Yes/No). Attributes: Weather (Sunny, Rainy, Cloudy), Temperature (Hot, Mild, Cold), Holiday (Yes, No). Show the hypothesis after each example, and state the final hypothesis in plain words.

Example	Weather	Temperature	Holiday	Picnic
1	Sunny	Hot	Yes	**Yes**
2	Rainy	Cold	No	**No**
3	Sunny	Mild	Yes	**Yes**
4	Cloudy	Cold	Yes	**No**

Exercise 2

Using the EnjoySport data of Table 2.1 above:

- (a) Suppose the examples arrive in the order 4, 2, 1, 3 instead of 1, 2, 3, 4. Trace FIND-S and give the final hypothesis. Is it the same as before?
- (b) Suppose Example 3 had been wrongly recorded as POSITIVE (a noisy label). Trace FIND-S on the order 1, 2, 3, 4 and give the final hypothesis. In one sentence, state what this shows about FIND-S and noise.
- (c) True or False, with one-line justification: "During a run of FIND-S, the hypothesis can become more specific after seeing a new example."

8. Answer Keys

Exercise 1 — Key

- **Start:** $h_0 = \langle \emptyset, \emptyset, \emptyset \rangle$.
- **Example 1 (+):** $h_1 = \langle$ Sunny, Hot, Yes$\rangle$.
- **Example 2 (–):** ignored; $h_2 = \langle$ Sunny, Hot, Yes$\rangle$.

- **Example 3 (+):** Weather Sunny ✓; Temperature Hot vs Mild ✗→ ?; Holiday Yes ✓. h_3 = ⟨ Sunny, ?, Yes⟩ .
- **Example 4 (–): ignored; FINAL h = ⟨ Sunny, ?, Yes⟩ .**
- **Plain words:** "Go for a picnic on a sunny holiday, whatever the temperature." (Check against the negatives: Example 2 fails Weather; Example 4 fails Weather. Consistent.)

Exercise 2 — Key

- **(a)** Order 4, 2, 1, 3: after Example 4 (+): ⟨ Sunny, Warm, High, Strong, Cool, Change⟩ . After Example 2 (+): Humidity High ✓stays; Water Warm vs Cool ✗→ ?; Forecast Same vs Change ✗→ ?: ⟨ Sunny, Warm, High, Strong, ?, ?⟩ . After Example 1 (+): Humidity Normal vs High ✗→ ?: ⟨ Sunny, Warm, ?, Strong, ?, ?⟩ . Example 3 (−): ignored. **FINAL: ⟨ Sunny, Warm, ?, Strong, ?, ?⟩ — the SAME hypothesis as before.** For this representation the final result is the same for any order of the same examples, because the output is determined by the set of positive examples (it is their most specific common generalization), not by their order.
- **(b)** Treating Example 3 ⟨ Rainy, Cold, High, Strong, Warm, Change⟩ as positive: after Examples 1, 2 we have ⟨ Sunny, Warm, ?, Strong, Warm, Same⟩ . "Example 3" now forces: Sky → ?, AirTemp → ?, Forecast → ?: ⟨ ?, ?, ?, Strong, Warm, ?⟩ . After Example 4 (+): Water Warm vs Cool ✗→ ?: **FINAL: ⟨ ?, ?, ?, Strong, ?, ?⟩** — "any strong-wind day." One wrong label wrecked the rule (it now wrongly accepts the true negative day), and FIND-S cannot even detect that anything went wrong. This is exactly Mitchell's noise objection.
- **(c)** FALSE. FIND-S only ever generalizes: each slot moves one-way, Ø → value → "?", and negative examples cause no change at all, so h can never become more specific during a run.

9. Five Quick MCQs (Answers Below)

Q1. FIND-S initializes h to: (A) ⟨ ?, ?, ?, ?, ?, ?⟩ (B) ⟨ Ø, Ø, Ø, Ø, Ø, Ø⟩ (C) a random hypothesis (D) the first training example.

Q2. On a NEGATIVE training example, FIND-S: (A) specializes h (B) generalizes h (C) ignores the example (D) restarts.

Q3. On Table 2.1, the final FIND-S hypothesis is: (A) ⟨ Sunny, Warm, ?, Strong, ?, ?⟩ (B) ⟨ Sunny, Warm, ?, ?, ?, ?⟩ (C) ⟨ Sunny, ?, ?, Strong, ?, ?⟩ (D) ⟨ ?, ?, ?, ?, ?, ?⟩ .

Q4. FIND-S's output is guaranteed consistent with the negative examples ONLY IF: (A) the data is large (B) c is in H and the training examples are correct (C) h is initialized to all "?" (D) always, unconditionally.

Q5. The "S" in FIND-S stands for the fact that at every stage h is the most: (A) simple (B) stable (C) specific consistent hypothesis so far (D) sensitive.

MCQ Keys: Q1: B. Q2: C. Q3: A. Q4: B — both provisos are stated by Mitchell. Q5: C.

10. One-Paragraph Summary to Close the Lecture

FIND-S searches the hypothesis space by climbing one chain of the general-to-specific ordering: start at ⟨ Ø,…,Ø⟩ , and for each positive example generalize each violated constraint minimally (Ø → value,

value → ?), ignoring every negative example. On Mitchell's Table 2.1 the trace is ⟨ Ø,Ø,Ø,Ø,Ø,Ø⟩ → ⟨ Sunny, Warm, Normal, Strong, Warm, Same⟩ → ⟨ Sunny, Warm, ?, Strong, Warm, Same⟩ → (unchanged) → ⟨ Sunny, Warm, ?, Strong, ?, ?⟩ . It is guaranteed to output the most specific hypothesis consistent with the positive examples, and it is safe on negatives only if the target concept is in H and the data is noise-free. Its four weaknesses — no convergence test, no reason to prefer the most specific hypothesis, no tolerance or detection of noise, and no policy when several maximally specific hypotheses exist — motivate the Candidate-Elimination algorithm.