Candidate Elimination Algorithm

A Complete Worked Example on the EnjoySport Concept

Introduction

The Candidate Elimination Algorithm (CEA) keeps track of **all hypotheses consistent with the training data**, not just one, by maintaining two boundaries: **S (the most specific boundary)** and **G (the most general boundary)**.

Positive examples push S to become more general, and negative examples push G to become more specific, until the two boundaries close in on the **version space** — the set of every rule that fits the data.

The algorithm helps by showing every hypothesis still consistent with the data — so you know exactly how much uncertainty remains, and which next example would best narrow it down.

The Training Data (EnjoySport)

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

Note: Example 3 is the only **negative** example (EnjoySport = No). All others are positive.

Initialization

The boundary sets are first initialized to the most specific and most general hypotheses possible:

$S_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$ (most specific boundary)

$G_0 = \langle ?, ?, ?, ?, ?, ? \rangle$ (most general boundary)

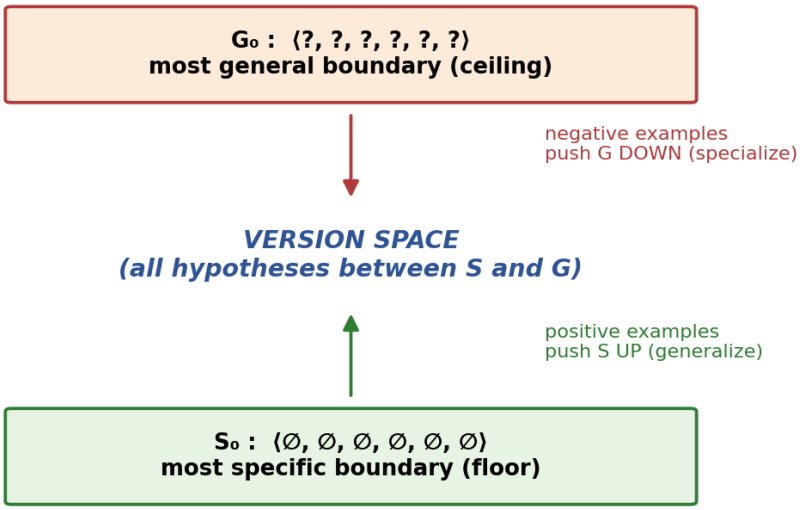


Figure 1: The two boundaries S and G, and the version space between them. Positive examples push S up; negative examples push G down.

The Algorithm

Initialize: S ← most specific hypothesis ⟨ ∅, …, ∅⟩ ; G ← most general hypothesis ⟨ ?, …, ?⟩

For each training example d:

If d is positive:

- Delete from G any rule inconsistent with d.
- Generalize S just enough to stay consistent with d.

If d is negative:

- Delete from S any rule inconsistent with d.
- Specialize G just enough to stay consistent with d.

Output: all rules consistent with the data (everything between S and G). If S = G, the exact concept has been found.

Remember: positives push S up (more general); negatives push G down (more specific).

Tracing the Algorithm Step by Step

Training Example 1 (Positive): ⟨Sunny, Warm, Normal, Strong, Warm, Same⟩ → Yes

When the first training example is presented, the algorithm checks the S boundary and finds that it is overly specific — it fails to cover this positive example. The boundary is therefore revised by moving it to the **least more general hypothesis** that covers the new example. There is no update to the G boundary, since G_0 already covers this example.

$S_1 = \{ \langle Sunny, Warm, Normal, Strong, Warm, Same \rangle \}$

$G_1 = G_0 = \langle ?, ?, ?, ?, ?, ? \rangle$

Training Example 2 (Positive): ⟨Sunny, Warm, High, Strong, Warm, Same⟩ → Yes

The second training example (also positive) has a similar effect of generalizing S further to S_2. The Humidity values differ (Normal vs. High), so that position is generalized to "?". G remains unchanged: $G_2 = G_1 = G_0$.

$S_2 = \{ \langle Sunny, Warm, ?, Strong, Warm, Same \rangle \}$

$G_2 = G_1 = G_0 = \langle ?, ?, ?, ?, ?, ? \rangle$

Note: up to this point, the algorithm behaves exactly the same as FIND-S.

Important Observations Before Example 3

• Positive examples may force the S boundary of the version space to become increasingly general.

• Negative training examples play the complementary role of forcing the G boundary to become increasingly specific.

• The third example is negative, and it shows that G is currently too general.

• Being too general, G wrongly labels (incorrectly predicts) the new example as positive.

• So G must be tightened until it calls the example negative. Therefore, we are next going to see several alternative minimally more specific hypotheses.

Training Example 3 (Negative): ⟨Rainy, Cold, High, Strong, Warm, Change⟩ → No

The negative day is ⟨Rainy, Cold, High, Strong, Warm, Change⟩. The positive days so far are summarised by S_2 = ⟨Sunny, Warm, ?, Strong, Warm, Same⟩.

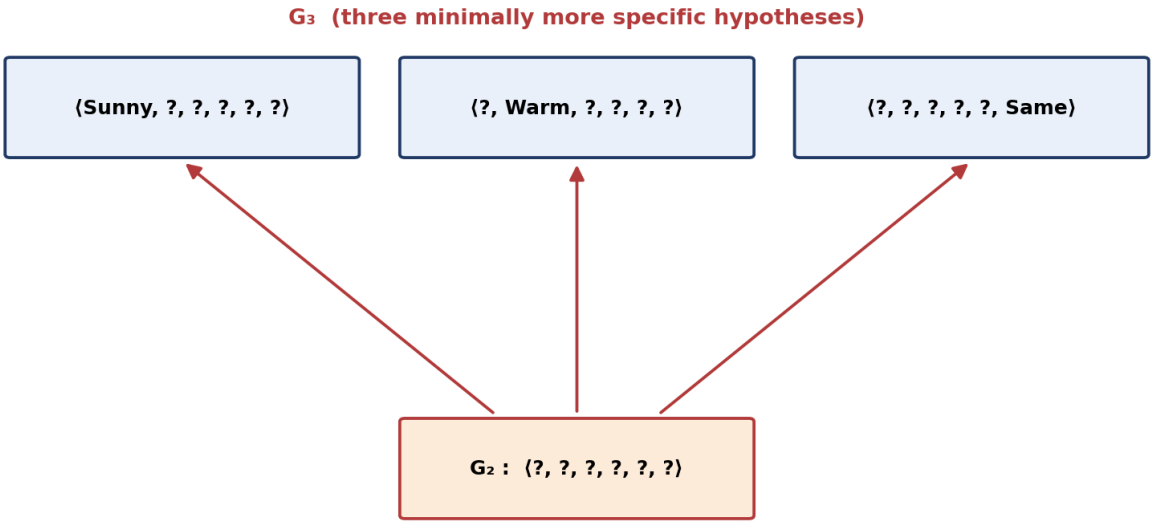
To specialize G_2 = ⟨?, ?, ?, ?, ?, ?⟩, we add **one constraint at a time**, using the value from the S-summary for that attribute (which guarantees the positive days still pass). Then we check whether it also rejects the negative day:

Add Constraint	Does it reject the negative day?	Kept?
Sky = Sunny	Yes — the negative day was Rainy	✓ Keep
AirTemp = Warm	Yes — the negative day was Cold	✓ Keep
Humidity = ?	No usable value — S_2 already has "?" for Humidity, so no constraint is available	✗Out
Wind = Strong	No — the negative day also had Strong wind, so it is not rejected	✗Out
Water = Warm	No — the negative day also had Warm water, so it is not rejected	✗Out
Forecast = Same	Yes — the negative day was Change	✓ Keep

So, out of six attributes, only **Sky, AirTemp, and Forecast** give a constraint that the positive days satisfy but the negative day violates. All three become members of the new G_3 boundary set. Regarding S_3: the negative example does not touch S, so $S_3 = S_2$.

$S_3 = S_2 = \{ \langle Sunny, Warm, ?, Strong, Warm, Same \rangle \}$

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



Negative example 3 ⟨Rainy, Cold, High, Strong, Warm, Change⟩ forces G_2 to be specialized to G_3.

Figure 2: Negative example 3 forces G_2 to be specialized — all three minimally more specific hypotheses become members of the new G_3 boundary set.

Training Example 4 (Positive): ⟨Sunny, Warm, High, Strong, Cool, Change⟩ → Yes

This positive example generalizes S further: Water differs (Warm vs. Cool) and Forecast differs (Same vs. Change), so both positions become "?".

It also affects G: the member ⟨?, ?, ?, ?, ?, Same⟩ is **inconsistent** with this positive example (its Forecast is Change), so that rule is deleted from G.

S_4 = { ⟨Sunny, Warm, ?, Strong, ?, ?⟩ }

G_4 = { ⟨Sunny, ?, ?, ?, ?, ?⟩, ⟨?, Warm, ?, ?, ?, ?⟩ }

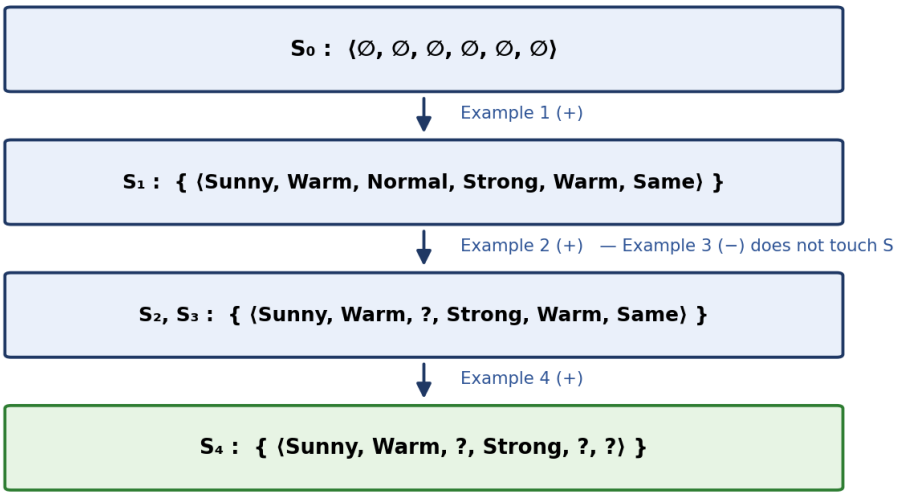


Figure 3: The complete evolution of the S boundary across all four training examples ($S_0 \rightarrow S_1 \rightarrow S_2, S_3 \rightarrow S_4$).

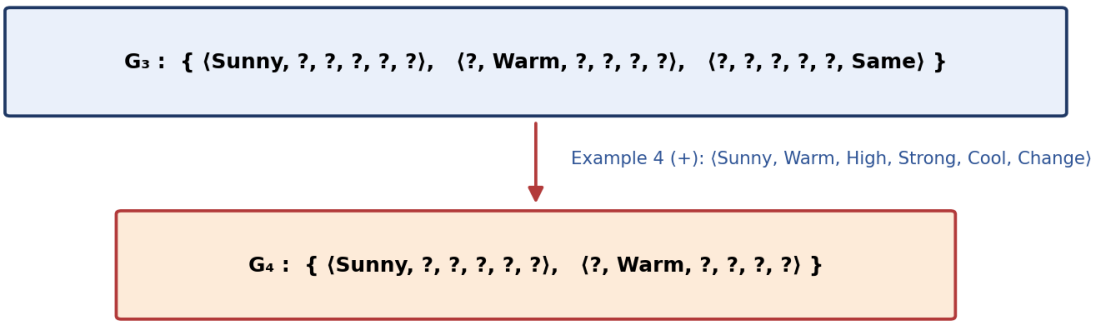


Figure 4: Positive example 4 prunes G_3 — the rule ⟨?, ?, ?, ?, ?, Same⟩ is deleted because it is inconsistent with this positive example.

The Final Version Space

After all four examples, S and G stop moving. This is the final result:

S_4 = { ⟨Sunny, Warm, ?, Strong, ?, ?⟩ } ← the most specific rule that fits

G_4 = { ⟨Sunny, ?, ?, ?, ?, ?⟩, ⟨?, Warm, ?, ?, ?, ?⟩ } ← the two most general rules that fit

Think of **S as the floor and G as the ceiling**. Any rule that sits between them — tighter than the ceiling, looser than the floor — is also a valid answer.

How to Find the Middle Rules

Relax S_4 one constraint at a time. S_4 has three real constraints: **Sunny, Warm, Strong**:

- Remove "Warm" → ⟨ Sunny, ?, ?, Strong, ?, ?⟩
- Remove "Strong" → ⟨ Sunny, Warm, ?, ?, ?, ?⟩
- Remove "Sunny" → ⟨ ?, Warm, ?, Strong, ?, ?⟩

The complete version space for the EnjoySport concept learning problem therefore contains six hypotheses:

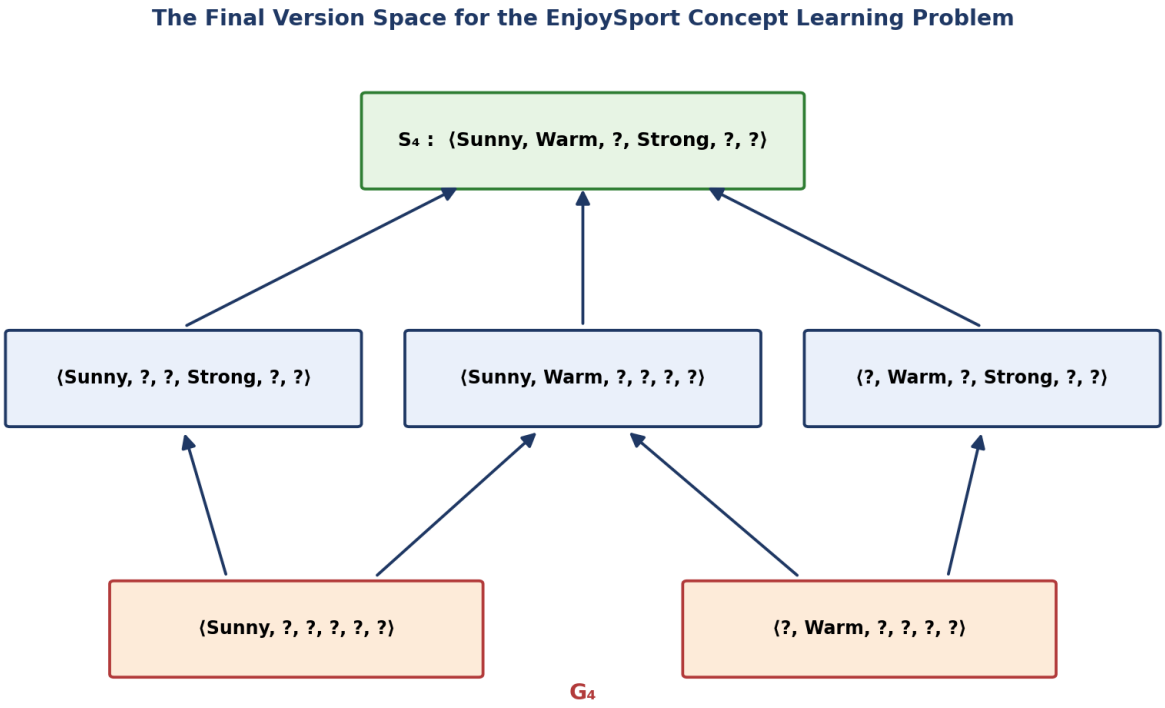


Figure 5: The final version space for the EnjoySport concept learning problem — six hypotheses, with S_4 as the floor and G_4 as the ceiling.

Every hypothesis in this diagram is consistent with all four training examples. The version space tells us exactly how much uncertainty remains — and any future training example that distinguishes between these six hypotheses would narrow it further, until S = G and the concept is found.

Key Takeaways

- CEA maintains the entire version space compactly via just two boundary sets, S and G.
- Positive examples generalize S and prune G; negative examples specialize G and prune S.
- Until the first negative example arrives, CEA's S boundary behaves identically to FIND-S.
- When S = G, the target concept has been exactly identified; while S ≠ G, the space between them measures the remaining uncertainty.