Keunwoo Lee
573 : Etzioni : Problem Set 4
November 20, 2000

1

Normalized into binary notation, the table is as follows:

	Sky	Temp	Humid	Wind	Water	Forecast	Enjoy
e_1	1	1	1	1	1	1	1
e_2	1	1	0	1	1	1	1
e_3	0	0	0	1	1	0	0
e_4	1	1	0	1	0	0	1

Resulting trace for CANDIDATE-ELIMINATION:

	S	G
(init)	$\{\ \langle\langle\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset\rangle \vee \langle\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset\rangle\rangle\ \}$	$\{\ \langle\langle ?,?,?,?,?,?\rangle \vee \langle ?,?,?,?,?,?\rangle\rangle\ \}$
e_1	$\{\ \langle\langle 1,1,1,1,1,1\rangle \vee \langle\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset\rangle\rangle\ \}$	$\{\ \langle\langle ?,?,?,?,?,?\rangle \vee \langle ?,?,?,?,?,?\rangle\rangle\ \}$
e_2	$\{\ \langle\langle 1,1,1,1,1,1\rangle \vee \langle 1,1,0,1,1,1\rangle\rangle\ \}$	$\{\ \langle\langle ?,?,?,?,?,?\rangle \vee \langle ?,?,?,?,?,?\rangle\rangle\ \}$
e_3	$\{\ \langle\langle 1,1,1,1,1,1\rangle \vee \langle 1,1,0,1,1,1\rangle\rangle\ \}$	$\{$ $\langle\langle 1,?,?,?,?,?\rangle \vee \langle ?,1,?,?,?,?\rangle\rangle,$ $\langle\langle 1,?,?,?,?,?\rangle \vee \langle ?,?,1,?,?,?\rangle\rangle,$ $\langle\langle 1,?,?,?,?,?\rangle \vee \langle ?,?,?,0,?,?\rangle\rangle,$ $\langle\langle 1,?,?,?,?,?\rangle \vee \langle ?,?,?,?,0,?\rangle\rangle,$ $\langle\langle 1,?,?,?,?,?\rangle \vee \langle ?,?,?,?,?,1\rangle\rangle,$ $\langle\langle ?,1,?,?,?,?\rangle \vee \langle 1,?,?,?,?,?\rangle\rangle,$ $\langle\langle ?,1,?,?,?,?\rangle \vee \langle ?,?,1,?,?,?\rangle\rangle,$ $\langle\langle ?,1,?,?,?,?\rangle \vee \langle ?,?,?,0,?,?\rangle\rangle,$ $\ldots$ $\langle\langle ?,?,?,?,?,1\rangle \vee \langle ?,?,?,?,0,?\rangle\rangle$ $\}$
e_4	$\{\ \langle\langle 1,1,?,1,1,1\rangle \vee \langle 1,1,0,1,0,0\rangle\rangle\ \}$	(same as above)

Notes:

- Unlike the purely conjunctive language we used in class, this language is redundant; for example, every disjunction $A \vee B$ is clearly equal to the commuted $B \vee A$. Also, the S-set after e_2 could be written $\{\langle\langle 1,1,?,1,1,1\rangle \vee \langle\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset\rangle\rangle\}$. Neither of the two representations is more specific than the other, but there is also no value in storing both in the S-set, because they mean exactly the same thing in this language.

- The trickiest step is the generalization after e_3, which results in a wide expansion in the representation of the G-set. In ZF set notation, the G-set is represented more compactly as follows:

$$\{\langle x \vee y\rangle \mid x \neq y, x \in T, y \in T, \quad T = \{ \begin{array}{l} \langle 1,?,?,?,?,?\rangle, \\ \langle ?,1,?,?,?,?\rangle, \\ \langle ?,?,1,?,?,?\rangle, \\ \langle ?,?,?,0,?,?\rangle, \\ \langle ?,?,?,?,0,?\rangle, \\ \langle ?,?,?,?,?,1\rangle \end{array} \quad \}$$

  In other words, G_3 is the set of all pairwise disjunctions of distinct hypotheses that disagree with the negative example in at least one position. Coincidentally, none of these pairs is more specific than the single hypothesis in S_3: the two S_3 subhypotheses disagree in only one position, which will always be a "don't care" in at least one half of the disjunction in each element of G_3.

- It is only by coincidence that the S-set always contains one hypothesis. It so happens that the 3 positive instances can be represented exactly by this language.

2

2.1 a,b

Algebraically, we can define hypotheses by a 4-tuple $\langle a, b, c, d\rangle$ representing the "left", "right", "bottom", and "top" of the hypotheses' rectangles. Points in version space are sets of these tuples. The S-set is $\{\langle 4,6,3,5\rangle\}$. The G-set is $\{\langle 2,7,2,5\rangle, \langle 3,7,2,7\rangle\}$.

Graphically, the set of most specific hypotheses (a unique set, in this case) is shown to the left of Figure 1, and the set of most general hypotheses is shown on the right.

c

The point $(4, 6)$ is guaranteed to reduce the size of the version space. It is strictly outside the S-set, and strictly inside the G-set. See Figure 2: if the trainer answers yes, then the S-set will expand to fit the diagram on the left; if the trainer answers no, the G-set will collapse to the one on the right.

On the other hand, any point that is either in the S-set (e.g., $(5, 4)$ and $(4, 3)$) or outside the G-set (e.g. $(8, 9)$) will not narrow either one of the sets (assuming the data are consistent, and that the "true" target concept can be represented in this hypothesis language).

d

We can teach an S-set which corresponds exactly to the target concept by specifying any two diagonally opposite corners of the rectangle as the first two points, as shown in the figure.

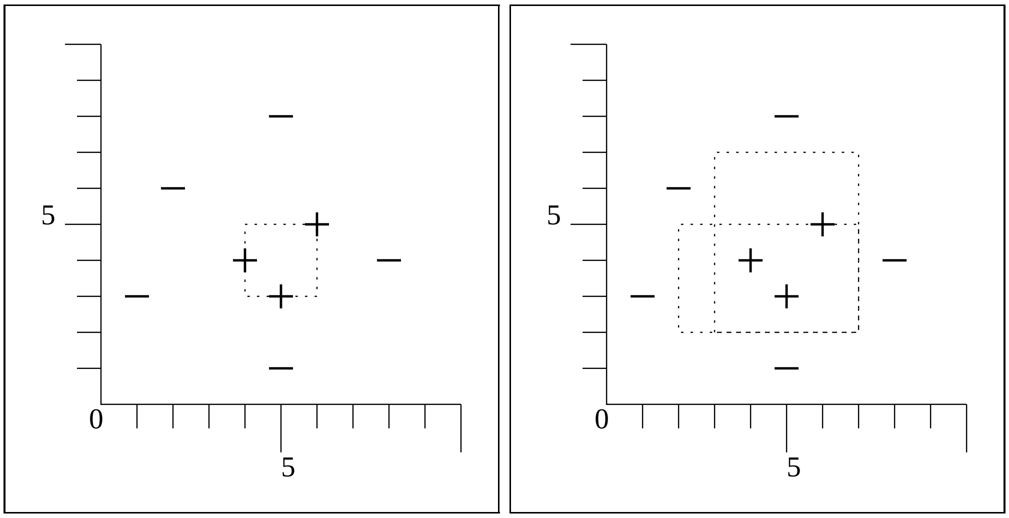

Figure 1: Most specific and most general hypotheses

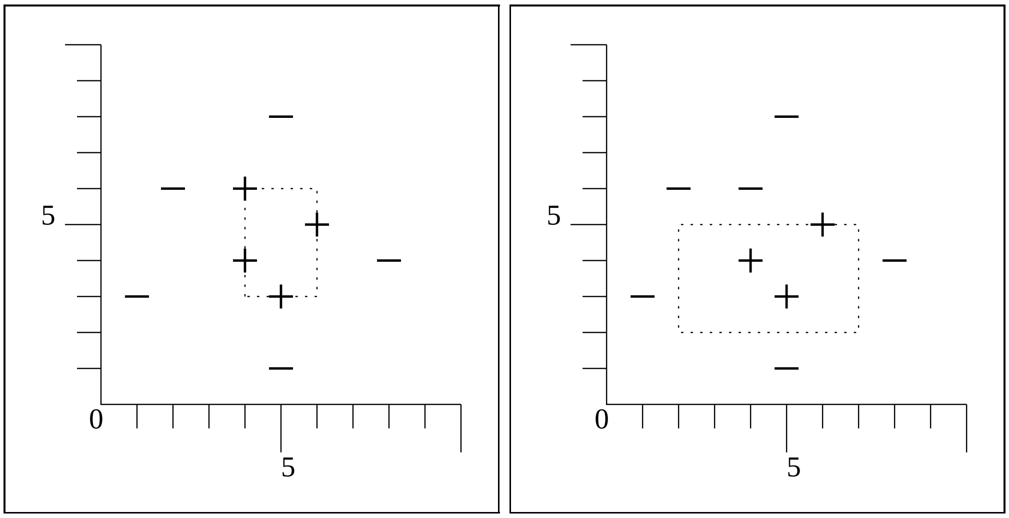

Figure 2: Result of testing $(4, 6)$ for + and - results

To bound the G-set, we need to use four examples. Each example can cut off at most half of an infinite plane, and we require four such cuts to make G exactly as specific as S, as shown in Figure 3. There are other ways to do this narrowing besides picking the negative points as shown, but all of them require four points.

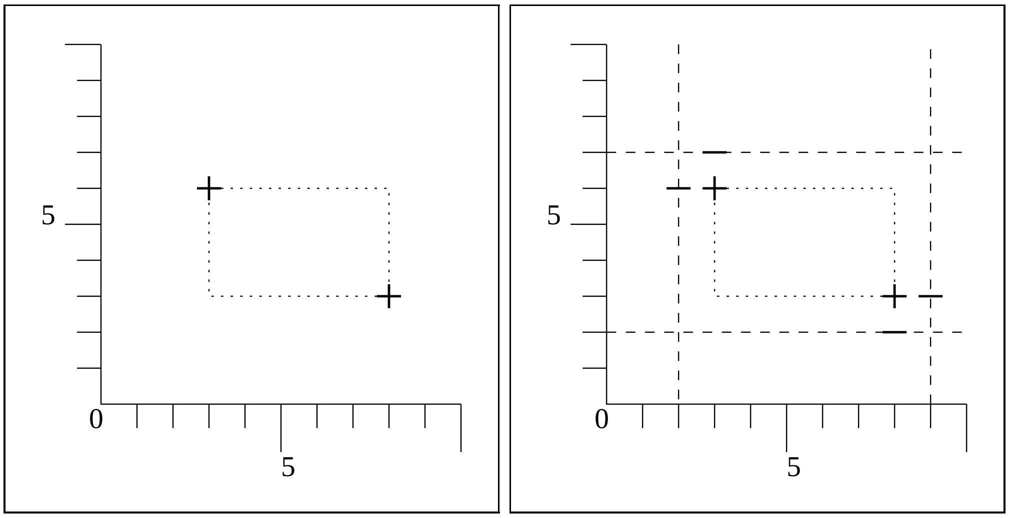

Figure 3: Minimal number of probes needed to learn a concept

3

a

There are a few different trees that could be constructed. If lexicographically earlier attributes are preferred when all else is equal, Figure 4 shows one possible tree.

b

The version space equivalent of this decision tree, for a conjunctive binary tuple language, would be $\langle 1, ?, ?, ?, ?, ? \rangle$. This tree therefore excludes certain hypotheses that would be con-

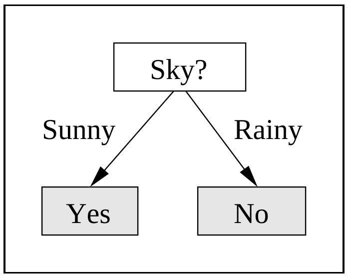

Figure 4: One possible tree for training set

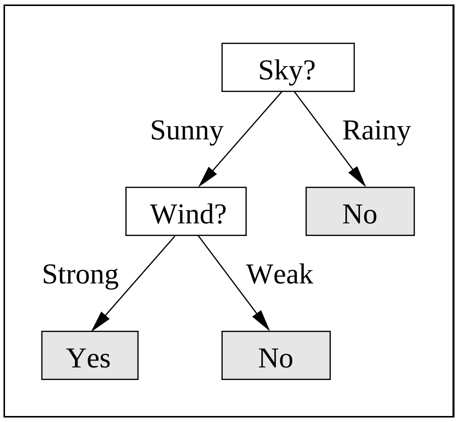

Figure 5: Decision tree for revised training set

sidered part of the "consistent set" by a version spaces algorithm—for example, $\langle ?, 1, ?, ?, ?, ? \rangle$. It also answers "yes" to some tuples that would not be accepted by the most specific hypotheses in the version spaces algorithm's S-set.

c

The new table is as follows:

	Sky	Temp	Humid	Wind	Water	Forecast	Enjoy
e_1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
e_2	Sunny	Warm	High	Strong	Warm	Same	Yes
e_3	Rainy	Cold	High	Strong	Warm	Change	No
e_4	Sunny	Warm	High	Strong	Cool	Change	Yes
e_5	Sunny	Warm	Normal	Weak	Warm	Same	No

See Figure 5 for one possible resulting tree.

d

See Figure 6 for the S and G sets; after the first training example, they both contain one element. Applying Candidate-Elimination to decision trees seems to have these drawbacks:

- Trees are not easily repesented compactly, as boolean tuples are.
- Generating all minimal generalizations of a tree, given a specific tree and a counterexample as input, is not straightforward. Even generating a single minimal generalization may be expensive: for example, if you wish to turn an attribute a into a "don't care", you must search an entire tree or subtree for nodes that branch on a.

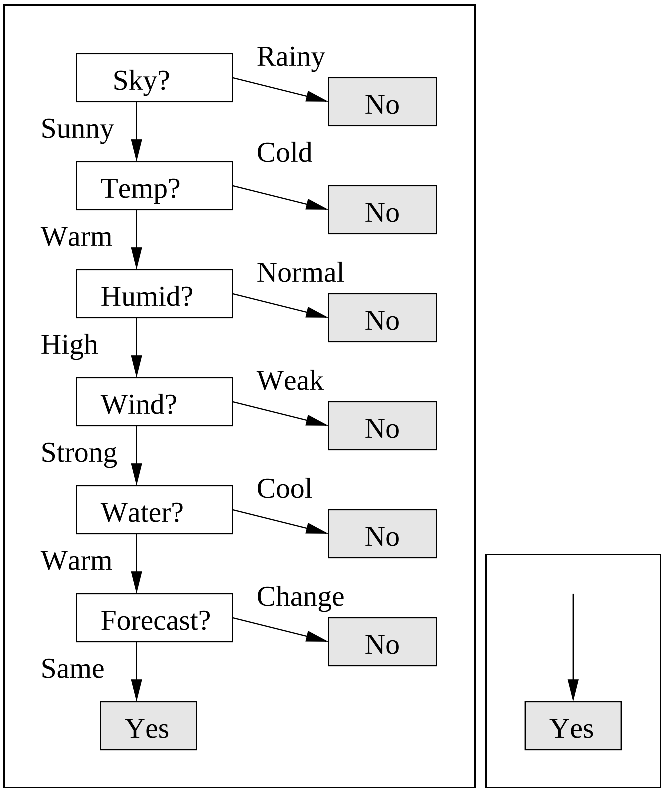

Figure 6: S-set (left) and G-set (right) for decision tree version space

- Computing the relative generality/specificity of two trees seems likely to be NP-hard (see below) unless you place strong restrictions on how the tree may be built. Preventing the S and G sets from "overlapping" each other will therefore be difficult.
- Most importantly, decision trees are more expressive than the purely conjunctive language with which we have been working. To be more precise, a decision tree can represent an arbitrary boolean predicate over the attributes. This can obviously result in huge expansion of the number of hypotheses in either the G or S sets.

4

Problem: For a conjunctive language over tuples of true/false values, what is the tightest upper bound on the number of training examples required to learn an arbitrary target concept?

For the version spaces algorithm, no concept can require more than $2 + n$ training examples, where n is the number of positions in the tuple. Each of the members in the target concept must be one of $\{1, 0, ?\}$. Consider how the trainer can force the learner to converge on each of these values for the S and then the G sets.

S-set:

First, note that the S-set in a purely conjunctive binary language always contains exactly one element. Given that:

1. To force a 1 into an S-element position, we must give the learner one example with a 1 in that position. We can teach the learner all 1 values by feeding the learner a single positive tuple with 1's in all the appropriate positions.

2. Forcing a 0 into the S-element is exactly like forcing a 1. We can force all desired 0 values into the S-set with the same tuple that we used to force the 1's.

3. To force a ? value in a tuple position, we give the learner a second positive tuple (after the one we used to teach the 1 and 0 values) that differs from the first exactly in the positions where we want ?.

Therefore, narrowing S to exactly the target concept takes only two tuples in the worst case (one if there are no ? values). This also corresponds to the number of examples required to teach an arbitrary concept for the FIND-S algorithm.

G-set

We now wish to make the G-set congruent with the S-set.

After receiving the one or two tuples we used to narrow the S-set, the G-set still contains the single tuple consisting of all "don't-care" (?) values. This is fine for all the positions where the target concept has ?, but in the worst case the target concept will be a single element and we must narrow every position to a 1 or 0.

Each training example can, at most, isolate an individual position in the tuple and determine its value. Intuitively, the learner can only receive one bit of information from each example, because it only gets a "yes" or "no" reply from the "world". If, for example, a negative training tuple differs from the current S-set in two positions, the learner cannot conclude which of the two positions caused the negative reply, and so must assume that either could have caused it. It will require a second training tuple to disambiguate the two and force the G-set to take the S-set's values for those positions.

The minimal number of training examples is therefore $n + 1$ if there are no "don't-care" values in the target concept, and $2 + (n - 1) = n + 1$ if there is at least one "don't-care" value.