Lecture 15

- Learning
  - PAC, Version space, Decision trees (part 1)

PAC: definition

Relax this requirement by not requiring that the learning program necessarily achieve a small error but only that it to keep the error small **with high probability**.

Probably approximately correct (PAC) with probability δ and error at most ε if, given any set of training examples drawn according to the fixed distribution, the program outputs a hypothesis f such that

$$\Pr(\mathrm{Error}(f) > \varepsilon) < \delta$$

PAC Training examples

Theorem:

If the number of hypotheses |H| is finite, then a program that returns an hypothesis that is consistent with

$$m = \ln(\delta / |H|) / \ln(1 - \varepsilon)$$

training examples (drawn according to Pr) is guaranteed to be PAC with probability δ and error bounded by ε.

We want….

- PAC (so far) describes accuracy of the hypothesis, and the chances of finding such a concept.
  - How may examples do we need to rule out the “really bad” hypotheses.
- We also want the process to proceed *quickly*.

PAC learnable spaces

A class of concepts C is said to be

PAC learnable for a hypothesis space H

if there exists an polynomial time algorithm *A* such that:

for any $c \in C$, distribution Pr, ε , and δ ,

if *A* is given a quantity of training examples polynomial in 1/ ε and 1/ δ,

then with probability 1- δ

the algorithm will return a hypothesis f from H such that

$$\text{error}(f) <= \varepsilon .$$

Observations on PAC

- PAC learnability doesn't tell us *how* to find the learning algorithm.
- The number of examples needed grows slowly as the concept space increases, and with other key parameters.

Example

- Target and learned concepts are conjunctions with up to *n* predicates. (This is our bias.)
  - Each predicate might be appear in either positive or negated form, or be absent: 3 options.
  - This gives 3^n possible conjunctions in the hypothesis space.

Result

- I have such a formula in mind.
- I'll give you some examples.
- You try to guess what the formula is.

A concept that matches all our examples will be PAC if m is at least

$$n/\varepsilon \ln (3/ \delta)$$

How

- How can we actually find a suitable concept?

- One key approach: start with the examples themselves, and try to generalize.

- E.g. Given f(3,5) and f(5,5).
  - We might try replacing the first argument with a variable X: f(X,5).
  - We might try replacing both arguments with variables: f(X,Y).
  - We want to get as general as possible, but not too general.
- The converse of this generalization is specialization.

Version Space [DAA 5.3]

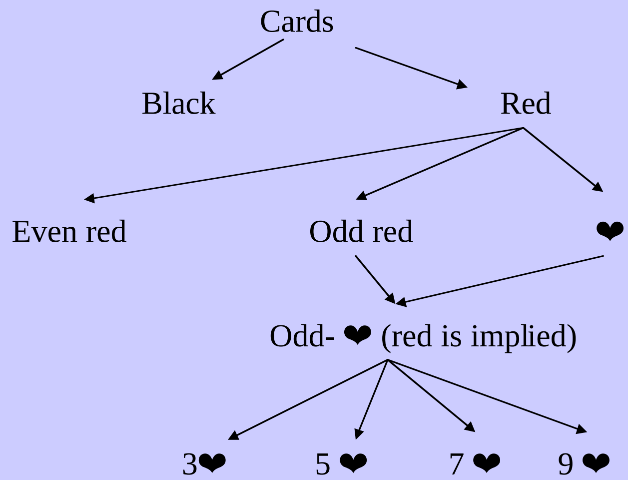
- Deals with *conjunctive concepts.*

- Consider a concept *C* as being identified with the set of positive examples it associated with.
  - C:even numbered hearts = {3❤,5 ❤,7 ❤,9 ❤}.

- A concept C_1 *is a* **specialization** *of concept* C_2 if the examples associated with C_1 are a subset of those associated with C_2 .

- **3-of-hearts** more specialized than **odd hearts**.

Specialization/Generalization



Immediate

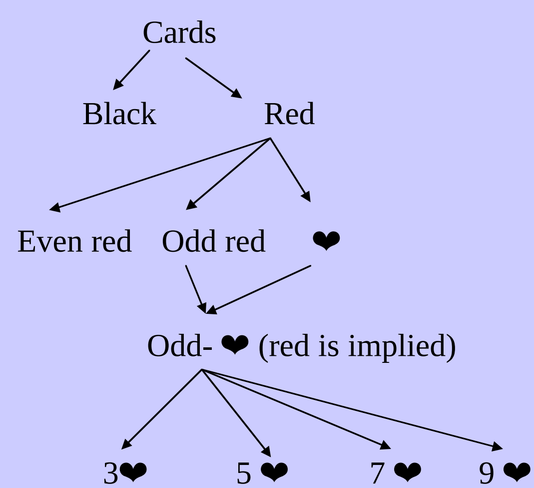
- <u>**Immediate Specialization**</u>: no intermediate.
- Red is <u>not</u> the immediate specialization of 2-of-hearts.
- Red <u>is</u> the immediate specialization of hearts and diamonds.
  – Note: This observation depends on knowing the hypothesis space restriction.

Algorithm outline

- Incrementally process training data.
- Keep list of <u>most</u> and <u>least</u> specific concepts consistent with the observed data.
  - For two concepts *A* and *B* that are consistent with the data,the concept *C*= (*A AND B)* will also be consistent yet more specific.
- Tied in a subtle way to **conjunctions.**
  - Disjunctive concepts can be obtained trivially by joining examples, but they're not interesting.

VS example

- 4 ❤ :no
- 5 ❤ :yes
- 5 ♣ :no
- 7 ❤ :yes
- 9 ♠ ---
- 3❤ yes



Algorithm specifics

- Maintain two bounding concepts:
  - The most specialized (DAA: *specific boundary)*
  - The broadest (DAA: *general boundary).*

- Each example we see is either positive (yes) or negative (no).
- Positive examples (+) tend to make the concept more general (or inclusive). Negative examples (-) are used to make the concept more exclusive (to reject them).

  \+ -> move “up” the specific boundary

  \- -> move “down” the general boundary.

Detailed algorithm: DAA p. 191-192.

Observations

- It allows you to ***<u>GENERALIZE</u>*** from a training set to examples never-before-seen !!!
  - In contrast, consider table lookup or rote learning.

- Why is that good?
  1 It allows you to infer things about new data (the whole point of learning)
  2 It allows you to (potentially) remember old data much more efficiently.
- Version space method is optimal for a conjunction of positive literals.
  - How does it perform with noisy data?

Learning Decision Trees

- **<u>Decision tree</u>** takes as input a set of properties and outputs yes/no "decisions".

- A tree structure such that
  - Internal nodes correspond to questions
  - Arcs are associated with answers to questions
  - Terminal (leaf) nodes are decisions.

- Example:

goal predicate: *WillWait*

do we want to wait for a table at a restaurant?

Restaurant Selector

Example attributes:

- 1. Alternate
- 2. Bar
- 3. Fri/Sat
- 4. Hungry
- 5. Patrons
- 6. Price
- etc.

forall r Patrons(r, Full) AND WaitEstimate(r,under_10) AND Hungry(r,N) -> WillWait(r)$

Example 2

Maybe we should have made a reservation? (using a decision tree)

- Restaurant lookup: you've heard Joe's is good.
- Lookup Joe's

Decision trees: issues

- Constructing a decision tree is easy… really easy!
  - Just add examples in turn.
- Difficulty: how can we extract a ***simplified*** decision tree?
  - This implies (among other things) establishing a preference order (bias) among alternative decision trees.
  - Finding the smallest one proves to be VERY hard. Improving over the trivial one is okay.