Foundations of Artificial Intelligence

CS472/3 — Fall 1999
Lecture #27

Bart Selman

Slide CS472–1

• today: formal learning results / **PAC**

Slide CS472–2

PAC

Introduce: **Probably Approximately Correct Learning**
That is,

For our learning procedures, we will try to prove that:

With **high probability** our learning algorithm will find an hypothesis that is **approximately** identical to the hidden target concept.

Note: the double "hedging" — probably ... approximately...
Why do you need both levels of uncertainty (in general)?

How Many Examples Are Needed?

(rather technical)

- **X** is the set of all possible examples.
- D the distribution with which we draw examples.
- **H** set of possible hypotheses.
- m number of examples in training set.

Assume, the true function f is in **H**.

error of a hypothesis h wrt f is defined as
the probability that h differs from f on a randomly picked example:

$$error(h) = P(h(x) \neq f(x) | x \text{ drawn from } D)$$

This is what we were trying to measure with our **test set** before.

But, we don't have f ...
Where do we get info about f?

Approximately Correct

h is **approximately correct** iff

$$error(h) \leq \epsilon.$$

Such an hypothesis lies withing an ϵ-ball of f. Rest of hypotheses $\mathbf{H}_{bad}$.
In words: We want h such that when we pick random examples, only **rarely** does h misclassify an example (i.e, with probability less than ϵ).

Idea: show that after seeing m examples, with high probability, all consistent hypothesis will be approximately correct.

I.e., chance of a "bad" hypothesis (*but consistent with the examples*) is small (less than δ).

Slide CS472–7

Let h_b be a bad hypothesis, i.e, $error(h_b) > \epsilon$.

So, chance h_b disagrees with an example $> \epsilon$.

So, prob. it agrees with a given example is $\leq (1 - \epsilon)$.

We have

$$P(h_b \text{ agrees with } m \text{ examples }) \leq (1-\epsilon)^m$$

$$P(\mathbf{H}_{bad} \text{ contains a hypth. consistent with } m \text{ exs. }) \leq |\mathbf{H}_{bad}|(1-\epsilon)^m \quad \leq |\mathbf{H}|(1-\epsilon)^m$$

We would like to make this unlikely ($\leq \delta$).

Slide CS472–8

So,

$$|\mathbf{H}|(1-\epsilon)^m \leq \delta.$$

It follows:

$$m \geq \frac{1}{\epsilon}(ln\frac{1}{\delta} + ln|\mathbf{H}|).\ (1)$$

So, how can we keep the number of examples we need down?

Slide CS472–9

(1) says that if a learning alg. returns a hypothesis that is consistent with this many examples, then with prob. at least $1-\delta$, the hypothesis has error of at most ϵ.

(1) as a function of ϵ and δ is called the **sample complexity** of the hypothesis space.

Note that we only ask from the learner to find some hypothesis consistent with the m examples.

Slide CS472–10

Consider: **H** space of all Boolean functions.

$|\mathbf{H}| = 2^{2^n}$.

Sample complexity grows with 2^n.

Same as number of **all possible** examples!

Therefore, learning algorithm cannot do better than lookup table, if it merely returns hypothesis consistent with given examples.

What is this saying intuitively about **H**?

What does this mean for *e.g* **neural nets**?

Learning in general?

Solution

1) Force algorithm to look for "smallest" consistent hypothesis.

We considered this for decision tree learning.

(often worst-case intractable)

2) Restrict form of Boolean function — size of hypotheses space.

E.g. if only hypotheses are **conjunctions of literals**, then we only need poly number of examples!

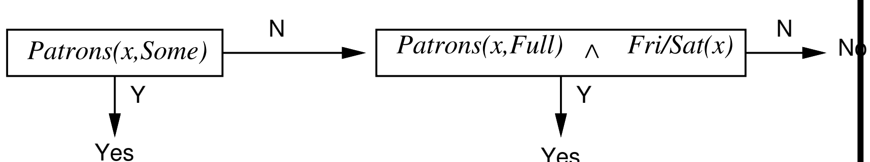
Example: decision lists (in R&N p. 555.)

Decision lists

Resemble decision trees, structure is simpler, decisions are more complex.

$$\forall\, WillWait(x, Some) \Leftrightarrow (Patrons(x, Some) \\ \vee (Patrons(x, Full) \wedge Fri/Sat(x)))$$

Slide CS472–13



Slide CS472–14

Each test is a conjunction of literals.
Labels can be "yes" or "no".

If you allow arbitrarily many lits per tests, then decision lists can express all Boolean functions

Consequence for PAC learning?

(Can view as "decision tree"; what form?)

Slide CS472–15

Limit expressiveness of tests:

involve at most k literals.

Becomes PAC learnable!

We have to show that we don't need too many examples to find a good k-DL(n) hypothesis.

n Boolean attributes.

Slide CS472–16

Limit expressiveness of tests:

Language of tests: $Conj(n, k)$

At most $3^{|Conj(n,k)|}$ sets of tests (Yes/No/absent).

All possible orders, so:

$$|k\text{-DL(n)}| \leq 3^{|Conj(n,k)|} \; |Conj(n, k)|!$$

Slide CS472–17

After some work, we get

$$|k\text{-DL(n)}| = 2^{O(n^k \; log_2(n^k))}$$

(useful exercise! try mathematica)
plug in (1), what is the key point here?

What if k approaches n?

Slide CS472–18

We get

$m \geq \frac{1}{\epsilon}\ (ln(\frac{1}{\delta}) + O(n^k\ log_2(n^k)))$

So, for fixed k, need only a polynomial number of examples!

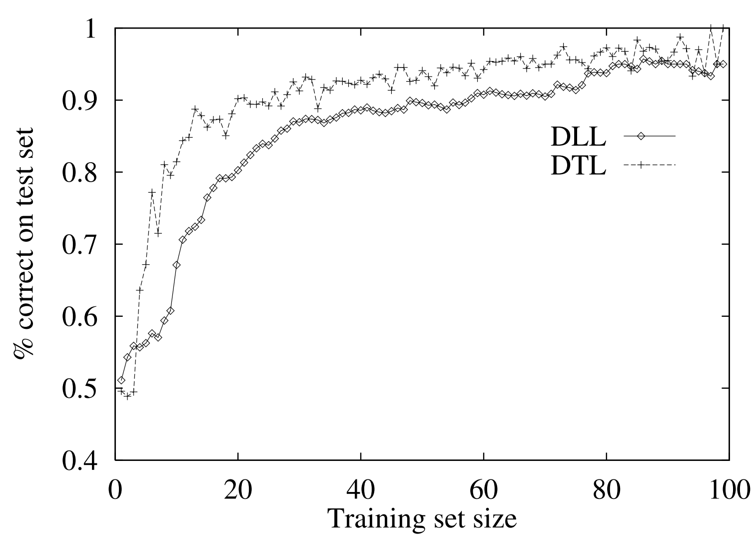
Slide CS472–19

Now we need an algorithm that can find a consistent hypothesis. Use simple greedy algorithm.

Slide CS472–20

function DECISION-LIST-LEARNING(*examples*) **returns** a decision list, *No* or failure

if *examples* is empty **then return** the value *No*
t ← a test that matches a nonempty subset $examples_t$ of *examples*
such that the members of $examples_t$ are all positive or all negative
if there is no such *t* **then return** failure
if the examples in $examples_t$ are positive **then** o ← *Yes*
else o ← *No*
return a decision list with initial test *t* and outcome *o*
and remaining elements given by DECISION-LIST-LEARNING($examples - examples_t$)



Why decision list somewhat worse than decision tree?

Can we actually be sure that our concept can be learned as a k-DL?

(Note not clear figure is for fixed k.)

Slide CS472–23

Learning a **conjunction of literals** is another example of a PAC learnable language.

PAC learning is an important advance in learning theory. Unfortunately, also many negative results. Practical algorithms somewhat limited.

Limitations? Is it the right model?

Slide CS472–24