EECS 219C: Computer-Aided Verification

Inductive Learning (Machine Learning Theory)

Sanjit A. Seshia

EECS, UC Berkeley

Acknowledgments: Avrim Blum

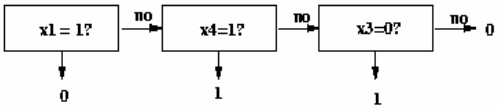
Outline

- Basic Concepts
- Batch vs. Online Learning
- More on Batch Learning
- More on Online Learning
- Teaching vs. Learning

(whiteboard material augments what's in the following slides)

2

Example: Decision Lists



Given a dataset S of m examples over n boolean features, drawn according to unknown distrib D, labeled by unknown target f:

1. Algorithm **A** will find a consistent DL if one exists, in time O(mn).
2. If $m > (1/\varepsilon)[n(2+\ln n) + \ln(1/\delta)]$, then
   Pr[exists consistent DL h with $err(h) > \varepsilon$] $< \delta$.

3

How can we find a consistent DL?

x_1	x_2	x_3	x_4	x_5	label
1	0	0	1	1	+
0	1	1	0	0	−
1	1	1	0	0	+
0	0	0	1	0	−
1	1	0	1	1	+
1	0	0	0	1	−

if (x_1=0) then -, else
if (x_2=1) then +, else
if (x_4=1) then +, else -

4

Decision List algorithm

- Start with empty list.
- Find if-then rule consistent with data.
  (and satisfied by at least one example)
- Put rule at bottom of list so far, and cross off examples covered. Repeat until no examples remain.

If algorithm fails, then:
- No DL consistent with remaining data.
- So, no DL consistent with original data.

OK, fine. Now why should we expect it to do well on future data?

5

Confidence/sample-complexity

- Consider some hypothesis h with err(h) > ε.
- Chance that h survives m examples is at most $(1-\varepsilon)^m$.
- Number of DLs over n Boolean features is at most $n!4^n$. (for each feature there are 4 possible rules, and no feature will appear more than once)

  $\Rightarrow$ Pr[some DL h with err(h)>ε is consistent] < $n!4^n(1-\varepsilon)^m$.

- This is < δ for $m > (1/\varepsilon)[n(2+\ln n) + \ln(1/\delta)]$

6

DL: summary

Suppose the target f is, in fact, a decision list.

Then with probability $\geq 1-\delta$, the hypothesis h produced by the algorithm has error $< \varepsilon$, so long as the number of examples m seen satisfies

$$m \geq \frac{1}{\varepsilon}\left[n(2+\ln n)+\ln\frac{1}{\delta}\right].$$

I.e., it's Probably Approximately Correct

7

Confidence / sample complexity

Nothing special about DLs in our argument.

- All we said was: "if not too many rules to choose from, then unlikely some bad one will fool you just by chance."
- Generalize to any hypothesis space H.
- After m examples, with probability $\geq 1-\delta$, all $h \in H$ with $err(h) \geq \varepsilon$ have $\widehat{err}(h) > 0$, for

$$m \geq \frac{1}{\varepsilon}\left[\log(|H|)+\log\left(\frac{1}{\delta}\right)\right].$$

Occam's razor

William of Occam ($\sim$ 1320 AD):

> "Entities should not be multiplied unnecessarily" (in Latin)

Which we interpret as: "in general, prefer simpler explanations".

Why? Is this a good policy? What if we have different notions of what's simpler?

9

Occam's razor (contd)

A computer-science-ish way of looking at it:

- Say "simple" = "short description".
- At most 2^s explanations that are $< s$ bits long.
- If number of examples seen satisfies

$$m \geq \frac{1}{\epsilon}\left[s \ln 2 + \ln\left(\frac{1}{\delta}\right)\right].$$

then it's unlikely a bad simple explanation will fool you just by chance.

10

Occam's razor (contd)[2]

Nice interpretation:

- Even if we have different notions of what's simpler (e.g., different representation languages), we can both use Occam's razor.
- Of course, there's no guarantee there will be a short explanation for the data. That depends on your representation.

11

Online Learning Setting

- View learning as a sequence of trials.
- In each trial, algorithm is given x, asked to predict f, and then is told the correct value.
- Make no assumptions about how examples are chosen.
- Goal is to minimize number of mistakes.

Note: can no longer talk about # examples needed to converge. Instead, we focus on number of mistakes. Need to "learn from our mistakes".

14

Simple example: learning an OR fn

- Suppose features are boolean: X = $\{0,1\}^n$.
- Target is an OR function, like $x_3 \vee x_9 \vee x_{12}$, with no noise.
- Can we find an on-line strategy that makes at most n mistakes?
- Sure.
  - Start with $h(x) = x_1 \vee x_2 \vee \ldots \vee x_n$
  - Invariant: {vars in h} contains {vars in f}
  - Mistake on negative: throw out vars in h set to 1 in x. Maintains invariant and decreases $|h|$ by 1.
  - No mistakes on postives. So at most n mistakes total.

15

N Experts Problem

- We have n "experts".
- One of these is perfect (never makes a mistake). We just don't know which one.
- Can we find a strategy that makes no more than lg(n) mistakes?

Answer: sure. Just take majority vote over all experts that have been correct so far. Called "halving algorithm".

Followup question: what if we have a "prior" p over the experts. Can we make no more than $\lg(1/p_i)$ mistakes, where expert i is the perfect one?

Sure, just take weighted vote according to p.

17

Relation to concept learning

- If computation time is no object, can have one "expert" per concept in C.
- If target in C, then number of mistakes at most lg(|C|).
- More generally, for any description language, number of mistakes is at most number of bits to write down f.

18

Back to expert-advice

What if no expert is perfect? Goal is to do nearly as well as the best one in hindsight.

Strategy #1:

- Iterated halving algorithm. Same as before, but once we've crossed off all the experts, restart from the beginning.
- Makes at most $\log(n)*OPT$ mistakes, where OPT is # mistakes of the best expert in hindsight.

Seems wasteful. Constantly forgetting what we've "learned". Can we do better? Yes.

19

Weighted Majority Algorithm

Intuition: Making a mistake doesn't completely disqualify an expert. So, instead of crossing off, just lower its weight.

Weighted Majority Alg:

- Start with all experts having weight 1.
- Predict based on weighted majority vote.
- Penalize mistakes by cutting weight in half.

20

Weighted Majority Algorithm

Weighted Majority Alg:

- Start with all experts having weight 1.
- Predict based on weighted majority vote.
- Penalize mistakes by cutting weight in half.

Example:

					prediction	correct
weights	1	1	1	1		
predictions	Y	Y	Y	N	Y	Y
weights	1	1	1	.5		
predictions	Y	N	N	Y	N	Y
weights	1	.5	.5	.5		
predictions	Y	N	N	N	N	N
weights	.5	.5	.5	.5		
predictions	N	Y	N	Y	either	N
weights	.5	.25	.5	.25		

21

Analysis: do nearly as well as best expert in hindsight

- M = # mistakes we've made so far.
- m = # mistakes best expert has made so far.
- W = total weight (starts at n).
- After each mistake, W drops by at least 25%. So, after M mistakes, W is at most $n(3/4)^M$.
- Weight of best expert is $(1/2)^m$. So,

$$\begin{aligned} (1/2)^m &\leq n(3/4)^M \\ (4/3)^M &\leq n2^m \\ M &\leq 2.4(m + \lg n) \end{aligned}$$

constant comp. ratio

22