General Setting

- A **concept** is a function from examples to $\{0, 1\}$.
  $c(x) = 1 \iff x$ is a positive example of c
  $c(x) = 0 \iff x$ is a negative example of c

- A **Concept Class** is a set of concepts, defined by some representation.
  E.g., conjunctions.

- The **Instance Space** is the set/space of possible examples.

Talk about algorithms to "learn a concept class C." Should be successful whenever target concept is in C.

Mistake-Bound model

A model for on-line/incremental learning.

Learning in stages. In each stage:

- Learner gets unlabeled example.
- Learner predicts classification.
- Learner is told correct answer.

Goal: bound the total number of mistakes.

Algorithm *learns* C in the Mistake-Bound model with mistake-bound p if: for any $c \in C$, any ordering of examples of c, # mistakes $\leq p(n, s)$.

n = size of examples, s = size (description length) of target concept.

Want: p and running time per stage to be low.

C **is learnable** in the mistake-bound model if there exists an algorithm to learn C where p and running time per stage are poly(n, s).

Learning conjunctions (monomials)

- Features are $x_1, x_2, \ldots, x_n$.
  For convenience, assume boolean. E.g.,
  $x_1 = 1 \iff$ sky $=$ sunny,
  $x_1 = 0 \iff$ sky $=$ rainy.
- Example is assignment of values to features.
- Concept class is all ANDs of "literals". E.g.,
  $(x_1 = 1) \wedge (x_4 = 0) \wedge (x_7 = 1) \wedge (x_9 = 0) \in C$.
  Write as: $x_1 \overline{x_4} x_7 \overline{x_9}$.

Algorithm: predict according to most specific conjunction consistent with data so far.

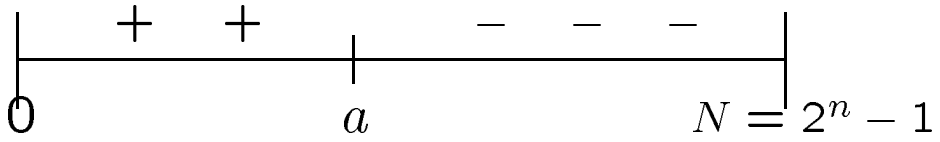
- Start: $h = x_1\overline{x_1}x_2\overline{x_2}\cdots x_n\overline{x_n}$.
- Mistake on positive x: remove from h all literals not satisfied by x.
- Mistake on negative x: say "no consistent conjunction."

Guarantee: $\{$ literals in target$\} \subseteq \{$ literals in $h\}$.

Max # of mistakes: $n + 1$

Another situation:

- Learn OK temperatures for some device.



  Say cold is good.

- Example: $x \in \{0, 1, \ldots, 2^n - 1\}$.
- Concept class is:
  {initial subintervals of $[0, 2^n - 1]$}.
  Specific concept is $[0, a]$.

How does the "pick most specific" alg do?

What is a good algorithm?

General Upper Bounds

If ignore computation time, then can learn any class C with at most $\log_2 |C|$ mistakes.

Use "Halving Algorithm."

Lower Bounds

- Conjunctions: $\geq n + 1$ mistakes.

- Single feature: $\geq \lfloor \log_2 n \rfloor$ mistakes.

- Conjunction (or disjunction) of r relevant features: $\geq r \lfloor \log_2 n/r \rfloor$ mistakes.

PAC model

- Sample from unknown but fixed distribution $\mathcal{D}$ on labeled examples.
- Error of hypothesis h w.r.t. target c is: $Prob_{x \in \mathcal{D}}[h(x) \neq c(x)]$
- Goal: error $< \epsilon$.
- Allowed to fail with probability δ.

Defn: Learner learns class C (by H) in PAC model if given any $\epsilon, \delta > 0$: for any $c \in C$ and any distrib $\mathcal{D}$, with high probability ($\geq 1 - \delta$) it finds a hypothesis h (in H) with low error ($\leq \epsilon$).

Allowed: Time, # examples poly$(n, s, 1/\epsilon, 1/\delta)$.

"probably approximately correct" (PAC)

Restricted ($H = C$) vs Unrestricted ($H =$ poly time algs)

Learning initial subintervals

Algorithm: Select m examples. Pick most specific interval. (I.e., interval $[0, b]$ where b is largest positive seen.)

Learning conjunctions

Algorithm: Select m examples. Pick most specific consistent conjunction.

Analysis:

- number of different conjunctions:
- Fix some conjunction with error $\geq \epsilon$.
  Prob[consistent with data] $\leq (1-\epsilon)^m$.
- Prob[exists any consistent conjunction with error $\geq \epsilon$] $\leq$

- If we can find a consistent $h \in C$, then just need:

$$\frac{1}{\epsilon}\left(\ln |C| \ + \ \ln \frac{1}{\delta}\right) \text{ examples.}$$

- Sometimes this is a big overcount.

PAC model and consistent hyps

Suppose all $c \in C$ are:

- $poly(n)$ bits long.
- poly-time computable.

Then, PAC-learning C by C is equivalent to solving:

> "Given a set S of labeled examples, find $h \in C$ consistent if one exists in (randomized) poly time."

Proof:

$\Longleftarrow$ Previous argument. Use: $\log |C| = poly(n)$.

$\Longrightarrow$ Let $\mathcal{D}$ have prob. $1/|S|$ on each example in S. Let $\epsilon = 1/(m+1)$, δ =desired failure prob.

Decision Lists

Example: If rainy then no else if warm then yes else if weekend then yes else if teaching-day then no else yes.

Defn: A decision list is a list of rules of the form $(A \rightarrow B)$ where A is a variable or its negation and B is yes or no, followed by a default value. The meaning is that given an example, we look at the first left-hand-side satisfied and output the corresponding right-hand-side. If none were satisfied we output the default.

- How many decision lists are there?

- How can you find a consistent one?

Notes on PAC calculations:

In order to simplify the formulas describing the number of examples needed, I used in class the fact that $-\ln(1-\epsilon) \geq \epsilon$. This comes from the Taylor series expansion: $-\ln(1-x) = x + x^2/2 + x^3/3 + ...$

So, in the case we were looking at, if $m \geq \frac{1}{\epsilon}(\ln(|C|) + \ln(\frac{1}{\delta}))$, then $-\ln(1-\epsilon)m \geq \ln(|C|) + \ln(\frac{1}{\delta})$, which implies that $(1-\epsilon)^{-m} \geq |C|/\delta$ or $|C|(1-\epsilon)^m \leq \delta$.

The analysis to get up to the equation above is: Since our algorithm finds a consistent hypothesis, we are certain to succeed if *all* bad hypotheses are inconsistent with the data. In other words, we can only fail if there exists a bad but consistent hypothesis. Since there are at most $|C|$ bad hypotheses, and each individual one has a probability at most $(1-\epsilon)^m$ of being consistent, by subadditivity we are OK if $|C|(1-\epsilon)^m \leq \delta$.