Recap: Learning conjunctions

Algorithm: Remove inconsistent literals.

Analysis:

- Fix some "bad" hypothesis (error $\geq \epsilon$).
  Prob[consistent with m data pts] $\leq (1-\epsilon)^m$.
- Prob[any bad hypothesis is consistent]
  $\leq |H|(1-\epsilon)^m$. So,

  $$m \geq \frac{1}{\epsilon}[\ln(|H|) + \ln(1/\delta)]$$

  examples sufficient to achieve guarantee.
- number of different conjunctions: 3^n.

Another convenient formulation: If have m examples, then can be confident if $|H| \leq \delta e^{\epsilon m}$.

PAC and consistency models

Suppose $|C| \leq 2^{poly(n)}$ and all $c \in C$ are poly-time computable.

Then, PAC-learning C by C is equivalent to solving consistency problem for C in randomized polynomial time.

Proof:

$\Leftarrow$ Previous argument. Give randomized alg a failure prob of $\delta/2$ and use $\delta/2$ in equation.

$\Rightarrow$ Let $\mathcal{D}$ have prob. $1/|S|$ on each example in S. Let $\epsilon = 1/(|S|+1)$, 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:

$$A_1 \to B_1,\ A_2 \to B_2,\ \ldots,\ A_{l-1} \to B_{l-1},\ True \to B_l$$

where A_i are literals and $B_i \in \{0, 1\}$. The meaning is: given an example, we look at the first left-hand-side satisfied and output the corresponding right-hand-side.

- How many decision lists are there?

- How can you find a consistent one?

What about more expressive hypotheses?

- What kind of guarantee can we give for algs that use decision trees or DNF formulas as hypotheses?

- More generally, when does data justify a hypothesis?

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

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

What can we say about Occam's razor?

Occam's razor: prefer simpler explanations.

Why?

One reason: not so many of them. Less likely fits data just by chance.

Definition: A *legal size function* on concepts is one such that $< 2^s$ concepts have size $< s$.

Thm: Given m examples, with probability $\geq 1 - \delta$ there are no bad (error $\geq \epsilon$) consistent hypotheses of size s where:

$$m > \frac{1}{\epsilon}\left(s \ln 2 + \ln \frac{1}{\delta}\right)$$

or

$$s < \frac{1}{\ln 2}\left(\epsilon m - \ln \frac{1}{\delta}\right).$$

"Learning is data compression"

Occam's razor, contd.

Defn: $\mathcal{A}$ is an *Occam* algorithm for learning C by H if given any set S of m examples, if $\exists$ consistent $c \in C$ then $\mathcal{A}$ finds consistent $h \in H$ with $size(h) \leq size(c)^{\alpha} m^{\beta}$, where α, β are constants and $\beta < 1$.

Thm: An Occam alg is a PAC-learning alg.

Proof: Just need # examples m to satisfy:

$$m/2 \geq \frac{1}{\epsilon}(s \ln 2) \quad \text{and} \quad m/2 \geq \frac{1}{\epsilon} \ln \frac{1}{\delta}.$$

So,

$$m \;\geq\; \max \left\{ \left[\frac{2 \ln 2}{\epsilon} size(c)^{\alpha} \right]^{\frac{1}{1-\beta}}, \; \frac{2}{\epsilon} \ln \frac{1}{\delta} \right\}$$

examples suffice.

Example: convex region in plane

$X = R^2$, with n bits of precision.

$C = \{$convex regions defined by boundary segments$\}$.

Algorithm:

1. Find line with positives all on one side and as many negatives as possible on the other.
2. Throw out those negatives and repeat 1.
3. End result is a "decision list of halfplanes".

Analysis: $s = |$boundary segments in target$|$.

(The actual description length of the concept is $2sn$ since we're using n bits of precision)

Each line separates at least $1/s$ fraction of negatives remaining. So, after L lines, at most $m(1 - 1/s)^L$ negs remain. Therefore

$$L = 1 + s \ln m$$

lines suffice.

Converses?

Suppose all hypotheses in H are *independent* with respect to $c, \mathcal{D}$:

Pr(all inconsistent with sample from $c, \mathcal{D}$) = $\prod_{h \in H}$Pr(h is inconsistent with sample)

and suppose all $h \in H$ have error ϵ and $|H| = 1/(1-\epsilon)^m$.

Then, E[# consistent with m examples] = 1.

Prob[At least 1 is consistent]
$= 1 - (1 - \text{Prob}[h \text{ is consistent}])^{|H|}$
$\approx 1 - 1/e.$

Note: If expected number of events is small then Prob(at least 1) is always small. We want that if expected number is large then prob(at least 1) is large too. This requires independence.