3.1 PAC Learning (Cont'd)

In the last lecture, we covered:

- Axis Aligned Rectangles,
- Disjunctions.

Those algorithms suggests the concept of "consistency."

Observation 1 *Learn some concept class by outputting $h \in C$ consistent with a set of m examples.*

3.1.1 General Condition for PAC learning

Though we checked the PAC learnability of those algorithms, it's not an easy task to follow such ways for every single algorithm respectively. So the question is:

"Is there any general purpose condition to confirm the PAC learnability?"

Claim 1 *If $|C|$ is bounded and m is sufficiently large, then with probability larger than $1-\delta$, h is ϵ-accurate hypothesis.*

To see this, we need to bound the probability that the ouput hypothesis h has error larger than ϵ with respect to any distribution D to be less than δ.

Firstly, let's say that a hypothesis h is bad if it has error larger than ϵ. Then, for any fixed bad h, the probability that h is consistent with m examples is at most $(1-\epsilon)^m$. Since there are at most $|C|$ hypotheses, the probability of bad hypotheses consistent with all m examples is at most $|C| \cdot (1-\epsilon)^m$.

Therefore,

$$\begin{aligned} & |C| \cdot (1-\epsilon)^m \leq \delta \\ \Longrightarrow \quad & |C| \cdot e^{-\epsilon m} \leq \delta \\ \Longrightarrow \quad & m \geq \frac{1}{\epsilon} \log\left(\frac{|C|}{\delta}\right). \end{aligned}$$

3.1.2 Occam Algorithms

Definition 1 *An* (α, β)**-Occam algorithm** *is one that draws* m *examples from* D*, outputs a hypothesis so that*

$$size(h) \leq (n \cdot size(c))^{\alpha} \cdot m^{\beta},$$

where $\alpha > 0$*,* $0 \leq \beta < 1$*,* n *is length of the examples, and* c *is unknown concept. (The size of a concept should be clear from the context.)*

Example 1 *Assume* $m \gg n$ and we have a dataset $\{(e_i, l_i)\}$, $i = 1, \cdots, m$ where e_i is an example and l_i is the corresponding label. Then the number of required bits to describe the dataset is $m \cdot n + m$. If an algorithm compresses the set of labels as a function which can take an examples and outputs the label, and if the number of bits to describe the function is at most $m^{\beta} \cdot (n \cdot size(c))^{\alpha}$, then this algorithm is an (α, β)-Occam algorithm.

3.1.3 Occam's Razor Bound

Let's consider the number of examples which makes an Occam algorithm accurate enough. In other words, how large do we have to choose m such that h generalize well?

Again, let's say a hypothesis h is bad if the error is larger than ϵ with respect to a distribution D. The probability that a fixed bad h is consistent with m examples is still at most $(1-\epsilon)^m$. But the number of possible hypotheses is at most $3^{(n \cdot size(c))^{\alpha} \cdot m^{\beta}} \cdot (1-\epsilon)^m$. Therefore,

$$\begin{aligned}
& 3^{(n \cdot size(c))^{\alpha} \cdot m^{\beta}} \cdot (1-\epsilon)^m \leq \delta, \\
\Longrightarrow \quad & 3^{(n \cdot size(c))^{\alpha} \cdot m^{\beta}} \cdot e^{-\epsilon m} \leq \delta, \\
\Longrightarrow \quad & m \geq \frac{a}{\epsilon} \cdot (n \cdot size(c))^{\alpha} \cdot m^{\beta} + \frac{1}{\epsilon} \log\left(\frac{1}{\delta}\right).
\end{aligned}$$

for some constant a. And here, $(n \cdot size(c))^{\alpha}$ is usually $n^{O(1)}$.

3.1.4 PAC Learning for Decision List

Let's consider the PAC learnability of decision list.

Definition 2 *A node is "useful" if it classifies some examples in the dataset and is consistent with all others.*

We'll start up with the following Occam algorithm for decision list.

- Search for a useful node, put this node at the head of the decision list.
- Remove all examples from the set classified by this node.
- Repeat until the dataset becomes empty.

To obtain a sample size bound, apply Occam's Razor Bound. Since any decision list over n variables can be encoded in $O(n \log n)$ bits,

$$m \approx O\left(\frac{n \log n}{\epsilon} + \frac{1}{\epsilon} \log\left(\frac{1}{\delta}\right)\right).$$

* MB - winnow M.B. of $O(k \log n)$ for disjunctions of length k.

3.1.5 Set Cover Problem

Given a set $U = \{1, \cdots m\}$, let $S_1, \cdots, S_t$ be subsets of U such that $\cup_{i=1}^{t} S_i = U$. Then the problem is finding smallest collection of sets H so that $\cup_{T \in H} T \supseteq U$.

This problem is known to be NP-hard, but there exists an approximation algorithm outputs $O(OPT \cdot \log m)$ sets where OPT stands for the number of sets in an optimal covering.

- Initialize H.
- Add the largest cover T to H.
- Update each $S_i \leftarrow S_i \backslash T$.
- Repeat until all the elements are covered by H.

To see the computational complexity, let $U_i \subseteq U$ be the set of elements not covered after i steps of the algorithm. Then,

$$|U_{i+1}| \leq |U_i| - \frac{|U_i|}{OPT} = |U_i|\left(1 - \frac{1}{OPT}\right),$$

$$\Longrightarrow |U_i| \leq \left(1 - \frac{1}{OPT}\right)^i m.$$

Therefore,

$$\left(1 - \frac{1}{OPT}\right)^i m \leq e^{-\frac{1}{OPT} i} m \leq 1.$$

gives the bound.