CSE 513T Theory of Artificial Intelligence & Machine Learning January 29, 2015

Lecture 6

Lecturer: Brendan Juba *Scribe: Connor Basich*

Today's Topics

1. PAC-learning using Occam's Razor
2. Learning Decision Lists

Warm-up Problem

How many bits does it take to express a function definable by a conjunction on n attributes?

Counting the number of distinct functions,(forgetting about contradicting variables), identify the conjunctions with $\{0, 1, *\}^n$
$3^n + 1$ (i.e. the functions plus one contradiction).
$log_2(3^n + 1) \leq$ k bits $\rightarrow 2^k - 1$ distinct strings.
$3^n + 1 \leq 2^{K+1} - 1$
$k \geq log_2(3^n + 2) - 1$
So, k $\sim$ O(n).

Occam's Razor

"Entities should not be multiplied without necessity."
But, why should it lead to a "good" scientific theory (i.e. with predictive power)?
1) Need to fix notion of "simplicity" (otherwise it's entirely subjective).
2) Need enough data (but how much is enough?)

Today we will show that a formal version of Occam's Razor, in which "simplicity" is measured by the encoding length of the hypothesis, is correct.

Recall: We have been talking about

Consis$^B_{\mathcal{H}}$: given a labeled data set, find $h \in \mathcal{H}$ such that h is consistent with the data, and size(h) in bits $\leq B(n, m)$ where n is the number of attributes, and m is the number of examples in the data set.

An "Occam Algorithm" for a class $\mathcal{C}$ solves Consis$^B_{\mathcal{H}}$ when the data is labeled by $\mathcal{C}$

Theorem (**Occam′s Razor**) : There is a constant $b > 0$ such that for any concept class $\mathcal{C} = U_{n \in N}$, and any distribution D over X_n, if the number of examples m is such that:

$$B(n, m) \leq b\epsilon m - log\tfrac{1}{\delta} - 1$$
$$\Updownarrow$$
$$m \geq \tfrac{1}{\epsilon b}(B(n, m) + log\tfrac{1}{\delta} + 1)$$

Then, with probability $1 - \delta$, any solution to Consis$^B_{\mathcal{H}}$ on the data set satisfies:

$$Pr_{x \in D}[c(x) = h(x)] \geq 1 - \epsilon$$

Thus, any Occam Algorithm that runs in polynomial time in $n, \frac{1}{\epsilon}, \frac{1}{\delta}, size(c)$ is a PAC-learning algorithm for $\mathcal{C}$ as long as $B(n, m)$ is small enough to satisfy the relation.

Proof.
We'll say that $h \in \mathcal{H}$ is bad if $Pr_{x \in D}[c(x) \neq h(x)] > \epsilon$
What is the probability that a bad hypothesis is consistent with m number of examples drawn from D?
The probability that h is consistent with one example is $< 1 - \epsilon$.
The examples are independent, so probability that all m are consistent is $< (1-\epsilon)^m$
Since $m \geq \frac{1}{\epsilon b}(B(n,m) + log\frac{1}{\delta} + 1)$, then:

$$Pr < (1-\epsilon)^{\frac{1}{\epsilon}\frac{1}{b}(B(n,m)+log\frac{1}{\delta}+1)}$$
$$Pr < e^{-\frac{1}{b}(B(n,m)+log\frac{1}{\delta}+1)}$$
let $b = \frac{1}{ln2}$
$$Pr < 2^{-(B(n,m)+log\frac{1}{\delta}+1)} = \frac{\delta}{2^{B(n,m)+1}}$$

On $B(n,m)$ bits, there are $\leq 2^{B(n,m)+1}$ strings.
Hence, there are $\leq 2^{B(n,m)+1}$ $h \in \mathcal{H}$ that can be written in that many bits.

Bad events: some bad $h \in \mathcal{H}$ of length $\leq B(n,m)$ bits is consistent with the data set, and therefore could be output by the Occam Algorithm.

There are $\leq 2^{B(n,m)+1}$ bad events, one for each $h \in \mathcal{H}$ of length $\leq B(n,m)$
Union Bound PR[any bad $h \in \mathcal{H}$ of length $\leq B(n,m)$ is consistent with m examples]

$$\leq \sum_{h \text{ of length } \leq B(n,m)} Pr[h \text{ is consistent on m examples}]$$
$$\leq 2^{B(n,m)+1}\frac{\delta}{2^{B(n,m)+1}}$$
$$\leq \delta$$

So in sum, the probability that every solution to $\text{Consis}^B_{\mathcal{H}}$ satisfies $Pr_{x \in D}[c(x) = h(x)] \geq 1 - \epsilon$ is at least $1 - \delta$
□

First Application of Theorem

Improve analysis of Elimination Algorithm.
Recall that the Elimination Algorithm is for learning conjunctions. It starts with conjunctions on all possible literals, scans over the data set, and when it encounters an example labeled "1", it deletes the literals that are false on that example from h.
Does this algorithm solve $\text{Consis}^{2n}_{conj}$ when the data is labeled by a conjunction?

For the conjunction c actually labeling the data, initially all literals of c are in h (h contains all literals) and we only delete literals from h that would make h inconsistent with the label given by c.
These literals can't have been in c since then we would have c output 0, but since we are removing literals at all, $c = 1$.
So all these literals of c remain in h.
h must therefore never make a mistake on an example c labeled 0, so it's enough to just have h consistent with the examples where $c = 1$.
We find that running the Elimination Algorithm on a data set of size

$$m \geq \frac{1}{\epsilon ln2}(\lceil nlog_2 3 \rceil + log_2\frac{1}{\delta} + 1)$$

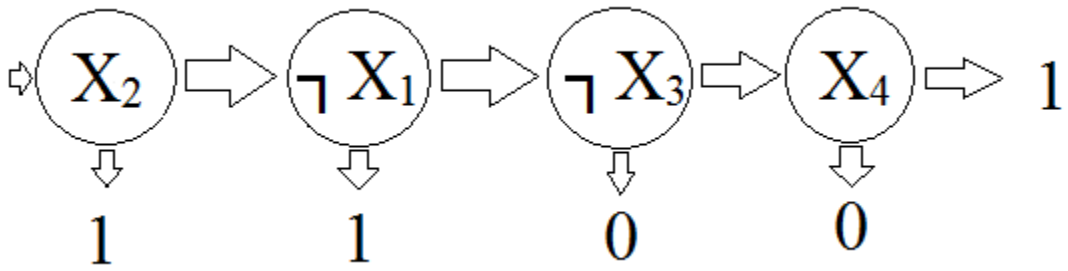
provides an output which, with probability $1 - \delta$ over the data, satisfies

$$Pr[c(x) = h(x)] \geq 1 - \epsilon$$

Compare: previously we needed $\sim$O($\frac{n}{\epsilon}(log(n) + log\frac{1}{\delta})$). In essence, we got rid of the log(n) using Occam's Razor.

Second Application of Theorem

Learning Decision Lists.
A decision list is a decision tree in which every internal node has at least one leaf as a child.



$x = (1000) \rightarrow 0$
$x = (1010) \rightarrow 1$

Theorem: There is an efficient (and proper) PAC-learning algorithm for decision lists.

Need to show an algorithm for finding a decision list that is consistent with the data, given that the data was actually labeled by a decision list.
How many bits does it take to write down a decision list on n attributes?
$\sim$O$(nlog(n))$ as we only need to test a literal once.

As long as the number of examples $\geq \Omega(\frac{1}{\epsilon}(nlog(n) + log\frac{1}{\delta}))$, we are guaranteed by Occam's Razor that our decision list has $Pr_{x \in D}[c(x) = h(x)] \geq 1 - \epsilon$ with probability $1 - \delta$.

For a literal l and a set of examples S, we will let $S_l = \}(x, b) \in S : l(x) = 1\}$. We'll say that l is useful for S if
(i) S_l is nonempty
(ii) Every $(x, b) \in S_l$ has the same label.

FindList(S)
Let h be the empty list.
Until every $(x, b) \in S_l$ has the same label
iterate over literals l until S_l is useful
for the label b of these examples in S_l, we add to h
$S \leftarrow S - S_l$
For the label b of S, we add the terminal leaf $\rightarrow b$ to h
Return (h)

Next time: We will show that there is always a useful literal. Correctness is then easy.