6.858/18.428 Machine Learning	**Lecture 6 : September 28, 1994**
Lecturer: Ron Rivest	*Scribe: Marcos Kiwi*

6.1 Outline

- Comments on learning 3-term DNF by 3-CNF.
- Learning k-DL.
- Hierarchies.
- Occam's Razor (general case).
- Learning conjunction with few literals.

6.2 Comments on learning 3-term DNF by 3-CNF.

Last lecture we showed that 3-term DNF could be rewritten as 3-CNF. Since the concept class 3-CNF is efficiently PAC learnable, we concluded that 3-term DNF are also efficiently PAC learnable using 3-CNF.

In general, the following holds:

Theorem 1 *For any constant $k \geq 2$, the concept class of k-*term DNF *is efficiently PAC learnable using k-*CNF.

In the next section we will introduce a concept class that contains both the concept classes k-CNF and k-DNF, which can still be efficiently learned.

6.3 Learning k-Decision Lists.

A decision list is basically an ordered sequence of if-then-else statements. The sequence of if-then-else conditions are tested in order, and the answer associated to the first satisfied condition is output (see Figure 6.1).

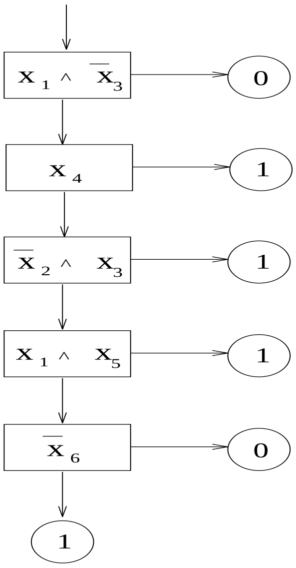

Figure 6.1: A 2-decision list.

More precisely,

Definition 1 *A k-*decision list *over the variables $x_1, \ldots, x_n$ is an ordered sequence $L = (c_1, b_1), \ldots, (c_l, b_l)$ and a bit b, in which each c_i is a conjunction of k literals over $x_1, \ldots, x_n$. The bit b is called the* default *value, and b_i is referred to as the bit* associated *with condition c_i. For any input $x \in \{0,1\}^n$, $L(x)$ is defined to be the bit b_j, where j is the smallest index satisfying $c_j(x) = 1$; if no such index exists, then $L(x) = b$.*

*We denote by k-*DL *the class of concepts that can be represented by a k-decision list.*

We now consider the issue of what is the expressive power of k-decision lists.

Lemma 1

$$k\text{-DNF} \cup k\text{-CNF} \subseteq k\text{-DL}.$$

Proof: First notice that if a concept c can be represented as a k-decision list, then, so can $\neg c$ (just complement the values b_i and b of the decision list representing the concept c). Thus, it suffices to show that k-DNF $\subseteq$ k-DL. But a k-DNF can be represented as a k-decision list by simply choosing an arbitrary ordering in which to evaluate the terms of the k-DNF, setting all the b_i's to 1 and the default to 0. ■

The containment of Lemma 1 is strict. That is, there exist functions that can be represented by a k-decision list, but not by either k-DNF or k-CNF.

The main result of this section is,

Theorem 2 *For any constant $k \geq 1$, the representation class of k-decision lists is efficiently PAC learnable.*

Proof: Consider an input sample S, where $|S| = m$ and S is consistent with some k-decision list. We build a decision list consistent with S as follows; we first find a conjunction $c(\cdot)$ of k literals (by going over all the $\Theta(n^k)$ conjunctions of k literals if necessary) such that the set $S_c = \{x \in S : c(x) = 1\}$ is both non-empty, and contains only positive examples, or negative examples. We call such conjunction $c(\cdot)$ a *good* conjunction. If $S_c = S$ then we set the default bit to 1 if S_c contains only positive examples, or to 0 if S_c contains only negative examples. Otherwise we add the conjunction $c(\cdot)$ (with the associated bit 1 if S_c contains only positive examples, and the associated bit 0 if S_c contains only negative examples) as the last condition in the current hypothesis decision list, update S to $S - S_c$, and iterate until $S = \phi$.

Clearly all the examples in the initial sample S are correctly classified by the hypothesis k-decision list built by the process described above. Also, the process will always succeed in finding a consistent hypothesis, because it succeeds at each iteration in correctly setting the value of the default bit or in finding a good conjunction. Note that the first conjunction of the target k-decision list is a good conjunction.

Since any k-decision list on n variables can be encoded in $\Theta(kn^k \log n)$ bits, we can apply the simple case of Occam's Razor to obtain a sample size bound of $m = \Omega((1/\epsilon)(\log(1/\delta) + kn^k \log n))$ for PAC learning. Since the described procedure runs in time polynomial in m, we get efficient PAC learning. ■

6.4 Hierarchies

In this section we review our understanding of the relationships between the representation classes that we have discussed. First, we need the following definitions:

Definition 2 *A* boolean circuit *over* $x_1, \ldots, x_n \in \{0,1\}$ *of size* N*, is a directed acyclic graph on* N *vertices, in which each vertex has indegree either* $0, 1$ *or* 2*, and unbounded outdegree. Each vertex of indegree* 0 *is labeled with one of the* input variables $x_1, \ldots, x_n$. *Each vertex of indegree* 1 *by the symbol* $\neg$*, and each vertex of indegree* 2 *is labeled by one of the symbols* $\vee$ *and* $\wedge$*. There is a single designated* output vertex *of outdegree* 0*. When the input variables are assigned values the graph computes a boolean function in the obvious way.*

Definition 3 *The class* Poly-size Circuits*, is the concept class* $\mathcal{C}$ *in which each concept in* $\mathcal{C}_n$ *is computed by a boolean circuit of size at most* $p(n)$*, where* $p(\cdot)$ *is some fixed polynomial.*

Our knowledge about the relationship between the concept classes that we have so far defined can be summarized as follows:

$$\begin{array}{ccccccccccc} \text{Conjunc.} & \subseteq & k\text{-term DNF} & \subseteq & k\text{-CNF} & \subseteq & k\text{-DL} & \subseteq & \text{Poly-size Circuits} & \subseteq & \text{DNF} \\ (easy) & & (hard) & & (easy) & & (easy) & & (hard) & & \end{array}$$

6.5 Occam's Razor (General Case).

The purpose of this section is to formalize the notion of Occam learning. In Occam learning, an algorithm takes a sample $S = \{< x_1, c(x_1) >, \ldots, < x_m, c(x_m) >\}$, and outputs a "succinct" hypothesis h that is consistent with the sample, *i.e.* $h(x_i) = c(x_i)$ for each i, and $size(h)$ is a slowly growing function of n, $size(c)$, and m. Formally,

Definition 4 *For constants* $\alpha \geq 0$ *and* $0 \leq \beta < 1$*.* L *is an* (α, β)*-Occam algorithm for the concept class* $\mathcal{C}$ *using* $\mathcal{H}$ *if on input sample* S *of cardinality* m *consistent with* $c \in \mathcal{C}_n$*,* L *outputs a hypothesis* $h \in \mathcal{H}$ *such that:*

- *h is consistent with S.*
- $size(h) \leq (n \cdot size(c))^{\alpha} m^{\beta}$.

We say that L *is an efficient* (α, β)*-Occam algorithm if its running time is bounded by a polynomial in* n*,* m *and* $size(c)$*.*

Observe that the definition of an Occam algorithm captures the notion of succinctness by requiring that for $m \gg n$, the size of the output is effectively $O(m^{\beta})$ for some $\beta < 1$, and thus $o(m)$. In other words, the hypothesis h is a succinct representation of the bits $c(x_1), \ldots, c(x_m)$.

Our main goal is to prove the following theorem, which shows that efficient Occam algorithms are also efficient PAC learning algorithms.

Theorem 3 (Occam's Razor) *Let L be an efficient (α, β)-Occam algorithm for $\mathcal{C}$ using $\mathcal{H}$. Let $\mathcal{D}$ be the target distribution over the instance space X, let $c \in \mathcal{C}_n$ be the target concept, and $\epsilon > 0$, $\delta \leq 1$. Then there is a constant $a > 0$ such that if L is given as input a random sample S of m examples drawn from $EX(c, \mathcal{D})$, where m satisfies*

$$m \geq a\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \left(\frac{1}{\epsilon}(n \cdot size(c))^{\alpha}\right)^{\frac{1}{1-\beta}}\right)$$

then, with probability at least $1-\delta$ the output h of L satisfies $error(h) \leq \epsilon$. Moreover, L runs in time polynomial in n, $size(c)$, $1/\epsilon$ and $1/\delta$.

The proof of Occam's Razor is a consequence of the result concerning the simple case of Occam's Razor covered in the preceding lecture which we state below in a slightly more general version.

Theorem 4 (Occam's Razor, Cardinality version) *Let $\mathcal{C}$ be a concept class and $\mathcal{H}$ a representation class. Let L be an algorithm such that for any n and any $c \in \mathcal{C}_n$, if L is given as input a sample S of m labeled examples of c, then L runs in time polynomial in n, m and $size(c)$, and outputs an $h \in \mathcal{H}_{n,m}$ that is consistent with S. Then, there is a constant $b > 0$ such that for any n, any distribution $\mathcal{D}$ over X_n, and any target concept $c \in \mathcal{C}_n$, if L is given as input a random sample from $EX(c, \mathcal{D})$ of m examples, where $|\mathcal{H}_{n,m}|$ satisfies*

$$m \geq \frac{1}{b\epsilon}\left(\log|\mathcal{H}_{n,m}| + \log\frac{1}{\delta}\right)$$

then, L is guaranteed to find a hypothesis $h \in \mathcal{H}_n$ that with probability at least $1-\delta$ satisfies $error(h) \leq \epsilon$.

We now prove Theorem 3.

Proof: Let $\mathcal{H}_{n,m}$ be the class of possible output hypothesis that the algorithm L can output on sample input S, where $|S| = m$. By Theorem 4 the output h of L is such that $error(h) \leq \epsilon$ with confidence $1-\delta$ given that

$$m \geq \frac{1}{b\epsilon}\left(\log|\mathcal{H}_{n,m}| + \log\frac{1}{\delta}\right).$$

The previous inequality is implied by the following inequality, if we require that for a sufficiently large constant a:

$$m \geq a\left(\frac{1}{\epsilon}\log\frac{1}{\delta} + \left(\frac{1}{\epsilon}(n \cdot size(c))^{\alpha}\right)^{\frac{1}{1-\beta}}\right)$$

and recalling that since L is an (α, β)-Occam algorithm, $\log|\mathcal{H}_{n,m}| \leq (n \cdot size(c))^{\alpha} m^{\beta}$.
■

6.6 Learning Conjunctions with few Literals.

Although we already have an efficient PAC algorithm for learning boolean conjunctions, we can imagine a situation in which the target conjunction of n literals, c, that we desire to learn, is such that $size(c) \ll n$. This situation could represent, for example, the case in which an object is described by referring to only a few of its attributes out of a long list of potential attributes. Our intention is to give an algorithm for learning conjunctions with few literals (relative to the total number of literals). The new algorithm will make use of the negative examples, as opposed to our previous algorithm for learning conjunctions.

The algorithm will use a well known approximation algorithm for the Set Cover Problem. Below we describe both the Set Cover Problem, and its approximation algorithm.

Set Cover Problem

- **Input:** A collection $\mathcal{S}$ of subsets of $U = \{1, \ldots, m\}$.
- **Output:** A collection $\mathcal{S}' \subseteq \mathcal{S}$, which covers U, *i.e.* $\bigcup_{s \in \mathcal{S}'} s = U$, and such that $|\mathcal{S}'|$ is minimized.

We denote by $opt(\mathcal{S})$ the number of sets in a minimum cardinality cover. It is well known that the Set Cover Problem is an NP-hard problem. However, there is an efficient greedy heuristic that is guaranteed to find a cover of cardinality at most $opt(\mathcal{S}) \cdot \ln m$.

The latter heuristics initializes $\mathcal{S}'$ to the empty collection. It then updates $\mathcal{S}'$ by adding the set $s \in \mathcal{S} - \mathcal{S}'$ which covers the largest number of elements of U that remain uncovered. The process is then repeated until all the elements of U are covered by $\mathcal{S}'$.

Let u_i denote the number of elements that remain uncovered after i iterations of the above described process. Clearly, $u_0 = m$. Furthermore,

$$u_i \le u_{i-1}\left(1 - \frac{1}{opt(\mathcal{S})}\right) \le m\left(1 - \frac{1}{opt(\mathcal{S})}\right)^i \le m \cdot \exp\left(-i/opt(\mathcal{S})\right).$$

The first inequality is a consequence of the fact that whatever the subset $U' \subseteq U$, since U has a set cover of size $opt(S)$, then there must be a set in S that covers at least a $1/opt(S)$ fraction of U'. Hence, if $i > opt(\mathcal{S}) \cdot \ln m$, then $u_i < 1$. That is the greedy heuristics outputs a cover of U of cardinality at most $opt(\mathcal{S}) \cdot \ln m$ as we wanted to prove.[1]

We now return to the problem of learning conjunctions with few literals. We shall give an Occam algorithm for this problem. This algorithm, given a sample S of size m, first applies the algorithm for learning conjunctions that we have already seen, and outputs a hypothesis h. Recall that h has the property that it is consistent with the positive examples. The new algorithm for learning conjunctions will now make use of the negative examples in S to exclude literals from h and produce a hypothesis h'. Thus, h' will be consistent with the positive examples. In particular, h' is built as follows;

- For each literal z of h, let $S_z = \{< x, 0 > \in S : z(x) = 0\}$, (*i.e.* S_z is the set of negative examples for which the value of z is 0).
- Then, find a collection of z's (using the greedy heuristics described above) such that the z's are literals of h and the S_z's cover the set of negative examples of S.
- Let h' be the conjunction of all such literals.

Observe that h' is consistent with the examples of S. Also, note that among the S_z's such that z is a literal of h, a cover of $size(c)$ sets exists. Thus, h' has at most $size(c) \log m$ literals. Hence, our hypothesis class $\mathcal{H}_{n,m}$ is the set of all conjunctions of at most $size(c) \log m$ literals, and $|\mathcal{H}_{n,m}| \le 2^{size(c)\log m \log n}$. It follows that for some sufficiently large constant c_1 we have that

$$m \ge \frac{c_1}{\epsilon}\left(\log\frac{1}{\delta} + size(c)\log n \log(size(c)\log n)\right)$$

[1] A tighter analysis of the described greedy heuristics shows that it actually is guaranteed to find a cover of cardinality at most $opt(S) \cdot H_{\max\{|s|:s\in S\}}$ where $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$, (see [1]).

implies that for $b > 0$

$$m \geq \frac{1}{b\epsilon}\left(\log\frac{1}{\delta} + size(c)\log m \log n\right) \geq \frac{1}{b\epsilon}\left(\log\frac{1}{\delta} + \log|\mathcal{H}_{n,m}|\right).$$

Thus, the sample size bound of Theorem 4 is insured and PAC learning implied.

References

[1] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, 1990.