A property of the hypothesis space

Aim: a property of the hypothesis space $\mathcal{H}$ that ensures that every consistent algorithm $\mathcal{L}$ that learns a hypothesis $H \in \mathcal{H}$ is PAC. $\mathcal{L}$ is consistent if given any training sample $s$, $\mathcal{L}$ produces a hypothesis that is consistent with $s$. Let $\mathcal{H}[s]$ the set of hypothesis consistent with $s$. 
Let $T$ be a target concept. The set of $\epsilon$-bad hypotheses is

$$B_\epsilon = \{ H \in \mathcal{H} \mid P(T \oplus H) \geq \epsilon \}.$$ 

A consistent $\mathcal{L}$ produces an output in $\mathcal{H}[s]$ starting from $s$ and the PAC property requires that it is unlikely that $H = \mathcal{L}[s]$ is $\epsilon$-bad.
A hypothesis space $\mathcal{H}$ is \textbf{potentially learnable} if, given real numbers $\delta$ and $\epsilon$, there is a positive integer $m_0(\delta, \epsilon)$ such that whenever $m \geq m_0(\delta, \epsilon)$

$$P(s \in S(m, T) \mid H[s] \cap B_\epsilon = \emptyset) > 1 - \delta,$$

for any probability distribution $P$. 
Theorem

If $\mathcal{H}$ is potentially learnable and $\mathcal{L}$ is a consistent learning algorithm, then $\mathcal{L}$ is PAC.

Proof: the proof is immediate because if $\mathcal{L}$ is consistent, $\mathcal{L}[s] \in \mathcal{H}[s]$. Thus, the condition $\mathcal{H}[s] \cap B_\epsilon = \emptyset$ implies that the error of $\mathcal{L}[s]$ is less than $\epsilon$, as required for PAC learning.
Theorem

Every finite hypothesis space is potentially learnable.

Proof: Suppose that $\mathcal{H}$ is a finite hypothesis space and let $\delta, \epsilon, C$ and $P$ are given. We prove that $P(\mathcal{H}[s] \cap B_\epsilon \neq \emptyset)$ can be made less than $\delta$ by choosing $m$ sufficiently large.

By the definition of $B_\epsilon$ it follows that for every $H \in B_\epsilon$:

$$P(x \mid H(x) = C(x)) \leq 1 - \epsilon.$$ 

Thus,

$$P(s \mid H(x_i) = C(x_i) \text{ for } 1 \leq i \leq m) \leq (1 - \epsilon)^m.$$ 

This is the probability that one $\epsilon$-bad hypothesis is in $\mathcal{H}[s]$. 
Proof (cont’d)

There is some $\epsilon$-bad hypothesis in $\mathcal{H}[s]$ iff there exists $s$ such that $\mathcal{H}[s] \cap B_\epsilon \neq \emptyset$. Therefore, the probability of the existence of such a hypothesis is $P(\{s \mid \mathcal{H}[s] \cap B_\epsilon\}$ is less than $|\mathcal{H}|(1 - \epsilon)^m$.

To have $|\mathcal{H}|(1 - \epsilon)^m < \delta$ we must have

$$|\mathcal{H}|(1 - \epsilon)^m < |\mathcal{H}|e^{-\epsilon m} < |\mathcal{H}|e^{\ln \frac{\delta}{|\mathcal{H}|}}$$

because $\delta = |\mathcal{H}|e^{\ln \frac{\delta}{|\mathcal{H}|}}$. Thus,

$$-\epsilon m < \ln \frac{\delta}{|\mathcal{H}|},$$

so $m > \frac{1}{\epsilon} \ln \frac{\delta}{|\mathcal{H}|}$, or

$$m \geq \left\lceil \frac{1}{\epsilon} \ln \frac{\delta}{|\mathcal{H}|} \right\rceil.$$
Observations

- the algorithm for learning monomials is PAC (hypothesis space has $3^n$ elements);
- practical limitations exists even for finite spaces; for example, there are $2^{2^n}$ Boolean functions, so the bound for the sample length is
  \[
  \left\lceil \frac{2^n}{\epsilon} \ln \frac{2}{\delta} \right\rceil;
  \]
- even for applications of moderate size (say $n = 50$) this is enormous!
A decision list is a sequence of pairs \( L = (((f_1, c_1), \ldots, f_r), (f_r, c_r)) \) and a bit \( c \), where \( f_i : \{0, 1\}^n \rightarrow \{0, 1\} \) for \( 1 \leq i \leq n \) and \( c_i \in \{0, 1\} \). The Boolean function defined by \( L \) is evaluated as shown below:

if \( f_1(\mathbf{x}) = 1 \) then set \( f(\mathbf{x}) = c_1 \)
else if \( f_2(\mathbf{x}) = 1 \) then set \( f(\mathbf{x}) = c_2 \)
\[ \vdots \]
else if \( f_r(\mathbf{x}) = 1 \) then set \( f(\mathbf{x}) = c_r \)
else set \( f(\mathbf{x}) = c \).

Given \( \mathbf{x} \in \{0, 1\}^n \) we evaluate \( f_1(\mathbf{x}) \). If \( f_1(\mathbf{x}) = 1 \), \( f(\mathbf{x}) \) has the value \( c_1 \). Otherwise, \( f_2(\mathbf{x}) \) is evaluated, etc.
If $K = (f_1, \ldots, f_r)$ is a sequence of Boolean functions we denote by $DL(K)$ the set of decision lists on $K$.

The value of the function defined by a decision list $((f_1, c_1), \ldots, (f_r, c_r))$, $c$ is

$$f(x) = \begin{cases} 
  c_j & \text{if } j = \min \{ i \mid f_i(x) = 1 \} \text{ exists} \\
  c & \text{otherwise.}
\end{cases}$$

There is no loss of generality in assuming that all functions $f_i$ are distinct, so the length of a decision list is at most $|K|$. 
Example

If $K = \text{MON}_{3,2}$, the set of monomials of length at most 2 in 3 variables, then the decision list

$$((u_2, 1), (u_1 \bar{u}_3, 0), (\bar{u}_1, 1)), 0$$

operates as follows:

- those examples for which $u_2$ is satisfied are assigned 1: 010, 011, 110, 111;
- the examples for which $u_1 \bar{u}_3$ is satisfied are assigned 0: the only remaining example is 100;
- the remaining examples for which $\bar{u}_1$ is satisfied are assigned 1: 000, 011; the remaining example, 101 is assigned 0.
Example

Let $K = (f_1, f_2)$. The decision list $((f_1, 1), (f_2, 1)), 0$ defines the function $f_1 \lor f_2$. 
Algorithm 2.1: A Consistent Algorithm for Decision Lists

Data: A sample $s = ((x_1, b_1), \ldots, (x_m, b_m))$, a sequence of Boolean functions $K = (g_1, \ldots, g_r)$ and a training sample

Result: A decision list

1. let $I = \{1, \ldots, m\}$;
2. let $j = 1$;
3. repeat
4.   if for all $i \in I$, $g_j(x_i) = 1$ implies $b_i = c$ for a fixed bit $c$ then
5.     select $(g_j, c)$ to include in the decision list;
6.     delete from $I$ all $i$ for which $g_j(x_i) = 1$;
7.     $j = 1$;
8.   else
9.     $j = j+1$;
10. end
11. until $I = \emptyset$;
12. return decision list;
Example

\( K = M_{5,2} \) is listed in lexicographic order based on the ordering

\[ u_1, \bar{u}_1, u_2, \bar{u}_2, u_3, \bar{u}_3, u_4, \bar{u}_4, u_5, \bar{u}_5. \]

The first few entries in the list are

\[ (), (u_1), (u_1u_2), (u_1\bar{u}_2), (u_1u_3), \ldots \]

Note that \((u_1\bar{u}_1)\) is not included.

Training sample \( s \) is:

\[ (x_1 = 10000, b_1 = 0), (x_2 = 01110, b_2 = 0), (x_3 = 11000, b_3 = 0), \\
(x_4 = 10101, b_4 = 1), (x_5 = 01100, b_5 = 1), (x_6 = 10111, b_6 = 1). \]
\[(x_1 = 10000, b_1 = 0), (x_2 = 01110, b_2 = 0), (x_3 = 11000, b_3 = 0),
(x_4 = 10101, b_4 = 1), (x_5 = 01100, b_5 = 1), (x_6 = 10111, b_6 = 1)\].

\[l = \{1, 2, 3, 4, 5, 6\}\] ( )
\[(u_1, 0)\] ( )
\[(u_1 u_2, 0)\] ( )
\[(u_1 \bar{u}_2)\] ( )
\[(u_1 u_3, 1)\] ( )
\[(u_1 \bar{u}_3)\] ( )
\[.\]
\[.\]
\[(u_1, 0)\] ( )
\[.\]
\[.\]
\[.\]
\[(\bar{u}_1 u_4, 0)\] ( )
\[() , 1\] ( )

no: all examples satisfy but some have 0 and others 1
no: both \(x_1\) and \(x_4\) satisfy it but have distinct \(b_i\)s
yes: this is satisfied only by \(x_3\), so add \((u_1 u_2, 0)\) and delete 3
no
yes: delete \(x_4\) and \(x_6\) and add \((u_1 u_3, 1)\)
no:
yes: delete \(x_1\) and add \((u_1, 0)\)
.
yes: delete \(x_3\) and add \((\bar{u}_1 u_4, 0)\)
yes: delete \(x_5\) and add 0
The resulting decision list:

$$\left((u_1 u_2), 0\right), \left((u_1 u_3), 1\right), \left((u_1), 0\right), \left(\bar{u}_1 u_4), 0\right), \left((), 0\right), 0$$

Claim: when we are given a sample $s$ for a target concept in $DL(K)$, then there is always a pair $(g, c)$ which has the required properties.
Theorem

Let $K$ be a sequence of Boolean function that contains the constant function of 1. If $f \in DL(K)$ and $S$ is a finite sample. There exists $g \in K$ and $c \in \{0, 1\}$ such that

- the set $S^g = \{x \in S \mid g(x) = 1\}$ is not empty;
- for all $x \in S^g$, $f(x) = c$. 
Proof: Since \( f \in DL(K) \) there is a representation of \( f \) as a decision list \(((f_1, c_1), \ldots, (f_r, c_r)), c\).

If \( f_i(x) = 0 \) for all \( x \) in \( S \) and all \( i, 1 \leq i \leq r \), then all examples of \( S \) are negative examples of \( f \). In this case we take \( g \) to be the constant function \( 1 \) and \( c = 0 \).

If there is \( i \) such that \( \{x \mid f_i(x) = 1\} \neq \emptyset \) let \( q = \min \{i \mid f_i(x) = 1\} \).

Then, \( f(x) = c_q \) for all \( x \) such that \( f_q(x) = 1 \). Select \( g = f_q \) and \( c = c_q \).

Thus, given a training example for \( f \in DL(K) \), there is a suitable choice of a pair \((g, c)\) for the first term in the decision list.

We followed here the paper [2] and the monograph [1].
M. Anthony and N. Biggs. 
*Computational Learning Theory.*  

R. L. Rivest. 
Learning decision lists. 