

# Agnostic PAC Learning Decision Lists is Hard

K.S. Ng

Computer Sciences Laboratory,  
Research School of Information Sciences and Engineering,  
The Australian National University

Draft, April 6, 2005\*

## 1 Introduction

In this paper, we consider the computational complexity of agnostic PAC learning the class of decision lists introduced by [4].

## 2 Basic Concepts and Notations

### Agnostic PAC Learning

Let  $X$  be the set of individuals and  $H$  a set of predicates over  $X$ . A learning algorithm  $L$  is said to be a *agnostic PAC learning algorithm* for  $H$  if it satisfies the following: given any  $\epsilon, \delta \in (0, 1)$ , there is an integer  $m(\epsilon, \delta)$  such that for all  $m \geq m(\epsilon, \delta)$ , for any  $t \in H$  and any probability distribution  $\mu$  on  $X \times \{0, 1\}$ , with probability at least  $1 - \delta$ , given a sample of size  $m$  drawn independently according to  $\mu$ , the error of the hypothesis  $h \in H$  output by  $L$  with respect to  $\mu$  defined by

$$er_{\mu}(h) = \mu\{(x, y) \in X \times \{0, 1\} : h(x) \neq y\}$$

is less than  $\epsilon$ . The number  $m(\epsilon, \delta)$  is called the sample complexity of learning  $H$ .

The algorithm  $L$  is said to be an *efficient PAC learning algorithm* if, in addition to the above, it runs in time polynomial in  $m, 1/\epsilon, 1/\delta$  and the encoding length of the target function  $size(t)$ .

The correct (and, in fact, only) strategy in the agnostic PAC setting is simple: find the  $h \in H$  that achieves the lowest empirical error on the training examples. See, for details, [2, Chap. 23].

### Decision Lists

Suppose  $n \in \mathbb{N}$ . Let  $\mathcal{G}_n = \{g_{i,j} : 1 \leq i \leq n, j \in \{0, 1\}\}$  be the class of functions where each

$$g_{i,j} : \{0, 1\}^n \rightarrow \{0, 1\}$$

is defined by

$$g_{i,j}(x) = \begin{cases} 1 & \text{if } x_i = j \\ 0 & \text{otherwise} \end{cases}.$$

A decision list over  $\{0, 1\}^n$  is a list  $L$  of pairs  $(f_1, v_1), \dots, (f_r, v_r)$  where each  $f_j$  is in  $\mathcal{G}_n$  ( $1 \leq j < r$ ), each  $v_i$  is in  $\{0, 1\}$  ( $1 \leq i \leq r$ ), and the last function  $f_r$  is the constant function  $\top$  that evaluates every  $x \in \{0, 1\}^n$  to 1.  $L$  defines a boolean function as follows: for any instance  $x \in \{0, 1\}^n$ ,  $L(x)$  is defined to be  $v_j$ , where  $j$  is the least index such that  $f_j(x) = 1$ .

---

\*Document first created 12 March 2005.

**Definition 1.** We define  $DL(n)$  to be the set of all boolean functions from  $\{0, 1\}^n$  to  $\{0, 1\}$  that can be represented as a decision list.

**Definition 2.** We define  $LT(n)$  to be the set of all boolean functions from  $\{0, 1\}^n$  to  $\{0, 1\}$  that can be represented in the form

$$f(x) = \text{sgn}\left(\sum_{i=1}^n w_i x_i + b\right)$$

where  $w_i, b \in \mathbb{R}$  and  $\text{sgn} : \mathbb{R} \rightarrow \{0, 1\}$  has the following definition:  $\text{sgn}(x) = 1$  iff  $x \geq 0$ .

**Lemma 3.**  $DL(n) \subseteq LT(n)$ .

*Proof.* We repeat a proof from [1] here. Consider an arbitrary  $f(\mathbf{x}) \in 1\text{-}DL(n)$ . We show that  $f(\mathbf{x})$  is equivalent to some  $g(\mathbf{x}) = \text{sgn}(\langle \mathbf{w} \cdot \mathbf{x} \rangle - b) \in LT(n)$ , proceeding by induction on the number of pairs  $n$  in  $f$ . When  $n = 1$ ,  $f$  is either  $(\top, 0)$  or  $(\top, 1)$ . In the first case, putting  $\mathbf{w} = \mathbf{0}$  and  $b = 1$  suffices. In the second case, we put  $\mathbf{w} = \mathbf{0}$  and  $b = -1$ . Consider now the case when  $n = r + 1$  for some  $r \in \mathbb{N}$ . That is,  $f = (f_1, v_1), (f_2, v_2), \dots, (f_{r+1}, v_{r+1})$ . By the inductive hypothesis, the sub-list  $(f_2, v_2), \dots, (f_{r+1}, v_{r+1})$  can be represented as a threshold function  $\text{sgn}(\langle \mathbf{w}' \cdot \mathbf{x} \rangle - b')$ . The following are four possible configurations for  $(f_1, v_1)$ :

$$(g_{i,1}, 1), (g_{i,1}, 0), (g_{i,0}, 1), (g_{i,0}, 0).$$

We'll assume that  $\mathbf{w}$  and  $b$  can be expressed in the following forms:  $\mathbf{w} = \mathbf{w}' + M\mathbf{e}_i$  and  $b = b' + k$  for some  $M$  and  $k$ , where  $\mathbf{e}_i$  is the vector that has 1 at the  $i$ -th position, and 0 everywhere else. The task is then to find the values of  $M$  and  $k$  for each of the four cases. We'll consider the first case here. Given an arbitrary input  $\mathbf{x}$ , if  $\mathbf{x}_i = 0$ , then we want

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle - b = \langle \mathbf{w}' \cdot \mathbf{x} \rangle - b'.$$

Since  $\mathbf{x}_i = 0$ ,  $\langle \mathbf{w} \cdot \mathbf{x} \rangle = \langle \mathbf{w}' \cdot \mathbf{x} \rangle$ , giving us the condition  $b = b'$ . That is,  $k = 0$ . If  $\mathbf{x}_i = 1$ , we need  $\langle \mathbf{w} \cdot \mathbf{x} \rangle - b \geq 0$ , giving the condition  $M + \langle \mathbf{w}' \cdot \mathbf{x} \rangle - b' \geq 0$ . Denoting by  $\|\mathbf{w}'\|_1$  the 1-norm of  $\mathbf{w}'$  and noting that  $-\|\mathbf{w}'\|_1 \leq \langle \mathbf{w}' \cdot \mathbf{x} \rangle \leq \|\mathbf{w}'\|_1$ , it is sufficient to put  $M = \|\mathbf{w}'\|_1 + |b'|$  here. Proceeding in a similar fashion for the remaining cases, we get  $M = -(\|\mathbf{w}'\|_1 + |b'| + 1)$ ,  $k = 0$  for the second case;  $M = -(\|\mathbf{w}'\|_1 + |b'|)$ ,  $k = -M$  for the third case; and  $M = \|\mathbf{w}'\|_1 + |b'| + 1$ ,  $k = M$  for the last case.  $\square$

### 3 Finding the Most Accurate Decision List is Hard

We adapt, with corrections, the proof of Theorem 24.2 in [2] here.

#### The DL-FIT Problem

The DL-FIT problem is as follows.

DL-FIT

Instance:  $z \in (\{0, 1\}^n \times \{0, 1\})^m$  and an integer  $k$  between 1 and  $m$ .

Question: Is there  $h \in DL(n)$  such that  $\hat{e}r(h, z) \leq k/m$ ?

Here,  $\hat{e}r(h, z) = |\{(x, y) \in z : h(x) \neq y\}|/m$ .

#### The Vertex-Cover Problem

A graph  $G = (V, E)$  consists of a set  $V$  of vertices and a set  $E$  of edges. We assume that each vertex in  $V$  is labelled with a number from  $\{1, 2, \dots, |V|\}$  and an edge in  $E$  connecting vertex  $i$  and vertex  $j$  is denoted  $ij$ . A vertex cover of a graph  $G = (V, E)$  is a set  $U \subseteq V$  of vertices such that for every edge  $ij$  in  $E$ , at least one of the vertices  $i, j$  belongs to  $U$ . The following VERTEX-COVER problem is known to be NP-hard [3].

VERTEX-COVER

Instance: A graph  $G = (V, E)$  and an integer  $k \leq |V|$ .

Question: Is there a vertex cover  $U \subseteq V$  such that  $|U| \leq k$ ?

### Reduction from Vertex-Cover to DL-Fit

We now show that given an instance of VERTEX-COVER, we can construct (in polynomial time) an instance of DL-FIT (of a size polynomially related to that of the instance of VERTEX-COVER) in such a way that the answer to the constructed DL-FIT problem is the same as the answer to the original VERTEX-COVER problem.

Consider an instance  $(G = (V, E), k)$  of VERTEX-COVER where  $|V| = n$  and  $|E| = r$ . The size of the instance is  $\Omega(r + n)$ . We construct  $z(G) \in (\{0, 1\}^n \times \{0, 1\})^{r+n}$  as follows. For any two integers  $i, j$  between 1 and  $n$ , let  $e_{i,j}$  denote the binary vector of length  $n$  with ones in positions  $i$  and  $j$  and zeroes everywhere else. The sample  $z(G)$  consists of the labelled examples  $(e_{i,i}, 1)$  for  $i = 1, 2, \dots, n$  and, for each edge  $ij \in E$ , the labelled example  $(e_{i,j}, 0)$ . The size of  $z$  is  $(r + n)(n + 1)$ , which is polynomial in the size of the original VERTEX-COVER instance.

**Example 4.** Consider the graph  $G = (\{1, 2, 3, 4\}, \{11, 12, 13, 14, 23, 33\})$ . Then we have

$$z(G) = \{(1000, 1), (0100, 1), (0010, 1), (0001, 1), \\ (1000, 0), (0010, 0), (1100, 0), (1010, 0), (1001, 0), (0110, 0)\}$$

**Lemma 5.** *Given any graph  $G = (V, E)$  with  $n$  vertices and  $r$  edges and any integer  $k \leq n$ , let  $z(G)$  be as defined above. Then there is  $h \in DL(n)$  such that  $\hat{e}r(h, z(G)) \leq k/(n + r)$  if and only if there is a vertex cover of  $G$  of cardinality at most  $k$*

*Proof.* ( $\rightarrow$ ) Suppose there is such an  $h \in DL(n)$ . By Lemma 3, there is an  $h' \in LT(n)$  that is equivalent to  $h$ . We represent  $h'$  by its weights  $w = (w_1, w_2, \dots, w_n, b)$ . We construct a subset  $U$  of  $V$  as follows.

1. For each  $(e_{i,i}, y) \in z(G)$ , if  $h'(e_{i,i}) = 0$ , then include  $i$  in  $U$ .
2. For each  $(e_{i,j}, 0) \in z(G)$ ,  $i \neq j$ , if  $h'(e_{i,j}) = 1$ , then include either one of  $i, j$  in  $U$ .

The set  $U$  so-constructed contains at most  $k$  vertices since  $\hat{e}r(h', z(G)) = \hat{e}r(h, z(G)) \leq k/(n + r)$ . We now show that  $U$  is a vertex cover for  $G$ . Consider an arbitrary edge  $ij$  in  $E$ . If either  $h'(e_{i,i}) = 0$  or  $h'(e_{j,j}) = 0$  then we're done. Suppose not, that is,  $h'(e_{i,i}) = h'(e_{j,j}) = 1$ . Then we may deduce that

$$w_i \geq b \text{ and } w_j \geq b.$$

This implies that  $h'(e_{i,j}) = 1$ . Because of the way  $U$  is constructed, it follows that at least one of the vertices  $i, j$  is in  $U$ . Since  $ij$  is an arbitrary edge, we conclude that  $U$  is indeed a vertex cover.

( $\leftarrow$ ) Suppose  $U = \{i_1, \dots, i_{|U|}\} \subseteq V$  is a vertex cover of  $G$  and  $|U| \leq k$ . We define  $h \in DL(n)$  to be

$$h = (g_{i_1,1}, 0), \dots, (g_{i_{|U|},1}, 0), (\top, 1).$$

We claim that  $\hat{e}r(h, z(G)) \leq k/(n + r)$ . Observe that if  $ij \in E$ , then since  $U$  is a vertex cover, one of  $i, j$  belongs to  $U$  and thus  $h(e_{i,j}) = 0$ . This means that all the examples in  $z(G)$  arising from the edges of  $G$  are correctly classified. Consider now examples in  $z(G)$  of the form  $(e_{i,i}, 1)$ . We have  $h(e_{i,i}) = 0$  if  $i \in U$  and  $h(e_{i,i}) = 1$  otherwise. It follows that

$$\hat{e}r(h, z(G)) = \frac{|U|}{n + r} \leq \frac{k}{n + r}.$$

□

Given that  $z(G)$  can be computed from  $G$  in time polynomial in the size of  $G$ , we have therefore establish the following hardness result.

**Theorem 6.** DL-FIT is NP-hard.

Now, if there is an algorithm that, given a set  $\mathcal{E}$  of examples, computes in polynomial time  $\arg \min_{h \in 1\text{-DL}(n)} \hat{e}r(h, \mathcal{E})$ , then it can be used to solve DL-FIT in polynomial time. By Theorem 6, such an algorithm cannot exist unless P=NP.

## References

- [1] Martin Anthony. Decision lists and threshold decision lists. Technical Report LSE-CDAM-2002-11, Department of Mathematics and Centre for Discrete and Applicable Mathematics, London School of Economics, 2002.
- [2] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.
- [3] Michael R. Garey and David S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W.H. Freeman, 1979.
- [4] Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.