

Hardness Results for Learning DNF

Nader H. Bshouty

Eyal Kushilevitz

Abstract

We give Online learning model.

1 Hardness Results

One of the main open problems in this area is whether the class DNF is learnable from membership queries and equivalence queries. On the positive side, we have seen several sub-classes which are learnable (for example, those DNF formulae who also have a short CNF representation). Here we show how to give an indication for the hardness of a sub-class of DNF: we will show that the sub-class is as hard to learn as the whole class of DNF. This is *not* to say that such a sub-class is impossible to learn but only that finding a learning algorithm for it might be hard and that it implies the learnability of DNF.

In particular, in this section we show that learning DNF from membership queries and equivalence queries is as hard as learning read-thrice-DNF. That is, the class of DNF formulae where each variable appears at most three times in the formula.

Theorem 1.0.1 *The class DNF is learnable if and only if the class of read-thrice-DNF is learnable.*

Proof: Obviously, if a class is learnable then so is any sub-class of it. Therefore, the learnability of DNF implies the learnability of read-thrice-DNF. For the other direction, suppose that there is an algorithm \mathcal{A} that learns read thrice DNF using membership queries and equivalence queries and construct an algorithm \mathcal{B} for learning the class DNF. Let f be any target DNF formula. Assume that s , the size of f (i.e., the number of terms) is given. Therefore, each variable appears at most $r \leq s$ times in f . We define the following read thrice DNF formula \hat{f} over the variables $y_{i,j}$ where $i = 1, \dots, n$ and $j = 1, \dots, r$: replace the j th appearance of x_i in f by $y_{i,j}$ and in addition, for every i , add to \hat{f} the terms:

$$y_{i,1}\bar{y}_{i,2} \vee y_{i,2}\bar{y}_{i,3} \vee \dots \vee y_{i,r-1}\bar{y}_{i,r} \vee y_{i,r}\bar{y}_{i,1}.$$

Notice that these additional terms are satisfied whenever $y_{i,1}, \dots, y_{i,r}$ are not all equal. Therefore, if there are i, j and k such that $a_{i,j} \neq a_{i,k}$ we get $\hat{f}(a) = 1$. Otherwise, we have

$\hat{f}(a) = f(a_{1,1}, a_{2,1}, \dots, a_{n,1})$. Finally note that, by its definition, the function \hat{f} is a read thrice DNF.

Algorithm \mathcal{B} will learn the DNF f by employing algorithm \mathcal{A} to learn the read-thrice-DNF \hat{f} . The difficulty of course is that we have available oracles for f and not for \hat{f} . Hence, we will simulate queries for \hat{f} as follows. To simulate $\text{EQ}(h)$ (i.e., to ask whether $h \equiv \hat{f}$) we ask $\text{EQ}(g)$ where g is defined as h when we substitute x_i for all $y_{i,j}$. If we get the answer YES then $g \equiv f$ and we accomplished the task of identifying f (note, in particular, that if $h \equiv \hat{f}$ then we get $g \equiv f$). Otherwise, if a is a counterexample (i.e., $f(a) \neq g(a)$) then define b to be an assignment such that $b_{i,j} = a_i$. Then, we have

$$h(b) = g(a) \neq f(a) = \hat{f}(b)$$

and so b is a counterexample for h which can be provided to \mathcal{A} .

To simulate a membership query $\text{MQ}(c)$ asked by \mathcal{A} we do the following: if there are i, j and k such that $c_{i,j} \neq c_{i,k}$ then by the definition of \hat{f} the answer to this membership query is 1 (so this membership query is answered without actually asking the oracle for f any query). Otherwise, if no such i, j and k exist then $\hat{f}(c) = f(c_{1,1}, c_{2,1}, \dots, c_{n,1})$ and so we can ask $\text{MQ}(c_{1,1}, c_{2,1}, \dots, c_{n,1})$ for f . \square

Exercise 1.0.1 *Modify the above algorithm \mathcal{B} to learn any DNF f even if its size is not given. Hint: use doubling to search for s .*

Exercise 1.0.2 *Show that the class read-once-DNF is learnable from membership queries and equivalence queries. Hint: Unate DNF.*