

Revising Projective DNF in the presence of noise

Robert H. Sloan
Univ. Illinois at Chicago
sloan@uic.edu

Balázs Szörényi
University of Szeged
szorenyi@inf.u-szeged.hu

1 Introduction

Machine learning in general is concerned with using computers to simulate the learning done by biological entities, especially human beings. Learning theory in particular is a research field that uses the tools of mathematics to study the design and analysis of algorithms to make predictions about the future based on past experiences. Many problems in learning theory have the following broad form (*learning from examples*): A fixed universe of interest U , called a domain, is specified (e.g., $\{0,1\}^n$). A *learner* is given some sort of access to examples of an unknown 0-1 function f on U , for instance a pair $(x_t, f(x_t))$ might be given at each time step $t = 0, 1, 2, \dots$. The goal of the learner is to determine f (or a “good approximation” to f). Of course, if we require any kind of efficiency, (e.g., limited number of example-label pairs, computational efficiency), then there must be some a priori constraints on f ; the usual constraint is that f is known to come from some particular class of functions.

This paper focuses on one particular problem in the learning theory of logical formulas: revising projective disjunctive normal forms (projective DNFs) in the presence of noise. This problem is chosen because, while it may sound quite specific, in fact, it ties together three central issues in machine learning. These issues are biological plausibility, revision, and noise. We address each in turn as we explain what our problem is in slightly more detail.

Our discussion of biological plausibility borrows heavily from Valiant [7–9], though these issues have been discussed in a number of other places. One very significant difference between human learning and typical current machine learning (see, e.g., Mitchell [5]) is that most machine learning requires some specific process to do *feature extraction* when the data has high dimensionality, whereas humans seem to be able to directly process data of very high dimensionality.

Humans learn from quite few examples, in spite of having an internal representation that is potentially extremely large—perhaps in the general vicinity of 10^{10} neurons with each neuron having up to about 10^4 parameters (synapses). This means first that the things humans learn must be “small” by some measure, or else learning from a small number of examples would be impossible on dimensionality grounds. Furthermore, human senses are very rich—that is, the input (i.e., training data) that humans receive has very high dimensionality.

Therefore the underlying algorithms must have a special property: *attribute efficiency* [1, 4]. In attribute-efficient learning, one is learning a (typically Boolean) function that depends on only k variables in a universe of n variables, where $k \ll n$. The algorithms’ learning complexity (e.g., amount of training data required) is polynomial in k , but much smaller, typically either logarithmic or polylogarithmic, in n . Littlestone’s Winnow [4] is a justly famous algorithm for the attribute efficient learning of Boolean disjunctions and conjunctions, which also has some other biologically appealing properties.

The purpose of *projection learning* [8] is to extend the class of Boolean functions for which we can exhibit a learning algorithm that is attribute efficient and otherwise biologically appealing. Projective classes are ones where one can hope to obtain simple, natural, two-level compositions of attribute-efficient algorithms such as Winnow. Perhaps the simplest is projective DNF, which we define formally in Section 2. Valiant gave a learning algorithm for this class [8]; Sloan et al. extended this to a revision algorithm [6]. In this paper, we will extend this to a revision algorithm that *tolerates noise*. Next we briefly discuss revision, and then noise.

The area of *theory revision* in machine learning is concerned with the revision, or correction of an initial theory (in this paper, a propositional logic formula used for classification) that is assumed to be roughly correct. A typical application area is refining a classifier obtained from a human expert in expert systems work. More broadly, it seems reasonable that in most cases learning starting from something that is close but not exactly right should be much easier than learning from scratch. Indeed, in the cases of human learning of natural language and of particular human faces, it may be that humans are born hardwired with a rough approximation, and in fact revise that, rather than learning from scratch.

The usual assumption of theory revision is that the correct theory can be obtained from the initial one by a small number of syntactic modifications, such as the deletion or the addition of a literal. An efficient revision algorithm is required to be polynomial in the number of literals that need to be modified and polylogarithmic in the total number of literals. Thus, theory revision is roughly an extension of attribute efficient learning—attribute efficient learning can be viewed as the special case of revising the null initial theory. Wrobel surveys the complete theory revision literature [10, 11]; efficient revision in the framework of learning propositional formulas with queries is discussed in detail in [2, 3]. Sloan et al. [6] show how to efficiently revise projective DNFs in the mistake bounded model of learning, where the learner first predicts the classification of a new instance presented by Nature, and then is informed of the correct classification.

However, the algorithm in Sloan et al., may fail badly if any wrong classifications are given to the learner. This is unacceptable, either for biological plausibility, or for application in any practical machine learning system. All real-world training data contains noise. Thus, in this paper, we complete the creation of a biologically plausible, potentially applicable learning of projective DNFs, by showing how to revise them in the presence of noise.

In Section 2 we will give some notations and technical definitions. Then in Section 3 we present our algorithm and its analysis. We conclude briefly in Section 4.

2 Preliminaries

To avoid confusion, vectors from $\{0, 1\}^n$ will always be written in bold face (e.g., \mathbf{w} or \mathbf{w}_t), and a component of a vector will be written in normal font (e.g., w_i or $w_{t,i}$).

Projection learning [8] is a tool for learning classes of functions that have a special representation as disjunctions with a particular special structure. Perhaps the most basic class are the *k-projective DNF* (*k-PDNF*) formulas. A DNF formula φ is a *k-projective DNF*, or *k-PDNF* if it is of the form

$$\varphi = \rho_1 c_1 \vee \cdots \vee \rho_\ell c_\ell,$$

where every ρ_i , referred to as a *k-conjunction*, is a conjunction of exactly *k* literals, c_i is a conjunction and for every *i* it holds that

$$\rho_i \varphi \equiv \rho_i c_i.$$

Each ρ_i is called a *projection*.

It is slightly easier to obtain and explain our results in this paper if we do the formal work with the negation of k -PDNFs, so we extend the notion of projection to its dual. A CNF formula $\varphi = (\rho_1 \vee c_1) \wedge \cdots \wedge (\rho_\ell \vee c_\ell)$ is called k -PCNF if each ρ_i is a disjunction (or *clause*) of size k , c_i is a disjunction and

$$\rho_i \vee \varphi \equiv \rho_i \vee c_i,$$

for $i = 1, \dots, \ell$. The latter condition means that $\varphi(\mathbf{x}) = c_i(\mathbf{x})$ for any vector \mathbf{x} that falsifies ρ_i . Again, each ρ_i is called a projection. It is easy to see that the negation of a k -PCNF is a k -PDNF and vice versa; thus any revision algorithm for one class can be easily transformed into a revision algorithm for the other.

To discuss revising a formula φ_0 to a formula φ , we need to define a measure of (syntactic) distance between two formulas. The distance of two terms t and t^* is $|t \oplus t^*|$, the number of literals occurring in exactly one of the two terms. The revision distance between an *initial* formula

$$\varphi_0 = \rho_1 t_1 \vee \cdots \vee \rho_\ell t_\ell \vee \rho_{\ell+1} t_{\ell+1} \vee \cdots \vee \rho_{\ell+d} t_{\ell+d}$$

and a *target* formula $\varphi = \rho_1 t_1^* \vee \cdots \vee \rho_\ell t_\ell^* \vee \rho'_1 t'_1 \vee \cdots \vee \rho'_a t'_a$ is

$$\text{dist}(\varphi_0, \varphi) = d + \sum_{i=1}^{\ell} |t_i \oplus t_i^*| + \sum_{i=1}^a \max(|t'_i|, 1).$$

The distance is *not* symmetric, and this reflects the fact that we are interested in the number of edit operations required to transform φ_0 to φ . These edit operations are the deletion of a literal or a term, and the addition of a literal. For example, the d term in the definition of *dist* corresponds to the deletion of the d terms $\rho_{\ell+1} t_{\ell+1}, \dots, \rho_{\ell+d} t_{\ell+d}$.

We use the standard model of *mistake bounded* learning [4], which proceeds in a sequence of trials. In each trial t the learner receives an instance \mathbf{x}_t , and produces a prediction \hat{y}_t . Then the algorithm receives a *label* y_t . If y_t is the correct classification of \mathbf{x}_t (i.e., it is equal to what the target formula evaluates on \mathbf{x}_t), we say that y_t is *correct*, otherwise it is *false*. $\text{FALSE}(t)$ is the indicator of the latter case. We denote by $\#\text{FALSELABELS}$ the number of false labels during a particular run, and by $\#\text{FALSENEG}$ the number of 0 labels that are false—that is, $\#\text{FALSELABELS} = \sum_t \text{FALSE}(t)$ and $\#\text{FALSENEG} = \sum_{t: y_t=0} \text{FALSE}(t)$. Of course it is sensible to assume that the number of false labels is relatively small compared to the number of trials, if we expect the learning to be successful. If $\hat{y}_t \neq y_t$ then the learner made a *mistake*. The mistake bound of the learning algorithm is the maximal number of mistakes, taken over all possible runs, that is, sequences of instances.

The goal of theory revision is that given an initial formula φ_0 , learn to simulate the unknown target formula φ making a number of mistakes that is bounded from above by a polynomial of $\text{dist}(\varphi_0, \varphi)$, the number of false labels, and the logarithm of the number of variables in the universe under consideration.

3 Algorithm and analysis

In this section, we present our main results—our algorithm and its analysis and proof of correctness. The overall algorithm has a two-level structure, with many instances of a revision version of Winnow on the lower level feeding their outputs to one instance of a revision version of Winnow on the top level. We first describe the component algorithms, and then the overall algorithm.

3.1 Revising disjunctions—Algorithm RevWinn

Algorithm REVWINN can be used to revise attribute efficiently a monotone disjunction. It can be applied to revise arbitrary disjunction by introducing extra variables for the negated literals, and this in turn can be used to revise arbitrary conjunctions by applying the De Morgan rules. We present REVWINN in a somewhat informal way; we will later assume without further discussion that we have versions available for arbitrary disjunctions and for conjunctions.

Algorithm REVWINN works by maintaining a weight vector \mathbf{w} of length n , where n is the total number of variables in the universe under consideration. In general for components we will use indices i and j , and for trials we will use t —accordingly $w_{t,i}$ denotes the value of the i -th component of vector \mathbf{w} in trial t .

The algorithm consists of three main parts: initialization of \mathbf{w} (which initializes the hypothesis), prediction (the hypothesis part), and the update part.

Let us now describe the three parts of REVWINN. The initialization part is done by function $\text{INIT}(n, \varphi_0; \delta)$, where n is a positive integer determining the number of variables in the universe, φ_0 is a disjunction over at most n variables and $\delta \in (0, 1)$. The output of INIT is an n -dimensional real vector \mathbf{w} which is defined by

$$w_i = \begin{cases} 1 & \text{if variable } x_i \text{ appears in } \varphi_0 \\ \delta & \text{otherwise} \end{cases}$$

for $i = 1, \dots, n$. This \mathbf{w} vector is a weight vector, which determines the current hypothesis, and is updated each time a mistake is made.

The hypothesis function, given instance (n -dimensional vector) \mathbf{x} , is

$$h(\mathbf{x}; \mathbf{w}) = \begin{cases} 0 & \text{if } \mathbf{w} \cdot \mathbf{x} < 1/2 \\ 1 & \text{otherwise} \end{cases}$$

where \mathbf{w} is the current weight vector. The hypothesis is used to make predictions; in trial t we predict that the label of \mathbf{x}_t is $\hat{y}_t = h(\mathbf{x}_t; \mathbf{w})$.

Finally the function $\text{UPDATE}(y, \mathbf{x}, \alpha; \mathbf{w})$, returns a vector \mathbf{w}' , a modification of the weight vector \mathbf{w} :

$$w'_i = w_i \left(1 + \frac{1}{4\alpha} \frac{x_i}{\mathbf{x} \cdot \mathbf{w}} (y - \hat{y}) \right).$$

The function UPDATE is intended to be called with \hat{y} being the output of the hypothesis function on \mathbf{x} (that is, $\hat{y} = h(\mathbf{x}; \mathbf{w})$), and normally UPDATE is called only if $\hat{y} \neq y$.

For illustrative purposes, we describe how REVWINN can be applied simply to revise a disjunction φ_0 . To use REVWINN for this purpose, put $\delta = \frac{1}{2n}$ and $\alpha = 1$ (we call this a *standard* setting), accordingly the initialization is $\mathbf{w} = \text{INIT}(n, \varphi_0; \frac{1}{2n})$, and in each trial perform an update only if $y_t \neq h(\mathbf{x}_t; \mathbf{w})$. In this case the update is given by $\mathbf{w} = \text{UPDATE}(y, \mathbf{x}, 1; \mathbf{w})$. We omit the analysis and the correctness of this algorithm, partly because of the lack of the space, but mostly because this particular case was analyzed in [6] and also would follow from the general analysis in Section 3.2 of this paper.

Let REVWINNC denote the variant of REVWINN that revises conjunctions, and INITC, hC, and UPDATEC denote its main functions.

3.2 Algorithm RevPCNF

For revising PCNFs, we use a two-level algorithm (see Algorithm 1) based on Valiant's construction ([8] and also in [6]). On the lower level, for each k -projection ρ , we run an instance of REVWINN

to find the appropriate clause for that projection. We call this the ρ instance of REVWINN, and we denote its weight vector by \mathbf{w}^ρ . The input and the label for each of these instances are \mathbf{x}_t and y_t . An update is applied to the ρ instance of REVWINN only when $\rho(\mathbf{x}_t) = 0$ (and additionally the top-level prediction of the label was wrong and agreed with the prediction of the ρ -instance of REVWINN) because in this case if ρ appears in the target formula accompanied with disjunction c , then the output of the target formula agrees with c —and this is the key of the whole algorithm. Intuitively, we hope that for each clause of the form $\rho \vee c$ in the target formula, where ρ is a k -projection, that the hypothesis of the ρ instance of REVWINN will converge to c .

The prediction of the ρ instance of REVWINN is denoted by \hat{y}^ρ and $\hat{y}_t^\rho = \text{h}(\mathbf{x}_t; \mathbf{w}_{t-1}^\rho)$.

For the top-level algorithm we introduce new Boolean variables v_ρ for each k -projection ρ . We denote by \mathbf{v} the vector formed by all these variables. In each trial \mathbf{v} is defined by

$$v_\rho := \rho(\mathbf{x}_t) \vee \text{h}(\mathbf{x}_t; \mathbf{w}^\rho).$$

The top level REVWINN algorithm learns a disjunction over variables v_ρ , which would ideally consist of exactly those variables that are indexed by projections appearing in the target formula.

To ease the description we introduce the following notations for the analysis. For each k -projection ρ let \mathbf{w}_t^ρ denote the weight vector \mathbf{w}^ρ after trial t , and accordingly let \mathbf{w}_0^ρ denote the initial weight vector, that is set by function INIT.

Similarly define \mathbf{w}_t and $\hat{y}_t = \text{hC}(\mathbf{x}_t, \mathbf{w}_{t-1})$ (and also \mathbf{v}_t) for the top level algorithm.

Algorithm 1 The procedure REVPCNF(φ_0)

```

1:  $\{\varphi_0 = (\rho_1 \vee c_1) \wedge \dots \wedge (\rho_{\ell+d} \vee c_{\ell+d})$  is the  $k$ -PCNF to be revised}
2:  $\mathbf{w} := \text{INITC}(2^k \binom{n}{k}, v_{\rho_1} \wedge \dots \wedge v_{\rho_{\ell+d}}; \frac{1}{2^k \binom{n}{k}})$ 
3: for each  $k$ -projection  $\rho$  do
4:   if  $\rho = \rho_i$  for an  $i \in \{1, \dots, \ell + d\}$  then
5:      $\mathbf{w}^\rho := \text{INIT}(n, c_i, \frac{1}{2n(\ell+d)})$ 
6:   else
7:      $\mathbf{w}^\rho := \text{INIT}(n, 0, \frac{1}{2n})$  // 0 is the empty disjunction
8:   for each trial  $t$  with input  $\mathbf{x}_t$  do
9:     evaluate  $\mathbf{v}$  by  $v_\rho := \rho(\mathbf{x}_t) \vee \text{h}(\mathbf{x}_t; \mathbf{w}^\rho)$ 
10:    In trial  $t$  predict  $\hat{y}_t = \text{hC}(\mathbf{v}; \mathbf{w})$ 
11:    if  $y_t = \hat{y}_t$  then
12:      don't update the weights
13:    else
14:       $\mathbf{w} := \text{UPDATEC}(y_t, \mathbf{v}, 1, \mathbf{w})$ 
15:       $m := |\{i : \rho_i(\mathbf{x}_t) = 0 \text{ and } \text{h}(\mathbf{x}_t; \mathbf{w}^{\rho_i}) \neq y_t, 1 \leq i \leq \ell + d\}|$ 
16:      for each  $\rho$  with  $\rho(\mathbf{x}_t) = 0$  and  $v_\rho \neq y_t$  do
17:        if  $\rho = \rho_i$  for an  $i \in \{1, \dots, \ell + d\}$  then
18:           $\mathbf{w}^\rho := \text{UPDATE}(y_t, \mathbf{x}_t, m; \mathbf{w}^\rho)$ 
19:        else
20:           $\mathbf{w}^\rho := \text{UPDATE}(y_t, \mathbf{x}_t, 1; \mathbf{w}^\rho)$ 

```

If one simply uses two levels of Littlestone's Winnow for PCNF revision [6], then noise may be a serious problem: even a single false label may cause undesired weight updates in so many projections' corresponding weight vectors that very many mistakes are forced in the rest of the run in order to gain back (resp. lose) all that lost (resp. gained) weight. So we need to limit somehow

the amount of weight update. But this also has its hazards: if the change of the weight during an update is too small, than again we might be forced to make too many mistakes to finally reach a sufficiently big weight.

Our solution to the problem, which is applied in algorithm REVPCNF (see algorithm 1) is the following. Let the initial formula be $\varphi_0 = (\rho_1 \vee c_1) \wedge \cdots \wedge (\rho_{\ell+d} \vee c_{\ell+d})$, and in trial t let

$$m_t = |\{i : \rho_i(\mathbf{x}_t) = 0 \text{ and } \hat{y}_t^{\rho_i} \neq y_t, 1 \leq i \leq \ell + d\}|$$

be the number of projections of the initial formula that makes a weight update in that trial (according to our previously introduced notations m_t is the value of m from algorithm 1 in trial t). For projections $\rho_1, \dots, \rho_{\ell+d}$ we use the update rule

$$w_{t,i}^{\rho_j} = w_{t-1,i}^{\rho_j} \left(1 + \frac{1}{4m_t} \frac{x_{t,i}}{\mathbf{x}_t \cdot \mathbf{w}_{t-1}^{\rho_j}} (y_t - \hat{y}_t^{\rho_j}) \right) \quad (1)$$

which causes exactly $\frac{1}{4m_t}$ change in the sum of the weights of projection ρ_j . Thus summing the change of the weights over all the projections of the initial formula gives exactly 1/4. Note that when $m_t = 0$ then by definition, we do not update the weight vectors of these projections, thus we don't have to bother that (1) would give infinity. For all the other projections that do not appear in the initial formula, a standard REVWINN instance is used. A standard REVWINNC algorithm is also used for the top-level algorithm.

Theorem 1 *Suppose that the initial formula φ_0 and the target formula φ are the k -PCNF formulas*

$$\begin{aligned} \varphi_0 &= (\rho_1 \vee c_1) \wedge \cdots \wedge (\rho_\ell \vee c_\ell) \wedge (\rho_{\ell+1} \vee c_{\ell+1}) \wedge \cdots \wedge (\rho_{\ell+d} \vee c_{\ell+d}) \\ \varphi &= (\rho_1 \vee c_1^*) \wedge \cdots \wedge (\rho_\ell \vee c_\ell^*) \wedge (\rho'_1 \vee c'_1) \wedge \cdots \wedge (\rho'_a \vee c'_a) , \end{aligned}$$

and let $e = \text{dist}(\varphi_0, \varphi)$. Then algorithm REVPCNF makes

$$O\left(\left(\#\text{FALSELABELS}(e + \#\text{FALSELABELS})k \log n\right)^2\right) \text{ mistakes.}$$

The proof is based on the following technical lemma.

Lemma 2 *The number of trials t for which one of $\mathbf{w}_t^{\rho_1}, \dots, \mathbf{w}_t^{\rho_\ell}$ is changed by an update is $O(e\#\text{FALSELABELS}(e + \#\text{FALSELABELS})k \log n)$.*

Proof: There are three cases for a trial t : no change is made to any of the weights $w^{\rho_1}, \dots, w^{\rho_\ell}$, one or more of those w^{ρ_i} is decreased (a *demotion step*), or one or more of those w^{ρ_i} is increased (a *promotion step*). (Since the same value of y_t is passed to all the lower-level REVWINN instances, it cannot be that some weights are increased while other weights are decreased.)

Now consider any run of the algorithm of length T . Let M^- denote the total number of demotions and M^+ the total number of promotions. We prove this lemma by using a potential function argument to obtain a bound on the sum $(M^- + M^+)$.

Let $P = \{\rho_1, \dots, \rho_\ell\}$. Let $P_t \subseteq P$ be the set of projections ρ that appear in both the initial and the target formula such that the ρ instance of REVWINN updates its weight in trial t , that is

$$P_t = \begin{cases} \emptyset & \text{if } \hat{y}_t = y_t \\ \{\rho_i : \rho_i(\mathbf{x}_t) = 0 \text{ and } \hat{y}_t^{\rho_i} \neq y_t, 1 \leq i \leq \ell\} & \text{if } \hat{y}_t \neq y_t \end{cases}$$

Also let $p_t = |P_t|$ (thus $m_t \geq p_t$). For $i = 1, \dots, \ell$ let I^{ρ_i} be the set of indices j of variables that appear in both c_i and c_i^* , and such that there is at least one trial t with $w_{t,j}^{\rho_i} < 1/2$. For $i = 1, \dots, \ell$

let J^{ρ_i} be the set of indices of variables that appear in c_i^* but not in c_i . Let us also introduce the notation $\overline{I^\rho \cup J^\rho} = \{1, \dots, n\} \setminus (I^\rho \cup J^\rho)$ for $\rho \in P$. When no confusion arises, we will sometimes refer to a variable x_i belonging to one of these sets when we really should say that the variable's index belongs to the set.

We use a potential function $\Phi(\mathbf{w}^{\rho_1}, \dots, \mathbf{w}^{\rho_\ell}) = \sum_{j=1}^{\ell} \sum_{i=1}^n \Phi_i^{\rho_j}(\mathbf{w}^{\rho_j})$, where

$$\Phi_i^{\rho}(\mathbf{w}) = \begin{cases} w_i - 1 + \gamma \ln \frac{1}{w_i} & \text{if } i \in I^\rho \cup J^\rho \\ w_i & \text{otherwise.} \end{cases}$$

where $\gamma > 1$ is an appropriate constant which satisfies $\gamma \frac{\ln(1+x)}{x} \geq 1$ for any $x \in (0, 0.5]$ —note that such γ exists¹. It can be verified that $\Phi_i^{\rho_j}(\mathbf{w}) \geq 0$ for any $\mathbf{w} \in (0, 1]^n$. Note that the potential function depends only on the weights corresponding to the projections that appear both in the initial and the target formula. Also note that it is defined with respect to a specific run of the algorithm—that is, not only its values, but even the function itself may vary in different runs.

Let us introduce the short notation² $\Phi_t = \Phi(\mathbf{w}_t^{\rho_1}, \dots, \mathbf{w}_t^{\rho_\ell})$, and let $\Delta\Phi_t = \Phi_{t-1} - \Phi_t$ denote the change of the potential function during trial t . We will derive both upper and lower bounds on $\sum_{t=1}^T \Delta\Phi_t$ that will allow us to relate the sum $(M^- + M^+)$ to e , n , and $\#\text{FALSELABELS}$.

First we derive an upper bound:

$$\begin{aligned} \sum_{t=1}^T \Delta\Phi_t &= \Phi_0 - \Phi_T \\ &\leq \Phi_0 - \sum_{\rho \in P} \sum_{i \in \overline{I^\rho \cup J^\rho}} \Phi_i^{\rho}(w_{T,i}^{\rho}) \\ &= \sum_{\rho \in P} \sum_{i \in I^\rho} \Phi_i^{\rho}(\mathbf{w}_0^{\rho}) + \sum_{\rho \in P} \sum_{j \in J^\rho} \Phi_j^{\rho}(\mathbf{w}_0^{\rho}) + \sum_{\rho \in P} \sum_{i \in \overline{I^\rho \cup J^\rho}} (w_{0,i}^{\rho} - w_{T,i}^{\rho}). \end{aligned}$$

For $i \in I^\rho$ we initialized $w_{0,i}^{\rho} = 1$ so $\Phi_i^{\rho}(\mathbf{w}_0^{\rho}) = 0$. Also, $\sum_{\rho \in P} |J^\rho| \leq e$, and $\Phi_j^{\rho}(\mathbf{w}_0^{\rho}) = \gamma \ln(2n(\ell+d)) - \frac{2n(\ell+d)-1}{2n(\ell+d)} < \gamma(k+1) \ln(2n)$ for $j \in J^\rho$, so the sum of the first two terms is at most $e\gamma(k+1) \ln(2n)$. Now we need to bound the third term. The elements of the third term can be divided into three groups according to their indices ρ_j and i :

- $i \notin c_j \cup c_j^*$. For these indices we have $w_{0,i}^{\rho_j} - w_{T,i}^{\rho_j} \leq w_{0,i}^{\rho_j} = \frac{1}{2n(\ell+d)}$, so altogether they can contribute at most $1/2$ to the sum.
- $i \in c_j \setminus c_j^*$. There are at most e of this kind, each at most 1, thus they contribute at most e to the sum.
- $i \in c_j \cap c_j^*$. By definition, any decrease of the weights corresponding to these index pairs is due to a false negative label, thus their contribution to the sum is at most $\frac{1}{4} \#\text{FALSENEG}$ (by the argument before Theorem 1).

Thus we get

$$\sum_{t=1}^T \Delta\Phi_t \leq e\gamma(k+1) \ln(2n) + e + \frac{1}{2} + \frac{1}{4} \#\text{FALSELABELS} . \quad (2)$$

¹For example $\gamma = 1/\ln 2$ suffices, noting that the function $\ln(1+x)$ is strictly concave, the $x \ln 2$ is linear, both are nonnegative on $(0, 0.5]$ and that they evaluate the same in 0 and in 1.

²Note that using Φ with one single lower index t denotes the value of the potential function in trial t , meanwhile using it with both an upper index ρ and a lower index i denotes the sub-sum of the potential function that corresponds to projection ρ and component i .

To get a lower bound on the sum, we begin by deriving a lower bound on the change in potential in one trial. Now

$$\begin{aligned}\Delta\Phi_t &= \sum_{\rho \in P} \left(\sum_{i \in I^\rho \cup J^\rho} \left(w_{t-1,i}^\rho - w_{t,i}^\rho + \gamma \ln \frac{w_{t,i}^\rho}{w_{t-1,i}^\rho} \right) + \sum_{i \in I^\rho \cup J^\rho} \left(w_{t-1,i}^\rho - w_{t,i}^\rho \right) \right) \\ &= \sum_{\rho \in P_t} \sum_{i=1}^n (w_{t-1,i}^\rho - w_{t,i}^\rho) + \sum_{\rho \in P_t} \sum_{i \in I^\rho \cup J^\rho} \gamma \ln \frac{w_{t,i}^\rho}{w_{t-1,i}^\rho} .\end{aligned}\quad (3)$$

since $w_{t,i}^\rho = w_{t-1,i}^\rho$ for each $\rho \in P \setminus P_t$ and all i . Also note that for any $\rho \in P_t$

$$\ln \frac{w_{t,i}^\rho}{w_{t-1,i}^\rho} = x_{t,i} \ln \left(1 \pm \frac{1}{4m_t} \frac{1}{\mathbf{x}_t \cdot \mathbf{w}_{t-1}^\rho} \right).$$

Now we analyze the change of the potential function during trial t depending on whether t was a demotion step or a promotion step—obviously, when no update is done in trial t (i.e., $y_t = \hat{y}_t$), then $\Delta\Phi_t = 0$.

In a demotion step, $\hat{y}_t = 1$ and $y_t = 0$. If $\rho \in P$ updates its weights in this trial (i.e. $\rho \in P_t$) and $|(I^\rho \cup J^\rho) \cap \mathbf{x}_t| > 0$ then, by the definition of I^ρ and J^ρ and recalling that the target formula is PCNF, the label y_t must be false, thus $|(I^\rho \cup J^\rho) \cap \mathbf{x}_t| = |(I^\rho \cup J^\rho) \cap \mathbf{x}_t| \text{FALSE}(t)$. Also, $\mathbf{x}_t \cdot \mathbf{w}_{t-1}^\rho \geq \frac{1}{2}$ for each $\rho \in P_t$, thus $1 - \frac{1}{4m_t} \frac{1}{\mathbf{x}_t \cdot \mathbf{w}_{t-1}^\rho} \geq \frac{1}{2}$. So, using (3),

$$\Delta\Phi_t \geq \frac{p_t}{4m_t} + \sum_{\rho \in P_t} |(I^\rho \cup J^\rho) \cap \mathbf{x}_t| \gamma \ln \left(\frac{1}{2} \right) \geq \frac{1}{4d} - \text{FALSE}(t) \gamma (e + \sum_{\rho \in P} |I^\rho|) \ln 2 .\quad (4)$$

In a promotion step, $\hat{y}_t = 0$ and $y_t = 1$, and $\mathbf{x}_t \cdot \mathbf{w}_{t-1}^\rho < 1/2$ for every $\rho \in P_t$, thus $4m_t \mathbf{x}_t \cdot \mathbf{w}_{t-1}^\rho < 2m_t$. Also note that $|(I^\rho \cup J^\rho) \cap \mathbf{x}_t| \geq 1$ for every such ρ , unless the label was false, thus

$$\begin{aligned}\sum_{\rho \in P_t} \sum_{i \in I^\rho \cup J^\rho} \ln \left(1 + \frac{1}{4m_t} \frac{x_{t,i}}{\mathbf{x}_t \cdot \mathbf{w}_{t-1}^\rho} \right) &\geq \sum_{\rho \in P_t} |(I^\rho \cup J^\rho) \cap \mathbf{x}_t| \ln \left(1 + \frac{1}{2m_t} \right) \\ &\geq \sum_{\rho \in P_t} (1 - \text{FALSE}(t)) \ln \left(1 + \frac{1}{2m_t} \right) \\ &= p_t (1 - \text{FALSE}(t)) \ln \left(1 + \frac{1}{2m_t} \right)\end{aligned}$$

consequently

$$\begin{aligned}\Delta\Phi_t &= -\frac{p_t}{4m_t} + \gamma \sum_{\rho \in P_t} \sum_{i \in I^\rho \cup J^\rho} \ln \left(1 + \frac{1}{4m_t} \frac{x_{t,i}}{\mathbf{x}_t \cdot \mathbf{w}_{t-1}^\rho} \right) \\ &\geq -\frac{p_t}{4m_t} + \gamma p_t (1 - \text{FALSE}(t)) \ln \left(1 + \frac{1}{2m_t} \right) \\ &\geq \frac{p_t}{4m_t} - \text{FALSE}(t) \gamma \ln \left(1 + \frac{1}{2m_t} \right) \geq \frac{1}{4d} - \text{FALSE}(t) \gamma .\end{aligned}\quad (5)$$

Observe furthermore that for any $\rho \in P$ and $i \in I^\rho$ all decrease of the weight of variable x_i in ρ 's REVWINN instance is due to a false label. The sum of these weight losses is at least $\frac{1}{2} \sum_{\rho \in P} |I^\rho|$

by the definition of I^ρ . Noting again that any false label y_t changes $\sum_{\rho \in P} \sum_{i=1}^n w_t^\rho, i$ with at most $\frac{1}{4}$, we get $\frac{1}{2} \sum_{\rho \in P} |I^\rho| \leq \frac{1}{4} \# \text{FALSENEG}$. Then (4) and (5) give us

$$\begin{aligned} \sum_{t=1}^T \Delta \Phi_t &\geq \sum_{\substack{\text{demotion} \\ \text{step } t}} \left(\frac{1}{4d} - \text{FALSE}(t) \gamma (e + \frac{1}{2} \# \text{FALSENEG}) \ln 2 \right) + \sum_{\substack{\text{promotion} \\ \text{step } t}} \left(\frac{1}{4d} - \text{FALSE}(t) \gamma \right) \\ &\geq M^- / 4d + M^+ / 4d - \gamma \# \text{FALSELABELS} (\# \text{FALSELABELS} + e) \ln 2 . \end{aligned}$$

Combining this with (2) gives the desired mistake bound noting that $d \leq \ell$. \square

Using a similar proof technique for the algorithms REVWINN and REVWINNC, we can derive the following results, which we state here without proof.

Lemma 3 *If at most F false labels are received, then the standard REVWINN algorithm makes at most $O(F(s + F)k \log n)$ mistakes in revising the empty disjunction to disjunction φ , where s is the size of φ .*

Lemma 4 *Let the initial conjunction be φ_0 and the target conjunction be φ . If at most F false labels are received, then algorithm REVWINNC makes at most $O(F(s + F)k \log n)$ mistakes, where $s = \text{dist}(\varphi_0, \varphi)$.*

Proof of Theorem 1: The top-level REVWINNC of REVPDNF is revising a conjunction over the variables denoted v_ρ in the overall algorithm. Lemma 4 tells us how to compute the mistake bound of that REVWINNC as a function of the revision distance and the number of false labels it receives in a trial. The revision distance for REVWINNC is $d + a \leq e$. There are two sources of false labels from the point of view of REVWINNC. One is any trial where y_t is a false label. The other are trials where one of the v_ρ that is in the target takes on the wrong value because $\rho(\mathbf{x}) = 0$ and the (lower-level) ρ instance of REVWINN had the wrong output for \mathbf{x} .

Let us constrain our attention to trials when the overall algorithm make a mistake (noting that in the rest of the cases no update is done, thus the bound we derive for this constrained case will hold for the whole run as well.).

By the projection property, for $1 \leq i \leq \ell$, when $\rho_i(\mathbf{x}) = 0$ we know that $c_i^*(\mathbf{x})$ must have the same value as the entire target k -PDNF. The same holds for ρ_j' and c_j^* for $1 \leq j \leq a$. Furthermore, the overall algorithm REVPDNF causes updates to a ρ instance of REVWINN only when the overall algorithm makes a mistake (i.e., $y_t \neq \hat{y}_t$) and $\rho(\mathbf{x}_t) = 0$. Therefore, whenever UPDATE is called for either the ρ_i , for $1 \leq i \leq \ell$, or the ρ_j' , for $1 \leq j \leq a$, instance of REVWINN, then the y_t passed down to that instance is the appropriate y_t to which that instance's output should be compared.

Excluding the trials where y_t is a false label, the top-level REVWINNC receives the correct value for $v_{t, \rho_i} = \rho_i(\mathbf{x}_t) \vee c_i^*(\mathbf{x}_t)$ for $i = 1, \dots, \ell$ anytime that the ρ_i instance of REVWINN does not update its weight vector. This is so because of the projection property if $\rho_i(\mathbf{x}_t) = 0$, and because v_{t, ρ_i} is always correct if $\rho_i(\mathbf{x}_t) = 1$. Lemma 2 gives a bound on the number of times that these REVWINN instances update their weights, noting that we can use the value of $\# \text{FALSELABELS}$ for the overall algorithm as an upper bound on the number of false labels that any ρ_i instance of REVWINN will receive.

Lemma 3 similarly gives a bound on the number of wrong outputs that the ρ_j' instance of REVWINN can pass up to REVWINNC as wrong $v_{\rho_j'}$ for $1 \leq j \leq a$.

Thus the overall mistake bound for the algorithm is the bound on REVWINNC from Lemma 4, where we use e as the revision distance and the sum of $\# \text{FALSELABELS}$ plus the sum of the two

mistake bounds from Lemmas 2 and 3 as an upper bound on F , noting that e is an upper bound for s in the mistake bound of Lemma 3.

Straightforward algebra gives the claimed bound.

QED

References

- [1] Nader Bshouty and Lisa Hellerstein. Attribute-efficient learning in query and mistake-bound models. *J. of Comput. Syst. Sci.*, 56(3):310–319, 1998.
- [2] Judy Goldsmith, Robert H. Sloan, B. Szörényi, and György Turán. Theory revision with queries: Horn, read-once, and parity formulas. Technical Report TR03-039, Electronic Colloquium on Computational Complexity (ECCC), 2003. Available at <http://www.eccc.uni-trier.de/eccc/>. Also submitted for journal publication.
- [3] Judy Goldsmith, Robert H. Sloan, and György Turán. Theory revision with queries: DNF formulas. *Machine Learning*, 47(2/3):257–295, 2002.
- [4] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.
- [5] Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, Boston, Massachusetts, 1997.
- [6] Robert H. Sloan, B. Szörényi, and György Turán. Projective DNF formulae and their revision. In *Proc. COLT 2003: 16th Annual Conf. on Learning Theory*. Springer Verlag, 2003. To appear.
- [7] Leslie G. Valiant. *Circuits of the Mind*. Oxford Univ. Press, 1994.
- [8] Leslie G. Valiant. Projection learning. *Machine Learning*, 37(2):115–130, 1999.
- [9] Leslie G. Valiant. A neuroidal architecture for cognitive computation. *Journal of the ACM*, 47(5):854–882, 2000.
- [10] S. Wrobel. *Concept Formation and Knowledge Revision*. Kluwer, 1994.
- [11] S. Wrobel. First order theory refinement. In L. De Raedt, editor, *Advances in ILP*, pages 14–33. IOS Press, Amsterdam, 1995.