ELSEVIER

Theoretical Computer Science 284 (2002) 109–142

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Direct and indirect algorithms for on-line learning of disjunctions ☆

D.P. Helmbold[a], S. Panizza[b], M.K. Warmuth[a]

[a] *Department of Computer Science, University of California at Santa Cruz, Santa Cruz, CA, USA*
[b] *Department of Computer Science, College of Staten Island/CUNY, NY, USA*

**Abstract**

It is easy to design on-line learning algorithms for learning $k$ out of $n$ variable monotone disjunctions by simply keeping one weight per disjunction. Such algorithms use roughly $O(n^k)$ weights which can be prohibitively expensive. Surprisingly, algorithms like Winnow require only $n$ weights (one per variable or attribute) and the mistake bound of these algorithms is not too much worse than the mistake bound of the more costly algorithms. The purpose of this paper is to investigate how exponentially many weights can be collapsed into only $O(n)$ weights. In particular, we consider probabilistic assumptions that enable the Bayes optimal algorithm's posterior over the disjunctions to be encoded with only $O(n)$ weights. This results in a new $O(n)$ algorithm for learning disjunctions which is related to the Bylander's BEG algorithm originally introduced for linear regression. Besides providing a Bayesian interpretation for this new algorithm, we are also able to obtain mistake bounds for the noise free case resembling those that have been derived for the Winnow algorithm. The same techniques used to derive this new algorithm also provide a Bayesian interpretation for a normalized version of Winnow.


*Keywords:* On-line learning; Bayes algorithm; Mistake bounds; Multiplicative updates

## 1. Introduction

We consider the problem of learning $k$ out of $n$ variable monotone disjunctions, where $k$ is typically much smaller than $n$, in an on-line setting. In this setting learning proceeds in a sequence of trials. On each trial the learning algorithm observes a Boolean instance, predicts the instance's classification, then is told the correct classification for the instance and incurs a mistake if its prediction differs from the correct

☆ The first and third authors are supported by NSF Grants CCR 9700201 and CCR 9821087. The second author is supported by a research fellowship from the University of Milan and by a Eurocolt grant.

*E-mail addresses:* dph@cse.ucsc.edu (D.P. Helmbold), panizza@cse.ucsu.edu (S. Panizza), manfred@cse.ucsu.edu (M.K. Warmuth).

0304-3975/02/$ - see front matter 
PII: S0304-3975(01)00081-0

classification. The goal is to minimize the number of mistakes of the algorithm relative to the sequence of examples being observed.

In this paper on-line learning algorithms always have three parts: a *prediction rule* which maps the instance and weights to a prediction, an *update function* which specifies how the algorithm's weights are modified, and an *update policy* indicating when the update function should be applied. The update policies considered in this paper are: (1) update after each trial, and (2) only update after trials where the algorithm makes an incorrect prediction. Algorithms with the latter policy are called *mistake-driven* (or conservative) [8, 9].

When learning monotone disjunctions, some algorithms keep one weight per disjunction (i.e., a total of $\binom{n}{k}$ weights). We call such algorithms *direct algorithms* since the weights directly encode the confidence in or likelihood of each individual disjunction.

There are other algorithms that learn disjunctions while maintaining only $O(n)$ weights. We call such algorithms *indirect algorithms* since they indirectly encode their confidences in the disjunctions using $O(1)$ weights per attribute. Surprisingly these more efficient algorithms learn disjunctions almost as well as the direct algorithms. The first such indirect algorithm was Littlestone's Winnow algorithm [7, 8].

In this paper, we are primarily interested in a performance criteria that makes no probabilistic assumptions about how the data is generated. On the contrary the examples can be chosen by an adversary and the goal is to make *relatively* few mistakes compared to the number of mistakes made by the best monotone disjunction on the sequence of examples being observed [7, 8].

The Bayesian approach is a popular way to design and analyze on-line algorithms. Bayes learning algorithms use probabilistic assumptions about the world and data observed in past trials to construct a posterior distribution over the class of disjunctions. These algorithms then predict the most likely classification with respect to the current posterior. It is well known that when the instances are generated and labeled according to the probabilistic assumptions, then Bayes algorithm minimizes the expected total number of mistakes.

By comparing the world model assumed by a Bayes algorithm to the actual situation, one can get important intuition about how well (or poorly) the algorithm will perform. Relative mistake bounds give a much different kind of intuition, and their worst-case nature may be overly pessimistic. Relating these two styles of algorithms will give important insight into existing algorithms and lead to new approaches for designing learning algorithms.

For many direct algorithms with good relative mistake bounds it is easy to work out a nice Bayesian interpretation for the algorithms' prediction rule and update function by making appropriate probabilistic assumptions on how the data is generated. For instance, the direct weighted majority (WM) [14] algorithm's weights are posterior probabilities over the set of disjunctions of up to $k$ attributes under the assumption of i.i.d. label noise with a known rate. The algorithm predicts with the label having the highest posterior probability. Although the direct WM algorithm has a clean Bayesian interpretation, until now, it has been unclear if there also exists a Bayesian inter-

pretation for the more efficient indirect algorithms which have good relative mistake bounds.

In this paper, we present a general technique for deriving indirect algorithms from Bayes optimal algorithms that make certain probabilistic assumptions about how the instances and labels are generated. In particular, we show that with some independence assumptions, the posterior distribution over monotone disjunctions kept by the direct Bayes algorithm can be encoded with only $O(n)$ weights. These assumptions lead to indirect algorithms whose updates and prediction functions have a clear Bayesian interpretation. Our technique has been applied to derive two indirect algorithms whose updates and prediction functions coincide with those used by Normalized Winnow [1] (first analyzed in [10]) and a new classification variant of Bylander's binary exponentiated gradient (BEG) algorithm [2] [2] (two indirect algorithms with good relative loss bounds). This suggests that there may be more indirect algorithms that combine the strengths of the Bayesian and relative mistake bound settings.

It is important to observe that the similarity between these algorithms does not extend to the update policy. All known indirect algorithms with good relative mistake bounds must use the mistake-driven update policy, and all Bayes algorithms update their posteriors after each trial.

The classical method for using independence assumptions to simplify the direct Bayes algorithm gives the indirect Naive Bayes algorithm. However, no relative loss bounds have been proven for Naive Bayes or its mistake-driven variant. Experimental results reported in [10] show that Naive Bayes is very sensitive to redundant attributes although its mistake-driven variant seems to handle redundant attributes quite well.

The precursor of this research is a paper by Nick Littlestone [11] (see also [12]) in which he uses a Bayesian approach to derive an indirect prediction algorithm, the singly variant Bayes (SVB) algorithm, for learning linearly separable functions (which include disjunctions). Rather than using a prior over the set of all monotone disjunctions, the SVB algorithm uses a uniform prior over the set of disjunctions of size one. This leads to a different style of indirect update than the ones considered in this paper. A good mistake bound for learning disjunctions with SVB has been proven for the noise-free case, although there is reason to expect that the bound for SVB could be generalized to deal with noise.

The next two sections review the on-line learning of disjunctions and the direct Bayes algorithm. Our general technique for deriving indirect algorithms from direct Bayes algorithms is presented in Section 4. To keep the presentation as simple as possible, we focus in Section 4 on the linear threshold classification algorithm

[1] Winnow and Normalized Winnow are identical except that the weights of Normalized Winnow are always scaled so that they sum to 1. See Fig. 2.

[2] Throughout this paper we call the algorithm using the update function of Fig. 1 BEG because it is related to the update used by Bylander's binary exponentiated gradient algorithm [2] for linear regression. When the gradient w.r.t. the square loss used in the derivation of Bylander's algorithm is replaced by the gradient w.r.t. the "linear hinge loss", we get the update function of Fig. 1. This "linear hinge loss" can be used to motivate other linear threshold classification algorithms such as the Perceptron algorithm and Winnow [5].

related to Bylander's BEG algorithm [2]. In Section 5, we then describe how the same technique can be applied to obtain a Bayesian interpretation of the normalized variant of Winnow.

## 2. An overview of on-line learning of disjunctions

In the Mistake Bound model introduced by Littlestone [7, 8], the goal of the learner is to make a number of mistakes not much greater than the best classifier in some comparison class. In this paper we use monotone disjunctions over $n$ attributes as the comparison class. Such disjunctions are Boolean formulas of the form $x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_k}$ where the indices $i_j$ belong to $\{1,\ldots,n\}$ and the size $k$ is at most $n$. It is natural to represent a monotone disjunction $\boldsymbol{d}$ by the $n$-dimensional binary vector indicating which attributes are in the disjunction. For example, when $n=5$ we will equate the disjunction $x_1 \vee x_3$ with the binary vector $(1,0,1,0,0)$. Given a monotone disjunction $\boldsymbol{d} \in \{0,1\}^n$ and an instance $\boldsymbol{x} \in \{0,1\}^n$, the prediction of $\boldsymbol{d}$ on $\boldsymbol{x}$ is defined to be the Boolean value

$$\boldsymbol{d}(\boldsymbol{x}) = 1 \quad \text{if } \boldsymbol{d} \cdot \boldsymbol{x} \geqslant 1 \text{ and } 0 \text{ otherwise.} \tag{2.1}$$

Good learning algorithms in the mistake bound model make a number of mistakes not much larger than twice[3] the number of mistakes made by the best disjunction on an *arbitrary* sequence of examples. This can be easily achieved for direct algorithms, such as direct WM. No known indirect algorithms achieve this goal. In fact, for indirect algorithms it is only possible to prove relative mistake bounds that are not much larger than twice[4] the number of attribute errors of the best disjunction. A disjunction's *attribute errors* are those bits in the instances that must be changed so that the disjunction correctly labels the modified instances. For disjunctions of size $k$, the number of attribute errors can be up to a factor of $k$ larger than the number of classification errors. These additional mistakes appear to be a necessary consequence of the indirect algorithm's improved computational efficiency. This penalty occurs only in the presence of noise; in the noise-free case both direct and indirect algorithms have similar $O(k \log n)$ mistake bounds (see [8]).

## 3. The direct Bayes algorithm for disjunctions

It is straightforward to apply Bayes methods (see, e.g. [4]) to the on-line learning of disjunctions in the presence of noise. For instance, we might assume that the unknown sequence is generated as follows. First, a "target" disjunction $\boldsymbol{d}$ is chosen at random

[3] The factor of two disappears when a probabilistic prediction is allowed, so that the direct algorithm's expected (w.r.t. its internal randomization) number of mistakes should not be much larger than the number of mistakes made by the best disjunction [14].

[4] Again, the factor of two multiplying the number of attribute errors disappears when a probabilistic prediction is allowed [1].

from some prior distribution $P(\cdot \mid \lambda)$ on the space of all monotone disjunctions over $n$ attributes, where $\lambda$ denotes the empty sequence. Second, each instance-label pair $(\boldsymbol{x}_t, y_t)$ of the sequence $S^\ell = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ is drawn at random according to some probability distribution $P(\cdot \mid \boldsymbol{d})$ such that $P((\boldsymbol{x}_t, y_t) \mid S^{t-1}, \boldsymbol{d}) = P(y_t \mid \boldsymbol{x}_t, \boldsymbol{d}) P(\boldsymbol{x}_t \mid \boldsymbol{d})$ where $P(\boldsymbol{x}_t \mid \boldsymbol{d}) = P(\boldsymbol{x}_t)$ and $P(y_t \mid \boldsymbol{x}_t, \boldsymbol{d}) = v^{|y_t - \boldsymbol{d}(\boldsymbol{x}_t)|} (1-v)^{(1-|y_t - \boldsymbol{d}(\boldsymbol{x}_t)|)}$. In other words, each label $y_1, \ldots y_\ell$ of the sequence of examples $S^\ell$ differs from that predicted by the selected disjunction with a probability that depends on an arbitrary but fixed "noise rate" $v \in (0, 1/2)$.

In this probabilistic setting, it is not too difficult to see that the Bayes prediction rule simply outputs the label $\hat{y}_t$ such that

$$\hat{y}_t = \arg \max_{y \in \{0,1\}} \left\{ \sum_{\boldsymbol{d} \in \{0,1\}^n} v^{|y - \boldsymbol{d}(\boldsymbol{x}_t)|} (1-v)^{(1-|y-\boldsymbol{d}(\boldsymbol{x}_t)|)} P(\boldsymbol{d} \mid S^{t-1}) \right\}. \tag{3.1}$$

At the end of every trial the current posterior distribution over the class of monotone disjunctions is then updated according to Bayes Rule.

Different choices of the noise rate $v$ produce different versions of the Bayes optimal predictor (3.1). For instance, if $\beta < 1$ and $v = \beta/(\beta + 1)$, then the Bayes prediction algorithm is identical (up to a trivial rescaling of the weights) to the direct WM algorithm that always updates with factor $\beta$.

## 4. A technique for deriving indirect algorithms

In this section, we present a general technique for deriving indirect prediction algorithms for learning disjunctions. In particular we show that when some independence assumptions are made regarding the generation of the instances and labels, then the posterior distribution over disjunctions kept by the direct Bayes algorithm can be encoded with only $O(n)$ weights. By appropriately fixing the unknown parameters of the model we obtain simple update rules for the $O(n)$ weights encoding the posterior. To simplify the presentation, we specialize our technique for the case where the update function is like the one used by Bylander's BEG algorithm [2]. We also show that when this update function is combined with the Bayes prediction rule, then the resulting mistake-driven indirect algorithms do provably well in the adversarial noise free setting when learning disjunctions.

It is not easy to encode the Bayes posterior over disjunctions with only $n$ weights. In particular, the normalization constants often make it difficult to represent the posterior probabilities of disjunctions as partial products of the $n$ weights. Our approach uses an expanded label space where each attribute has its own label bit. This vector-label prediction problem enables us to sidestep the normalization constant that would otherwise appear when the successive posterior distributions are computed. Combining this expansion with a natural loss function yields Bayes algorithms that predict the bit 1 whenever the posterior probability of the all-1 label vector $1^n$ is greater than the posterior probability of the all-0 vector $0^n$. Thus, these Bayes algorithms for the vector-

label problem can be used for the original disjunction problem by simply converting the binary feedback into the $1^n$ or $0^n$ vector-labels.

So far we have been unable to obtain interesting algorithms without going through this vector-label problem. Neither considering the label as a stochastic function of the attributes, nor considering the attributes as corrupted versions of the label seemed to work. In the first case, we were unable to decompose the problem because of a normalizing factor depending on all of the components. Although the posteriors factored in the second case, the resulting algorithms were not Winnow-like and we were not able to prove reasonable bounds for them.

### 4.1. The Bayesian framework

In this section, we consider a *vector-label* prediction problem where the sequence of examples $S^\ell = (\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell)$ consists of instances $\boldsymbol{x}_t = (x_{t,1}, \ldots, x_{t,n}) \in \{0,1\}^n$ and vector-labels $\boldsymbol{y}_t = (y_{t,1}, \ldots y_{t,n}) \in \{0,1\}^n$. We will use a natural loss function between Boolean predictions and vector-labels, so that algorithms for this problem make the same Boolean predictions required for the disjunction problem.

In Section 3, we assumed that the unknown sequence $S^\ell = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ is generated by first selecting a "target" disjunction $\boldsymbol{d}$ according to some prior distribution $P(\cdot \mid \lambda)$ over the class of all monotone disjunctions and then by drawing each instance-label pair $(\boldsymbol{x}_t, y_t)$ of the sequence $S^\ell$ at random according to some probability distribution $P(\cdot \mid \boldsymbol{d})$ on $\{0,1\}^n \times \{0,1\}$. However, here we assume that the probability distribution $P(\cdot \mid \boldsymbol{d})$ is on $\{0,1\}^n \times \{0,1\}^n$ since each label is now a vector $\boldsymbol{y}_t$. Furthermore, we also make the following assumptions on the probability model

**Model** $\mathscr{M}$

**AS1** $P(\boldsymbol{d} \mid \lambda) = \prod_{i=1}^n P(d_i \mid \lambda)$,
**AS2** $P((\boldsymbol{x}_t, \boldsymbol{y}_t) \mid S^{t-1}, \boldsymbol{d}) = P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{d}) P(\boldsymbol{x}_t \mid \boldsymbol{d})$,
**AS3** $P(\boldsymbol{x}_t \mid \boldsymbol{d}) = P(\boldsymbol{x}_t)$,
**AS4** $P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{d}) = \prod_{i=1}^n P(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{d}) = \prod_{i=1}^n P(y_{t,i} \mid x_{t,i}, d_i)$.

The assumptions of "model $\mathscr{M}$" are designed so that the posterior probabilities over disjunctions have the following product form.

**Lemma 1.** *Under model* $\mathscr{M}$, *for any sequence* $S^t$ *we have that*

$$P(\boldsymbol{d} \mid S^t) = \prod_{i=1}^n P(d_i \mid S_i^t), \tag{4.1}$$

*where* $S_i^t = (x_{1,i}, y_{1,i}), \ldots, (x_{t,i}, y_{t,i})$.

**Proof** (*Sketch*). The proof is by induction on $t$ and is rather straightforward using the assumptions of Model $\mathscr{M}$. For completeness, the proof is provided in Appendix A. □

Thus maintaining the posterior $P(\boldsymbol{d} \mid S^t)$ reduces to maintaining the $n$ independent posteriors $P(d_i \mid S_i^t)$, each of which can be encoded with a single weight.

Before proceeding further in the analysis of our Bayesian framework it is important to point out an important difference between our set of assumptions and the ones used by the popular Naive Bayes algorithm. Naive Bayes simply assumes that the attribute values are conditionally independent given the label, i.e., for any instance $\boldsymbol{x}_t$ and label $y_t$,

$$P(\boldsymbol{x}_t \mid y_t) = \prod_{i=1}^{n} p(x_{t,i} \mid y_t).$$

Despite this simple assumption, Domingos and Pazzani [3] show that if the instances are drawn uniformly at random, then Naive Bayes is optimal for learning disjunctions in the average case setting. However, experimental results reported by Littlestone [7] show that Naive Bayes is not optimal in the relative mistake bound setting even when it is run in a conservative way. Thus, in the mistake bound setting it may be advantageous to track the posterior probabilities of the various disjunctions as done by the direct algorithms with good mistake bounds. However, efficiently tracking these posteriors is difficult with binary labels. Our approach uses an expanded label space that allows the algorithm to track the posteriors over disjunctions with only $O(n)$ time per trial. It is difficult to directly relate our vector-label prediction problem to the standard setting for disjunctions. However, we can show that indirect algorithms derived for the vector-label problem have good mistake bounds in the standard setting.

*An update rule for the posterior probabilities*: Lemma 1 shows that (under model $\mathscr{M}$) the posterior distribution $P(\boldsymbol{d} \mid S^t)$ is a simple function of the $n$ single-component posteriors $P(d_i \mid S_i^t)$. We now consider a particular family of component-wise distributions $\{P_{\beta_0,\beta_1,\gamma}(y \mid x,d)\}_{x,y,d\in\{0,1\}}$, for which these single-component posteriors are easily updated. This family has the noise parameters $0<\gamma<1$, $0\leqslant\beta_0<1$, and $\beta_1>1$, and is defined as follows:

$$P_{\beta_0,\beta_1,\gamma}(y \mid x,d) =$$

$$\begin{cases} \left(\left(\frac{\beta_1-1}{\beta_1-\beta_0}\right)^{1-d}\left(\beta_0\frac{\beta_1-1}{\beta_1-\beta_0}\right)^{d}\right)^{1-y}\left(\left(\frac{1-\beta_0}{\beta_1-\beta_0}\right)^{1-d}\left(\beta_1\frac{1-\beta_0}{\beta_1-\beta_0}\right)^{d}\right)^{y}, & \text{if } x=1 \\ \gamma^{1-y}(1-\gamma)^{y}, & \text{otherwise.} \end{cases}$$

Parameter $1-\gamma$ is the probability that the label is flipped to 1 when $x=0$, and the $\beta$ parameters jointly encode different noise probabilities for the cases when $x=1,\ d=1$, and $x=1,\ d=0$.

After seeing a new example, the weights encoding the posterior are updated as in Bylander's BEG algorithm.

**Theorem 1.** *Let $S^{t-1}$ be the sequence of examples through trial $t-1$, $w_{t,i}=P(d_i=1 \mid S_i^{t-1})$ and $(\boldsymbol{x}_t,\boldsymbol{y}_t)$ be the example received at trial $t$. If for each $i=1,\ldots,n$, the probability $P(y_{t,i} \mid x_{t,i},d_i)$ is equal to $P_{\beta_0,\beta_1,\gamma}(y_{t,i} \mid x_{t,i},d_i)$, then in model $\mathscr{M}$*

$$P(\boldsymbol{d} \mid S^{t-1}) = \prod_{i=1}^{n} w_{t,i}^{d_i}(1-w_{t,i})^{1-d_i},$$

$$P(\boldsymbol{d} \mid S^{t-1}, (\boldsymbol{x}_t, \boldsymbol{y}_t)) = \prod_{i=1}^{n} w_{t+1,i}^{d_i}(1 - w_{t+1,i})^{1-d_i}, \tag{4.2}$$

*where*

$$w_{t+1,i} = w_{t,i} \frac{(\beta_{y_{t,i}})^{x_{t,i}}}{1 - w_{t,i} + w_{t,i}(\beta_{y_{t,i}})^{x_{t,i}}}. \tag{4.3}$$

*Furthermore, the distribution $P_{\beta_0,\beta_1,\gamma}(y_{t,i} \mid x_{t,i}, d_i)$ is the only distribution under model $\mathscr{M}$ for which the identity* (4.3) *holds.*

**Proof** (**Sketch**). By using assumption **AS3**, it is not difficult to see that the Bayes rule for computing successive posterior probabilities for the components $\{d_i\}_{i=1}^{n}$ is

$$P(d_i \mid S_i^{t-1}, (x_{t,i}, y_{t,i})) = P(d_i \mid S_i^{t-1}) \frac{P(y_{t,i} \mid x_{t,i}, d_i)}{\sum_{d_i' \in \{0,1\}} P(y_{t,i} \mid x_{t,i}, d_i') P(d_i' \mid S_i^{t-1})}. \tag{4.4}$$

Without loss of generality let $d_i = 1$. Since $w_{t+1,i} = P(d_i = 1 \mid S_i^{t-1}, (x_{t,i}, y_{t,i}))$ and $w_{t,i} = P(d_i = 1 \mid S_i^{t-1})$, Eq. (4.4) can equivalently be written as

$$w_{t+1,i} = w_{t,i} \frac{P(y_{t,i} \mid x_{t,i}, d_i = 1)}{\sum_{d_i' \in \{0,1\}} P(y_{t,i} \mid x_{t,i}, d_i') P(d_i' \mid S_i^{t-1})}. \tag{4.5}$$

For the first part of the theorem, a case analysis shows that for the distribution $P(y_{t,i} \mid x_{t,i}, d_i)$ assumed in the theorem, the Bayes Rule (4.5) reduces to Eq. (4.3). Eq. (4.2) then follows by combining this Bayesian single component update rule with the product decomposition (4.1). The second part of the theorem is proven in Appendix B. □

Notice that update rule (4.3) is independent of the parameter $\gamma$ that specifies the distribution $P_{\beta_0,\beta_1,\gamma}$. This is because when $x = 0$ the disjunction cannot evaluate to 1, and the probability that $y = 1$ is the (unknown) noise rate. This value can be set to any value $0 < \gamma < 1$ without affecting the update rule.

*A Bayes predictor for bit-labels*: The next step is to map the posterior distribution (4.2) and the current instance into a prediction. Since the disjunction problem requires single bit predictions (rather than vector-labels), we define a natural loss function between vector-labels and bit-labels that is 1 if and only if some component of the vector-label differs from the bit-label, i.e., for $\boldsymbol{y}_t \in \{0,1\}^n$ and $y \in \{0,1\}$, the function $L^n(\boldsymbol{y}_t, y) = 1$ if $\exists j$ such that $y_{t,j} \neq y$, and $L^n(\boldsymbol{y}_t, y) = 0$ otherwise.

The Bayes optimal algorithm for this loss and probabilistic model $\mathscr{M}$ predicts the binary label $\hat{y}_t$ that minimizes the expected loss, i.e.,

$$\hat{y}_t = \arg \min_{y \in \{0,1\}} \sum_{\boldsymbol{y}_t \in \{0,1\}^n} L^n(\boldsymbol{y}_t, y) P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, S^{t-1}). \tag{4.6}$$

It is not difficult to see that this simplifies to

$$\hat{y}_t = \arg \max_{y \in \{0,1\}} P(y^n \mid \boldsymbol{x}_t, S^{t-1}), \tag{4.7}$$

where $y^n$ is the $n$-dimensional vector with each component set to $y$. Thus, the Bayes optimal prediction is the bit $y$ for which the corresponding vector $y^n$ is more likely.

Prediction (4.7) can be expressed in a dot product form over a transformed weight space as shown by the following result.

**Theorem 2.** *Let $S^{t-1}=(\boldsymbol{x}_1,\boldsymbol{y}_1),\ldots,(\boldsymbol{x}_{t-1},\boldsymbol{y}_{t-1})$ be the sequence of examples through trial $t-1$, $w_{t,i}=P(d_i=1\mid S_i^{t-1})$ and $\boldsymbol{x}_t$ be the instance received at trial $t$. If $P(y\mid x_{t,i}, d_i)$ is some $P_{\beta_0,\beta_1,\gamma}(y\mid x_{t,i},d_i)$, then under model $\mathscr{M}$ decision rule (4.7) can be expressed in the following form*:

$$\hat{y}_t=\begin{cases}1, & if\ \boldsymbol{x}_t\cdot\boldsymbol{z}_t>\theta,\\ 0, & otherwise,\end{cases}$$

*where $\theta=n\ln(\gamma/(1-\gamma))$ and*

$$z_{t,i}=\ln\left(\frac{\gamma(1-\beta_0)}{(1-\gamma)(\beta_1-1)}\frac{1+w_{t,i}(\beta_1-1)}{1+w_{t,i}(\beta_0-1)}\right).$$

**Proof.** Under model $\mathscr{M}$ it is not difficult to see that

$$P(\boldsymbol{y}_t\mid\boldsymbol{x}_t,S^{t-1})=\sum_{\boldsymbol{d}\in\{0,1\}^n}\prod_{i=1}^{n}(P(y_{t,i}\mid x_{t,i},d_i)P(d_i\mid S_i^{t-1}))$$

$$=\prod_{i=1}^{n}\left(\sum_{d_i\in\{0,1\}}P(y_{t,i}\mid x_{t,i},d_i)P(d_i\mid S_i^{t-1})\right),$$

where the second equality follows from the fact that we are summing over all $\boldsymbol{d}\in\{0,1\}^n$ and thus sum and product can be switched. The Bayes Decision rule (4.7) is then equivalent to predict $\hat{y}_t=1$ iff

$$\prod_{i=1}^{n}\left(\sum_{d_i\in\{0,1\}}P(y=1\mid x_{t,i},d_i)P(d_i\mid S_i^{t-1})\right)$$

$$>\prod_{i=1}^{n}\left(\sum_{d_i\in\{0,1\}}P(y=0\mid x_{t,i},d_i)P(d_i\mid S_i^{t-1})\right). \tag{4.8}$$

Now, using the $P(y\mid x_{t,i},d_i)$ given in the theorem and the identity $P(d_i\mid S_i^{t-1})=w_{t,i}^{d_i}(1-w_{t,i})^{(1-d_i)}$, it is quite straightforward to see that the summations in (4.8) reduce to

$$\sum_{d_i\in\{0,1\}}P(y=1\mid x_{t,i},d_i)P(d_i\mid S_i^{t-1})$$

$$=\left(\frac{1-\beta_0}{\beta_1-\beta_0}\right)^{x_{t,i}}(1+w_{t,i}(\beta_1^{x_{t,i}}-1))(1-\gamma)^{1-x_{t,i}}$$

$$=\left(\frac{1-\beta_0}{\beta_1-\beta_0}(1+w_{t,i}(\beta_1-1))\right)^{x_{t,i}}(1-\gamma)^{1-x_{t,i}},$$

$$\sum_{d_i\in\{0,1\}} P(y=0\,|\,x_{t,i},d_i)P(d_i\,|\,S_i^{t-1}) = \left(\frac{\beta_1-1}{\beta_1-\beta_0}\right)^{x_{t,i}}(1+w_{t,i}(\beta_0^{x_{t,i}}-1))\gamma^{1-x_{t,i}}$$

$$= \left(\frac{\beta_1-1}{\beta_1-\beta_0}(1+w_{t,i}(\beta_0-1))\right)^{x_{t,i}}\gamma^{1-x_{t,i}},$$

where we have used the fact that each $x_{t,i}\in\{0,1\}$. Thus, prediction rule (4.8) can be rewritten as predict $\hat{y}_t=1$ iff

$$\prod_{i=1}^{n}\left(\frac{1-\beta_0}{\beta_1-\beta_0}(1+w_{t,i}(\beta_1-1))\right)^{x_{t,i}}(1-\gamma)^{1-x_{t,i}}$$

$$> \prod_{i=1}^{n}\left(\frac{\beta_1-1}{\beta_1-\beta_0}(1+w_{t,i}(\beta_0-1))\right)^{x_{t,i}}\gamma^{1-x_{t,i}},$$

$$\prod_{i=1}^{n}\left(\frac{1-\beta_0}{\beta_1-1}\frac{1+w_{t,i}(\beta_1-1)}{1+w_{t,i}(\beta_0-1)}\right)^{x_{t,i}} > \prod_{i=1}^{n}\left(\frac{\gamma}{1-\gamma}\right)^{1-x_{t,i}},$$

$$\sum_{\{i:\,x_{t,i}=1\}}\ln\left(\frac{1-\beta_0}{\beta_1-1}\frac{1+w_{t,i}(\beta_1-1)}{1+w_{t,i}(\beta_0-1)}\right) > \sum_{\{i:\,x_{t,i}=0\}}\ln\frac{\gamma}{1-\gamma},$$

$$\sum_{\{i:\,x_{t,i}=1\}}\ln\left(\frac{\gamma(1-\beta_0)}{(1-\gamma)(\beta_1-1)}\frac{1+w_{t,i}(\beta_1-1)}{1+w_{t,i}(\beta_0-1)}\right) > \sum_{i=1}^{n}\ln\frac{\gamma}{1-\gamma}.$$

The dot product prediction rule then immediately follows by setting $z_{t,i}=\ln((\gamma(1-\beta_0)(1+w_{t,i}(\beta_1-1)))/((1-\gamma)(\beta_1-1)(1+w_{t,i}(\beta_0-1))))$ and $\theta=n\ln(\gamma/(1-\gamma))$. This concludes the proof. □

We call the indirect algorithm using the prediction rule of Theorem 2 and the update function (4.3) described in Theorem 1 the `Bayes-BEG` algorithm. The algorithm is summarized in Fig. 1. Its always-update version minimizes the probability of a mistake with respect to the discrete loss $L^n$ when the vector-labels are generated by $P_{\beta_0,\beta_1,\gamma}(y\,|\,x,d)$ as per model $\mathscr{M}$. However, when learning disjunctions it will only see the vector-labels $1^n$ and $0^n$.

### 4.2. The mistake-driven Bayes-BEG algorithm

We now turn from the probabilistic setting to the adversarial setting where we analyze `MD-Bayes-BEG`, the mistake-driven version of `Bayes-BEG`, assuming the algorithm only sees the vector-labels $1^n$ and $0^n$ that correspond to the labels for the disjunction problem. We use $M_{\mathtt{alg}}(S^\ell)$ to denote the number of mistakes made by algorithm "alg" on sequence $S^\ell$, and we say that a sequence $S^\ell$ and a disjunction $\boldsymbol{d}\in\{0,1\}^n$ are consistent if $\boldsymbol{d}$ correctly labels all the instances of $S^\ell$.

We start by proving a mistake bound for the `MD-Bayes-BEG` algorithm when $\beta_0=0$.

Bayes-BEG

**Input:** $0 \leqslant \beta_0 < 1$, $\beta_1 > 1$, $0 < \gamma < 1$, and $n \geqslant 1$

**Initialization:** Let $(w_{1,1}, \ldots, w_{1,n})$ be a weight vector in $[0,1]^n$

For $t = 1, 2, \ldots$

**Prediction Rule:** Upon receiving the instance $\boldsymbol{x}_t$, if $\boldsymbol{x}_t \cdot \mathbf{z}_t > \theta$ then predict 1, otherwise predict 0, where

$$z_{t,i} = \ln\left(\frac{\gamma(1-\beta_0)}{(1-\gamma)(\beta_1-1)}\frac{1+w_{t,i}(\beta_1-1)}{1+w_{t,i}(\beta_0-1)}\right) \text{ and } \theta = n\ln(\gamma/(1-\gamma)).$$

**Update Function:** Observe vector-label $\boldsymbol{y}_t$ and for each $i = 1, \ldots, n$ set

$$w_{t+1,i} = w_{t,i}\frac{(\beta_{y_{t,i}})^{x_{t,i}}}{1 - w_{t,i} + w_{t,i}(\beta_{y_{t,i}})^{x_{t,i}}}. \tag{4.9}$$

**Update Policy:** Update in all trials.

Fig. 1. The Bayes-BEG algorithm.

**Theorem 3.** *Let $n \geqslant 2$, $c = ((e+1)/(e-1))^{1/n}$ and set $\gamma = c/(1+c)$, $\beta_1 = 1+c$ and $\beta_0 = 0$. Furthermore, let $w_{1,i} = 1/n$ for $i = 1, \ldots, n$. For all sequences $S^\ell = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ consistent with a monotone disjunction $\boldsymbol{d} \in \{0,1\}^n$, we have*

$$M_{\text{MD-Bayes-BEG}}(S^\ell) \leqslant 6.41 + 2.48k\left(1 + \ln_{c+1}\left(\frac{2(n-1)}{e-1}\right)\right), \tag{4.10}$$

*where $k \geqslant 1$ is the size of the target disjunction $\boldsymbol{d}$ and $e$ is the base of the natural logarithm.*

**Proof.** The bound is proven by following the same approach used in Section 5 of Littlestone [7]. As in [7], we call every trial where the algorithm predicts $\hat{y}_t = 0$ and $y_t = 1$ a *promotion* step and every trial where $\hat{y}_t = 1$ but $y_t = 0$ an *elimination* step. Since the MD-Bayes-BEG prediction algorithm is mistake-driven, its number of mistakes is equal to the number of promotion steps, $p$, plus the number of elimination steps, $d$, made by the algorithm. The theorem is then proved by bounding the number of promotion and elimination steps.

To avoid confusion, in the proof we will call the weights $z$ used in the dot product prediction the "$z$-weights", and the weights $w$ associated with the attributes the "$w$-weights".

Let $Z_t = \sum_{i=1}^n z_{t,i}$ be the total $z$-weight at the beginning of trial $t$. Since for any $i \in \{1, \ldots, n\}$ and $t = 1, \ldots, \ell$, $z_{t,i} \geqslant 0$ and $z_{t,i}$ only increases/decreases during promotion/elimination steps it follows that

$$Z_1 + pZ_{\text{gain}} - dZ_{\text{lost}} \geqslant 0, \tag{4.11}$$

where $Z_{\text{gain}}$ is an upper bound on the total $z$-weight gained during a promotion step and $Z_{\text{lost}}$ is a lower bound on the total $z$-weight lost during an elimination step. By solving (4.11) with respect to $d$, we obtain $d \leqslant (Z_1/Z_{\text{lost}}) + p(Z_{\text{gain}}/Z_{\text{lost}})$. Hence, the number of mistakes made by `MD-Bayes-BEG` can be upper bounded by

$$M_{\texttt{MD-Bayes-BEG}}(S^{\ell}) \leqslant \frac{Z_1}{Z_{\text{lost}}} + p\left(1 + \frac{Z_{\text{gain}}}{Z_{\text{lost}}}\right). \tag{4.12}$$

We now estimate the quantities in the right hand side of (4.12). It is easy to see that $Z_1 = n\ln(1+(\beta_1/(n-1))) \leqslant 2\beta_1$ and $Z_{\text{lost}} = \theta$. For $Z_{\text{gain}}$, the analysis is more involved. First note that if $x_{t,i} = 1$ the corresponding weight $w_{t,i}$ is updated to $w_{t+1,i} = w_{t,i}\beta_1/(1 - w_{t,i} + w_{t,i}\beta_1)$. Substituting $w_{t+1,i}$ into $z_{t+1,i}$ and observing that for $w_{t,i} \in [0,1]$ the ratio $z_{t+1,i}/z_{t,i}$ is decreasing with respect to $w_{t,i}$ and $\lim_{w_{t,i}\to 0} z_{t+1,i}/z_{t,i} = (1+c)$, we obtain $z_{t+1,i}/z_{t,i} \leqslant (1+c)$. Now, the total $z$-weight gained at trial $t$ is

$$Z_{\text{gain},t} = \sum_{i=1}^{n} x_{t,i}(z_{t+1,i} - z_{t,i}) \leqslant c\sum_{i=1}^{n} x_{t,i}z_{t,i} \leqslant c\theta,$$

where the last inequality follows from the fact that during a promotion step we have $\sum_{i=1}^{n} x_{t,i}z_{t,i} \leqslant \theta$. Hence, we can set $Z_{\text{gain}} = c\theta$. Finally, we need to bound the number of promotion steps incurred by the algorithm. First, observe that an adversary can force the algorithm to incorrectly predict 0 on an appropriately chosen positive example only when the $w$-weight assigned to a relevant attribute is $\leqslant w_{\text{threshold}}$ where $w_{\text{threshold}} = 2/(ec - c + e + 1)$. This implies that the weight of any relevant attribute is always less than or equal to $w_{\text{threshold}}(1+c)/(1 - w_{\text{threshold}} + w_{\text{threshold}}(1+c)) = 2/(e+1)$. Now, by simple manipulations it is not difficult to see that if $x_{t,i} = 1$ and $w_{t,i} = 1 - 1/(1 + (1+c)^{-k})$ then at the end of a promotion step the updated weight $w_{t+1,i}$ is $w_{t+1,i} = 1 - 1/(1 + (1+c)^{-k+1})$. By expressing the initial and final weights of a relevant attribute in this form, we have that the number of promotion steps per relevant attribute can be upper bounded by $\lceil \ln_{c+1}((2(n-1))/(e-1)) \rceil$ and thus,

$$p \leqslant k\left\lceil \ln_{c+1}\left(\frac{2(n-1)}{e-1}\right)\right\rceil \leqslant k\left(1 + \ln_{c+1}\left(\frac{2(n-1)}{e-1}\right)\right). \tag{4.13}$$

Bound (4.10) then immediately follows by plugging the above estimates for $Z_1$, $Z_{\text{lost}}$, $Z_{\text{gain}}$ and $p$ into (4.12) and by observing that $1 < c \leqslant \sqrt{(1+e)/(e-1)}$. □

The `Bayes-BEG` update function with $\beta_0 = 0$ sets $w_{t+1,i}$ to 0 whenever $x_{t,i} = 1$ and $y_t = 0$, and the multiplicative nature of the update ensures that the weight of the attribute remains 0 thereafter. Therefore, if an example has the label 0 but all the attributes are 1, then all of the weights get set to zero and the algorithm is no longer able to predict 1. This indicates that the $\beta_0 = 0$ version of `Bayes-BEG` is unable to tolerate noise.

On the other hand, if $\beta_0 > 0$ then the weights will always be positive. Even if the weight of an attribute in the best disjunction gets driven down by noisy examples, the multiplicative update will allow it to recover before the algorithm makes too many

additional mistakes. Although the exact analysis with noise is difficult, the next result shows that noise tolerant versions of the algorithm (with $\beta_0 > 0$) also have similar noise-free mistake bounds.

**Theorem 4.** *Let* $n \geqslant 2$, $q = 16/10$, $\varepsilon = 9/10$ *and set* $\gamma/(1-\gamma) = q^{1/(n-1)}$, $\beta_0 = 1 - (q^{3/2} - 1 + \varepsilon)/((1+q)(q^{1/(2(n-1))}))$ *and* $\beta_1 = 1 + (q^{3/2} - 1 + \varepsilon)/(1+q)$. *Furthermore, let* $w_{1,i} = 1/n$ *for* $i = 1, \ldots, n$. *For all sequences* $S^\ell = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ *consistent with a monotone disjunction* $\boldsymbol{d} \in \{0,1\}^n$, *we have*

$$M_{\texttt{MD-Bayes-BEG}}(S^\ell) \leqslant 29.96 + 9.82k \ln(n-1) + 12.14k, \tag{4.14}$$

*where* $k \geqslant 1$ *is the size of the target disjunction* $\boldsymbol{d}$.

**Proof** (**Sketch**). It is similar to the proof of Theorem 3, but now, rather than analyzing the change in the total weight $Z_t = \sum_{i=1}^n z_{t,i}$, where the $z_{t,i}$s are the weights used in the dot product prediction at trial $t$, we analyze the change in the total weight $\Psi_t = \sum_{i=1}^n \psi_{t,i}$ where $\psi_{t,i} = \ln((1 + w_{t,i}(\beta_1 - 1))/(1 - w_{t,i}(1 - \beta_0)))$. Since the details of the proof are more involved than the ones of Theorem 3, the complete proof of Theorem 4 is provided in Appendix C. □

### 4.3. The mistake-driven thresholded-BEG algorithm

An indirect algorithm related to `MD-Bayes-BEG` results when the update function of `MD-Bayes-BEG` is combined with the simple thresholded dot product prediction rule used by Winnow. That is, rather than predicting with the prediction rule of Fig. 1, the algorithm predicts 1 if $\boldsymbol{x}_t \cdot \boldsymbol{w}_t > \theta$, and 0 otherwise. We call the resulting algorithm `MD-Thresholded-BEG`.

This algorithm is much easier to analyze with the existing relative mistake bound techniques [8, 1], and it is not difficult to get relative mistake bounds for it even in the noisy case. For instance, if no information besides the number $n$ of attributes or the size $k$ of the target disjunction is given, then the following bounds on the number of mistakes made by the algorithm on any sequence of examples where the best disjunction incurs at most $A$ attribute errors can be proven. Recall that the attribute errors of a disjunction $\boldsymbol{d}$ are those bits in the Boolean instances that have to be changed so that $\boldsymbol{d}$ correctly labels the modified instances.

**Theorem 5.** *Let* $\alpha > 1$ *and set* $\beta_1 = \alpha$, $\beta_0 = 1/\alpha$ *and* $\theta = (\alpha \ln \alpha)/(\alpha^2 - 1)$. *Also, let* $w_{1,i} > 0$ *for* $i = 1, \ldots, n$. *For all sequences* $S^\ell = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ *having at most* $A$ *attribute errors with respect to some monotone disjunction* $\boldsymbol{d} \in \{0,1\}^n$, *we have*

$$M_{\texttt{MD-Thresholded-BEG}}(S^\ell) \leqslant (\alpha+1)\frac{\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_1) + A \ln \alpha}{\ln \alpha}, \tag{4.15}$$

*where* $\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_1) = \sum_{i=1}^n d_i \ln(d_i/w_{1,i}) + (1-d_i)\ln((1-d_i)/(1-w_{1,i}))$ *is the binary relative entropy between the binary vector* $\boldsymbol{d}$ *and the initial weight vector* $\boldsymbol{w}_1$ *used by the algorithm.*

*Further, if the algorithm knows ahead of time the size $k \geqslant 1$ of the target disjunction $\boldsymbol{d}$, then by setting $\theta = (\alpha \ln \alpha)/(k(\alpha^2 - 1))$ we have*

$$M_{\texttt{MD-Thresholded-BEG}}(S^\ell) \leqslant (\alpha + 1)\frac{k \operatorname{dist}_{\text{bre}}(\boldsymbol{u}, \boldsymbol{w}_1) + A \ln \alpha}{\ln \alpha}, \tag{4.16}$$

*where $\boldsymbol{u} = (1/k)\boldsymbol{d}$.*

**Proof.** The proof of the relative mistake bounds for `MD-Thresholded-BEG` is similar to the proof given in [1] for Winnow. The main idea, due to Littlestone [8], is to derive upper and lower bounds on the total progress made by the algorithm toward a comparison vector specifying the target disjunction which, together, imply an upper bound on the number of mistakes incurred by the algorithm.

For $1 \leqslant t \leqslant \ell$, let $\boldsymbol{w}_t \in [0,1]^n$ be the weight vector of `MD-Thresholded-BEG` at the beginning of trial $t$. Then the updated weight at the end of the trial is

$$w_{t+1,i} = w_{t,i}\frac{(\beta_{y_t})^{x_{t,i}}}{1 - w_{t,i} + w_{t,i}(\beta_{y_t})^{x_{t,i}}} \quad \text{for } t = 1, \ldots, n. \tag{4.17}$$

We measure the amount of progress made by the algorithm at trial $t$ towards a comparison vector $\boldsymbol{\xi}$ specifying the target disjunction by the difference $(\operatorname{dist}_{\text{bre}}(\boldsymbol{\xi}, \boldsymbol{w}_t) - \operatorname{dist}_{\text{bre}}(\boldsymbol{\xi}, \boldsymbol{w}_{t+1}))$ where $\operatorname{dist}_{\text{bre}}(\boldsymbol{\xi}, \boldsymbol{w}) = \sum_{i=1}^n \xi_i \ln(\xi_i/w_i) + (1 - \xi_i)\ln(1 - \xi_i)/(1 - w_i)$ is the binary relative entropy between $\boldsymbol{\xi}$ and $\boldsymbol{w}$. The method of using a difference of relative entropies as a measure of progress was introduced in Littlestone's Ph.D. thesis [8].

We only provide a proof for the mistake bound (4.15). The mistake bound (4.16) can be derived in a similar way by using the probability vector $\boldsymbol{u}$ as the comparison vector rather than the binary vector $\boldsymbol{d}$.

We begin with a lower bound on the total progress made by the algorithm. Throughout the proof we assume that there exists a monotone disjunction $\boldsymbol{d} \in \{0,1\}^n$ with at most $A$ attribute errors on $S^\ell$. Furthermore, for any instance-label pair $(\boldsymbol{x}_t, y_t)$ in the sequence $S^\ell$, let $a_t$ be the minimal number of bits in the Boolean instance $\boldsymbol{x}_t$ that have to be changed so that $\boldsymbol{d}$ correctly labels the modified instance $\boldsymbol{x}'_t$, i.e., $a_t = \sum_{i=1}^n |x_{t,i} - x'_{t,i}|$ where $\boldsymbol{x}'_t$ satisfies $\boldsymbol{d}(\boldsymbol{x}'_t) = y_t$. Note that $A = \sum_{t=1}^\ell a_t$. Using the update function (4.17), we obtain by simple manipulations

$$\begin{aligned}
&\operatorname{dist}_{\text{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \operatorname{dist}_{\text{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) \\
&\quad = (\boldsymbol{d} \cdot \boldsymbol{x}_t)\ln \beta_{y_t} - \sum_{i=1}^n \ln(1 + w_{t,i}((\beta_{y_t})^{x_{t,i}} - 1)) \\
&\quad = (\boldsymbol{d} \cdot \boldsymbol{x}_t)\ln \beta_{y_t} - \sum_{i=1}^n x_{t,i}\ln(1 + w_{t,i}(\beta_{y_t} - 1)) \\
&\quad = (\boldsymbol{d} \cdot \boldsymbol{x}_t{}')\ln \beta_{y_t} + (\ln \beta_{y_t})\sum_{i=1}^n d_i(x_{t,i} - x'_{t,i}) \\
&\qquad - \sum_{i=1}^n x_{t,i}\ln(1 + w_{t,i}(\beta_{y_t} - 1)),
\end{aligned} \tag{4.18}$$

where the second equality follows from the fact that each $x_{t,i} \in \{0,1\}$. Since $\boldsymbol{x}'_t$ is obtained from $\boldsymbol{x}_t$ by removing from it its attribute errors and $a_t = \sum_{i=1}^{n} |x_{t,i} - x'_{t,i}|$, it is not difficult to see that for $\beta_0$ and $\beta_1$ assumed in the theorem we have $(\ln \beta_{y_t}) \sum_{i=1}^{n} d_i(x_{t,i} - x'_{t,i}) = -a_t \ln \alpha$ and $(\boldsymbol{d} \cdot \boldsymbol{x}'_t) \ln \beta_{y_t} \geqslant y_t \ln \alpha$. Thus,

$$\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) \geqslant y_t \ln \alpha - a_t \ln \alpha - \sum_{i=1}^{n} x_{t,i} \ln(1 + w_{t,i}(\beta_{y_t} - 1)). \tag{4.19}$$

Summing (4.19) over trials $t = 1, \ldots, \ell$ we get

$$\begin{aligned}
&\sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) \\
&\quad \geqslant -A \ln \alpha - \sum_{\{t: y_t = 0, \hat{y}_t = 1\}} \left( \sum_{i=1}^{n} x_{t,i} \ln(1 - w_{t,i}(1 - 1/\alpha)) \right) \\
&\qquad - \sum_{\{t: y_t = 1, \hat{y}_t = 0\}} \left( -\ln \alpha + \sum_{i=1}^{n} x_{t,i} \ln(1 + w_{t,i}(\alpha - 1)) \right) \\
&\quad \geqslant -A \ln \alpha + \sum_{\{t: y_t = 0, \hat{y}_t = 1\}} (1 - 1/\alpha)(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) \\
&\qquad + \sum_{\{t: y_t = 1, \hat{y}_t = 0\}} (\ln \alpha - (\alpha - 1)(\boldsymbol{w}_t \cdot \boldsymbol{x}_t)) \\
&\quad \geqslant -A \ln \alpha + \sum_{\{t: y_t = 0, \hat{y}_t = 1\}} (1 - 1/\alpha)\theta + \sum_{\{t: y_t = 1, \hat{y}_t = 0\}} (\ln \alpha - (\alpha - 1)\theta).
\end{aligned} \tag{4.20}$$

Here the second inequality follows from approximating the logarithms. The last inequality follows by observing that when $\hat{y}_t = 1$ we have $\boldsymbol{w}_t \cdot \boldsymbol{x}_t > \theta$ and when $\hat{y}_t = 0$ we have $\boldsymbol{w}_t \cdot \boldsymbol{x}_t \leqslant \theta$. Finally, for the choice of $\theta$ given in the theorem $(1 - 1/\alpha)\theta = \ln \alpha - (\alpha - 1)\theta = (\ln \alpha)/(\alpha + 1)$ and inequality (4.20) yields

$$\sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) \geqslant \frac{\ln \alpha}{\alpha + 1} M_{\texttt{MD-Thresholded-BEG}}(S^{\ell}) - A \ln \alpha. \tag{4.21}$$

Now, note that

$$\begin{aligned}
\sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) &= \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_1) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{\ell+1}) \\
&\leqslant \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_1).
\end{aligned} \tag{4.22}$$

This is all that is needed to complete our analysis. The theorem now follows by combining the lower bound (4.21) with the upper bound (4.22) and solving for $M_{\texttt{MD-Thresholded-BEG}}(S^{\ell})$. □

It is interesting to observe that bound (4.15) of Theorem 5 has the same form as the bound derived for Winnow in [1], except that in the latter the binary relative entropy of (4.15) is replaced by the un-normalized relative entropy $\mathrm{dist}_{\mathrm{ure}}(\boldsymbol{d}, \boldsymbol{w}_1) = \sum_{i=1}^{n} d_i \ln(d_i/w_{1,i}) + w_{1,i} - d_i$. Furthermore, bound (4.15) and bound (4.16) of Theorem 5 have the same form, except that in the latter the binary relative entropy $\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_1)$ of (4.15) is replaced by $k\,\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{u}, \boldsymbol{w}_1)$. Since a natural choice for the initial weight vector is $\boldsymbol{w}_1 = (1/n, \ldots, 1/n)$, and for this choice we have $\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_1) \leqslant 1 + k \ln(n)$ and $k\,\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{u}, \boldsymbol{w}_1) \leqslant 2k + k \ln(n/k)$, it follows that in many cases bound (4.16) is better than bound (4.15). This is not at all surprising since for bound (4.16) to hold the algorithm needs to know the size $k$ of the target disjunction ahead of time in order to tune the value of $\theta$.

Better results can be obtained if the algorithm has some further information regarding the sequence to be predicted. For instance, if the number $A$ of attribute errors of the best disjunction is known in advance, then the parameters of the algorithm can be optimally tuned to obtain bounds similar to those derived in [1] (although with slightly worse constants). For example, when the algorithm knows ahead of time that there exists a monotone disjunction consistent with $S^{\ell}$, i.e., $A = 0$, we get the following bound.

**Theorem 6.** *Let $n \geqslant 2$ and set $\beta_0 = 0$, $\beta_1 = e$ and $\theta = 1/e$. Also, let $w_{1,i} = 1/n$ for $i = 1, \ldots, n$. For all sequences $S^{\ell} = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{\ell}, y_{\ell})$ consistent with a monotone disjunction $\boldsymbol{d} \in \{0,1\}^n$, we have*

$$M_{\texttt{MD-Thresholded-BEG}}(S^{\ell}) \leqslant e + e\,k\,\ln(n), \tag{4.23}$$

*where $k \geqslant 1$ is the size of the target disjunction $\boldsymbol{d}$ and $e$ is the base of the natural logarithm. Furthermore, if the size $k$ is known in advance to the learning algorithm, then by setting $\theta = 1/(e\,k)$ we have*

$$M_{\texttt{MD-Thresholded-BEG}}(S^{\ell}) \leqslant 2\,e\,k + e\,k \ln\left(\frac{n}{k}\right). \tag{4.24}$$

**Proof (Sketch).** Follows the same approach used to prove Theorem 5, but now, the progress of the algorithm is measured towards a comparison vector that correctly classifies the instances in $S^{\ell}$. Details of the proof are given in Appendix D. □

The above theorem and Theorem 3 both bound the performance in the noise free case. Although bound (4.23) in Theorem 6 is always smaller, the Bayesian predictions used by the `MD-Bayes-BEG` algorithm analyzed in Theorem 3 are less well studied than the simpler dot product predictions used by the `MD-Thresholded-BEG` algorithm, and we feel that the approximations used in the analysis of `MD-Bayes-BEG` can probably be tightened, leading to improvements in the bound of Theorem 3.

Normalized Winnow

**Input:** $0 \leqslant \beta_0 < 1$, $\beta_1 > 1$, $\theta > 0$, and $n \geqslant 1$

**Initialization:** Let $(w_{1,1}, \ldots, w_{1,n})$ be a probability vector in $[0,1]^n$

For $t = 1, 2, \ldots$

**Prediction Rule**: Upon receiving the instance $\boldsymbol{x}_t$, if $\boldsymbol{x}_t \cdot \boldsymbol{w}_t > \theta$ then predict 1, otherwise predict 0.

**Update Function:** Observe the label $y_t$ and for each $i = 1, \ldots, n$ set

$$w_{t+1,i} = w_{t,i} \frac{(\beta_{y_t})^{x_{t,i}}}{\sum_{j=1}^{n} w_{t,j} (\beta_{y_t})^{x_{t,j}}}. \tag{4.25}$$

**Update Policy:** Only after trials where the algorithm makes an incorrect prediction.

Fig. 2. The Normalized Winnow algorithm.

## 5. The normalized Winnow algorithm

The normalized Winnow algorithm [10] is another mistake-driven linear thresholded algorithm for on-line learning disjunctions with a good relative mistake bound (see Fig. 2).[5] The original (un-normalized) Winnow algorithm [7] is almost identical. The only difference is that the normalization in (4.25) is omitted. Techniques like those employed in Subsection 4.1 show that Normalized Winnow is also closely related to Bayesian methods.

Whereas the prior on disjunctions used in Subsection 4.1 was a product of $n$ Bernoulli distributions, the prior corresponding to Normalized Winnow is the $n$-fold product of a distribution over $\{1, 2, \ldots, n\}$. Since sampling this prior gives a vector in $\{1, 2, \ldots, n\}^n$, most of the $2^n$ disjunctions will be represented by several vectors. For example, those vectors in $\{1, 3, 7\}^n$ that contain at least one 1, at least one 3, and at least one 7 all represent the disjunction $x_1 \vee x_3 \vee x_7$. As before, the vector-label prediction problem is used to decouple the attributes and obtain the corresponding Bayes algorithm.

Given a sequence of examples $S^t = (\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_t, \boldsymbol{y}_t)$ where $\boldsymbol{x}_t, \boldsymbol{y}_t \in \{0,1\}^n$, throughout this section we will denote by $S_i^t \in (\{0,1\}^n \times \{0,1\})^t$ the sequence $S_i^t = (\boldsymbol{x}_1, y_{1,i}), \ldots, (\boldsymbol{x}_t, y_{t,i})$. Furthermore, recall that a random vector $\mathscr{Z} = (Z_1, \ldots, Z_k)$ has a multinomial distribution with parameters $(p_1, \ldots, p_k)$ and $n$ if $P(Z_1 = z_1, \ldots, Z_k = z_k) = (n!/((z_1!) \cdots (z_k!))) p_1^{z_1} \cdots p_k^{z_k}$ where $(p_1, \ldots, p_k)$ is a probability vector and $\sum_{i=1}^{k} z_i = n$. We denote such distribution by $Mult((p_1, \ldots, p_k), n)$.

To derive a Bayesian interpretation for Normalized Winnow, throughout the section we will assume that the unknown sequence $S^\ell = (\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell)$ is gener-

[5] The threshold equal $\frac{1}{2}$ version of Normalized Winnow was first analyzed in Littlestone's Ph.D. thesis [8] where it was called the WM algorithm.

ated by first selecting a "target" vector $\boldsymbol{q} \in \{1,2,\ldots,n\}^n$ according to some prior distribution $P(\cdot \mid \lambda)$ over the space $\{1,2,\ldots,n\}^n$ and then by drawing each instance-label pair $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ of the sequence $S^\ell$ at random according to some probability distribution $P(\cdot \mid \boldsymbol{q}) \in \{0,1\}^n \times \{0,1\}^n$ where the prior $P(\cdot \mid \lambda)$ and the distribution $P(\cdot \mid \boldsymbol{q})$ satisfy the following assumptions. Here we use "$\boldsymbol{e}_{q_i}$" to denote the unit $n$-vector containing a 1 in position $q_i$ and 0s elsewhere.

**Model** $\mathscr{M}_n$

**AS1n** $P(\boldsymbol{q} \mid \lambda) = \prod_{i=1}^n P(\boldsymbol{e}_{q_i} \mid \lambda)$ where $\boldsymbol{e}_{q_i} \sim Mult((p_1,\ldots,p_n),1)$,

**AS2n** $P((\boldsymbol{x}_t, \boldsymbol{y}_t) \mid S^{t-1}, \boldsymbol{q}) = P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{q}) P(\boldsymbol{x}_t \mid \boldsymbol{q})$,

**AS3n** $P(\boldsymbol{x}_t \mid \boldsymbol{q}) = P(\boldsymbol{x}_t)$,

**AS4n** $P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{q}) = \prod_{i=1}^n P(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i})$.

As in Subsection 4.1, the assumptions of "Model $\mathscr{M}_n$" are designed so that the posterior probabilities over the space $\{1,2,\ldots,n\}^n$ can also be represented as an $n$-fold product distribution.

**Lemma 7.** *Under Model $\mathscr{M}_n$, for any sequence $S^t$ we have that*

$$P(\boldsymbol{q} \mid S^t) = \prod_{i=1}^n P(\boldsymbol{e}_{q_i} \mid S_i^t). \tag{5.1}$$

**Proof.** Omitted since similar to the proof of Lemma 1 in Subsection 4.1. □

Thus maintaining the posterior $P(\boldsymbol{q} \mid S^{t-1})$ reduces to maintaining the $n$ independent posteriors $\{P(\boldsymbol{e}_{q_i} \mid S_i^{t-1})\}_{i=1}^n$, each of which is a multinomial distribution $P(\boldsymbol{e}_{q_i} \mid S_i^{t-1}) = Mult((w_{t,1}^{(i)},\ldots,w_{t,n}^{(i)}),1)$. Hence, the $n^n$ posteriors $\{P(\boldsymbol{q} \mid S^{t-1})\}_{\boldsymbol{q} \in \{1,2,\ldots,n\}^n}$ can be encoded with only $n$ probability vectors (with $n(n-1)$ free parameters). Note that if the sequence $S^{t-1}$ contains only vector-labels $1^n$ and $0^n$ then $S_1^{t-1} = \cdots = S_n^{t-1}$ and the $n$ probability vectors $\{(w_{t,1}^{(i)},\ldots,w_{t,n}^{(i)})\}_{i=1}^n$ are all the same and we can drop the superscript. Thus, when learning disjunctions where the feedback is always either $1^n$ or $0^n$, the algorithm will keep for each trial $t$ only one $n$-dimensional probability vector $\boldsymbol{w}_t = (w_{t,1},\ldots,w_{t,n})$.

Proceeding similarly to the derivation of `Bayes-BEG`, we next show that when the distribution $P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{q})$ as defined by assumption **AS4n** is appropriately chosen, then the posterior probabilities/weights of the integer in $\{1,2,\ldots,n\}$ are easily updated.

Specifically, let $0 \leqslant \beta_0 < 1$, $\beta_1 > 1$, $y \in \{0,1\}$, $\boldsymbol{x} \in \{0,1\}^n$ and $q \in \{1,2,\ldots,n\}$ and consider the family of distributions $P_{\beta_0,\beta_1}(y \mid \boldsymbol{x}, \boldsymbol{e}_q)$ defined by

$$P_{\beta_0,\beta_1}(y \mid \boldsymbol{x}, \boldsymbol{e}_q) = \frac{1}{\beta_1 - \beta_0}((1-\beta_0)\beta_1^{x_q})^y((\beta_1 - 1)\beta_0^{x_q})^{1-y}.$$

The $\beta$ parameters jointly encode different noise parameters for the cases when $x_q = 1/0$ and $q \in \{1,2,\ldots,n\}$. For the probability $P_{\beta_0,\beta_1}(y \mid \boldsymbol{x}, \boldsymbol{e}_q)$, the weights encoding the Bayes posteriors are updated as in the Normalized Winnow algorithm.

**Theorem 7.** *Let $S^{t-1}$ be the sequence of examples through trial $t-1$ and let $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ be the example received at trial $t$. If for each $i = 1,\ldots,n$ and $q_i \in \{1,\ldots,n\}$, the prob-*

*ability* $P(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i})$ *is equal to* $P_{\beta_0,\beta_1}(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i})$ *and* $P(\boldsymbol{e}_{q_i} \mid S_i^{t-1}) = Mult((w_{t,1}^{(i)}, \ldots, w_{t,n}^{(i)}), 1)$, *then in Model* $\mathscr{M}_n$

$$P(\boldsymbol{q} \mid S^{t-1}) = \prod_{i=1}^{n} w_{t,q_i}^{(i)}, \qquad P(\boldsymbol{q} \mid S^{t-1}, (\boldsymbol{x}_t, \boldsymbol{y}_t)) = \prod_{i=1}^{n} w_{t+1,q_i}^{(i)}, \tag{5.2}$$

*where*

$$w_{t+1,q_i}^{(i)} = w_{t,q_i}^{(i)} \frac{(\beta_{y_{t,i}})^{x_{t,q_i}}}{\sum_{j=1}^{n} w_{t,j}^{(i)} (\beta_{y_{t,i}})^{x_{t,j}}}. \tag{5.3}$$

**Proof** (**Sketch**). By using assumption **AS3n**, a case analysis shows that for the distribution $P(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i})$ assumed in the theorem, the Bayes Rule for computing successive posterior probabilities for the unit vectors $\{\boldsymbol{e}_{q_i}\}_{i=1}^{n}$ reduces to Eq. (5.3). The theorem then follows by combining this Bayesian single component update rule with the product decomposition (5.1). □

To map the posterior distribution (5.2) and the current instance into a bit prediction for the disjunction problem, we again consider the loss function $L^n(\boldsymbol{y}_t, y) = 1$ if $\exists j$ such that $y_{t,j} \neq y$, and $L^n(\boldsymbol{y}_t, y) = 0$ otherwise. As argued in Subsection 4.1, the Bayes optimal algorithm for this loss function and the probability model $\mathscr{M}_n$ predicts the bit $y$ for which the corresponding vector $y^n$ is more likely where $y^n$ is the $n$-dimensional vector with each component set to $y$. That is, for $t = 1, \ldots, \ell$

$$\hat{y}_t = \arg \max_{y \in \{0,1\}} P(y^n \mid \boldsymbol{x}_t, S^{t-1}). \tag{5.4}$$

Prediction (5.4) can be expressed in a more suitable form as shown by the following result.

**Theorem 8.** *Let* $S^{t-1} = (\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})$ *be the sequence of examples observed through trial* $t-1$ *and let* $\boldsymbol{x}_t$ *be the instance received at trial t. If* $P(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i})$ *is some* $P_{\beta_0,\beta_1}(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i})$ *and* $P(\boldsymbol{e}_{q_i} \mid S_i^{t-1}) = Mult((w_{t,1}^{(i)}, \ldots, w_{t,n}^{(i)}), 1)$, *then under model* $\mathscr{M}_n$ *decision rule* (5.4) *can be expressed in the following form*:

$$\hat{y}_t = \arg \max_{y \in \{0,1\}} \prod_{i=1}^{n} \left( \sum_{q_i \in \{1,2,\ldots,n\}} ((1-\beta_0)\beta_1^{x_{t,q_i}})^y ((\beta_1 - 1)\beta_0^{x_{t,q_i}})^{1-y} w_{t,q_i}^{(i)} \right). \tag{5.5}$$

*Furthermore, if the sequence* $S^{t-1}$ *only contains vector-labels* $1^n$ *and* $0^n$, *then prediction* (5.5) *can be expressed in the following dot product form*:

$$\hat{y}_t = \begin{cases} 1, & \text{if } \boldsymbol{x}_t \cdot \boldsymbol{w}_t > \theta, \\ 0, & \text{otherwise}, \end{cases} \tag{5.6}$$

*where* $\theta = (\beta_1 + \beta_0 - 2)/(2(\beta_1 - 1)(1 - \beta_0))$ *and* $P(\boldsymbol{e}_{q_i} \mid S_i^{t-1}) = Mult(\boldsymbol{w}_t, 1)$ *for* $i = 1, \ldots, n$.

**Proof.** Using model $\mathscr{M}_n$ it is not difficult to see that

$$P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, S^{t-1}) = \sum_{\boldsymbol{q}\in\{1,2,\ldots,n\}^n} \prod_{i=1}^{n} P(\boldsymbol{y}_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i}) P(\boldsymbol{e}_{q_i} \mid S_i^{t-1}). \tag{5.7}$$

Since in (5.7) we are summing over all $\boldsymbol{q} \in \{1,2,\ldots,n\}^n$, the sum and product can be switched and thus (5.7) can be rewritten as

$$P(\boldsymbol{y}_t|\boldsymbol{x}_t, S^{t-1}) = \prod_{i=1}^{n} \left( \sum_{q_i\in\{1,2,\ldots,n\}} P(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i}) P(\boldsymbol{e}_{q_i} \mid S_i^{t-1}) \right). \tag{5.8}$$

Prediction rule (5.5) then immediately follows by substituting the expression for $P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, S^{t-1})$ given in (5.8) in the Bayes Decision rule (5.4) and using the $P(y_{t,i} \mid \boldsymbol{x}_t, \boldsymbol{e}_{q_i})$ given in the theorem.

Now, to prove the second part of the theorem we argue as follows. First, let us rewrite prediction rule (5.5) conveniently as

$$\text{Predict } 1 \quad \text{iff} \quad \prod_{i=1}^{n} \left( \sum_{j=1}^{n} \beta_1^{x_{t,j}} w_{t,j}^{(i)} \right) > \left( \frac{\beta_1 - 1}{1 - \beta_0} \right)^n \prod_{i=1}^{n} \left( \sum_{j=1}^{n} \beta_0^{x_{t,j}} w_{t,j}^{(i)} \right). \tag{5.9}$$

Since $S^{t-1}$ contains only vector-labels $1^n$ and $0^n$ for any $j \in \{1,2,\ldots,n\}$, $P(\boldsymbol{e}_j \mid S_1^{t-1}) = \cdots = P(\boldsymbol{e}_j \mid S_n^{t-1}) = w_{t,j}$, and (5.9) reduces to

$$\text{Predict } 1 \quad \text{iff} \quad \sum_{j=1}^{n} \beta_1^{x_{t,j}} w_{t,j} > \gamma \sum_{j=1}^{n} \beta_0^{x_{t,j}} w_{t,j}, \tag{5.10}$$

where for ease of notation, we have set $\gamma = (\beta_1 - 1)/(1 - \beta_0)$. Splitting the sum on both sides of (5.10) into the sums over $\{j\colon\ x_{t,j} = 1\}$ and $\{j\colon\ x_{t,j} = 0\}$ we obtain

$$\text{Predict } 1 \quad \text{iff} \quad (\beta_1 - \gamma\beta_0) \sum_{\{j:x_{t,j}=1\}} w_{t,j} > (\gamma - 1) \sum_{\{j:x_{t,j}=0\}} w_{t,j}. \tag{5.11}$$

Now, setting $\sum_{\{j:x_{t,j}=1\}} w_{t,j} = r$ and thus $\sum_{\{j:\,x_{t,j}=0\}} w_{t,j} = 1 - r$, we obtain by simple manipulations

$$\text{Predict } 1 \quad \text{iff} \quad (\beta_1 - \gamma\beta_0 + \gamma - 1)r > (\gamma - 1). \tag{5.12}$$

Substituting the expression for $\gamma$ into (5.12) and observing that $\beta_1-\gamma\beta_0+\gamma-1=2(\beta_1-1)>0$ and $\gamma-1=(\beta_1+\beta_0-2)/(1-\beta_0)$ we finally obtain

$$\begin{aligned}
\text{Predict } 1 \quad &\text{iff } 2(\beta_1-1)r > \frac{\beta_1+\beta_0-2}{1-\beta_0} \\
&\text{iff } r > \frac{\beta_1+\beta_0-2}{2(\beta_1-1)(1-\beta_0)} \\
&\text{iff } \sum_{\{j:x_{t,j}=1\}} w_{t,j} > \frac{\beta_1+\beta_0-2}{2(\beta_1-1)(1-\beta_0)} \\
&\text{iff } \boldsymbol{w}_t \cdot \boldsymbol{x}_t > \theta,
\end{aligned}$$

where $\theta=(\beta_1+\beta_0-2)/(2(\beta_1-1)(1-\beta_0))$. This concludes the proof. □

We call the indirect algorithm using the prediction rule (5.5) of Theorem 8 and the update function (5.3) described in Theorem 7 the `Bayes-NW` algorithm. Its always-update version minimizes the probability of a mistake with respect to the discrete loss $L^n$ when the vector-labels are generated by $P_{\beta_0,\beta_1}(y \mid \boldsymbol{x}, \boldsymbol{e}_q)$ as per model $\mathscr{M}_n$. Observe that on each trial $t$ the algorithm needs to keep $n^2$ weights $\{(w_{t,1}^{(i)},\ldots,w_{t,n}^{(i)})\}_{i=1}^n$ where each $(w_{t,1}^{(i)},\ldots,w_{t,n}^{(i)})$ is a probability vector. However, when learning disjunctions the algorithm will only see the vector-labels $1^n$ and $0^n$. In this case, the Bayes optimal predictions given by (5.6) are the same thresholded dot products between the instances and weights used by the Normalized Winnow algorithm. The only difference with respect to Normalized Winnow is that in (5.6) the threshold $\theta$ is a specific function of the noise parameters $\beta_0$ and $\beta_1$. This establishes a close correspondence between Normalized Winnow and Bayes methods.

Using techniques similar to those used in Subsection 4.3 it is not difficult to get relative mistake bounds for `MD-Bayes-NW`, the mistake-driven version of `Bayes-NW`, assuming the algorithm only sees the vector-labels $1^n$ and $0^n$ that correspond to the labels for the disjunction problem. However, here we use the relative entropy function between probability vectors as our main tool for proving the bounds. Recall that given two probability vectors $\boldsymbol{u}$ and $\boldsymbol{w}$, the relative entropy $\mathrm{dist}_{\mathrm{re}}(\boldsymbol{u},\boldsymbol{w})$ between $\boldsymbol{u}$ and $\boldsymbol{w}$ is defined by $\mathrm{dist}_{\mathrm{re}}(\boldsymbol{u},\boldsymbol{w})=\sum_{i=1}^n u_i \ln(u_i/w_i)$.

Until now we have specified a monotone disjunction mainly by a $n$-dimensional binary vector where the components with value 1 correspond to the attributes of the disjunction. In this case, the prediction of a disjunction $\boldsymbol{d}\in\{0,1\}^n$ on an instance $\boldsymbol{x}$ was defined as the binary outcome of $\boldsymbol{d}\cdot\boldsymbol{x}\geqslant 1$ (see (2.1)). Since now the weights of `MD-Bayes-NW` are probability vector, it is natural to set the $k$ non-zero components of the binary vector specifying the disjunction of size $k\geqslant 1$ to $1/k$ and to define the prediction of such a probability vector as follows. Given a probability vector $\boldsymbol{u}$ representing a disjunction of size $k$ and an instance $\boldsymbol{x}\in\{0,1\}^n$, the prediction of $\boldsymbol{u}$ on $\boldsymbol{x}$ is defined to be the Boolean value

$$\boldsymbol{u}(\boldsymbol{x})=1 \quad \text{if } \boldsymbol{u}\cdot\boldsymbol{x}\geqslant 1/k \text{ and } 0 \text{ otherwise.} \tag{5.13}$$

We start by proving a bound for MD-Bayes-NW that depends on the number $A$ of attribute errors incurred by the best disjunction on the sequence of examples being observed. To obtain tighter bounds we have used a threshold $\theta$ which is different from the one suggested by Theorem 8.

**Theorem 9.** *Let $\alpha > 1$ and set $\beta_1 = \alpha$, $\beta_0 = 1/\alpha$ and $\theta = (\alpha \ln \alpha)/(k(\alpha^2 - 1))$. Furthermore, let $\boldsymbol{w}_1$ be a probability vector such that $w_{1,i} > 0$ for $i = 1, \ldots, n$. For all sequences $S^\ell$ having at most $A$ attribute errors with respect to some monotone disjunction $\boldsymbol{d} \in \{0,1\}^n$, we have*

$$M_{\texttt{MD-Bayes-NW}}(S^\ell) \leqslant (\alpha + 1)\frac{k \operatorname{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_1) + A \ln \alpha}{\ln \alpha}, \tag{5.14}$$

*where $k \geqslant 1$ is the size of the target disjunction $\boldsymbol{d}$ and $\operatorname{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_1) = \sum_{i=1}^n u_i \ln(u_i/w_{1,i})$ is the relative entropy between the probability vector $\boldsymbol{u} = (1/k)\boldsymbol{d}$ and the initial probability vector $\boldsymbol{w}_1$ used by the algorithm.*

**Proof.** We follow the same approach used to prove Theorem 5 except that now we measure the progress made by the algorithm at each trial $t$ in terms of the difference $(\operatorname{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \operatorname{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}))$ rather than the difference $(\operatorname{dist}_{\mathrm{bre}}(\boldsymbol{u}, \boldsymbol{w}_t) - \operatorname{dist}_{\mathrm{bre}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}))$.

For $1 \leqslant t \leqslant \ell$, let $\boldsymbol{w}_t$ be the probability vector used by MD-Bayes-NW at the beginning of trial $t$. Then the updated weight at the end of the trial is

$$w_{t+1,i} = w_{t,i} \frac{(\beta_{y_t})^{x_{t,i}}}{\sum_{j=1}^n w_{t,j}(\beta_{y_t})^{x_{t,j}}} \quad \text{for } t = 1, \ldots, n. \tag{5.15}$$

Throughout the proof we assume that there exists a monotone disjunction $\boldsymbol{d} \in \{0,1\}^n$ with at most $A$ attribute errors on the sequence $S^\ell$ and we let $\boldsymbol{u} = (1/k)\boldsymbol{d}$ be the probability vector specifying the target disjunction $\boldsymbol{d}$. As done in Theorem 5, for any instance-label pair $(\boldsymbol{x}_t, y_t)$ in the sequence $S^\ell$, let $a_t$ be the minimal number of bits in the Boolean instance $\boldsymbol{x}_t$ that have to be changed so that the target disjunction correctly labels the modified instance $\boldsymbol{x}'_t$. That is, $a_t = \sum_{i=1}^n |x_{t,i} - x'_{t,i}|$ where $\boldsymbol{x}'_t$ satisfies $\boldsymbol{u}(\boldsymbol{x}'_t) = y_t$ and $A = \sum_{t=1}^\ell a_t$. Using the update function (5.15) we obtain by simple manipulations

$$\begin{aligned}
\operatorname{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \operatorname{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) &= (\boldsymbol{u} \cdot \boldsymbol{x}_t) \ln \beta_{y_t} - \ln \left( \sum_{j=1}^n w_{t,j}(\beta_{y_t})^{x_{t,j}} \right) \\
&= (\boldsymbol{u} \cdot \boldsymbol{x}_t) \ln \beta_{y_t} - \ln \left( \sum_{j=1}^n w_{t,j}(1 - x_{t,j}(1 - \beta_{y_t})) \right) \\
&= (\boldsymbol{u} \cdot \boldsymbol{x}'_t) \ln \beta_{y_t} + (\ln \beta_{y_t}) \sum_{i=1}^n u_i(x_{t,i} - x'_{t,i}) \\
&\quad - \ln(1 + (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)(\beta_{y_t} - 1)).
\end{aligned} \tag{5.16}$$

Note that in the second equality we have used the fact that $x_{t,j} \in \{0,1\}$ and thus $(\beta_{y_t})^{x_{t,j}} = 1 - x_{t,j}(1-\beta_{y_t})$. Now, since $\boldsymbol{x}'_t$ is obtained from $\boldsymbol{x}_t$ by removing from it its attribute errors and $a_t = \sum_{i=1}^{n} |x_{t,i} - x'_{t,i}|$, it is not difficult to see that for the choice of $\beta_0$ and $\beta_1$ given in the theorem we have $(\ln \beta_{y_t}) \sum_{i=1}^{n} u_i(x_{t,i} - x'_{t,i}) = -(a_t/k) \ln \alpha$ and $(\boldsymbol{u} \cdot \boldsymbol{x}'_t) \ln \beta_{y_t} \geqslant (y_t/k) \ln \alpha$. Thus,

$$\mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) \geqslant \frac{y_t \ln \alpha}{k} - \frac{a_t \ln \alpha}{k} - \ln(1 + (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)(\beta_{y_t} - 1)). \tag{5.17}$$

Summing (5.17) over trials $t = 1, \ldots, \ell$ we get

$$\begin{aligned}
&\sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) \\
&\quad \geqslant -\frac{A \ln \alpha}{k} - \sum_{\{t: y_t = 0, \hat{y}_t = 1\}} \ln(1 - (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)(1 - 1/\alpha)) \\
&\qquad + \sum_{\{t:\, y_t = 1, \hat{y}_t = 0\}} \left( \frac{\ln \alpha}{k} - \ln(1 + (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)(\alpha - 1)) \right) \\
&\quad \geqslant -\frac{A \ln \alpha}{k} + \sum_{\{t:\, y_t = 0, \hat{y}_t = 1\}} (1 - 1/\alpha)(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) \\
&\qquad + \sum_{\{t: y_t = 1, \hat{y}_t = 0\}} \left( \frac{\ln \alpha}{k} - (\alpha - 1)(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) \right) \\
&\quad \geqslant -\frac{A \ln \alpha}{k} + \sum_{\{t: y_t = 0, \hat{y}_t = 1\}} (1 - 1/\alpha)\theta + \sum_{\{t: y_t = 1, \hat{y}_t = 0\}} \left( \frac{\ln \alpha}{k} - (\alpha - 1)\theta \right).
\end{aligned} \tag{5.18}$$

Here the second inequality follows from approximating the logarithms. The last inequality follows by observing that when $\hat{y}_t = 1$ we have $\boldsymbol{w}_t \cdot \boldsymbol{x}_t > \theta$ and when $\hat{y}_t = 0$ we have $\boldsymbol{w}_t \cdot \boldsymbol{x}_t \leqslant \theta$. Finally, for the choice of $\theta$ given in the theorem, $(1 - 1/\alpha)\theta = ((\ln \alpha)/k) - (\alpha - 1)\theta = (\ln \alpha)/(k(\alpha + 1))$ and inequality (5.18) yields

$$\sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) \geqslant \frac{\ln \alpha}{k(\alpha + 1)} M_{\texttt{MD-Bayes-NW}}(S^{\ell}) - \frac{A \ln \alpha}{k}. \tag{5.19}$$

Now, note that

$$\begin{aligned}
\sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) &= \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_1) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{\ell+1}) \\
&\leqslant \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_1).
\end{aligned} \tag{5.20}$$

The theorem now follows by combining the lower bound (5.19) with the upper bound (5.20) and solving for $M_{\texttt{MD-Bayes-NW}}(S^{\ell})$. □

Note that bound (5.14) of Theorem 9 has the same form as bound (4.16) derived for MD-Thresholded-BEG except that in the latter the relative entropy of (5.14) is replaced by the binary relative entropy.

To understand better how the algorithm's performance is affected by this different measure of progress, we now prove a bound for MD-Bayes-NW in the noise-free case. That is, we assume that the algorithm knows in advance that there exists a monotone disjunction which correctly classifies all the instances of $S^\ell$. For this case, the following theorem gives a bound of $O(k \ln n/k)$ for MD-Bayes-NW when learning monotone disjunctions of size $k$. A similar bound was first shown to us by Nick Littlestone [10] (see also [6, 13] for alternative proofs). The key insight in Littlestone's proof was to set the threshold $\theta$ proportional to $1/k$ rather than $\frac{1}{2}$. In contrast, the $\frac{1}{2}$ threshold version of MD-Bayes-NW[6] has a mistake bound of $O(k^2 \ln(n/k))$.

**Theorem 10.** *Let $n \geqslant 2$ and set $\beta_0 = 0$, $\beta_1 = e$ and $\theta = 1/(e\,k)$. Furthermore, let $w_{1,i} = 1/n$ for $i = 1, \ldots, n$. For all sequences $S^\ell = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ consistent with a monotone disjunction $\boldsymbol{d} \in \{0,1\}^n$, we have*

$$M_{\texttt{MD-Bayes-NW}}(S^\ell) \leqslant e\,k \ln \frac{n}{k}, \tag{5.21}$$

*where $k \geqslant 1$ is the size of the target disjunction $\boldsymbol{d}$ and $e$ is the base of the natural logarithm.*

**Proof** (**Sketch**). Similar to the proof of Theorem 9. For completeness, the proof is provided in Appendix E. □

By comparing the above bound with the bound (4.24) derived for MD-Thresholded-BEG in Theorem 6, we can see that bound (5.21) is smaller by $2\,e\,k$ than the bound for MD-Thresholded-BEG. Thus, at least in the noise-free case and when the algorithm knows in advance the size $k$ of the disjunction being learned, the bound for MD-Bayes-NW is better than the bound for MD-Thresholded-BEG.

Although the relationship between Normalized Winnow and its corresponding Bayes algorithm is analogous to the relationship between indirect Bayes-BEG and BEG, there are two subtle differences. Indirect Bayes-BEG uses a logarithmic function of the weights in its dot-product prediction rule, while Normalized Winnow and its corresponding Bayes algorithm both predict with the simple thresholded dot-product between the weights (or probabilities) and the instance. However, the predictions of the Bayes algorithm remain a simple thresholded dot-product only as long as the vector-labels are either $1^n$ or $0^n$. Vector-labels containing both 1s and 0s break the symmetry and the Bayes optimal prediction is no longer a dot product.

The technical details for relating Normalized Winnow to Bayesian methods are more complex than for the new classification variant of BEG, but the basic approach is the

[6] The MD-Bayes-NW algorithm with threshold $\theta = \frac{1}{2}$ was first analyzed in Nick Littlestone's thesis [8] and there it was called the Weighted Majority algorithm.

same. It is now natural to ask if the original (un-normalized) Winnow algorithm also has a corresponding Bayesian interpretation. Our attempts in this direction have been unsuccessful. Using Poisson distributions to encode the prior and posteriors over disjunctions appear promising since they correspond to the un-normalized relative entropy used to analyze Winnow [8, 1]. However, Winnow's weights do not seem to encode the proper Poisson posteriors.

## 6. Conclusions

The Winnow family of algorithms is surprisingly good at learning disjunctions in the relative mistake bound model. These algorithms are very efficient, using only $O(n)$ weights. The goal of this research is to gain a better understanding of this family by exploring its relationship to Bayesian methods. Although we have not yet answered this question for Winnow itself, we do have a Bayesian interpretation for the prediction and update rules used by Normalized Winnow and a new classification variant of BEG.

We started by investigating the assumptions necessary to encode the posteriors over monotone disjunctions kept by Bayes algorithms with only $O(n)$ weights. Our methods lead to computationally efficient algorithms which are motivated by a Bayesian analysis. For one of these algorithms, indirect `Bayes-BEG`, we have examined how its mistake-driven variant performs in the relative mistake bound model when learning disjunctions. In the noise free case we have shown that this variant has mistake bounds with the same form as the best known indirect algorithms for learning disjunctions. Further results imply that this algorithm can tolerate noise, but the complexity of its predictions makes the analysis difficult.

## Appendix A. Proof of Lemma 1

The proof is by induction on $t$. If $t=0$ then $S^t=\lambda$ and the thesis holds by **AS1**. Assume that $P(\boldsymbol{d}\mid S^{t-1})=\prod_{i=1}^{n}P(d_i\mid S_i^{t-1})$. We now show that the decomposition also holds for $S^t$. Using Bayes Rule and assumptions **AS2** through **AS4** it is not difficult to see that

$$\begin{aligned}
P(\boldsymbol{d}\mid S^{t-1},(\boldsymbol{x}_t,\boldsymbol{y}_t)) &= \frac{P(\boldsymbol{d}\mid S^{t-1})P((\boldsymbol{x}_t,\boldsymbol{y}_t)\mid S^{t-1},\boldsymbol{d})}{\sum_{\boldsymbol{d}'\in\{0,1\}^n}P((\boldsymbol{x}_t,\boldsymbol{y}_t)\mid S^{t-1},\boldsymbol{d}')P(\boldsymbol{d}'\mid S^{t-1})}\\
&= \frac{P(\boldsymbol{d}\mid S^{t-1})P(\boldsymbol{y}_t\mid \boldsymbol{x}_t,\boldsymbol{d})\,P(\boldsymbol{x}_t\mid \boldsymbol{d})}{\sum_{\boldsymbol{d}'\in\{0,1\}^n}P(\boldsymbol{y}_t\mid \boldsymbol{x}_t,\boldsymbol{d}')P(\boldsymbol{x}_t\mid \boldsymbol{d}')P(\boldsymbol{d}'\mid S^{t-1})}\\
&= \frac{\prod_{i=1}^{n}P(d_i\mid S_i^{t-1})P(y_{t,i}\mid x_{t,i},d_i)}{\sum_{\boldsymbol{d}'\in\{0,1\}^n}\prod_{i=1}^{n}P(y_{t,i}\mid x_{t,i},d_i')P(d_i'\mid S_i^{t-1})},
\end{aligned}\tag{A.1}$$

where the first equality follows from Bayes Rule, the second equality from assumption **AS2**, and the third equality follows from assumptions **AS3**, **AS4** and the inductive hypothesis. Now, since in the denominator of (A.1) we are summing over all $\boldsymbol{d}' \in \{0,1\}^n$, sum and product can be switched and thus (A.1) can equivalently be written as

$$P(\boldsymbol{d} \mid S^{t-1}, (\boldsymbol{x}_t, \boldsymbol{y}_t)) = \prod_{i=1}^{n} \left( \frac{P(d_i \mid S_i^{t-1}) P(y_{t,i} \mid x_{t,i}, d_i) P(x_{t,i} \mid d_i)}{\sum_{d_i' \in \{0,1\}} P(y_{t,i} \mid x_{t,i}, d_i') P(x_{t,i} \mid d_i') P(d_i' \mid S_i^{t-1})} \right)$$

$$= \prod_{i=1}^{n} \frac{P(S_i^{t-1} \mid d_i) P((x_{t,i}, y_{t,i}) \mid d_i) P(d_i \mid \lambda)}{\sum_{d_i' \in \{0,1\}} P(S_i^{t-1} \mid d_i') P((x_{t,i}, y_{t,i}) \mid d_i') P(d_i' \mid \lambda)},$$

where the first equality follows by observing that under the assumptions of Model $\mathscr{M}$ we have $P(x_{t,i} \mid d_i) = P(x_{t,i})$ and the second equality follows by applying Bayes Rule to $P(d_i \mid S_i^{t-1})$. Finally, observing that $P((x_{t,i}, y_{t,i}) \mid d_i) = P((x_{t,i}, y_{t,i}) \mid S_i^{t-1}, d_i)$ and that $P(S_i^{t-1} \mid d_i) P((x_{t,i}, y_{t,i}) \mid S_i^{t-1}, d_i) = P(S_i^{t-1}(x_{t,i}, y_{t,i}) \mid d_i)$ we obtain by simple manipulations

$$P(\boldsymbol{d} \mid S^{t-1}, (\boldsymbol{x}_t, \boldsymbol{y}_t)) = \prod_{i=1}^{n} \frac{P(S_i^{t-1}(x_{t,i}, y_{t,i}) \mid d_i) P(d_i \mid \lambda)}{\sum_{d_i' \in \{0,1\}} P(S_i^{t-1}(x_{t,i}, y_{t,i}) \mid d_i') P(d_i' \mid \lambda)}$$

$$= \prod_{i=1}^{n} \frac{P(S_i^{t-1}(x_{t,i}, y_{t,i}), d_i)}{P(S_i^{t-1}(x_{t,i}, y_{t,i}))} = \prod_{i=1}^{n} P(d_i \mid S_i^{t-1}(x_{t,i}, y_{t,i})).$$

This concludes the proof. □

## Appendix B. Details of the Proof of Theorem 1

Let us assume that $P(y_{t,i} \mid x_{t,i}, d_i)$ is a distribution for which identity (4.3) holds, i.e.

$$\frac{P(y_{t,i} \mid x_{t,i}, d_i = 1)}{\sum_{d_i' \in \{0,1\}} P(y_{t,i} \mid x_{t,i}, d_i') P(d_i' \mid S_i^{t-1})} = \frac{(\beta_{y_{t,i}})^{x_{t,i}}}{1 - w_{t,i} + w_{t,i} (\beta_{y_{t,i}})^{x_{t,i}}}. \tag{B.1}$$

We need to show that $P(y_{t,i} \mid x_{t,i}, d_i) = P_{\beta_0, \beta_1, \gamma}(y_{t,i} \mid x_{t,i}, d_i)$. For ease of notation, we let $a_{x_{t,i}} = P(y_{t,i} = 0 \mid x_{t,i}, d_i = 1)$ and $b_{x_{t,i}} = P(y_{t,i} = 0 \mid x_{t,i}, d_i = 0)$. Clearly $P(y_{t,i} = 1 \mid x_{t,i}, d_i = 1) = 1 - a_{x_{t,i}}$ and $P(y_{t,i} = 1 \mid x_{t,i}, d_i = 0) = 1 - b_{x_{t,i}}$. For (B.1) to hold $a_{x_{t,i}}$ and $b_{x_{t,i}}$ must satisfy identities (B.2) and (B.3) below

$$\frac{1 - a_{x_{t,i}}}{(1 - a_{x_{t,i}}) w_{t,i} + (1 - b_{x_{t,i}})(1 - w_{t,i})} = \frac{\beta_1^{x_{t,i}}}{1 - w_{t,i} + w_{t,i} \beta_1^{x_{t,i}}}, \tag{B.2}$$

$$\frac{a_{x_{t,i}}}{a_{x_{t,i}} w_{t,i} + b_{x_{t,i}} (1 - w_{t,i})} = \frac{\beta_0^{x_{t,i}}}{1 - w_{t,i} + w_{t,i} \beta_0^{x_{t,i}}}, \tag{B.3}$$

which follow from (B.1) by setting respectively $y_{t,i} = 1$ and $y_{t,i} = 0$. Now, two cases must be considered depending on the value of $x_{t,i}$.

We start by analyzing the case $x_{t,i}=0$. When in Eqs. (B.2) and (B.3) we set $x_{t,i}=0$ we obtain

$$\frac{1-a_0}{(1-a_0)w_{t,i}+(1-b_0)(1-w_{t,i})}=1 \quad \text{and} \quad \frac{a_0}{a_0 w_{t,i}+b_0(1-w_{t,i})}=1. \tag{B.4}$$

It is not difficult to see that for (B.4) to hold we need $a_0=b_0$ where $0<a_0,\ b_0<1$ which, in turn, implies

$$P(y_{t,i}=0\,|\,x_{t,i}=0,d_i=1)=P(y_{t,i}=0\,|\,x_{t,i}=0,d_i=0)=\gamma, \tag{B.5}$$

$$P(y_{t,i}=1\,|\,x_{t,i}=0,d_i=1)=P(y_{t,i}=1\,|\,x_{t,i}=0,d_i=0)=1-\gamma, \tag{B.6}$$

for some $0<\gamma<1$.

Next, we consider the case $x_{t,i}=1$. Again, by setting $x_{t,i}=1$ in (B.2) and (B.3) we obtain

$$\frac{1-a_1}{(1-a_1)w_{t,i}+(1-b_1)(1-w_{t,i})}=\frac{\beta_1}{1-w_{t,i}+w_{t,i}\beta_1}, \tag{B.7}$$

$$\frac{a_1}{a_1 w_{t,i}+b_1(1-w_{t,i})}=\frac{\beta_0}{1-w_{t,i}+w_{t,i}\beta_0}. \tag{B.8}$$

Solving (B.8) with respect to $a_1$ yields $a_1=\beta_0 b_1$. Substituting this value in (B.7) and solving with respect to $b_1$ we then obtain

$$b_1=\frac{\beta_1-1}{\beta_1-\beta_0} \quad \text{and} \quad a_1=\beta_0\frac{\beta_1-1}{\beta_1-\beta_0}. \tag{B.9}$$

Thus, by definition of $a_1$ and $b_1$ it follows that

$$P(y_{t,i}=0\,|\,x_{t,i}=1,d_i=1)=\beta_0\frac{\beta_1-1}{\beta_1-\beta_0},$$

$$P(y_{t,i}=1\,|\,x_{t,i}=1,d_i=1)=\beta_1\frac{1-\beta_0}{\beta_1-\beta_0},$$

$$P(y_{t,i}=0\,|\,x_{t,i}=1,d_i=0)=\frac{\beta_1-1}{\beta_1-\beta_0},$$

$$P(y_{t,i}=1\,|\,x_{t,i}=1,d_i=0)=\frac{1-\beta_0}{\beta_1-\beta_0}.$$

It is now easy to see that the distribution $P(y_{t,i}\,|\,x_{t,i},d_i)$ defined by (B.5), (B.6) and the above equations is equal to $P_{\beta_0,\beta_1,\gamma}(y_{t,i}\,|\,x_{t,i},d_i)$ and this concludes the proof. □

## Appendix C. Details of the Proof of Theorem 4

For ease of notation we let $c=(q^{3/2}-1+\varepsilon)/(1+q)$. Throughout the proof the parameters $\beta_1$ and $\gamma$ are fixed to the values given in the theorem, i.e.,

$$\beta_1=1+c \ \text{ and } \ \frac{\gamma}{1-\gamma}=q^{1/(n-1)},$$

where $q = \frac{16}{10}$ and $\varepsilon = \frac{9}{10}$. By letting $Z_{\text{low}} = \ln(\gamma(1-\beta_0)/(c(1-\gamma)))$ and $\psi_{t,i} = \ln((1 + w_{t,i}c)/(1-w_{t,i}(1-\beta_0)))$, we can rewrite the weight $z_{t,i}$ used in the dot product prediction conveniently as $z_{t,i} = Z_{\text{low}} + \psi_{t,i}$. Since $Z_{\text{low}}$ is constant throughout the learning process, it follows that $z_{t,i}$ increases/decreases if and only if the same holds for the corresponding $\psi_{t,i}$. Thus, rather than analyzing the change in the total weight $Z_t = \sum_{i=1}^{n} z_{t,i}$ we can more conveniently analyze the change in the total weight $\Psi_t = \sum_{i=1}^{n} \psi_{t,i}$. Furthermore, observing that for any $i \in \{1,\ldots,n\}$ and $t = 1,\ldots,\ell$, $\psi_{t,i} > 0$, $\lim_{w_{t,i}\to 0} \psi_{t,i} = 0$ and $\psi_{t,i}$ only increases/decreases during promotion/demotion steps it follows that

$$0 < \Psi_t \leqslant \Psi_1 + p\Psi_{\text{gain}} - d\Psi_{\text{lost}}, \tag{C.1}$$

where $\Psi_{\text{gain}}$ is an upper bound on the total $\psi$-weight gained during a promotion step, $\Psi_{\text{lost}}$ is a lower bound on the total $\psi$-weight lost during a demotion step and $p$ and $d$ are, respectively, the number of promotion and demotion steps incurred by the algorithm. Now, solving (C.1) with respect to $d$ we obtain $d \leqslant (\Psi_1/\Psi_{\text{lost}}) + p(\Psi_{\text{gain}}/\Psi_{\text{lost}})$. Hence, the number of mistakes made by `MD-Bayes-BEG` can be upper bounded by

$$M_{\texttt{MD-Bayes-BEG}}(S^{\ell}) \leqslant \frac{\Psi_1}{\Psi_{\text{lost}}} + p\left(1 + \frac{\Psi_{\text{gain}}}{\Psi_{\text{lost}}}\right). \tag{C.2}$$

We now estimate the quantities in the right hand side of (C.2). For the total initial $\psi$-weight, $\Psi_1 = \sum_{i=1}^{n} \psi_{1,i}$, it is not difficult to see that when $w_{1,i} = 1/n$ we have

$$\begin{aligned}\Psi_1 &= n\ln\left(1 + \frac{c+1-\beta_0}{n-1+\beta_0}\right)\\ &\leqslant \frac{n(c+1-\beta_0)}{n-1+\beta_0}\\ &\leqslant \frac{2nc}{n-c} = \frac{2n(q^{3/2}-1+\varepsilon)}{nq+n-q^{3/2}+1-\varepsilon},\end{aligned} \tag{C.3}$$

where the first inequality follows by observing that for $x \geqslant 0$ we have $\ln(1+x) \leqslant x$ and the second inequality from the fact that $(c+1-\beta_0)/(n-1+\beta_0)$ is decreasing in $\beta_0$ and for $\beta_0$ assumed in the theorem we have $\beta_0 = 1 - c/q^{1/(2(n-1))} > (1-c)$. Furthermore, since $2nc/(n-c)$ is decreasing in $n$, the maximizing $n$ of (C.3) is $n = 2$ and this yields

$$\Psi_1 \leqslant \frac{4(q^{3/2}-1+\varepsilon)}{2q+3-q^{3/2}-\varepsilon}. \tag{C.4}$$

Next, we consider the total $\psi$-weight lost during a demotion step. To derive a lower bound for it we argue as follows. Recall that during a demotion step the algorithm predicts 1 (i.e., $\boldsymbol{x}_t \cdot \boldsymbol{z}_t > \theta$) while the outcome is $y_t = 0$. Since $z_{t,i} = Z_{\text{low}} + \psi_{t,i}$, $\boldsymbol{x}_t \cdot \boldsymbol{z}_t > \theta$ implies

$$\sum_{i=1}^{n} x_{t,i}\psi_{t,i} > \theta - \sum_{i=1}^{n} x_{t,i} Z_{\text{low}}. \tag{C.5}$$

Using the fact that for $\beta_0$ assumed in the theorem we have $Z_{\text{low}} = (1/2)\ln(\gamma/(1-\gamma)) > 0$ and that the number of irrelevant attributes is at most $(n-k)$, it is not difficult to see that

$$\sum_{i=1}^{n} x_{t,i} Z_{\text{low}} \leqslant (n-k) Z_{\text{low}} = \frac{n-k}{2} \ln\left(\frac{\gamma}{1-\gamma}\right).$$

This in conjunction with (C.5) implies

$$\begin{aligned}\sum_{i=1}^{n} x_{t,i}\psi_{t,i} &> n \ln\frac{\gamma}{1-\gamma} - \frac{n-k}{2}\ln\frac{\gamma}{1-\gamma} \\ &= n\ln\frac{\gamma}{1-\gamma}\left(1 - \frac{n-k}{2n}\right) \\ &= \theta\left(1 - \frac{n-k}{2n}\right) = \theta\left(\frac{1}{2} + \frac{k}{2n}\right). \end{aligned} \tag{C.6}$$

Notice that the lower bound $\theta(1/2 + k/(2n))$, rather than $\theta$, results by using in the prediction rule the weights $z_{t,i} = Z_{\text{low}} + \psi_{t,i}$ where $Z_{\text{low}} > 0$. Now, if we can show that for any attribute $x_{t,i}$ such that $x_{t,i} = 1$, $\psi_{t+1,i} \leqslant d\psi_{t,i}$ where $d < 1$, then the total weight lost at trial $t$ can be lower bounded by

$$\begin{aligned}\Psi_{\text{lost},t} &= \sum_{i=1}^{n} x_{t,i}(\psi_{t,i} - \psi_{t+1,i}) \geqslant \sum_{i=1}^{n} x_{t,i}(\psi_{t,i} - d\psi_{t,i}) \\ &= (1-d)\sum_{i=1}^{n} x_{t,i}\psi_{t,i} \geqslant (1-d)\theta\left(\frac{1}{2} + \frac{k}{2n}\right), \end{aligned} \tag{C.7}$$

where the second inequality follows from (C.6). Observe that in (C.7) the factor $(1-d) < 1$ is the result of using a $\beta_0 > 0$ rather than a $\beta_0 = 0$. We now show that for any attribute $x_{t,i}$ such that $x_{t,i} = 1$, $\psi_{t+1,i} \leqslant d\psi_{t,i}$ where $d < 1$.

First note that when $x_{t,i} = 1$ the corresponding weight $w_{t,i}$ is updated to $w_{t+1,i} = w_{t,i}\beta_0/(1 - w_{t,i} + w_{t,i}\beta_0)$. Substituting $w_{t+1,i}$ into $\psi_{t+1,i}$ we obtain

$$\frac{\psi_{t+1,i}}{\psi_{t,i}} = \frac{\ln((1 - w_{t,i} + w_{t,i}\beta_0(1+c))/(1 - w_{t,i}(1-\beta_0^2)))}{\ln((1 + w_{t,i}c)/(1 - w_{t,i}(1-\beta_0)))}. \tag{C.8}$$

Since for $1 - c < \beta_0 \leqslant 1 - c/\sqrt{q}$ and $w_{t,i} \in (0,1)$ the derivative of $\psi_{t+1,i}/\psi_{t,i}$ with respect to $\beta_0$ is strictly positive, it follows that $\psi_{t+1,i}/\psi_{t,i}$ is maximized when $\beta_0 = 1 - c/\sqrt{q}$ and we obtain $\psi_{t+1,i}/\psi_{t,i} \leqslant G(w_{t,i}, \varepsilon, q)$ where

$$G(w_{t,i}, \varepsilon, q) = \frac{\ln((1 - w_{t,i} + w_{t,i}(1+c)(1 - \frac{c}{\sqrt{q}}))/(1 - w_{t,i}(1 - (1 - \frac{c}{\sqrt{q}})^2)))}{\ln((1 + w_{t,i}c)/(1 - w_{t,i}\frac{c}{\sqrt{q}}))}. \tag{C.9}$$

Next, observe that an adversary can force the algorithm to incorrectly predict 0 on an appropriately chosen positive example only when the $w$-weight assigned to a relevant attribute is $\leqslant w_{\text{threshold}}$ where $w_{\text{threshold}} = (cq - 1 + \beta_0)/(c(1-\beta_0)(1+q))$. This implies that at each trial $t$ the weight $w_{t,i}$ of a relevant attribute $x_{t,i}$ can be upper bounded by

$$\begin{aligned} w_{t,i} &\leqslant \frac{w_{\text{threshold}}\beta_1}{1 - w_{\text{threshold}} + w_{\text{threshold}}\beta_1} \\ &\leqslant \frac{(q^{3/2}-1)(1+c)}{c(1+\sqrt{q})q} \\ &= \frac{(q+q^{3/2}+\varepsilon)(q^{3/2}-1)}{q(1+\sqrt{q})(q^{3/2}-1+\varepsilon)} = w_{\text{up}}(q,\varepsilon), \end{aligned} \tag{C.10}$$

where the second inequality follows from the fact that $w_{\text{threshold}}\beta_1/(1 - w_{\text{threshold}} + w_{\text{threshold}}\beta_1)$ is an increasing function of $\beta_0$ and that for $\beta_0$ assumed in the theorem we have $\beta_0 \leqslant 1 - c/\sqrt{q}$. Furthermore, since for $w_{t,i} \in (0,1)$ we have $\partial/\partial w_{t,i} G(w_{t,i},\varepsilon,q) > 0$, the function $G(w_{t,i},\varepsilon,q)$ is maximized when $w_{t,i} = w_{\text{up}}(q,\varepsilon)$. Substituting this value in $G(w_{t,i},\varepsilon,q)$ and using the fact that $\psi_{t+1,i}/\psi_{t,i} \leqslant G(w_{t,i},\varepsilon,q)$, we obtain $\psi_{t+1,i} \leqslant H(q,\varepsilon)\psi_{t,i}$ where

$$H(q,\varepsilon) = \frac{\ln\left(\dfrac{1 - w_{\text{up}}(q,\varepsilon) + w_{\text{up}}(q,\varepsilon)(1+c)(1-\frac{c}{\sqrt{q}})}{1 - w_{\text{up}}(q,\varepsilon)(1-(1-\frac{c}{\sqrt{q}})^2)}\right)}{\ln\left(\dfrac{1 + w_{\text{up}}(q,\varepsilon)c}{1 - w_{\text{up}}(q,\varepsilon)\frac{c}{\sqrt{q}}}\right)}.$$

Setting $d = H(q,\varepsilon)$ in (C.7) we obtain

$$\Psi_{\text{lost}} = (1 - H(q,\varepsilon))\left(\frac{1}{2} + \frac{k}{2n}\right)\theta. \tag{C.11}$$

We now bound the total $\psi$-weight gained during a promotion step. Proceeding similarly to the previous case but with a promotion step, i.e., $\boldsymbol{x}_t \cdot \boldsymbol{z}_t \leqslant \theta$ while $y_t = 1$, we obtain

$$\begin{aligned} \sum_{i=1}^{n} x_{t,i}\psi_{t,i} &\leqslant \theta - \sum_{i=1}^{n} x_{t,i} Z_{\text{low}} \\ &\leqslant \theta - Z_{\text{low}} = \theta\left(1 - \frac{1}{2n}\right), \end{aligned} \tag{C.12}$$

where the second inequality follows by observing that $\sum_{i=1}^{n} x_{t,i} \geqslant 1$ and that for $\beta_0$ assumed in the theorem we have $Z_{\text{low}} > 0$ and the equality from the fact that $Z_{\text{low}} = \theta/(2n)$. As done for the previous case, if we can show that for each attribute $x_{t,i}$ such that $x_{t,i} = 1$, $\psi_{t+1,i} \leqslant d\psi_{t,i}$ where $d > 1$, then the total weight gained at trial $t$ can be upper

bounded by

$$\Psi_{\text{gain},t} = \sum_{i=1}^{n} x_{t,i}(\psi_{t+1,i} - \psi_{t,i}) \leqslant \sum_{i=1}^{n} x_{t,i}(d-1)\psi_{t,i}$$

$$= (d-1)\sum_{i=1}^{n} x_{t,i}\psi_{t,i} \leqslant (d-1)\left(1 - \frac{1}{2n}\right)\theta, \tag{C.13}$$

where the right hand side of (C.13) follows directly from (C.12). Again, if $x_{t,i} = 1$ the corresponding weight $w_{t,i}$ is updated to $w_{t+1,i} = w_{t,i}(1+c)/(1+w_{t,i}c)$. Substituting $w_{t+1,i}$ into $\psi_{t+1,i}$ we obtain

$$\frac{\psi_{t+1,i}}{\psi_{t,i}} = \frac{\ln((1+w_{t,i}c(2+c))/(1-w_{t,i}+w_{t,i}\beta_0(1+c)))}{\ln((1+w_{t,i}c)/(1-w_{t,i}(1-\beta_0)))}. \tag{C.14}$$

Since $\frac{\partial}{\partial w_{t,i}}(\frac{\psi_{t+1,i}}{\psi_{t,i}}) < 0$, the ratio $\psi_{t+1,i}/\psi_{t,i}$ can be upper bounded by $\lim_{w_{t,i}\to 0} \psi_{t+1,i}/\psi_{t,i} = 1+c$ and thus $\psi_{t+1,i} \leqslant (1+c)\psi_{t,i}$. Then setting $d = 1+c$ in (C.13) yields

$$\Psi_{\text{gain}} = c\left(1 - \frac{1}{2n}\right)\theta. \tag{C.15}$$

We now bound the number of promotion steps incurred by the algorithm. First recall that for any attribute $i \in \{1,\ldots,n\}$ and $t = 1,\ldots,\ell$, $w_{t,i} \leqslant w_{\text{up}}(q,\varepsilon)$ where $w_{\text{up}}(q,\varepsilon)$ is given by (C.10). By simple manipulation it is not difficult to see that if $x_{t,i} = 1$ and $w_{t,i} = 1 - 1/(1+(1+c)^{-\xi})$ then at the end of a promotion step the updated weight is $w_{t+1,i} = 1 - 1/(1+(1+c)^{-\xi+1})$. By expressing the initial and final weights of a relevant attribute in this form, we have that the number of promotion steps per relevant attribute can be upper bounded by $\ln_{c+1}((n-1)/(1/w_{\text{up}}(q,\varepsilon) - 1)) + 1$ and thus

$$p \leqslant k\left(1 + \ln_{c+1}\frac{n-1}{1/w_{\text{up}}(q,\varepsilon) - 1}\right). \tag{C.16}$$

Substituting (C.4), (C.11), (C.15) and (C.16) in (C.2) by tedious but simple manipulation we finally obtain

$$M_{\texttt{MD-Bayes-BEG}}(S^{\ell}) \leqslant C_1 + C_2 k \ln(n-1) + C_3 k, \tag{C.17}$$

where

$$C_1 = \frac{8(q^{3/2} - 1 + \varepsilon)(n-1)}{(2q + 3 - q^{3/2} - \varepsilon)(n+k)(1 - H(q,\varepsilon))\ln q},$$

$$C_2 = \frac{1}{\ln(1+c)}\left(1 + \frac{c(2n-1)}{(1-H(q,\varepsilon))(n+k)}\right),$$

$$C_3 = \left(1 - \frac{\ln(1/w_{\text{up}}(q,\varepsilon) - 1)}{\ln(1+c)}\right)\left(1 + \frac{c(2n-1)}{(1-H(q,\varepsilon))(n+k)}\right).$$

Bound (4.14) then immediately follows by setting $q = \frac{16}{10}$ and $\varepsilon = \frac{9}{10}$ in (C.17). □

## Appendix D. Details of the Proof of Theorem 6

We only prove the first bound. The second bound can be derived in a similar way by using the probability vector $\boldsymbol{u} = (1/k)\boldsymbol{d}$ as the comparison vector for the target disjunction rather than the binary vector $\boldsymbol{d}$.

Throughout the proof we assume that there exists a monotone disjunction $\boldsymbol{d} \in \{0,1\}^n$ that correctly classifies the instances in $S^\ell$. Thus, for $t = 1,\ldots,\ell$ we have $\boldsymbol{d}(\boldsymbol{x}_t) = y_t$. When $\hat{y}_t = 1$ and $y_t = 0$, we have $\boldsymbol{w}_t \cdot \boldsymbol{x}_t > \theta$ and $\boldsymbol{d} \cdot \boldsymbol{x}_t = 0$, respectively. Since $\beta_0 = 0$, the progress of the `MD-Thresholded-BEG` algorithm given by Eq. (4.18) reduces to

$$\begin{aligned} \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) &= -\sum_{i=1}^{n} \ln(1 - w_{t,i}x_{t,i}) \\ &\geqslant \sum_{i=1}^{n} w_{t,i}x_{t,i} > \theta. \end{aligned} \tag{D.1}$$

When $\hat{y}_t = 0$ and $y_t = 1$, we have $\boldsymbol{w}_t \cdot \boldsymbol{x}_t \leqslant \theta$ and $\boldsymbol{d} \cdot \boldsymbol{x}_t \geqslant 1$, respectively. Since $\beta_1 > 1$, Eq. (4.18) reduces to

$$\begin{aligned} \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) &= (\boldsymbol{d} \cdot \boldsymbol{x}_t) \ln \beta_1 - \sum_{i=1}^{n} \ln(1 + w_{t,i}x_{t,i}(\beta_1 - 1)) \\ &\geqslant \ln \beta_1 - (\beta_1 - 1)(\boldsymbol{w}_t \cdot \boldsymbol{x}_t) \\ &\geqslant \ln \beta_1 - (\beta_1 - 1)\theta. \end{aligned} \tag{D.2}$$

Now, setting $\theta = \ln \beta_1 - (\beta_1 - 1)\theta$ so that the progress for both type of mistakes is the same, yields $\theta = \ln(\beta_1)/\beta_1$ and with this choice of $\theta$

$$\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) \geqslant \frac{\ln \beta_1}{\beta_1}.$$

Maximizing the right hand side with respect to $\beta_1$ gives $\beta_1 = e$. With this choice of $\beta_1$ we obtain $(\mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1})) \geqslant 1/e$, and summing over trials $t = 1,\ldots,\ell$ we get

$$\sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) \geqslant \frac{M_{\texttt{MD-Thresholded-BEG}}(S^\ell)}{e}. \tag{D.3}$$

Bound (4.23) then follows from (D.3) and from the following upper bound on the total progress made by the algorithm.

$$\begin{aligned} \sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{t+1}) &= \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_1) - \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_{\ell+1}) \\ &\leqslant \mathrm{dist}_{\mathrm{bre}}(\boldsymbol{d}, \boldsymbol{w}_1) \\ &= (n-k)\ln\frac{n}{n-1} + k \ln n \\ &\leqslant 1 + k \ln n. \qquad \square \end{aligned}$$

## Appendix E. Details of the Proof of Theorem 10

It is similar to the proof of Theorem 9, except that now the progress of the algorithm is measured towards a comparison vector $\boldsymbol{u} = (1/k)\boldsymbol{d}$ that correctly classifies the instances in $S^\ell$. When $\hat{y}_t = 1$ and $y_t = 0$ we have $\boldsymbol{w}_t \cdot \boldsymbol{x}_t > \theta$ and $\boldsymbol{u} \cdot \boldsymbol{x}_t = 0$, respectively. Since $\beta_0 = 0$, the progress of the `MD-Bayes-NW` algorithm given by Eq. (5.16) reduces to

$$\begin{aligned}
\mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) &= -\ln\left(\sum_{j=1}^{n} w_{t,j}(1 - x_{t,j})\right) \\
&= -\ln(1 - \boldsymbol{w}_t \cdot \boldsymbol{x}_t) \\
&\geqslant \boldsymbol{w}_t \cdot \boldsymbol{x}_t > \theta. \qquad \text{(E.1)}
\end{aligned}$$

When $\hat{y}_t = 0$ and $y_t = 1$, we have $\boldsymbol{w}_t \cdot \boldsymbol{x}_t \leqslant \theta$ and $\boldsymbol{u} \cdot \boldsymbol{x}_t \geqslant 1/k$, respectively. Since $\beta_1 > 1$ Eq. (5.16) reduces to

$$\begin{aligned}
\mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) &= (\boldsymbol{u} \cdot \boldsymbol{x}_t)\ln\beta_1 - \ln\left(\sum_{j=1}^{n} w_{t,j}\beta_1^{x_{t,j}}\right) \\
&= (\boldsymbol{u} \cdot \boldsymbol{x}_t)\ln\beta_1 - \ln(1 + (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)(\beta_1 - 1)) \\
&\geqslant (\boldsymbol{u} \cdot \boldsymbol{x}_t)\ln\beta_1 - (\boldsymbol{w}_t \cdot \boldsymbol{x}_t)(\beta_1 - 1) \\
&\geqslant \frac{1}{k}\ln\beta_1 - (\beta_1 - 1)\theta. \qquad \text{(E.2)}
\end{aligned}$$

Now, setting $(1/k)\ln\beta_1 - (\beta_1 - 1)\theta = \theta$ so that the progress for both type of mistakes is the same, yields $\theta = \ln(\beta_1)/(k\beta_1)$ and thus,

$$\mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) \geqslant \frac{\ln\beta_1}{k\beta_1}.$$

Finally, maximizing the right hand side gives $\beta_1 = e$. With this choice of $\beta_1$ we obtain $(\mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1})) \geqslant 1/(ek)$, and summing over trials $t = 1, \ldots, \ell$ we get

$$\sum_{t=1}^{\ell} \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1}) \geqslant \frac{M_{\texttt{MD-Bayes-NW}}(S^\ell)}{ek}. \qquad \text{(E.3)}$$

Bound (5.21) then follows from (E.3) and from the fact that

$$\begin{aligned}
\sum_{t=1}^{\ell} (\mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_t) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{t+1})) &= \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_1) - \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_{\ell+1}) \\
&\leqslant \mathrm{dist}_{\mathrm{re}}(\boldsymbol{u}, \boldsymbol{w}_1) = \ln\frac{n}{k}. \qquad \square
\end{aligned}$$

## References

[1] P. Auer, M.K. Warmuth, Tracking the best disjunction, J. Mach. Learning, Special issue on concept drift 32(2) (1998).
[2] T. Bylander, The binary exponentiated gradient algorithm for learning linear functions. Unpublished manuscript, private communication, 1997.
[3] P. Domingo, M. Pazzani, Beyond independence: conditions for the optimality of the simple Bayesian classifier, Proc. 13th International Conference on Machine Learning, 1996, pp. 105–112.
[4] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[5] C. Gentile, M.K. Warmuth, Hinge Loss and Average Margin. Advances in Neural Information Processing Systems, Morgan Kaufmann, Los Altos, CA, 1999, pp. 225–231.
[6] A. Grove, N. Littlestone, D. Schuurmans, General Convergence results for linear discriminant updates, Mach. Learning, to appear (earlier version in Proceedings of COLT 1997).
[7] N. Littlestone, Learning when irrelevant attributes abound: a new linear-threshold algorithm, Mach. Learning 2 (1988) 285–318 (earlier version in FOCS87).
[8] N Littlestone, Mistake bounds and logarithmic linear-threshold learning algorithms, Ph.D. Thesis, Technical Report UCSC-CRL-89-11, University of California Santa Cruz, 1989.
[9] N. Littlestone, Comparing several linear-threshold learning algorithms on tasks involving superfluous attributes, Proc. 12th International Conference on Machine Learning, Morgan Kaufmann, Los Altos, CA, 1995, pp. 353–361.
[10] N. Littlestone, Private Communication.
[11] N. Littlestone, Mistake-driven Bayes sports: bounds for symmetric apobayesian learning algorithms, Technical report, NEC Research Institute, Princeton, NJ, 1996.
[12] N. Littlestone, C. Mesterharm, An Apobayesian Relative of Winnow, in: Advances in Neural Information Processing Systems, Morgan Kaufmann, Los Altos, CA, 1997, pp. 204–210.
[13] N. Littlestone, C. Mesterharm, A simulation study of winnow and related learning algorithms. Unpublished manuscript, 1999.
[14] N. Littlestone, M.K. Warmuth, The weighted majority algorithm, Inform. Comput. 108 (1994) 212–261 (earlier version in FOCS89).