Direct and Indirect Algorithms for On-line Learning of Disjunctions*

D. P. Helmbold, S. Panizza, and M. K. Warmuth

Department of Computer Science
University of California at Santa Cruz
dph/panizza/manfred@cse.ucsc.edu

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) 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 the 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. Beside 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.

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, and then is told the correct classification for the instance.

Most on-line learning algorithms use a set of weights or parameters to represent their current hypothesis. 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

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

P. Fischer and H.U. Simon (Eds.): EuroCOLT'99, LNAI 1572, pp. 138–152, 1999.

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) [Lit89,Lit95].

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 variable. Surprisingly these more efficient algorithms learn disjunctions almost as well as the direct algorithms. The first such indirect algorithm was Littlestone's Winnow algorithm [Lit88,Lit89].

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 [Lit88,Lit89].

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) [LW94] algorithm's weights are posterior probabilities over the set of disjunctions of up to k variables 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 interpretation 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 [Lit95a]) and a new classification variant of Bylander's BEG algorithm[2] [Byl97] (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. The mistake-driven variant has performed better in experiments, but both versions are very sensitive to redundant attributes and neither performs as well as Winnow [Lit95].

The precursor of this research is a paper by Nick Littlestone [Lit96] (see also [LM97]) in which he uses a Bayesian approach to derive an indirect prediction algorithm, the Singly Variant Bayes algorithm (SVB), 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 only for the noise-free case, and Winnow's bound is much better when learning disjunctions.

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 specialize the presentation to derive the linear threshold classification algorithm related to Bylander's BEG algorithm [Byl97]. In Section 5 we briefly describe how the same technique can be applied to obtain a Bayesian interpretation of the normalized variant of Winnow.

[1] Normalized Winnow is identical to Winnow except that for computing its linear threshold predictions it uses the normalized instead of the un-normalized weights.

[2] Throughout this paper we call the algorithm using the update function of Figure 1 BEG because it is related to the update used by Bylander's Binary Exponentiated Gradient algorithm [Byl97] 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 Figure 1. This "linear hinge loss" can be used to motivate other linear threshold classification algorithms such as the Perceptron algorithm and Winnow [GW98].

2 An Overview of On-line Learning of Disjunctions

In the Mistake Bound model introduced by Littlestone [Lit88,Lit89], 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 variables as the comparison class. Such disjunctions are boolean formulas of the form $x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k}$ 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 variables are in the disjunction. For example, when $n = 5$ we will specify the disjunction $x_1 \vee x_3$ by 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 $\boldsymbol{d}(\boldsymbol{x}) = 1$ if $\boldsymbol{d} \cdot \boldsymbol{x} \geq 1$ and 0 otherwise.

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 [Lit89]).

3 The Direct Bayes Algorithm for Disjunctions

It is straightforward to apply Bayes methods (see, e.g. [DH73]) 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 from some prior distribution $P(\cdot \mid \lambda)$ on the space of all monotone disjunctions over n variables, where λ 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 $P(\cdot|\boldsymbol{d})$ 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 $P(y_t \mid \boldsymbol{x}_t, \boldsymbol{d}) = \nu^{|y_t - \boldsymbol{d}(\boldsymbol{x}_t)|}\ (1 - \nu)^{(1 - |y_t - \boldsymbol{d}(\boldsymbol{x}_t)|)}$. In

[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 [LW94].

[4] Again, the factor of two multiplying the number of attribute errors disappears when a probabilistic prediction is allowed [AW98].

other words, each label $y_1, \ldots y_\ell$ of the sequence of examples S^ℓ differs from that predicted by the selected disjunction with a probability that depends on an arbitrary but fixed "noise rate" $\nu \in (0, 1/2)$.

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

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

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 ν produce different versions of the Bayes optimal predictor (1). For instance, if $\beta < 1$ and $\nu = \beta/(\beta + 1)$, 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 β.

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 [Byl97]. 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. Our approach uses an expanded label space where each variable 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, allowing an easy factorization of the posteriors. 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 to solve the original disjunction problem by simply converting the binary labels 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 unable to prove relative loss 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 Booleans and vector-labels so that the predictions made by the algorithm are the Boolean predictions required for the disjunction problem. In Section 3 we assumed that the unknown sequence $S^\ell = (\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell)$ 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 $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ of the sequence S^ℓ at random according to some probability distribution $P(\cdot|\boldsymbol{d})$ on $\{0,1\}^n \times \{0,1\}^n$. However, here we assume that the probability distributions of the model satisfy the following assumptions.

Model $\mathcal{M}$

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 $\mathcal{M}$" are designed so that the posterior probabilities over disjunctions have the following product form.

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

$$P(\boldsymbol{d} \mid S^t) = \prod_{i=1}^n P(d_i \mid S_i^t), \tag{2}$$

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

Proof. 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 **AS1** through **AS4** it is not difficult to see that

$$\begin{aligned} P(\boldsymbol{d} \mid S^{t-1}, (\boldsymbol{x}_t, \boldsymbol{y}_t)) &= \frac{P(\boldsymbol{d} \mid S^{t-1})\; P((\boldsymbol{x}_t, \boldsymbol{y}_t) \mid S^{t-1}, \boldsymbol{d})}{\sum_{\boldsymbol{d}' \in \{0,1\}^n} P((\boldsymbol{x}_t, \boldsymbol{y}_t) \mid S^{t-1}, \boldsymbol{d}')\; P(\boldsymbol{d}' \mid S^{t-1})} \\ &= \frac{P(\boldsymbol{d} \mid S^{t-1})\; P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{d})\; P(\boldsymbol{x}_t \mid \boldsymbol{d})}{\sum_{\boldsymbol{d}' \in \{0,1\}^n} P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, \boldsymbol{d}')\; P(\boldsymbol{x}_t \mid \boldsymbol{d}')\; P(\boldsymbol{d}' \mid S^{t-1})} \\ &= \frac{\prod_{i=1}^n P(d_i \mid S_i^{t-1})\; P(y_{t,i} \mid x_{t,i}, d_i)}{\sum_{\boldsymbol{d}' \in \{0,1\}^n} \prod_{i=1}^n P(y_{t,i} \mid x_{t,i}, d_i')\; P(d_i' \mid S_i^{t-1})}, \end{aligned} \tag{3}$$

where the first equality follows from Bayes Rule, the second equality from assumptions **AS2** and the third equality follows from assumptions **AS3**, **AS4** and the inductive hypothesis. Now, since in the denominator of (3) we are summing over all $\boldsymbol{d}' \in \{0,1\}^n$, sum and product can be switched and thus (3) 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 in the first equality we have used assumption **AS3** and in the second equality we have applied 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. □

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 further proceeding 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.

Both methodologies make some simplifying assumptions regarding the generation of the instances and labels that allow them to use only $O(n)$ time per trial when learning disjunctions. 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,

$$P(\boldsymbol{x}_t \mid y_t) = \prod_{i=1}^{n} p(x_{t,i} \mid y_t). \tag{4}$$

However, Naive Bayes makes no use of the fact that the examples are generated by some target disjunctions. Our model allows the algorithm to track the posterior probabilities of the various disjunctions. Despite the simple assumption (4), Domingos and Pazzani [DP96] 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 [Lit95] show that Naive Bayes is not optimal in the relative mistake bound setting even when it is run in a conservative way.

An Update Rule for the Posterior Probabilities We now consider a particular family of distributions, $\{P_{\beta_0,\beta_1,\gamma}(y \mid x,d)\}_{x,y,d\in\{0,1\}}$, for which the weights encoding the Bayes posteriors are easily updated. This family has the noise parameters $0 < \gamma < 1$, $0 < \beta_0 < 1$, 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 γ is the probability that the label is flipped to 1 when $x = 0$, and the β parameters jointly encode different noise probabilities for the case when $x = 1,\ d = 1$, and the case when $x = 1,\ d = 0$.
After seeing a new example, the weights encoding the posterior are updated as in Bylander's BEG algorithm.

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

$$P(\boldsymbol{d} \mid S) = \prod_{i=1}^{n} w_i^{d_i}(1-w_i)^{1-d_i}, \qquad P(\boldsymbol{d} \mid S,(\boldsymbol{x},\boldsymbol{y})) = \prod_{i=1}^{n} \tilde{w}_i^{d_i}(1-\tilde{w}_i)^{1-d_i} \quad (5)$$

$$\textit{where} \quad \tilde{w}_i = w_i\frac{(\beta_{y_i})^{x_i}}{1-w_i+w_i(\beta_{y_i})^{x_i}}. \quad (6)$$

Proof Sketch. By using assumption **AS3**, a case analysis shows that for the distribution $P(y_i \mid x_i,d_i)$ assumed in the theorem, the Bayes rule for computing successive posterior probabilities for the components $\{d_i\}_{i=1}^n$ reduces to equation (6). In fact, a more tedious analysis can be used to show that the distribution $P_{\beta_0,\beta_1,\gamma}(y_i \mid x_i,d_i)$ is the *only* distribution (under our four assumptions) for which identity (6) holds. The theorem then follows by combining this Bayesian single component update rule with the product decomposition (2). □

Notice that update rule (6) is independent of the parameter γ 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 (5) 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 $\mathcal{M}$ predicts the binary label $\hat{y}$ that minimizes the expected loss, i.e.

$$\hat{y}_t = \arg \min_{y \in \{0,1\}} \sum_{\boldsymbol{y}_t \in \{0,1\}^n} L^n(\boldsymbol{y}_t, y) P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, S^{t-1}). \tag{7}$$

It is not difficult to see that this simplifies to

$$\hat{y}_t = \arg \max_{y \in \{0,1\}} P(y^n \mid \boldsymbol{x}_t, S^{t-1}), \tag{8}$$

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

Theorem 2. *Let $S = (\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})$ be the sequence of examples observed before trial t and let $\boldsymbol{x}_t$ be the instance received at the beginning of 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 $\mathcal{M}$ decision rule (8) can be expressed in the following form:*

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

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

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

$$P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, S) = \frac{P(\boldsymbol{x}_t)}{P(\boldsymbol{x}_t \mid S)} \prod_{i=1}^{n} \left(\sum_{d_i \in \{0,1\}} P(y_{t,i} \mid x_{t,i}, d_i) P(d_i \mid S_i) \right). \tag{9}$$

Substituting the expression for $P(\boldsymbol{y}_t \mid \boldsymbol{x}_t, S)$ given in (9) in the Bayes Decision rule (8) we obtain

$$\hat{y}_t = \arg \max_{y \in \{0,1\}} \prod_{i=1}^{n} \left(\sum_{d_i \in \{0,1\}} P(y \mid x_{t,i}, d_i) P(d_i \mid S_i) \right). \tag{10}$$

The thesis then follows, by simple manipulations, from (10) and the facts

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

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

Bayes-BEG
Input: $0 \leq \beta_0 < 1$, $\beta_1 > 1$, $0 < \gamma < 1$, and $n \geq 1$
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 $\boldsymbol{x}_t \cdot \boldsymbol{z}_t > \theta$ 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

$$w_{t+1,i} = w_{t,i}\frac{\beta_{y_{t,i}}^{x_{t,i}}}{1 - w_{t,i} + w_{t,i}\beta_{y_{t,i}}^{x_{t,i}}}. \tag{11}$$

Update Policy: Update in all trials.

Fig. 1. The *Bayes*-BEG algorithm.

□

We call the indirect algorithm using the prediction rule of Theorem 2 and the update function (6) described in Theorem 1 the *Bayes-BEG* algorithm. The algorithm is summarized in Figure 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 $P_{\beta_0,\beta_1,\gamma}(y \mid x, d)$ as per model $\mathcal{M}$. 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 $M_{\texttt{alg}}(S)$ to denote the number of mistakes made by algorithm "alg" on sequence S.
We start by proving a mistake bound for the MD-*Bayes*-BEG algorithm when $\beta_0 = 0$.

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

$$M_{\textit{MD-}\texttt{Bayes-}\textit{BEG}}(S) \leq 6.48 + 2.48k\left(1 + \left|\ln_2\left(\frac{2(n-1)}{(1+c)(e-1)}\right)\right|\right), \tag{12}$$

where k is the number of relevant variables in the target disjunction.

Proof. The bound is proven by following the same approach used in Section 5 of Littlestone [Lit88]. As in [Lit88], 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 rest of 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 $t = 1, \ldots, \ell$, $z_{t,i} \geq 0$ and $z_{t,i}$ only increases/decreases during promotion/elimination steps it follows that $Z_1 + p\, Z_{gain} - d\, Z_{lost} \geq 0$, where Z_{gain} is an upper bound on the total z-weight gained during a promotion step and Z_{lost} is a lower bound on the total z-weight lost during an elimination step. By solving it with respect to d we obtain that $d \leq (Z_1/Z_{lost}) + p(Z_{gain}/Z_{lost})$. Hence, the number of mistakes made by MD-*Bayes*-BEG can be upper bounded by

$$M_{\texttt{MD-}\textit{Bayes}\texttt{-BEG}}(S) \leq \frac{Z_1}{Z_{lost}} + p\left(1 + \frac{Z_{gain}}{Z_{lost}}\right). \tag{13}$$

We now estimate the quantities in the right hand side of (13). For the total initial z-weight, $Z_1 = \sum_{i=1}^{n} z_{1,i}$, it is easy to see that when $c = (e+1)/(e-1)$ and $w_{1,i} = 1/n$ $(i = 1, \ldots, n)$, we have $Z_1 = n \ln\left(1 + \frac{\beta_1}{n-1}\right) \leq 2\,\beta_1$, where the inequality follows from the fact that for $r > 0$ we have $\ln(1+r) \leq r$ and $n/(n-1) \leq 2$ for $n \geq 2$. Similarly, $Z_{lost} > \theta$ since during each elimination step we have $\sum_{i=1}^{n} x_{t,i} z_{t,i} > \theta$ and $z_{t+1,i} = 0$ for any attribute $x_{t,i}$ such that $x_{t,i} = 1$. For Z_{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 $z_{t+1,i}/z_{t,i} \leq (1+c)$. Now, $Z_{gain} = \sum_{i=1}^{n} x_{t,i}(z_{t+1,i} - z_{t,i}) \leq c \sum_{i=1}^{n} x_{t,i} z_{t,i} \leq c\theta$, where the last inequality follows from the fact that during a promotion step we have $\sum_{i=1}^{n} x_{t,i} z_{t,i} \leq \theta$.

We now bound the number of promotion steps incurred by the algorithm. We first observe that if the w-weight assigned to each relevant attribute is $> 1/g(c)$ where $g(c) = 1 + ((1+c)(e-1)/2)$, then $z_{t,i} > n \ln(\gamma/(1-\gamma))$ and positive examples are correctly classified by MD-*Bayes*-BEG. By simple manipulation 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 $\lceil |\, \ln_{c+1}((2(n-1))/((1+c)(e-1)))\, | \rceil$ and thus,

$$p \leq k \left\lceil \left| \ln_{c+1}\left(\frac{2(n-1)}{(1+c)(e-1)}\right) \right| \right\rceil \leq k \left(1 + \left| \ln_2\left(\frac{2(n-1)}{(1+c)(e-1)}\right) \right| \right) \tag{14}$$

Bound (12) then immediately follows by plugging the above estimates for Z_1, Z_{lost}, Z_{gain} and p into (13) and by observing that $1 < c < \sqrt{(1+e)/(e-1)}$. □

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 w_i remains 0 thereafter. Therefore, if an example has the label 0 but all the variables 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 $\beta = 0$ 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 a variable 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 $n \geq 2$, $q = 16/10, \epsilon = 9/10$ and set $\gamma/(1-\gamma) = q^{1/(n-1)}, \beta_0 = 1-(q^{3/2}-1+\epsilon)/((1+q)(q^{1/(2(n-1))}))$ and $\beta_1 = 1+(q^{3/2}-1+\epsilon)/(1+q)$. Then for all sequences $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ such that there exists a monotone disjunction consistent with S we have*

$$M_{\texttt{MD-Bayes-BEG}}(S) \leq 24.79 + 8.44k\ln(n-1) + 5.76k, \tag{15}$$

where k is the number of relevant variables in the target disjunction.

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 $z_{t,i}$ are the weights used in the dot product prediction, we analyze the change in the total weight $Y_t = \sum_{i=1}^n y_{t,i}$ where $y_{t,i} = \ln((1+w_{t,i}(\beta_1-1))/(1-w_{t,i}(1-\beta_0)))$. Details of the proof are omitted. □

4.3 The Thresholded-BEG Algorithm

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

This algorithm is much easier to analyze with the existing relative mistake bound techniques [Lit89,AW98], 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 is given, then the following bound on the number of mistakes made by the algorithm on any sequence of examples where the best disjunctions 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 $\theta = (\alpha \ln \alpha)/(\alpha^2 - 1)$. Let $w_1 > 0$. Then for all sequences S such that there exists a monotone disjunction with at most A attribute errors on S, we have*

$$M_{Thresholded-BEG}(S) \leq (\alpha + 1)\frac{\mathrm{dist}_{bre}(\boldsymbol{u}, \boldsymbol{w}_1) + A \ln \alpha}{\ln \alpha}, \tag{16}$$

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

Proof Sketch. The technique used to derive the mistake bound is similar to Auer and Warmuth's [AW98]. The analysis proceeds by showing that the distance between the weight vector $\boldsymbol{w}_t$ used by the algorithm and a target weight vector $\boldsymbol{u}$ decreases whenever the algorithm makes a mistakes. However, our analysis uses the binary relative entropy as a potential function. Details of the proof are omitted. □

It is interesting to observe that bound (16) of Theorem 5 has the same form as the bound derived for Winnow in [AW98], except that in the latter the binary relative entropy of (16) is replaced by the un-normalized relative entropy $\mathrm{dist}_{ure}(\boldsymbol{u}, \boldsymbol{w}_{1,i}) = \sum_{i=1}^{n} u_i \ln(u_i/w_{1,i}) + w_{1,i} - u_i$.
Better results can be obtained if the algorithm has some additional 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 [AW98] (although with slightly worse constants). For example, in the noise-free case, i.e. when the algorithm knows ahead of time that there exists a monotone disjunction consistent with S ($A = 0$), we get a bound that is incomparable to Theorem 3.

Corollary 1. *Let $n \geq 2$ and set $\beta_0 = 0$, $\beta_1 = e$ and $\theta = 1/e$. Furthermore, let $w_{1,i} = 1/n$ for $i = 1, \ldots, n$. Then for all sequences $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ such that there exists a monotone disjunction consistent with S we have*

$$M_{\texttt{Thresholded-BEG}}(S) \leq 3.76 + 2.72k \ln(n), \tag{17}$$

where k is the number of relevant variables in the target disjunction.

5 The Normalized Winnow Algorithm

The Normalized Winnow algorithm [Lit95a] is another mistake-driven linear threshold algorithm for on-line learning disjunctions with a good relative mistake bound. This algorithm is identical to Winnow except that it normalizes its weights before computing the linear threshold prediction. 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 we found useful for Normalized Winnow is the n-fold product of a distribution over $\{1, 2, \ldots, n\}$. Since sampling this prior gives a vector in $\{1, 2, ..., n\}^n$, most of the 2^n disjunctions will be represented by several possible outcomes. For example, the disjunction $x_1 \vee x_3 \vee x_7$ is represented by all vectors containing only 1's, 3's, and 7's, and at least one of each.

Under model $\mathcal{M}$ (with a slight modification to assumption **AS4** for the new prior) the posterior probabilities over the space $\{1, 2, \ldots, n\}^n$ can also be represented as an n-fold product distribution. As before, the vector-label problem is used to decouple the attributes and obtain this result. It turns out that the posterior probabilities/weights of the integers in $\{1, 2, \ldots, n\}$ are updated in the same way as the weights of the Normalized Winnow algorithm. With the loss function defined in Section 4.1, the Bayes-optimal predictions are the same thresholded dot products between the instances and weights used by the Normalized Winnow algorithm. This establishes a close correspondence between Normalized Winnow and Bayes methods.

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 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 appears promising since it corresponds to the un-normalized relative entropy used to analyze Winnow [Lit89,AW98]. 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.

References

AW98. Auer, P., Warmuth, M. K.: Tracking the best disjunction. Journal of Machine Learning, Special issue on concept drift. **32** (2) (1998)

Byl97. Bylander, T.: The binary exponentiated gradient algorithm for learning linear functions. Unpublished manuscript, private communication, (1997)

DP96. Domingo, P., Pazzani, M.: Beyond independence: conditions for the optimality of the simple Bayesian classifier. Proc. 13th International Conference on Machine Learning, (1996) 105–112

DH73. Duda, R. O., Hart, P. E.: Pattern Classification and Scene Analysis. Publisher Wiley (1973)

GW98. Gentile, C., Warmuth, M. K.: Hinge Loss and Average Margin. Advances in Neural Information Processing Systems. Morgan Kaufmann, (1998)

Lit88. Littlestone, N.: Learning when Irrelevant Attributes Abound: A New Linear-threshold Algorithm. Machine Learning. **2** (1988) 285–318 (Earlier version in FOCS87)

Lit89. Littlestone, N.: Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms. PhD thesis, Technical Report UCSC-CRL-89-11, University of California Santa Cruz (1989)

Lit95. Littlestone, N.: Comparing several linear-threshold learning algorithms on tasks involving superfluous attributes. In Proc. 12th International Conference on Machine Learning. Morgan Kaufmann, (1995) 353–361

Lit95a. Littlestone, N.: "Private Communication"

Lit96. Littlestone, N.: Mistake-driven Bayes sports: bounds for symmetric apobayesian learning algorithms. Technical report, NEC Research Institute, Princeton, NJ. (1996)

LM97. Littlestone, N., Mesterharm, C.: An Apobayesian Relative of Winnow. Advances in Neural Information Processing Systems. Morgan Kaufmann, (1997)

LW94. Littlestone, N., Warmuth, M. K.: The weighted majority algorithm. Information and Computation. **108** (1994) 212–261. (Earlier version in FOCS89)