Tracking the best disjunction

Peter Auer* Manfred K. Warmuth†
Department of Computer Science
University of California at Santa Cruz
Santa Cruz, CA 95064 (USA)

Abstract

Littlestone developed a simple deterministic on-line learning algorithm for learning k-literal disjunctions. This algorithm (called Winnow) keeps one weight for each of the n variables and does multiplicative updates to its weights. We develop a randomized version of Winnow and prove bounds for an adaptation of the algorithm for the case when the disjunction may change over time. In this case a possible target disjunction schedule *T is a sequence of disjunctions (one per trial) and the* shift size *is the total number of literals that are added/removed from the disjunctions as one progresses through the sequence.*

We develop an algorithm that predicts nearly as well as the best disjunction schedule for an arbitrary sequence of examples. This algorithm that allows us to track the predictions of the best disjunction is hardly more complex than the original version. However the amortized analysis needed for obtaining worst-case mistake bounds requires new techniques. In some cases our lower bounds show that the upper bounds of our algorithm have the right constant in front of the leading term in the mistake bound and almost the right constant in front of the second leading term. By combining the tracking capability with existing applications of Winnow we are able to enhance these applications to the shifting case as well.

1 Introduction

One of the most significant successes of the Computational Learning Theory community has been Littlestone's formalization of an on-line model of learning and the development of his algorithm Winnow for learning disjunctions [Lit89, Lit88]. The key feature of Winnow is that when learning disjunctions, the number of mistakes of the algorithm grows only logarithmically with the input dimension. For many other standard algorithms such as the Perceptron Algorithm [Ros58], the number of mistakes can grow linearly in the dimension [KW95]. In the meantime a number of algorithms similar to Winnow have been developed that also show the logarithmic growth of the loss bounds in the dimension [LW94, Vov90, CBFH+94, HKW94, KW94].

In this paper we give a refined analysis of Winnow, develop a randomized version of the algorithm, give lower bounds that show that both the deterministic and the randomized version are close to optimal, and adapt both versions so that they can be used to track the predictions of the best disjunction.

Consider the following by now standard on-line learning model [Lit89, Lit88, Vov90, CBFH+94]. Learning proceeds in trials. In trial $t \geq 1$ the algorithm is presented with an instance $\vec{x}_t$ (in our case an n-dimensional binary vector) that is used to produce a binary prediction $\hat{y}_t$. The algorithm then receives a binary classification y_t of the instance and incurs a mistake if $\hat{y}_t \neq y_t$. The goal is to minimize the worst-case number of mistakes of the algorithm for an arbitrary sequence of examples $\langle(\vec{x}_t, y_t)\rangle$. This is of course a hopeless scenario. For any deterministic algorithm an adversary can always choose the sequence so that the algorithm makes a mistake in each trial. A more reasonable goal is to minimize the number of mistakes of the algorithm compared to the minimum number of mistakes made by any concept from a comparison class.

In this paper we use monotone[1] k-literal disjunctions as the comparison class. If the dimension (number of Boolean attributes/literals) is n then 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 lie in $\{1, \ldots, n\}$. The number of classification errors of such a disjunction with respect to a sequence of examples is simply the total number of misclassifications that this disjunction produces on the sequence. The goal is to develop algorithms whose number of mistakes is not much larger than the number of classification errors of the best disjunction on any sequence of examples.

In this paper we consider the case where the mistakes of

*P. Auer is supported by grant J01028-MAT of the Fonds zur Förderung der wissenschaftlichen Forschung, Austria.

†M. K. Warmuth acknowledges the support of the NSF grant IRI-9123692.

[1] By expanding the dimension to $2n$, Learning non-monotone disjunctions reduces to the monotone case.

the best ("target") disjunction are caused by attribute errors. The number of attribute errors of an example $(\vec{x}, y) \in \{0,1\}^n \times \{0,1\}$ with respect to a target disjunction $\vec{u}$ is the minimum number of attributes/bits of $\vec{x}$ that have to be changed so that for the resulting $\tilde{x}$, $\vec{u}(\tilde{x}) = y$. The number of attribute errors for a sequence of examples with respect to a target concept is simply the total number of such errors for all examples of the sequence. Note that if the target $\vec{u}$ is a k-literal monotone disjunction then the number of attribute errors is at most k times the number of classification errors with respect to $\vec{u}$ (i.e. the number of examples $(\vec{x}, y)$ in the sequence for which $\vec{u}(\vec{x}) \neq y$).

Winnow can be tuned as a function of k so that it makes at most $O(A + k \ln(n/k))$ mistakes on any sequence of examples where the best disjunction makes at most A attribute errors [Lit88]. We give a randomized version of Winnow and give improved tunings of the original algorithm. The new algorithm can be tuned based on k and A so that its expected mistake bound is at most $A + (2 + o(1))\sqrt{Ak \ln(n/k)}$ on any sequence of examples for which there is a monotone k-literal disjunction with at most A attribute errors. We also show how the original deterministic algorithm can be tuned so that its number of mistakes is at most $2A + (2\sqrt{2} + o(1))\sqrt{Ak \ln(n/k)}$ for the same set of sequences.

Our lower bounds show that these bounds are very close to optimal. We show that for any algorithm the expected number of mistakes must be at least $A + (1 + o(1))\sqrt{Ak \ln(n/k)}$. So our upper bound has the correct constant on the leading term and almost the optimal constant on the second term. For deterministic algorithms our lower bounds show that the constant on the leading term is optimal.

Our lower bounds for both the deterministic and the randomized case cannot be improved significantly because there are essentially matching upper bounds achieved by non-efficient algorithms with the correct factors on the first *and* the second term. These algorithms use $\binom{n}{\leq k}$ experts [CBFH+94]. Each expert simply computes the value of a particular k-literal disjunction and one weight is kept per expert. This amounts to expanding the n-dimensional Boolean inputs into $\binom{n}{\leq k}$ Boolean inputs and then using single literals (experts) [LW94, Vov90, CBFH+94] as the comparison class instead of k-literal monotone disjunctions. The expected number of mistakes of the probabilistic algorithm is at most $Q + \sqrt{Qk \ln(n/k)} + k \log_2(n/k)/2$ where Q is a bound on the number of classification errors of the best k-literal disjunction. The mistake bound of the deterministic algorithm is exactly twice as high. Observe that these algorithms have to use about n^k weights, and that they need that much time in each trail to calculate their prediction and update the weights. Thus their run time is exponential in k.

In contrast, our algorithm uses only n weights. On the other hand the noise in the upper bounds of our efficient algorithm is measured in attribute errors rather than classification errors. This naturally arises since we are using one weight per attribute. Recall that a classification error with respect to a k-literal disjunction can equate to up to k attribute errors. To capture errors that affect up to k attributes efficiently the expansion to $\binom{n}{\leq k}$ experts seems to be unavoidable. Nevertheless, it is surprising that our simple algorithm is able to get the right factor before the number of attribute errors A and for the probabilistic version almost the right factor before the square root term. In some sense our algorithm compresses $\binom{n}{\leq k}$ weights to only n weights. At this point we don't have a computational interpretation of our weights. Such an interpretation was only found for the single literal (expert) case [CBFHW94].

As Littlestone [Lit91] we use an amortized analysis with an entropic potential function to obtain our worst-case loss bounds. However besides the more careful tuning of the bounds we take the amortized analysis method a significant step further by proving mistake bounds of our algorithm as compared to the best shifting disjunction. Assume that a disjunction $\vec{u}$ is specified by a n-dimensional binary vector, where the components with value 1 correspond to the monotone literals of the disjunction. For two disjunctions $\vec{u}$ and $\vec{u}'$ the Hamming distance $||\vec{u} - \vec{u}'||_1$ measures how many literals have to be "shifted" to obtain $\vec{u}'$ from $\vec{u}$. A disjunction schedule $\mathcal{T}$ for a sequence of examples of length T is simply a sequence of T disjunctions $\vec{u}_t$. The *(shift) size* of the schedule $\mathcal{T}$ is $\sum_{t=1}^{T} ||\vec{u}_{t-1} - \vec{u}_t||_1$ ($\vec{u}_0$ is the all zero vector). In the original non-shifting case all $\vec{u}_t$ ($t \geq 1$) are equal to some k-literal disjunction $\vec{u}$.

At trial t the schedule $\mathcal{T}$ predicts with disjunction $\vec{u}_t$. We define the number of attribute errors of a sequence with respect to a schedule $\mathcal{T}$ as the total number of attributes that have to be changed in the sequence of examples to make it consistent with the schedule $\mathcal{T}$. Note that the loss bound for the non-shifting case can be written as $cA + O(\sqrt{AB})$, where $B = O(k + k \ln(n/k))$ is the number of bits it takes to describe a k-literal monotone disjunction, and where $c = 1$ for the randomized and $c = 2$ for the deterministic algorithm. Surprisingly, we were able to prove bounds of the same form for the shifting disjunction case. B is now the number of bits it takes to describe the best schedule $\mathcal{T}$ and A is the number of attribute errors of this schedule. If Z is the size of the best schedule (essentially the number of shifts of literals) then it takes $O(Z \ln(An/Z))$ bits to describe a target schedule in respect to a given sequence of examples.[2]

[2] Essentially one has to describe when a shift occurs and which literal is shifted. Obviously there is no necessity to shift if the current disjunction

Our worst-case mistake bounds are similar to bounds obtained for "competitive algorithms" in that we compare the number of mistakes of our algorithm against the number of attribute errors of the best off-line algorithm that is given the whole sequence ahead of time. The off-line algorithm still incurs A attribute errors and here we bound the additional loss of the on-line algorithm over the number of attribute errors of the best schedule (as opposed to the less accurate method of bounding the ratio of on-line over off-line).

Winnow does multiplicative updates to its weights. Whenever the algorithm makes a mistake then the weights of all the literals for which the corresponding bit in the current input instance is one are multiplied by a factor. In the case of Winnow2, the version of Winnow this paper is based on, this factor is either α or $1/\alpha$, where $\alpha > 1$ is a parameter of the algorithm. The multiplicative weight updates might cause the weights of the algorithm to decay rather rapidly. Since any literal might become part of the disjunction schedule even when it was misleading during the early part of the sequence of examples, any algorithm that is to predict well as compared to the best disjunction schedule must be able to recover weights quickly. Our extension of Winnow2 simply adds a step to the original algorithm that resets a weight to β/n whenever it drops below this boundary. Similar methods for lower bounding the weights were used in the algorithm WML of [LW94] which was designed for predicting as well as the best shifting single literal/expert. In addition to generalizing the work of [LW94] to arbitrary size disjunctions we were able to optimize the constant in the leading term of the mistake bound of Winnow and develop a probabilistic version of the algorithm.

In [HW95] the work of [LW94] was generalized in a different direction. The focus there is to predict as well as the best shifting expert, where "well" is measured in terms of other loss functions than the discrete loss (counting mistakes) which is the loss function used in this paper. Again the basic building block is a simple on-line algorithm that uses multiplicative weight updates [Vov90, HKW94] but now the predictions and the feedback in each trial are real-valued and lie in the interval $[0, 1]$. The class of loss functions includes the natural loss functions of log loss, square loss and Hellinger loss. In this cases more sophisticated methods are needed for recovering small weights quickly [HW95] than simply lower bounding the weights.

is correct on the current example. Thus only in some of the trials in which the current disjunction would make a mistake the disjunction is shifted. Since the target schedule might make up to A mistakes due to attribute errors and there are up to Z shifts, we get $A + Z$ trials which are candidates for shifts. Choosing Z of them and choosing one literal for each shift gives $\binom{A+Z}{Z} n^Z$ possibilities which need about $Z \log \frac{An}{Z}$ bits to be encoded.

Besides doing experiments on practical data that exemplify the merits of our worst-case mistake bounds, this research also leaves a number of theoretical open problems. Winnow is an algorithm for learning arbitrary linear threshold functions and our methods for tracking the best disjunction still need to be generalized to learning this more general class of concepts.

There is a natural competitor to Winnow which is the well known Perceptron algorithm [Ros58] for learning linear threshold functions. This algorithm does additive instead of multiplicative updates. The classical Perceptron Convergence Theorem gives a mistake bound for this algorithm [DH73, Hay93]. The proof of this theorem can also be seen as an amortized analysis. However the potential function needed for the perceptron algorithm is quite different from the potential function used for the analysis of Winnow. If $\vec{w}_t$ is the weight vector of the algorithm in trial t and $\vec{u}$ is a target weight vector, then for the perceptron algorithm $||\vec{u} - \vec{w}_t||_2^2$ is the potential function where $||.||_2$ is the Euclidean length of a vector. In contrast the potential function used for the analysis of Winnow [Lit88, Lit89] that is also used in this paper is the following generalization[3] of relative entropy [Cov65]: $\sum_{i=1}^{n}[w_i - u_i + u_i \ln(u_i/w_i)]$.

In the case of linear regression a framework was developed [KW94] for deriving updates from the potential function used in the amortized analysis. The same framework can be adapted to derive both the Perceptron algorithm and Winnow. The different potential functions for the algorithms lead to the additive and multiplicative algorithms, respectively. The Perceptron algorithm is seeking a weight vector that is consistent with the examples but otherwise minimizes some Euclidean length. Winnow instead minimizes a relative entropy and is thus rooted in the Minimum Relative Entropy Principle of Kullback [KK92, Jum90].

We believe that the techniques developed here for learning how to predict as well as the best shifting disjunction will be useful in other settings such as developing algorithms that predict nearly as well as the best shifting linear combination. Now the discrete loss has to be replaced by a continuous loss function such as the square loss, which makes this problem more challenging.

Why are disjunctions so important? Whenever a richer class is built by (small) unions of a large number of simple basic concepts, our methods can be applied. Simply expand the original input into as many inputs as there are basic concepts. Since our mistake bounds only depend logarithmically on the number of basic concepts, we can even allow exponentially many basic concepts and still have polynomial mistake bounds. This method was previously used for developing noise robust algorithms for predicting

[3]For this potential function the weights must be positive. Negative weights are handled via a reduction [Lit88, Lit89, KW94].

nearly as well as the best discretized d-dimensional axis-parallel box [MW95, Aue93] or as well as the best pruning of a decision tree [HS95]. In these cases a multiplicative algorithm maintains one weight for each of the exponentially many basic concepts. However for the above examples, the multiplicative algorithms with the exponentially many weights can still be simulated efficiently. Now, for example, the methods of this paper immediately lead to an efficient algorithm for predicting as well as the best shifting d-dimensional box. Thus by combining our methods with existing algorithms, we can design efficient learning algorithms with provably good worst-case loss bounds for more general shifting concepts than disjunctions.

1.1 Notations

A target schedule $\mathcal{T} = \langle \vec{u}_1, \ldots, \vec{u}_T \rangle$ is a sequence of disjunctions represented by n-ary bit vectors $\vec{u}_t = (u_{t,1}, \ldots, u_{t,n}) \in \{0,1\}^n$. The size of the shift from disjunction $\vec{u}_{t-1}$ to disjunction $\vec{u}_t$ is $z_t = ||\vec{u}_{t-1} - \vec{u}_t||_1 = \sum_{i=1}^n |u_{t-1,i} - u_{t,i}|$. The total shift size of schedule $\mathcal{T}$ is $Z = \sum_{t=1}^T z_t$ where we assume that $\vec{u}_0 = (0, \ldots, 0)$. If $\vec{u}_1 = \cdots = \vec{u}_T$ then $Z = k = ||\vec{u}_0 - \vec{u}_1||_1 = \sum_{i=1}^n u_{1,i}$.

A sequence of examples $S = \langle (\vec{x}_1, y_1), \ldots, (\vec{x}_T, y_T) \rangle$ consists of attribute vectors $\vec{x}_t = (x_{t,1}, \ldots, x_{t,n}) \in \{0,1\}^n$ and classifications $y_t \in \{0,1\}$. The prediction of disjunction $\vec{u}_t$ for attribute vector $\vec{x}_t$ is $\vec{u}_t(\vec{x}_t) = 1$ if $\vec{u}_t \cdot \vec{x}_t = \sum_{i=1}^n u_{t,i} x_{t,i} \geq 1$ and $\vec{u}_t(\vec{x}_t) = 0$ if $\vec{u}_t \cdot \vec{x}_t = 0$. The number of attribute errors a_t at trial t with respect to a target schedule $\mathcal{T}$ is the minimal number of attributes that have to be changed, resulting in $\tilde{x}_t$, such that $\vec{u}_t(\tilde{x}_t) = y_t$. That is $a_t = \min_{\tilde{x} \in \{0,1\}^n} \{||\tilde{x}_t - \vec{x}_t||_1 : \vec{u}_t(\tilde{x}_t) = y_t\}$. The total number of attribute errors of sequence S with respect to schedule $\mathcal{T}$ is $A = \sum_{t=1}^T a_t$. We denote by $\mathcal{S}(Z, A, n)$ the class of example sequences S with n attributes which are consistent with some target schedule $\mathcal{T}$ with shift size at most Z and with at most A attribute errors. By $\mathcal{S}^0(k, A, n)$ we denote the class of example sequences S with n attributes which are consistent with some non-shifting target schedule $\mathcal{T} = \langle \vec{u}, \ldots, \vec{u} \rangle$ of size at most k (i.e. $\sum_{i=1}^n u_i \leq k$) and with at most A attribute errors.

The loss of a learning algorithm L on an example sequence S is the number of misclassifications

$$M(L, S) = \sum_{t=1}^{T} |\hat{y}_t - y_t|$$

where $\hat{y}_t$ is the prediction of the learning algorithm L in trial t.

2 The algorithm

We present algorithm SWIN, see Figure 1, an extension of Littlestone's Winnow2 algorithm [Lit91]. Our extension incorporates a randomization of the algorithm, and it guarantees a lower bound on the weights used by the algorithm. The algorithm maintains a vector of n weights for the n attributes. By $\vec{w}_t = (w_{t,1}, \ldots, w_{t,n})$ we denote the weights at the end of trial t, and $\vec{w}_0$ denotes the initial value of the weight vector. In trial t the algorithm predicts using the weight vector $\vec{w}_{t-1}$. The prediction of the algorithm depends on $r_t = \vec{w}_{t-1} \cdot \vec{x}_t = \sum_{i=1}^n w_{t-1,i} x_{t,i}$, and a function $p : \mathbf{R} \to [0,1]$. The algorithm predicts 1 with probability $p(r_t)$, and it predicts 0 with probability $1 - p(r_t)$. (To obtain a deterministic algorithm one has to choose a function $p : \mathbf{R} \to \{0,1\}$.) Then it receives the classification y_t and updates the weights, obtaining $\vec{w}_t$. The updates of the weights are performed in two steps. The first step is the original WINNOW update, and the second step guarantees that no weight is smaller than $\frac{\beta}{n}$ for some parameter β (the same approach was taken in [LW94]). Observe that the weights are changed only if the probability of making a mistake was non-zero. For the deterministic algorithm this means that the weights are changed only if the algorithm made a mistake. Furthermore the i-th weight is modified only if $x_{t,i} = 1$. The weight is increased (multiplied by α) if $y_t = 1$, and it is decreased (divided by α) if $y_t = 0$. The parameters α, β, w_0, and the function $p(\cdot)$, have to be set appropriately. Good choices of the parameters and the corresponding mistake bounds are given in the next section, and the proofs are given in Section 4. Corresponding lower bounds are shown in Section 5.

3 Results

In this section we give bounds on the (expected) number of mistakes for specific choices of α, β, w_0, and $p(\cdot)$. We will always choose some $\beta < \frac{\ln \alpha}{\alpha - 1}$ and function $p(\cdot)$ as

$$p(r) = \begin{cases} 0 & \text{if} \quad r \leq \frac{\alpha \ln \alpha + (\alpha - 1)\beta}{\alpha^2 - 1} \\ 1 & \text{if} \quad r > \frac{\alpha \ln \alpha + (\alpha - 1)\beta}{\alpha^2 - 1} \end{cases} \qquad \text{(DET)}$$

for the deterministic version of the algorithm, and as

$$p(r) = \begin{cases} 0 & \text{if} \quad r \leq \beta \\ \frac{(r - \beta)(\alpha - 1)}{\ln \alpha - (\alpha - 1)\beta} & \text{if} \quad \beta < r < \frac{\ln \alpha}{\alpha - 1} \\ 1 & \text{if} \quad r \geq \frac{\ln \alpha}{\alpha - 1} \end{cases} \qquad \text{(PROB)}$$

for the probabilistic version of the algorithm. Observe that $p(r) = 1$ if $r \geq 1$ for both choices of $p(\cdot)$.

At first we give results on the number of mistakes of SWIN, if no information besides n, the total number of attributes, is given.

Parameters:

The algorithm uses parameters α, β, w_0 and a function $p : \mathbf{R} \to [0,1]$. These parameters must satisfy:

$$\alpha > 1 \text{ and } 0 \leq \frac{\beta}{n} \leq w_0.$$

Initialization:

Set the weights to initial values $w_{0,1} = \cdots = w_{0,n} = w_0$.

Prediction:

In each trial $t \geq 1$ set $r_t = \vec{w}_{t-1} \cdot \vec{x}_t$ and predict

$$\hat{y}_t = \begin{cases} 1 & \text{with probability } p(r_t) \\ 0 & \text{with probability } 1 - p(r_t) \end{cases}$$

Update:

Receive the binary classification y_t.
If $y_t = p(r_t)$ then set $\vec{w}_t = \vec{w}_{t-1}$.
If $y_t \neq p(r_t)$ then for all $i = 1, \ldots, n$ set

1. $w'_{t,i} = w_{t-1,i}\alpha^{x_{t,i}(2y_t-1)}$,
2. $w_{t,i} = \max\left\{w'_{t,i}, \frac{\beta}{n}\right\}$.

Figure 1: Algorithm SWIN

Theorem 3.1 *Let $n \geq 8$, $\alpha = 2.7$, $\beta = \frac{2}{5}$, $w_0 = \frac{\beta}{n}$, and $p(\cdot)$ be as in (DET). Then for all $S \in \mathcal{S}(Z, A, n)$*

$$M(SWIN, S) \leq 11.9Z \ln n + 11.8A + 4.8.$$

Let $n \leq 7$, $\alpha = 2.5$, $\beta = \frac{n}{e^2\alpha}$, $w_0 = \frac{1}{e^2\alpha}$, and $p(\cdot)$ be as in (DET). Then for all $S \in \mathcal{S}(Z, A, n)$

$$M(SWIN, S) \leq 19.3Z + 9.3A + 3.9.$$

Let $\alpha = 2.4$, $\beta = 0$, $w_0 = \frac{2}{5n}$, and $p(\cdot)$ be as in (DET). Then for all $S \in \mathcal{S}^0(k, A, n)$

$$M(SWIN, S) \leq 3.9k \ln n + 3.4A + 1.6.$$

In Section 5 we will show that these bounds are optimal up to constants. If A and Z are known in advance then the parameters of the algorithm can be tuned to obtain even better results. If for example in the non-shifting case the number k of attributes in the target concept is known we get

Theorem 3.2 *Let $\alpha = e$, $\beta = 0$, $k \leq \frac{n}{e}$, $w_0 = \frac{k}{n}$, and $p(\cdot)$ be as in (DET). Then for all $S \in \mathcal{S}^0(k, A, n)$*

$$M(SWIN, S) \leq (e+1)\left(k \ln \frac{n}{k} + A\right),$$

and with $p(\cdot)$ as in (PROB)

$$\mathbf{E}M(SWIN, S) \leq e\left(k \ln \frac{n}{k} + A\right).$$

If $k \geq \frac{n}{e}$ we get with $w_0 = \frac{1}{e}$

$$M(SWIN, S) \leq (e+1)\left(\frac{n}{e} + A\right),$$

and

$$\mathbf{E}M(SWIN, S) \leq n + eA.$$

Of particular interest is the case when A is the dominant term, i.e. $A \gg k \ln \frac{n}{k}$.

Theorem 3.3 *Let $\alpha = 1 + \sqrt{\frac{2k}{A} \ln \frac{n}{k}}$, $\beta = 0$, $k \leq \frac{n}{e}$, $w_0 = \frac{k}{n}$, and $p(\cdot)$ be as in (DET). Then for all $S \in \mathcal{S}^0(k, A, n)$*

$$\begin{aligned} & M(SWIN, S) \\ & \leq \quad 2A + 2\sqrt{2Ak \ln \frac{n}{k}}\left(1 + o(1)\right), \end{aligned}$$

and with $\alpha = 1 + \sqrt{\frac{k}{A} \ln \frac{n}{k}}$ and $p(\cdot)$ as in (PROB)

$$\mathbf{E}M(SWIN, S)$$

$$\leq \quad A + 2\sqrt{Ak \ln \frac{n}{k}}\left(1 + o(1)\right),$$

for $\frac{A}{k \ln \frac{n}{k}} \to \infty$.

In the shifting case we get for dominant $A \gg Z \ln n$

Theorem 3.4 *Let* $\epsilon = \sqrt{\frac{2Z}{A} \ln \frac{An}{Z}}$, $\alpha = 1 + \epsilon$, $\beta = \frac{\epsilon}{\ln \epsilon - 1}$, $w_0 = \frac{\beta}{n}$, *and* $p(\cdot)$ *be as in (DET). Then for all* $S \in \mathcal{S}(Z, A, n)$

$$M(SWIN, S) \leq 2A + 2\sqrt{2AZ \ln \frac{An}{Z}}\left(1 + o(1)\right),$$

and for $\epsilon = \sqrt{\frac{Z}{A} \ln \frac{An}{Z}}$ *and* $p(\cdot)$ *as in (PROB)*

$$\mathbf{E}M(SWIN, S) \leq A + 2\sqrt{AZ \ln \frac{An}{Z}}\left(1 + o(1)\right),$$

for $\frac{A}{Z \ln n} \to \infty$.

In Section 5 we will show that in Theorems 3.3 and 3.4 the constants on A are optimal. Furthermore we can show for the probabilistic algorithm that also the magnitude of the second order term in Theorem 3.3 is optimal.

4 Amortized analysis

The analysis of the algorithm proceeds by showing that the distance between the weight vector of the algorithm $\vec{w}_t$, and vector $\vec{u}_t$ representing the disjunction at trial t, decreases, if the algorithm makes a mistake. The potential/distance function used for the previous analysis of Winnow [Lit88, Lit89, Lit91] is the following generalization of relative entropy to arbitrary non-negative weight vectors:

$$D(\vec{w}, \vec{u}) = \sum_{i=1}^{n}\left[w_i - u_i + u_i \ln \frac{u_i}{w_i}\right] .$$

By taking derivatives it is easy to see that the distance is minimal and equal to 0 if and only if $\vec{w}_t = \vec{u}_t$. With the convention that $0 \ln 0 = 0$ and the assumption that $\vec{u} \in \{0,1\}^n$ the distance function simplifies to

$$D(\vec{w}, \vec{u}) = \sum_{i=1}^{n}\left[w_i - u_i - u_i \ln w_i\right] .$$

The following analysis is mainly for the probabilistic algorithm with shifting target disjunctions. The other cases will be derived easily from this analysis. We start by calculating how much the distance $D(\vec{w}_t, \vec{u}_t)$ changes between trials:

$$\begin{aligned} & D(\vec{w}_{t-1}, \vec{u}_{t-1}) - D(\vec{w}_t, \vec{u}_t) \\ = \quad & D(\vec{w}_{t-1}, \vec{u}_{t-1}) - D(\vec{w}_{t-1}, \vec{u}_t) && (1) \\ & + D(\vec{w}_{t-1}, \vec{u}_t) - D(\vec{w}'_t, \vec{u}_t) && (2) \\ & + D(\vec{w}'_t, \vec{u}_t) - D(\vec{w}_t, \vec{u}_t). && (3) \end{aligned}$$

Observe that term (1) might be non-zero in any trial, but that terms (2) and (3) are non-zero only if the weights are updated in trial t. For any

$$\gamma \geq \max\{|1 + \ln w_{t,i}| : 0 \leq t \leq T-1, 1 \leq i \leq n\}$$

we can lower bound term (1) by

$$\begin{aligned} & D(\vec{w}_{t-1}, \vec{u}_{t-1}) - D(\vec{w}_{t-1}, \vec{u}_t) \\ = \quad & \sum_{i=1}^{n}(u_{t,i} - u_{t-1,i})(1 + \ln w_{t-1,i}) \\ \geq \quad & -z_t\gamma. \end{aligned}$$

If the weights are updated in trial t, term (2) is bounded by

$$\begin{aligned} & D(\vec{w}_{t-1}, \vec{u}_t) - D(\vec{w}'_t, \vec{u}_t) \\ = \quad & \sum_{i=1}^{n}\left[w_{t-1,i} - w'_{t,i} + u_{t,i} \ln \frac{w'_{t,i}}{w_{t-1,i}}\right] \\ = \quad & \sum_{i=1}^{n} w_{t-1,i}(1 - \alpha^{x_{t,i}(2y_t - 1)}) \\ & + \sum_{i=1}^{n} u_{t,i}x_{t,i}(2y_t - 1)\ln\alpha \\ = \quad & \sum_{i=1}^{n} w_{t-1,i}x_{t,i}(1 - \alpha^{2y_t - 1}) \\ & + \sum_{i=1}^{n} u_{t,i}(x_{t,i} - \tilde{x}_{t,i})(2y_t - 1)\ln\alpha \\ & + \sum_{i=1}^{n} u_{t,i}\tilde{x}_{t,i}(2y_t - 1)\ln\alpha \\ \geq \quad & r_t(1 - \alpha^{2y_t - 1}) \\ & - a_t \ln\alpha \\ & + y_t(2y_t - 1)\ln\alpha. \end{aligned}$$

Remember that $\tilde{x}_t$ is obtained from $\vec{x}_t$ by removing the attribute errors from $\vec{x}_t$. The last inequality follows from the fact that $\sum_{i=1}^{n} u_{t,i}\tilde{x}_{t,i} \geq 1$ if $y_t = 1$ and $\sum_{i=1}^{n} u_{t,i}\tilde{x}_{t,i} = 0$ if $y_t = 0$.

At last observe that $w_{t,i} \neq w'_{t,i}$ only if $y_t = 0$ and $w'_{t,i} < \frac{\beta}{n}$. In this case $w'_{t,i} = w_{t-1,i}\alpha^{-1} \geq \frac{\beta}{\alpha n}$ and we get

for term (3)

$$\begin{aligned} & D(\vec{w}'_t, \vec{u}_t) - D(\vec{w}_t, \vec{u}_t) \\ &= \sum_{i=1}^{n} \left[w'_{t,i} - w_{t,i} + u_{t,i} \ln \frac{w_{t,i}}{w'_{t,i}} \right] \\ &\geq -\beta(1-\alpha^{-1}). \end{aligned}$$

Summing over all trials we have to consider the trials where the weights are updated and we have to distinguish between trials with $y_t = 0$ and trials with $y_t = 1$. Let

$$\begin{aligned} \mathcal{M}_0 &= \{1 \leq t \leq T : y_t = 0, p(r_t) > 0\}, \\ \mathcal{M}_1 &= \{1 \leq t \leq T : y_t = 1, p(r_t) < 1\}, \end{aligned}$$

denote these trials. Then by the above considerations we have

$$\begin{aligned} & \sum_{t=1}^{T} D(\vec{w}_{t-1}, \vec{u}_{t-1}) - D(\vec{w}_t, \vec{u}_t) \\ &\geq \sum_{t=1}^{T} [-z_t\gamma - a_t \ln\alpha] \\ &\quad + \sum_{t\in\mathcal{M}_0} \left[r_t(1-\alpha^{-1}) - \beta(1-\alpha^{-1}) \right] \\ &\quad + \sum_{t\in\mathcal{M}_1} \left[r_t(1-\alpha) + \ln\alpha \right]. \end{aligned}$$

Now we want to lower bound the sum over $\mathcal{M}_0$ and $\mathcal{M}_1$ by the expected (or total) number of mistakes of the algorithm. We can do this by choosing an appropriate function $p(\cdot)$. We denote by p_t the probability that the algorithm makes a mistake in trial t. Then the expected number of mistakes is $\sum_{t=1}^{T} p_t$. Observe that $p_t = 0$ for any $t \notin \mathcal{M}_0 \cup \mathcal{M}_1$, since in this case $y_t = p(r_t)$. Furthermore $p_t = p(r_t)$ if $t \in \mathcal{M}_0$ and $p_t = 1 - p(r_t)$ if $t \in \mathcal{M}_1$. Thus it is sufficient to find a function $p(\cdot)$ and a constant C with

$$\forall r : p(r) > 0 : r(1-\alpha^{-1}) - \beta(1-\alpha^{-1}) \geq Cp(r) \quad (4)$$

and

$$\forall r : p(r) < 1 : r(1-\alpha) + \ln\alpha \geq C(1-p(r)). \quad (5)$$

For such a function $p(\cdot)$ satisfying (4) and (5) we get

$$\begin{aligned} & \sum_{t=1}^{T} D(\vec{w}_{t-1}, \vec{u}_{t-1}) - D(\vec{w}_t, \vec{u}_t) \\ &\geq -Z\gamma - A\ln\alpha + C\mathbf{E}M(\text{SWIN}, S), \end{aligned}$$

assuming that $S \in \mathcal{S}(Z, A, n)$. Since

$$\begin{aligned} & \sum_{t=1}^{T} \left(D(\vec{w}_{t-1}, \vec{u}_{t-1}) - D(\vec{w}_t, \vec{u}_t) \right) \\ &= D(\vec{w}_0, \vec{u}_0) - D(\vec{w}_T, \vec{u}_T) \\ &\leq nw_0 \end{aligned}$$

we can upper bound the expected number of mistakes by

$$\mathbf{E}M(\text{SWIN}, S) \leq \frac{Z\gamma + A\ln\alpha + nw_0}{C}.$$

Hence we want to choose $p(\cdot)$ and C such that C is as big as possible. Some calculations show that the best choice is

$$C = \frac{\ln\alpha - (\alpha-1)\beta}{\alpha}$$

which satisfies (4) and (5) for $p(\cdot)$ as in (PROB). Of course we have to choose β small enough such that $(\alpha-1)\beta < \ln\alpha$. Putting everything together we have the following lemma.

Lemma 4.1 *Let $\beta < \frac{\ln\alpha}{\alpha-1}$ and assume that*

$$\gamma \geq \max\{|1 + \ln w_{t,i}| : 0 \leq t \leq T-1, 1 \leq i \leq n\}$$

where $w_{t,i}$ are the weights used by algorithm SWIN. Then for all $S \in \mathcal{S}(Z, A, n)$

$$\mathbf{E}M(SWIN, S) \leq \alpha \frac{Z\gamma + A\ln\alpha + nw_0}{\ln\alpha - (\alpha-1)\beta}$$

if SWIN uses the function $p(\cdot)$ given by (PROB).

For the deterministic variant we have to use a function $p : \mathbf{R} \to \{0, 1\}$ and therefore cannot use (PROB). The optimal choice satisfying (4) and (5) is

$$C = \frac{\ln\alpha - (\alpha-1)\beta}{\alpha+1}$$

with $p(\cdot)$ as in (DET), and we get

Lemma 4.2 *Let $\beta < \frac{\ln\alpha}{\alpha-1}$ and assume that*

$$\gamma \geq \max\{|1 + \ln w_{t,i}| : 0 \leq t \leq T-1, 1 \leq i \leq n\}$$

where $w_{t,i}$ are the weights used by algorithm SWIN. Then for all $S \in \mathcal{S}(Z, A, n)$

$$M(SWIN, Z, A) \leq (\alpha+1) \frac{Z\gamma + A\ln\alpha + nw_0}{\ln\alpha - (\alpha-1)\beta}$$

if SWIN uses the function $p(\cdot)$ given by (DET).

Now we are going to calculate a bound γ on $|1 + \ln w_{t,i}|$. We get this bound by lower and upper bounding $w_{t,i}$. Obviously $w_{t,i} \geq \frac{\beta}{n}$ for all t and i. The upper bound on $w_{t,i}$ is derived from the observation that $w_{t,i} > w_{t-1,i}$ only if $y_t = 1$, $p(r_t) < 1$, and $x_{t,i} = 1$. Since $p(r) = 1$ for $r \geq 1$ with the $p(\cdot)$ as in (PROB) or (DET), and $r_t \geq w_{t-1,i}x_{t,i}$ we find that $w_{t,i} \leq \alpha$. Thus $\ln\frac{n}{e\beta} \geq |1 + \ln w_{t,i}|$ whenever $\beta \leq \frac{n}{e^2\alpha}$.

Lemma 4.3 *If $\frac{\beta}{n} \leq w_0 \leq \alpha$ then for all $t = 0, \ldots, T$ and $i = 1, \ldots, n$ the weights $w_{t,i}$ of algorithm SWIN with function $p(\cdot)$ as in (PROB) or (DET) satisfy*

$$\frac{\beta}{n} \leq w_{t,i} \leq \alpha.$$

Furthermore

$$|1 + \ln w_{t,1}| \leq \ln \frac{n}{e\beta}$$

for $\beta \leq \frac{n}{e^2\alpha}$.

4.1 The non-shifting case

In the non-shifting case where $\vec{u}_1 = \cdots = \vec{u}_T = \vec{u}$ and $\vec{u}_0 = (0, \ldots, 0)$ term (1) is 0 for all $t \geq 2$, and it is

$$\begin{aligned} & D(\vec{w}_0, \vec{u}_0) - D(\vec{w}_0, \vec{u}_1) \\ = \;& k + k \ln w_0 \end{aligned}$$

for $t = 1$ where $k = \sum_{i=1}^{n} u_i$ is the number of attributes in the target disjunction $\vec{u}$. Thus in the non-shifting case the term $Z\gamma$ in the upper bounds of Lemmas 4.1 and 4.2 can be replaced by $k \ln \frac{1}{ew_0}$, provided that $ew_0 \leq 1$, and we get

Lemma 4.4 *Let $\beta < \frac{\ln \alpha}{\alpha - 1}$ and $w_0 \leq \frac{1}{e}$. Then for all $S \in \mathcal{S}^0(k, A, n)$*

$$\begin{aligned} & \mathbf{E}M(SWIN, S) \\ \leq \;& \alpha \frac{k \ln \frac{1}{ew_0} + A \ln \alpha + nw_0}{\ln \alpha - (\alpha - 1)\beta} \end{aligned}$$

if SWIN uses the function $p(\cdot)$ given by (PROB), and

$$\begin{aligned} & M(SWIN, S) \\ \leq \;& (\alpha + 1) \frac{k \ln \frac{1}{ew_0} + A \ln \alpha + nw_0}{\ln \alpha - (\alpha - 1)\beta} \end{aligned}$$

if SWIN uses the function $p(\cdot)$ given by (DET).

4.2 Proofs of the upper bounds

The bounds given in Theorems 3.1–3.4 are derived from the lemmas above. After plugging in the parameters given in the theorems, some tedious calculations yield the upper bounds.

5 Lower bounds

We start by proving a lower bound for the shifting case. We show that for any learning algorithm L there are example sequences S for which the learning algorithm makes "many" mistakes. Although not expressed explicitly in the following theorems we will show that these sequences S can be generated by target schedules $\mathcal{T} = \langle \vec{u}_1, \ldots, \vec{u}_T \rangle$ where each disjunction $\vec{u}_t$ consists of exactly one literal, i.e. $\vec{u}_t = \vec{e}_j$ for some j where $\vec{e}_j$ is the jth unit vector.

Theorem 5.1 *For any deterministic learning algorithm L, any $n \geq 2$, any $Z \geq 1$, and any $A \geq 0$, there is an example sequence $S \in \mathcal{S}(Z, A, n)$ such that*

$$M(L, S) \geq 2A + \left\lfloor \frac{Z+1}{2} \right\rfloor \lfloor \log_2 n \rfloor .$$

Proof. For notational convenience we assume that $n = 2^\nu$, $\nu \geq 1$, and $Z = 2R - 1$, $R \geq 1$. We construct the example sequence S depending on the predictions of the learning algorithm such that the learning algorithm makes a mistake in each trial. We partition the trials into R rounds. The first $R - 1$ rounds have length ν, the last round has length $\nu + 2A$. Attribute errors will occur only within the last $2A + 1$ trials. We choose the target schedule such that during each round the target disjunction does not change and is equal to some $\vec{e}_j$.

At the beginning of each round there are $n = 2^\nu$ disjunctions consistent with the examples of this round. After l trials in this round there are still $2^{\nu - l}$ consistent disjunctions: we construct the attribute vector by setting half of the attributes which correspond to consistent disjunctions to 1, and the other attributes to 0. Furthermore we set $y_t = 1 - \hat{y}_t$ where $\hat{y}_t$ is the prediction of the algorithm for this attribute vector. Obviously half of the disjunctions are consistent with this example, and thus the number of consistent disjunctions is divided by 2 in each trial. Thus in each of the first $R - 1$ rounds there is a disjunction consistent with all ν examples of this round.

After $\nu - 1$ trials in the last round there are two disjunctions consistent with the examples of this round. For the remaining $2A + 1$ trials we fix some attribute vector for which these two disjunctions predict differently, and again we set $y_t = 1 - \hat{y}_t$. Thus one of these disjunctions disagrees at most A times with the classifications in these $2A + 1$ trials. This disagreement can be seen as caused by A attribute errors, so that the disjunction is consistent with all the examples in the last round up to A attribute errors. □

Remark 5.2 *Observe that a lower bound for deterministic algorithms like*

$$\forall L \; \exists S : M(L, S) \geq m$$

implies the following lower bound on probabilistic algorithms:

$$\forall L \; \exists S : \mathbf{E}M(L, S) \geq \frac{m}{2}.$$

This follows from the fact that any probabilistic learning algorithm can be turned into a deterministic learning algorithm which makes at most twice as many mistakes as the probabilistic algorithm makes in the average. This means that Theorem 5.1 implies for any probabilistic algorithm L that there are sequences $S \in \mathcal{S}(Z, A, n)$ with $\mathbf{E}M(L, S) \geq A + \lfloor \frac{Z}{4} \rfloor \lfloor \log_2 n \rfloor$.

Now we turn to the non-shifting case. For $k = 1$ there are already lower bounds known.

Lemma 5.3 ([LW94]) *For any deterministic learning algorithm L, any $n \geq 2$, and any $A \geq 0$, there is an example sequence $S \in \mathcal{S}^0(1, A, n)$ such that*

$$M(L, S) \geq 2A + \log_2 n.$$

A slight modification of results in [CBFH+94] gives

Lemma 5.4 ([CBFH+94]) *There are functions $n(\eta)$ and $A(n, \eta)$ such that for any $\eta > 0$, any probabilistic learning algorithm L, any $n \geq n(\eta)$, and any $A \geq A(n, \eta)$, there is an example sequence $S \in \mathcal{S}^0(1, A, n)$ such that*

$$\mathbf{E}M(L, S) \geq A + (1 - \eta)\sqrt{A \ln n}.$$

We extend these results and obtain the following theorems.

Theorem 5.5 *For any deterministic learning algorithm L, any $k \geq 1$, any $n \geq 2k$, and any $A \geq 0$, there is an example sequence $S \in \mathcal{S}^0(k, A, n)$ such that*

$$M(L, S) \geq 2A + k \log_2 \left\lfloor \frac{n}{k} \right\rfloor.$$

Theorem 5.6 *There are functions $n(\eta)$ and $A(n, \eta)$ such that for any $\eta > 0$, any probabilistic learning algorithm L, any $k \geq 1$, any $n \geq kn(\eta)$, and any $A \geq kA(n, \eta)$, there is an example sequence $S \in \mathcal{S}^0(k, A, n)$ such that*

$$\mathbf{E}M(L, S) \geq A + (1 - \eta)\sqrt{Ak \ln \left\lfloor \frac{n}{k} \right\rfloor}.$$

Proof of Theorems 5.5 and 5.6. The proof is by a reduction to the case $k = 1$. The n attributes are divided into k groups $G_i, i = 1, \ldots, k$, such that each group consists of $n_i \geq \lfloor \frac{n}{k} \rfloor$ attributes. Furthermore we choose numbers $a_i \geq \lfloor \frac{A}{k} \rfloor$, $i = 1, \ldots, k$, with $\sum_{i=1}^{k} a_i = A$. For each group G_i we choose a sequence $S_i \in \mathcal{S}^0(1, a_i, n_i)$, accordingly to Lemmas 5.3 and 5.4, respectively, such that for any learning algorithm L_i

$$M(L_i, S_i) \geq 2a_i + \log_2 n_i \tag{6}$$

and

$$\mathbf{E}M(L_i, S_i) \geq a_i + (1 - \eta)\sqrt{a_i \ln n_i}. \tag{7}$$

These sequences S_i can be extended to sequences S_i' with n attributes by setting all the attributes not in group i to 0. Concatenating the expanded sequences S_i' we get a sequence S. It is easy to see that $S \in \mathcal{S}(k, A, n)$. On the other hand any learning algorithm for sequences with n attributes can be transformed into a learning algorithm for sequences with a smaller number of attributes by setting the missing attributes to 0. Thus on each subsequence S_i' of S learning algorithm L makes at least as many mistakes as given in (6) and (7). Hence

$$M(L, S) \geq 2A + k \log \left\lfloor \frac{n}{k} \right\rfloor$$

and

$$\begin{aligned}
\mathbf{E}M(L, S) &\geq 2A + (1 - \eta) \sum_{i=1}^{k} \sqrt{\left\lfloor \frac{A}{k} \right\rfloor \ln \left\lfloor \frac{n}{k} \right\rfloor} \\
&\geq 2A + (1 - \eta)\sqrt{Ak \ln \left\lfloor \frac{n}{k} \right\rfloor} \\
&\quad - k\sqrt{\frac{\ln \left\lfloor \frac{n}{k} \right\rfloor}{\frac{A}{k} - 1}} \\
&\geq 2A + (1 - 2\eta)\sqrt{Ak \ln \left\lfloor \frac{n}{k} \right\rfloor}
\end{aligned}$$

if the function $A(n, \eta)$ is chosen appropriately. □

Acknowledgments

We would like to thank Mark Herbster and Nick Littlestone for valuable discussions.

References

[Aue93] P. Auer. On-line learning of rectangles in noisy environments. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 253–261. ACM Press, New York, NY, 1993.

[CBFH+94] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth. How to use expert advice. Technical Report UCSC-CRL-94-33, Univ. of Calif. Computer Research Lab, Santa Cruz, CA, 1994. An extended abstract appeared in STOC '93.

[CBFHW94] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, and M. Warmuth. On-line prediction and conversion strategies. In *Computational Learning Theory: Eurocolt '93*, volume New Series Number 53 of *The Institute*

of Mathematics and its Applications Conference Series, pages 205–216, Oxford, 1994. Oxford University Press.

[Cov65] T. Cover. Behavior of sequential predictors of binary sequences. In *Proceedings of the 4th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pages 263–272. Publishing House of the Czechoslovak Academy of Sciences, 1965.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[Hay93] S. Haykin. *Neural Networks: a Comprehensive Foundation*. Macmillan, New York, NY, 1993.

[HKW94] D. Haussler, J. Kivinen, and M. K. Warmuth. Tight worst-case loss bounds for predicting with expert advice. Technical Report UCSC-CRL-94-36, University of California, Santa Cruz, Computer Research Laboratory, November 1994. An extended abstract appeared in Eurocolt 1995. To appear subject to revision in *IEEE Transaction on Information Theory*.

[HS95] D. P. Helmbold and R. E. Schapire. Predicting nearly as well as the best pruning of a decision tree. In *Proc. 6th Annu. Conf. on Comput. Learning Theory*, pages 61–68. ACM Press, New York, NY, July 1995.

[HW95] M. Herbster and M. K. Warmuth. Tracking the best expert. In *Machine Learning: Proceedings of the Twelfth International Conference*, San Francisco, CA., 1995. Morgan Kaufmann Publishers.

[Jum90] G. Jumarie. *Relative information*. Springer-Verlag, 1990.

[KK92] J. N. Kapur and H. K Kesavan. *Entropy Optimization Principles with Applications*. Academic Press, Inc., 1992.

[KW94] J. Kivinen and M. Warmuth. Using experts for predicting continuous outcomes. In *Computational Learning Theory: Eurocolt '93*, volume New Series Number 53 of *The Institute of Mathematics and its Applications Conference Series*, pages 109–120, Oxford, 1994. Oxford University Press.

[KW95] J. Kivinen and M. K. Warmuth. The perceptron algorithm vs. winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 289–300. ACM Press, New York, NY, 1995.

[Lit88] N. Littlestone. Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

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

[Lit91] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 147–156, San Mateo, CA, 1991. Morgan Kaufmann.

[LW94] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[MW95] M. Maass and K. Warmuth, M. Efficient learning with virtual threshold gates. In *Proc. 12th International Conf. on Machine Learning*, pages 378–386, San Francisco, CA, July 1995. Morgan Kaufmann.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

[Vov90] V. Vovk. Aggregating strategies. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.