Machine Learning, 32, 127–150 (1998)

Tracking the Best Disjunction*

PETER AUER pauer@igi.tu-graz.ac.at
Institute for Theoretical Computer Science, University of Technology Graz, Klosterwiesgasse 32/2, A-8010 Graz, Austria

MANFRED K. WARMUTH manfred@cse.ucsc.edu
Department of Computer Science, University of California at Santa Cruz, Applied Sciences Building, Santa Cruz, CA 95064

Editors: Gerhard Widmer and Miroslav Kubat

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* $\mathcal{T}$ 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. Computer experiments support our theoretical findings.

Keywords: on-line learning, prediction, concept drift, WINNOW, computational learning theory, amortized analysis

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 (Littlestone, 1989, 1988). The key feature of WINNOW is that when learning disjunctions of constant size, the number of mistakes of the algorithm grows only logarithmically with the input dimension. For many other standard algorithms such as the Perceptron Algorithm (Rosenblatt, 1958), the number of mistakes can grow linearly in the dimension (Kivinen, Warmuth, & Auer, 1997). 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 (Littlestone & Warmuth, 1994; Vovk, 1990; Cesa-Bianchi et al., 1997; Haussler, Kivinen, & Warmuth, 1994).

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.

* An extended abstract appeared in (Auer & Warmuth, 1995).

Consider the following by now standard on-line learning model (Littlestone, 1989, 1988; Vovk, 1990; Cesa-Bianchi et al., 1997). Learning proceeds in trials. In trial $t \geq 1$ the algorithm is presented with an instance $\mathbf{x}_t$ (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 number of mistakes of the algorithm for an arbitrary sequence of examples $\langle(\mathbf{x}_t, y_t)\rangle$. 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.

1.1. The (non-shifting) basic setup

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 (distinct) 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, for any sequence of examples.

In this paper we consider the case where the mistakes of the best ("target") disjunction are caused by attribute errors. The number of attribute errors of an example $(\mathbf{x}, y) \in \{0,1\}^n \times \{0,1\}$ with respect to a target disjunction $\mathbf{u}$ is the minimum number of attributes/bits of $\mathbf{x}$ that have to be changed so that for the resulting $\mathbf{x}'$, $\mathbf{u}(\mathbf{x}') = y$. 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 $\mathbf{u}$ 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 $\mathbf{u}$ (i.e., k times the number of examples $(\mathbf{x}, y)$ in the sequence for which $\mathbf{u}(\mathbf{x}) \neq y$).

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 incurs at most A attribute errors (Littlestone, 1988). 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 $A + (2 + o(1))\sqrt{Ak\ln(n/k)}$ (for $A \gg k\ln(n/k)$) 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 $\binom{n}{k}$ experts (Cesa-Bianchi et al., 1997): 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 $\binom{n}{k}$ Boolean inputs and then using single literals (=experts) (Littlestone & Warmuth, 1994; Vovk, 1990; Cesa-Bianchi et al., 1997) as the comparison class instead of k-literal disjunctions. The expected number of mistakes of the randomized 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 trial 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 arises since we are using just 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 $\binom{n}{k}$ experts seems to be unavoidable. Nevertheless, it is surprising that our version of WINNOW is able to get the right factor before the number of attribute errors A and for the randomized version almost the right factor before the square root term. In some sense, WINNOW compresses $\binom{n}{k}$ weights to only n weights. At this point we don't have a combinatorial interpretation of our weights. Such an interpretation was only found for the single literal (expert) case (Cesa-Bianchi, Freund, Helmbold, & Warmuth, 1996).

As Littlestone (1991) 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.

1.2. Shifting disjunctions

Assume that a disjunction $\mathbf{u}$ is specified by an n-dimensional binary vector, where the components with value 1 correspond to the monotone literals of the disjunction. For two disjunctions $\mathbf{u}$ and $\mathbf{u}'$ the Hamming distance $||\mathbf{u} - \mathbf{u}'||_1$ measures how many literals have to be "shifted" to obtain $\mathbf{u}'$ from $\mathbf{u}$. A disjunction schedule $\mathcal{T}$ for a sequence of examples of length T is simply a sequence of T disjunctions $\mathbf{u}_t$. The *(shift) size* of the schedule $\mathcal{T}$ is $\sum_{t=1}^{T} ||\mathbf{u}_{t-1} - \mathbf{u}_t||_1$ ($\mathbf{u}_0$ is the all zero vector). In the original non-shifting case all $\mathbf{u}_t$ ($t \geq 1$) are equal to some k-literal disjunction $\mathbf{u}$, and according to the above definition the "shift size" is k.

At trial t the schedule $\mathcal{T}$ predicts with disjunction $\mathbf{u}_t$. We define the number of attribute errors of an example sequence $\langle(\mathbf{x}_t, y_t)\rangle$ 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 $\mathcal{T}$, i.e., for which the changed instances $\mathbf{x}'_t$ satisfy $\mathbf{u}_t(\mathbf{x}'_t) = y_t$.

Note, that the loss bounds for the non-shifting case can be written as $cA + O(\sqrt{AB} + B)$, where $B = \log_2 \binom{n}{k}$ is the number of bits it takes to describe a disjunction with k literals, 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 shift size of schedule $\mathcal{T}$ then it takes $\log_2 \binom{A+Z}{Z} + Z \log_2 N$ bits to describe a schedule $\mathcal{T}$ in respect to a given sequence of examples.[2]

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 coarser 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 (Littlestone, 1988), 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 (Littlestone & Warmuth, 1994) which was designed for predicting as well as the best shifting single literal (which is called expert in (Cesa-Bianchi et al., 1997)). In addition to generalizing the work of (Littlestone & Warmuth, 1994) to arbitrary size disjunctions we were able to optimize the constant in the leading term of the mistake bound of WINNOW and develop a randomized version of the algorithm.

In (Herbster & Warmuth, 1998) the work of (Littlestone & Warmuth, 1994) 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 (Vovk, 1990; Haussler et al., 1994) 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. Now the loss does not occur in "large" discrete units. Instead the loss in a trial may be arbitrarily small and thus more sophisticated methods are needed for recovering small weights quickly (Herbster & Warmuth, 1998) than simply lower bounding the weights.

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 nearly as well as the best discretized d-dimensional axis-parallel box (Maass & Warmuth, 1998; Auer,

1993) or as well as the best pruning of a decision tree (Helmbold & Schapire, 1997). 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.

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.

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.

1.3. Related work

There is a natural competitor to WINNOW which is the well known Perceptron algorithm (Rosenblatt, 1958) 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 (Duda & Hart, 1973; Haykin, 1994), but this bound is *linear* in the number of attributes (Kivinen et al., 1997) whereas the bounds for the WINNOW-like algorithms are *logarithmic* in the number of attributes. The proof of the Perceptron Convergence 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 $\mathbf{w}_t$ is the weight vector of the algorithm in trial t and $\mathbf{u}$ is a target weight vector, then for the perceptron algorithm $||\mathbf{u} - \mathbf{w_t}||_\mathbf{2}^\mathbf{2}$ is the potential function where $||.||_2$ is the Euclidean length of a vector. In contrast the potential function used for the analysis of WINNOW (Littlestone, 1988, 1989) that is also used in this paper is the following generalization[3] of relative entropy (Cover, 1965): $\sum_{i=1}^{n}[w_i - u_i + u_i \ln(u_i/w_i)]$.

In the case of linear regression, a framework was developed (Kivinen & Warmuth, 1997) 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 (Kapur & Kesavan, 1992; Jumarie, 1990).

1.4. Organization of the paper

In the next section we formally define the notation we will use throughout the paper. Most of it has already been discussed in the introduction. Section 3 presents our algorithm and Section 4 gives the theoretical results for this algorithm. In Section 5 we consider some more practical aspects, namely how the parameters of the algorithm can be tuned to achieve good performance. Section 6 reports some experimental results. The analysis of our algorithm and the proofs for Section 4 are given in Section 7. Lower bounds on the number of mistakes made by any algorithm are shown in Section 8 and we conclude in Section 9.

2. Notation

A target schedule $\mathcal{T} = \langle \mathbf{u}_1, \ldots, \mathbf{u}_T \rangle$ is a sequence of disjunctions represented by n-ary bit vectors $\mathbf{u}_t = (u_{t,1}, \ldots, u_{t,n}) \in \{0,1\}^n$. The size of the shift from disjunction $\mathbf{u}_{t-1}$ to disjunction $\mathbf{u}_t$ is $z_t = ||\mathbf{u}_{t-1} - \mathbf{u}_t||_1 = \sum_{i=1}^n |u_{t-1,i} - u_{t,i}|$. The total shift size of schedule $\mathcal{T}$ is $Z = \sum_{t=1}^T z_t$ where we assume that $\mathbf{u}_0 = (0, \ldots, 0)$. If $\mathbf{u}_1 = \cdots = \mathbf{u}_T$ then $Z = k = ||\mathbf{u}_0 - \mathbf{u}_1||_1 = \sum_{i=1}^n u_{1,i}$. To get more precise bounds for the case when there are shifts in the target schedule we will distinguish between shifts where a literal is added to the disjunction and shifts where a literal is removed from the disjunction. Thus, we define $z_t^+ = |\{1 \le i \le n : u_{t-1,i} = 0 \text{ and } u_{t,i} = 1\}|$, $z_t^- = |\{1 \le i \le n : u_{t-1,i} = 1 \text{ and } u_{t,i} = 0\}|$, and then $Z^+ = \sum_{t=1}^T z_t^+$ as the number of times a literal is switched on, and $Z^- = \sum_{t=1}^T z_t^-$ as the number of times a literal is switched off. Clearly $z_t^+ + z_t^- = z_t$ and $Z^+ + Z^- = Z$.

A sequence of examples $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T) \rangle$ consists of attribute vectors $\mathbf{x}_t = (x_{t,1}, \ldots, x_{t,n}) \in \{0,1\}^n$ and classifications $y_t \in \{0,1\}$. The prediction of disjunction $\mathbf{u}_t$ for attribute vector $\mathbf{x}_t$ is $\mathbf{u}_t(\mathbf{x}_t) = 1$ if $\mathbf{u}_t \cdot \mathbf{x}_t = \sum_{i=1}^n u_{t,i} x_{t,i} \ge 1$ and $\mathbf{u}_t(\mathbf{x}_t) = 0$ if $\mathbf{u}_t \cdot \mathbf{x}_t = 0$. 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 $\mathbf{x}'_t$, such that $\mathbf{u}_t(\mathbf{x}'_t) = y_t$. That is $a_t = \min_{\mathbf{x}' \in \{0,1\}^n} \{||\mathbf{x}'_t - \mathbf{x}_t||_1 : \mathbf{u}_t(\mathbf{x}'_t) = y_t\}$. 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 Z and with at most A attribute errors. If we wish to distinguish between positive and negative shifts we denote the corresponding class by $\mathcal{S}(Z^+, Z^-, A, n)$ where Z^+ and Z^- are the numbers of literals added and removed, respectively, in the target schedule. 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 $\mathcal{T} = \langle \mathbf{u}, \ldots, \mathbf{u} \rangle$ of size k (i.e., $\sum_{i=1}^n u_i = k$) and with at most A attribute errors. For the case that only upper bounds on Z, Z^+, Z^-, or k are known we denote the corresponding classes by $\mathcal{S}^{\le}(Z, A, n) = \bigcup_{z \le Z} \mathcal{S}(z, A, n)$, $\mathcal{S}^{\le}(Z^+, Z^-, A, n) = \bigcup_{z^+ \le Z^+, z^- \le Z^-} \mathcal{S}(z^+, z^-, A, n)$, and $\mathcal{S}_0^{\le}(k, A, n) = \bigcup_{\kappa \le k} \mathcal{S}_0(\kappa, A, n)$, respectively.

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

Table 1. Algorithm SWIN.

Parameters:

The algorithm uses parameters $\alpha > 1, \beta \geq 0, w_0 > 0$, and a function $p : \mathbf{R} \to [0, 1]$.

Initialization:

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

Prediction:

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

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

Receive the binary classification y_t.

If $y_t = p(r_t)$ then set $\mathbf{w}_t = \mathbf{w}_{t-1}$, else

Update:

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. $w_{t,i} = \max\left\{w'_{t,i}, \frac{\beta}{n}\right\}$.

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

where $\hat{y}_t \in \{0, 1\}$ is the binary prediction of the learning algorithm L in trial t.

3. The Algorithm

We present algorithm SWIN ("Shifting WINNOW"), see Table 1, an extension of Littlestone's WINNOW2 algorithm (Littlestone, 1991). 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 $\mathbf{w}_t = (w_{t,1}, \ldots, w_{t,n})$ we denote the weights at the end of trial t, and $\mathbf{w}_0$ denotes the initial value of the weight vector. In trial t the algorithm predicts using the weight vector $\mathbf{w}_{t-1}$. The prediction of the algorithm depends on $r_t = \mathbf{w}_{t-1} \cdot \mathbf{x}_t = \sum_{i=1}^{n} w_{t-1,i} x_{t,i}$, 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 $p : \mathbf{R} \to \{0, 1\}$.) After predicting the algorithm receives the classification y_t. If $y_t = p(r_t)$ then $\mathbf{w}_t = \mathbf{w}_{t-1}$, i.e. the weight vector is not modified. Since $y_t \in \{0, 1\}$ and $p(r_t) \in [0, 1]$ this can only occur when the prediction was deterministic, i.e., $p(r_t) \in \{0, 1\}$, and correct. An update occurs in all other cases when the prediction was wrong or $p(r_t) \in (0, 1)$.

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 (a similar approach was taken in (Littlestone & Warmuth, 1994)). 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. A good choice for function $p(\cdot)$ is the following: for a randomized prediction let

$$p(r) = \begin{cases} 0 & \text{if } r \le \frac{\ln\alpha+2\beta}{\alpha+1} \\ \frac{(\alpha-1)(r(\alpha+1)-\ln\alpha-2\beta)}{2(\ln\alpha-(\alpha-1)\beta)} & \text{if } \frac{\ln\alpha+2\beta}{\alpha+1} < r < \frac{\ln\alpha}{\alpha-1} \\ 1 & \text{if } r \ge \frac{\ln\alpha}{\alpha-1}. \end{cases} \tag{RAND}$$

and for a deterministic version of the algorithm let

$$p(r) = \begin{cases} 0 \text{ if } r \le \frac{\alpha\ln\alpha+(\alpha-1)\beta}{\alpha^2-1} \\ 1 \text{ if } r > \frac{\alpha\ln\alpha+(\alpha-1)\beta}{\alpha^2-1}. \end{cases} \tag{DET}$$

For the randomized version one has to choose $\beta < \frac{\ln\alpha}{\alpha-1}$. Observe that (DET) is obtained from (RAND) by choosing the threshold $\Theta = \frac{\alpha\ln\alpha+(\alpha-1)\beta}{\alpha^2-1}$ such that $p(\Theta) = 1/2$ in (RAND). This corresponds to the straightforward conversion from a randomized prediction algorithm into a deterministic prediction algorithm.

Theoretically good choices of the parameters α, β, and w_0 are given in the next section and practical issues for tuning the parameters are discussed in Section 5.

4. Results

In this section we give rigorous bounds on the (expected) number of mistakes of SWIN, first in general and then for specific choices of α, β, and w_0, all with $p(\cdot)$ chosen from (RAND) or (DET). These bounds can be shown to be close to optimal for adversarial example sequences; for details see Section 8.

THEOREM 1 (RANDOMIZED VERSION) *Let $\alpha > 1$, $0 \le \beta \le \frac{\ln\alpha}{\alpha-1}$, $w_0 = \frac{\beta}{n}$, and $p(\cdot)$ as in (RAND). Then for all $S \in \mathcal{S}(Z^+, Z^-, A, n)$*

$$\mathbf{E}M(\text{SWIN}, S) \le \alpha\frac{A\ln\alpha + Z^+\ln\left(\frac{n}{e\beta}\right) + Z^-\ln(e\alpha) + \beta}{\ln\alpha - (\alpha-1)\beta}.$$

If $\beta \le \frac{n}{e}$ then the bound holds for all $S \in \mathcal{S}^{\le}(Z^+, Z^-, A, n)$.

THEOREM 2 (DETERMINISTIC VERSION) *Let $\alpha > 1$, $0 \le \beta \le \frac{\ln\alpha}{\alpha-1}$, $w_0 = \frac{\beta}{n}$, and $p(\cdot)$ as in (DET). Then for all $S \in \mathcal{S}(Z^+, Z^-, A, n)$*

$$M(\text{SWIN}, S) \le (\alpha+1)\frac{A\ln\alpha + Z^+\ln\left(\frac{n}{e\beta}\right) + Z^-\ln(e\alpha) + \beta}{\ln\alpha - (\alpha-1)\beta}.$$

If $\beta \le \frac{n}{e}$ then the bound holds for all $S \in \mathcal{S}^{\le}(Z^+, Z^-, A, n)$.

THEOREM 3 (NON-SHIFTING CASE) *Let $\alpha > 1$, $\beta = 0$ and $w_0 > 0$. Then for all $S \in \mathcal{S}_0(k, A, n)$*

$$\mathbf{E}M(\text{SWIN}, S) \leq \alpha \frac{k \ln \frac{1}{ew_0} + A \ln \alpha + nw_0}{\ln \alpha}$$

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

$$M(\text{SWIN}, S) \leq (\alpha + 1) \frac{k \ln \frac{1}{ew_0} + A \ln \alpha + nw_0}{\ln \alpha}$$

if SWIN *uses the function $p(\cdot)$ given by (DET).*
If $w_0 \leq \frac{1}{e}$ then the bounds hold for all $S \in \mathcal{S}_0^{\leq}(k, A, n)$.

Remark. The usual conversion of a bound M for the randomized algorithm into a bound for the deterministic algorithm would give $2M$ as the deterministic bound.[4] But observe that our deterministic bound is just $1 + 1/\alpha$ times the randomized bound.

Since at any time a disjunction cannot contain more than n literals we have $Z^+ - Z^- \leq \min\{n, Z\}$ which gives the following corollary.

COROLLARY 1 *Let $\alpha > 1$, $0 \leq \beta \leq \min\left\{\frac{\ln \alpha}{\alpha - 1}, \frac{n}{e^2 \alpha}\right\}$, and $w_0 = \frac{\beta}{n}$. If $p(\cdot)$ as in (RAND) then for all $S \in \mathcal{S}^{\leq}(Z, A, n)$*

$$\mathbf{E}M(\text{SWIN}, S) \leq \alpha \frac{A \ln \alpha + Z \ln\left(\frac{\alpha n}{\beta}\right) \Big/ 2 + \min\{n, Z\} \cdot \ln\left(\frac{n}{e^2 \alpha \beta}\right) \Big/ 2 + \beta}{\ln \alpha - (\alpha - 1)\beta}.$$

If $p(\cdot)$ as in (DET) then for all $S \in \mathcal{S}^{\leq}(Z, A, n)$

$$M(\text{SWIN}, S) \leq (\alpha + 1) \frac{A \ln \alpha + Z \ln\left(\frac{\alpha n}{\beta}\right) \Big/ 2 + \min\{n, Z\} \cdot \ln\left(\frac{n}{e^2 \alpha \beta}\right) \Big/ 2 + \beta}{\ln \alpha - (\alpha - 1)\beta}.$$

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

THEOREM 4 *Let $\alpha = 1.44$, $\beta = 0.125$, $w_0 = \frac{\beta}{n}$, and $p(\cdot)$ be as in (RAND). Then for all $S \in \mathcal{S}^{\leq}(Z, A, n)$*

$$\begin{aligned} \mathbf{E}M(\text{SWIN}, S) \;\leq\; & 1.7 \cdot A + 2.33 \cdot Z \cdot (\ln n + 2.44) \\ & + 2.33 \cdot \min\{n, Z\} \cdot \ln n + 0.581. \end{aligned}$$

Let $\alpha = 1.32$, $\beta = 0.0269$, $w_0 = \frac{\beta}{n}$, and $p(\cdot)$ be as in (DET). Then for all $S \in \mathcal{S}^{\leq}(Z, A, n)$

$$\begin{aligned} M(\text{SWIN}, S) \;\leq\; & 2.4 \cdot A + 4.32 \cdot Z \cdot (\ln n + 3.89) \\ & + 4.32 \cdot \min\{n, Z\} \cdot (\ln n + 1.34) + 0.232. \end{aligned}$$

Let $\alpha = 1.44$, $\beta = 0$, $w_0 = \frac{1}{n}$, and $p(\cdot)$ be as in (RAND). Then for all $S \in \mathcal{S}_0(k, A, n)$

$$\mathbf{E}M(\text{SWIN}, S) \leq 1.44 \cdot A + 3.95 \cdot k \cdot (\ln n - 1) + 3.95.$$

If $n \geq 3$ then the above bound holds for all $S \in \mathcal{S}_0^{\leq}(k, A, n)$. For $n \leq 2$ we have $\mathbf{E}M(\text{SWIN}, S) \leq 1.44 \cdot A + 3.95$ for all $S \in \mathcal{S}_0^{\leq}(n, A, n)$.

Let $\alpha = 1.75$, $\beta = 0$, $w_0 = \frac{1}{n}$, and $p(\cdot)$ be as in (DET). Then for all $S \in \mathcal{S}_0(k, A, n)$

$$M(\text{SWIN}, S) \leq 2.75 \cdot A + 4.92 \cdot k \cdot (\ln n - 1) + 4.92.$$

If $n \geq 3$ then the above bound holds for all $S \in \mathcal{S}_0^{\leq}(k, A, n)$. For $n \leq 2$ we have $M(\text{SWIN}, S) \leq 2.75 \cdot A + 4.92$ for all $S \in \mathcal{S}_0^{\leq}(n, A, n)$.

In Section 8 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 5 *Let $\alpha = 1.44$, $\beta = 0$, $w_0 = \frac{k}{n}$, and $p(\cdot)$ be as in (RAND). Then for all $S \in \mathcal{S}_0(k, A, n)$*

$$\mathbf{E}M(\text{SWIN}, S) \leq 1.44 \cdot A + 3.95 \cdot k \cdot \ln \frac{n}{k}.$$

If $k \leq \frac{n}{e}$ then the above bound holds for all $S \in \mathcal{S}_0^{\leq}(k, A, n)$. For $k \geq \frac{n}{e}$ we set $w_0 = \frac{1}{e}$ and get $\mathbf{E}M(\text{SWIN}, S) \leq 1.44 \cdot A + 3.95 \cdot \frac{n}{e}$ for all $S \in \mathcal{S}_0^{\leq}(n, A, n)$.

Let $\alpha = 1.75$, $\beta = 0$, $w_0 = \frac{k}{n}$, and $p(\cdot)$ be as in (DET). Then for all $S \in \mathcal{S}_0(k, A, n)$

$$M(\text{SWIN}, S) \leq 2.75 \cdot A + 4.92 \cdot k \cdot \ln \frac{n}{k}.$$

If $k \leq \frac{n}{e}$ then the above bound holds for all $S \in \mathcal{S}_0^{\leq}(k, A, n)$. For $k \geq \frac{n}{e}$ we set $w_0 = \frac{1}{e}$ and get $M(\text{SWIN}, S) \leq 2.75 \cdot A + 4.92 \cdot \frac{n}{e}$ for all $S \in \mathcal{S}_0^{\leq}(k, A, n)$.

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

THEOREM 6 *Let $A \geq k \ln \frac{n}{k}$, $\alpha = 1 + \sqrt{\frac{k}{A} \ln \frac{n}{k}}$, $\beta = 0$, $w_0 = \frac{k}{n}$, and $p(\cdot)$ be as in (RAND). Then for all $S \in \mathcal{S}_0(k, A, n)$*

$$\mathbf{E}M(\text{SWIN}, S) \leq A + 2\sqrt{Ak \ln \frac{n}{k}} + 3k \ln \frac{n}{k}.$$

If $k \leq \frac{n}{e}$ then the above bound holds for all $S \in \mathcal{S}_0^{\leq}(k, A, n)$. For $k \geq \frac{n}{e}$, $A \geq \frac{n}{e}$, $\alpha = 1 + \sqrt{\frac{n}{Ae}}$, $w_0 = \frac{1}{e}$, we have $\mathbf{E}M(\text{SWIN}, S) \leq A + 2\sqrt{An/e} + 3\frac{n}{e}$ for all $S \in \mathcal{S}_0^{\leq}(n, A, n)$.

If $A \geq 2k \ln \frac{n}{k}$, $\alpha = 1 + \sqrt{\frac{2k}{A} \ln \frac{n}{k}}$, $\beta = 0$, $w_0 = \frac{k}{n}$, and $p(\cdot)$ be as in (DET), then for all $S \in \mathcal{S}_0(k, A, n)$

$$M(\text{SWIN}, S) \leq 2A + 2\sqrt{2Ak \ln \frac{n}{k}} + 4k \ln \frac{n}{k}.$$

If $k \leq \frac{n}{e}$ *then the above bound holds for all* $S \in \mathcal{S}_0^{\leq}(k, A, n)$. *For* $k \geq \frac{n}{e}$, $A \geq 2\frac{n}{e}$, $\alpha = 1 + \sqrt{\frac{2n}{Ae}}$, $w_0 = \frac{1}{e}$, *we have* $M(\text{SWIN}, S) \leq 2A + 2\sqrt{2An/e} + 4\frac{n}{e}$ *for all* $S \in \mathcal{S}_0^{\leq}(n, A, n)$.

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

THEOREM 7 *Let* $\epsilon = \sqrt{\frac{Z+\min\{n,Z\}}{2A} \ln \frac{An}{Z}}$, *and* A *and* Z*such that* $\epsilon \leq \frac{1}{10}$. *Then for* $\alpha = 1 + \epsilon$, $\beta = \frac{\epsilon}{\ln \epsilon^{-1}}$, $w_0 = \frac{\beta}{n}$, *and* $p(\cdot)$ *as in (RAND), and for all* $S \in \mathcal{S}^{\leq}(Z, A, n)$,

$$\mathbf{E}M(\text{SWIN}, S) \leq A + \sqrt{2A(Z + \min\{n, Z\}) \ln \frac{An}{Z}} \left(1 + \frac{2}{\ln \epsilon^{-1}} + \frac{3\epsilon}{2}\right)$$

If $\epsilon = \sqrt{\frac{Z+\min\{n,Z\}}{A} \ln \frac{An}{Z}} \leq \frac{1}{10}$, $\alpha = 1 + \epsilon$, $\beta = \frac{\epsilon}{\ln \epsilon^{-1}}$, $w_0 = \frac{\beta}{n}$, *and* $p(\cdot)$ *as in (DET), then for all* $S \in \mathcal{S}^{\leq}(Z, A, n)$,

$$M(\text{SWIN}, S) \leq 2A + 2\sqrt{A(Z + \min\{n, Z\}) \ln \frac{An}{Z}} \left(1 + \frac{3}{\ln \epsilon^{-1}} + \epsilon\right).$$

In Section 8 we will show that in Theorems 6 and 7 the constants on A are optimal. Furthermore, we can show for the randomized algorithm that also the magnitude of the second order term in Theorem 6 is optimal.

5. Practical Tuning of the Algorithm

In this section we give some thoughts on how the parameters α, β, and w_0 of SWIN should be chosen for particular target schedules and sequences of examples. Our recommendations are based on our mistake bounds which hold for *any* target schedule and for *any* sequence of examples with appropriate bounds on the number of shifts and attribute errors. Thus, it has to be mentioned that, since many target schedules and many example sequences are not worst case, our bounds usually overestimate the number of mistakes made by SWIN. Therefore, parameter settings different from our recommendations might result in a smaller number of errors for a specific target schedule and example sequence. On the other hand, SWIN is quite insensitive to small changes in the parameters (see Section 6) and the effect of such changes should be benign.

If little is known about the target schedule or the example sequence, then the parameter settings of Theorems 4 or 5 are advisable since they balance well between the effect of target shifts and attribute errors. If good estimates for the number of target shifts and the number of attribute errors are known then good parameters can be calculated by numerically minimizing the bounds in Theorems 1, 2, 3 or Corollary 1, respectively.

If the average rate of target shifts and attribute errors is known such that $Z \approx r_Z T$ and $A \approx r_A T$ with $r_Z > 0, r_A \geq 0$, then for large T the error rate $r_T = M(\text{SWIN}, S)/T$ is by Corollary 1 approximately upper bounded by

$$r_T \preceq \alpha \frac{r_A \ln \alpha + r_Z \ln \left(\frac{\alpha n}{\beta}\right) \Big/ 2}{\ln \alpha - (\alpha - 1)\beta}$$

for randomized predictions and by

$$r_T \preceq (\alpha + 1) \frac{r_A \ln \alpha + r_Z \ln \left(\frac{\alpha n}{\beta}\right) \Big/ 2}{\ln \alpha - (\alpha - 1)\beta}$$

for deterministic predictions. Again, optimal choices for α and β can be obtained by numerical minimization.

6. Experimental Results

The experiment reported in this section is not meant to give a rigorous empirical evaluation of algorithm SWIN. Instead, it is intended as an illustration of the typical behavior of SWIN, compared with the theoretical bound and also with a version of WINNOW which was not modified to adapt to shifts in the target schedule.

In our experiment we used $n = 100$ boolean attributes and a target schedule $\mathcal{T}$ of length

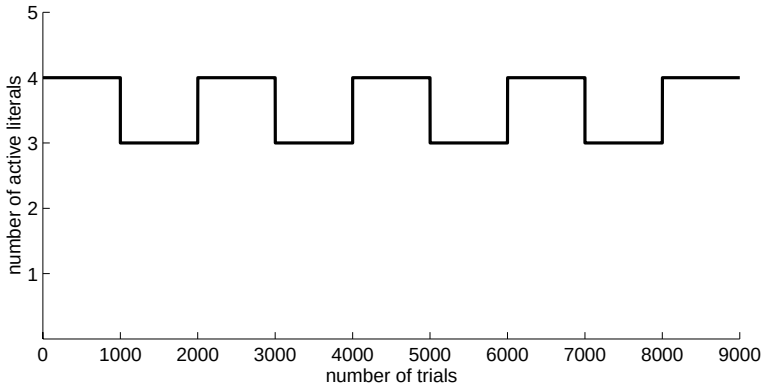

Figure 1. Shifts in the target schedule used in the experiment.

$T = 9,000$ which starts with 4 active literals. After 1,000 trials one of the literals is switched off and after another 1,000 trials another literal is switched on. This switching on and switching off of literals continues as depicted in Figure 1. Thus $Z^- = 4$ and $Z^+ = 8$ (since there are initially 4 active literals).

The example sequence $\langle(\mathbf{x}_t, y_t)\rangle$ was chosen such that for half of the examples $y_t = 1$ and for the other half $y_t = 0$. The values of attributes not appearing in the target schedule were chosen at random such that $x_{t,i} = 1$ with probability 1/2. For examples with $y_t = 1$ exactly one of the active attributes (chosen at random) was set to 1. For examples with attribute errors all relevant attributes were either set to 1 (for the case $y_t = 0$ where

$a_t = \sum_{i=1}^{n} u_{t,i}$) or set to 0 (for the case $y_t = 1$ where $a_t = 1$). Attribute errors occurred at trials $t = 1500, 2250, 3500, 4250, 5500, 6250, 7500, 8250$ with $y_t = 1$ and $a_t = 1$ and at trials $t = 2500, 4500, 6500, 8500$ with $y_t = 0$ and $a_t = 4$.

Figure 2 shows the performance of SWIN compared with the theoretical bound where the parameters were set by numerically minimizing the bound of Corollary 1 as described in the previous section, which yielded $\alpha = 2$, $\beta = 0.067$, $w_0 = \beta/n$. The theoretical bound

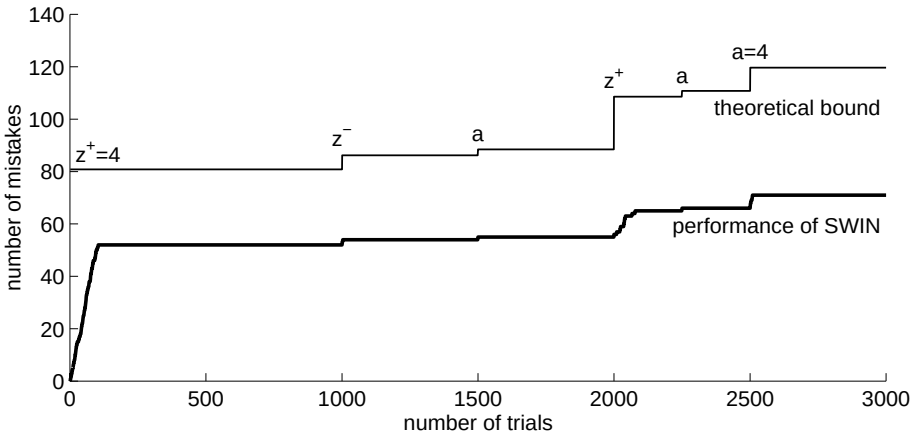

Figure 2. Comparison of SWIN with the theoretical bound for a particular target schedule and sequence of examples. Shifts and attribute errors are indicated by z^+, z^-, and a.

at trial t is calculated from the actual number of shifts and attribute errors up to this trial. Thus, an increase of the bound is due to a shift in the target schedule or an attribute error at this trial. In Figure 2 the reasons for these increases are indicated by z^+ for a literal switched on, z^- for a literal switched off, and a for attribute errors.

Figure 2 shows that the theoretical bound very accurately depicts the behavior of SWIN, although it overestimates the actual number of mistakes by some amount. It can be seen that switching off a literal causes far less mistakes than switching on a literal, as predicted by the bound. Also, the relation between attribute errors and mistakes can be seen.

The performance of SWIN for the whole sequence of examples is shown in Figure 3 and it is compared with the performance of a version of WINNOW which was not modified for target shifts. As can be seen SWIN adapts very quickly to shifts in the target schedule. On the other hand, the unmodified version of WINNOW makes more and more mistakes for each shift.

The unmodified version of WINNOW we used is just SWIN with $\beta = 0$. Thus, the weights are not lower bounded and can become arbitrarily small which causes a large number of mistakes if the corresponding literal becomes active. We used the same α for the unmodified version but we set $w_0 = 4/n$ which is optimal for the initial part of the target schedule. Therefore, the unmodified WINNOW adapts very quickly to this initial part, but then it

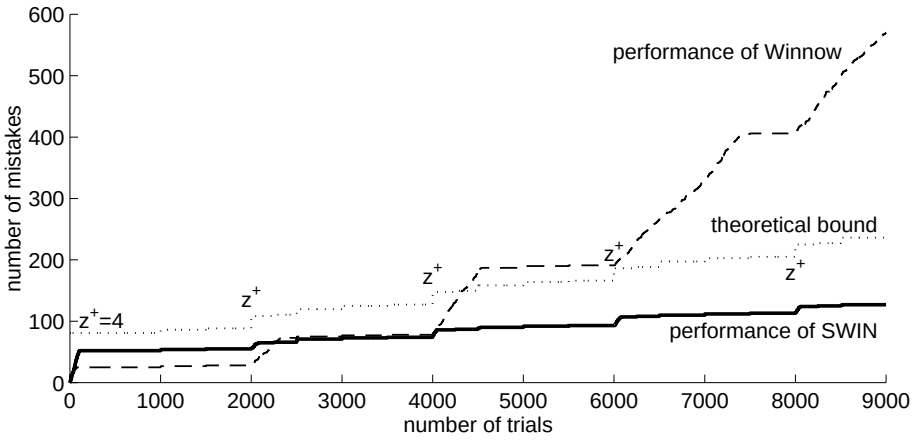

Figure 3. A version of WINNOW which does not lower bound the weights makes many more mistakes than SWIN.

makes an increasing number of mistakes for each shift in the target schedule. For each shift the number of mistakes made approximately doubles.

In the last plot, Figure 4, we compare the performance of SWIN with tuned parameters to the performance of SWIN with the generic parameter setting given by Theorem 4. Although

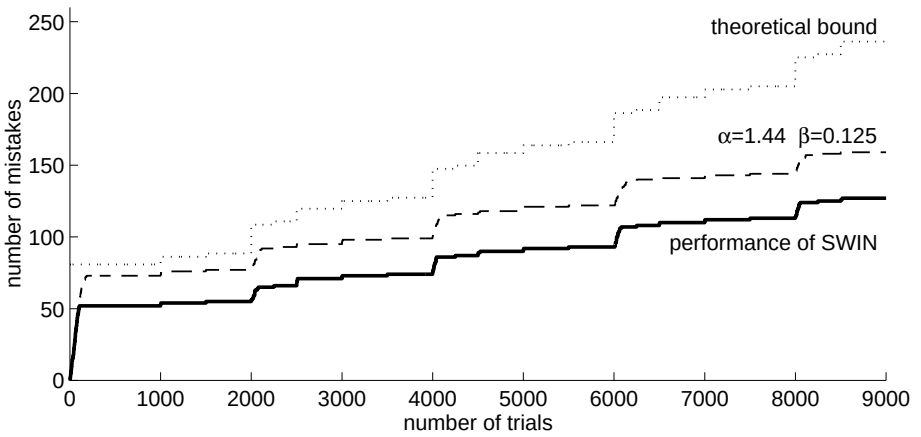

Figure 4. Tuned parameters of SWIN versus the generic parameters $\alpha = 1.44$, $\beta = 0.125$.

the tuned parameters do perform better the difference is relatively small.

The overall conclusion of our experiment is that, first, the theoretical bounds capture the actual performance of the algorithm quite well, second, that some mechanism of lower

bounding the weights of WINNOW is necessary to make the algorithm adaptive to target shifts, and third, that moderate changes in the parameters do not change the qualitative behavior of the algorithm.

7. Amortized Analysis

In this section we first prove the general bounds given in Theorems 1 and 2 for the randomized and for the deterministic version of SWIN. Then, from these bounds we calculate the bounds given in Theorems 4-7 for specific choices of the parameters.

The analysis of the algorithm proceeds by showing that the distance between the weight vector of the algorithm $\mathbf{w}_t$, and vector $\mathbf{u}_t$ representing the disjunction at trial t, decreases, if the algorithm makes a mistake. The potential/distance function used for the previous analysis of WINNOW (Littlestone, 1988, 1989, 1991) is the following generalization of relative entropy to arbitrary non-negative weight vectors:

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

(This distance function was also used for the analysis of the EGU regression algorithm (Kivinen & Warmuth, 1997), which shows that WINNOW is related to the EGU algorithm.) By taking derivatives it is easy to see that the distance is minimal and equal to 0 if and only if $\mathbf{w}_t = \mathbf{u}_t$. With the convention that $0 \ln 0 = 0$ and the assumption that $\mathbf{u} \in \{0,1\}^n$ the distance function simplifies to

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

We start with the analysis of the randomized algorithm with shifting target disjunctions. The other cases will be derived easily from this analysis. At first we calculate how much the distance $D(\mathbf{u}_t, \mathbf{w}_t)$ changes between trials:

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

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 γ, δ with

$$\gamma \leq 1 + \ln w_{t,i} \leq \delta$$

for all $1 \leq i \leq n$ and $0 \leq t \leq T-1$ we can lower bound term (1) by

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

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

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

The third equality holds since each $x_{t,i} \in \{0, 1\}$. Remember that $\mathbf{x}'_t$ is obtained from $\mathbf{x}_t$ by removing the attribute errors from $\mathbf{x}_t$. The last inequality follows from the fact that $\sum_{i=1}^{n} u_{t,i} x'_{t,i} \geq 1$ if $y_t = 1$ and $\sum_{i=1}^{n} u_{t,i} x'_{t,i} = 0$ 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)

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

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 bounds on terms (1), (2), and (3) we have

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

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 $p(\cdot, \mathbf{w}_t)$. 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

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

and

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

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

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

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

$$\sum_{t=1}^{T} (D(\mathbf{u}_{t-1}, \mathbf{w}_{t-1}) - D(\mathbf{u}_t, \mathbf{w}_t)) = D(\mathbf{u}_0, \mathbf{w}_0) - D(\mathbf{u}_T, \mathbf{w}_T) \leq nw_0$$

we can upper bound the expected number of mistakes by

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

Hence, we want to choose $p(\cdot)$ and C such that C is as big as possible. For that fix $p(\cdot)$ and let r^* be a value where $p(r^*)$ becomes 1.[5] Since the left hand sides of equations (4) and (5) are continuous we get $(r^* - \beta)(1 - 1/\alpha) \geq C$ and $r^*(1 - \alpha) + \ln \alpha \geq 0$, and combining these two we have

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

This can be achieved by choosing $p(\cdot)$ as in (RAND) which satisfies (4) and (5) for $C = (\ln \alpha - (\alpha - 1)\beta)/\alpha$. Of course we have to choose $\beta < \frac{\ln \alpha}{\alpha - 1}$. Putting everything together we have the following lemma.

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

$$\gamma \leq 1 + \ln w_{t,i} \leq \delta$$

for all $1 \leq i \leq n$ and $0 \leq t \leq T - 1$ where $w_{t,i}$ are the weights used by algorithm SWIN. *Then for all $S \in \mathcal{S}(Z^+, Z^-, A, n)$*

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

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

For the deterministic variant of our algorithm the function $p(\cdot)$ has to take values in $\{0,1\}$. Thus, we get from (4) and (5) that $(r-\beta)(1-1/\alpha) \geq C$ and $r(1-\alpha)+\ln\alpha \geq C$ which yields

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

This we get by choosing $p(\cdot)$ as in (DET) which satisfies (4) and (5) for $C = (\ln\alpha - (\alpha-1)\beta)/(\alpha+1)$.

LEMMA 2 *Let $\beta < \frac{\ln\alpha}{\alpha-1}$ and assume that*

$$\gamma \leq 1 + \ln w_{t,i} \leq \delta$$

for all $1 \leq i \leq n$ and $0 \leq t \leq T-1$ where $w_{t,i}$ are the weights used by algorithm SWIN. *Then for all $S \in \mathcal{S}(Z^+, Z^-, A, n)$*

$$M(\text{SWIN}, Z, A) \leq (\alpha+1)\frac{\delta Z^- - \gamma Z^+ + A\ln\alpha + n w_0}{\ln\alpha - (\alpha-1)\beta}$$

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

Now we are going to calculate bounds γ, δ on $1 + \ln w_{t,i}$. We get these bounds 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 (RAND) or (DET), and $r_t \geq w_{t-1,i}x_{t,i}$ we find that $w_{t,i} \leq \alpha$. Thus $\ln\frac{e\beta}{n} \leq 1 + \ln w_{t,i} \leq \ln(e\alpha)$.

LEMMA 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 (RAND) or (DET) satisfy*

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

and

$$\ln\frac{e\beta}{n} \leq 1 + \ln w_{t,i} \leq \ln(e\alpha).$$

Proof of Theorems 1 and 2. By Lemmas 1, 2, and 3. □

Proof of Theorem 3 In the non-shifting case where $\mathbf{u}_1 = \cdots = \mathbf{u}_T = \mathbf{u}$ and $\mathbf{u}_0 = (0, \ldots, 0)$ term (1) is 0 for all $t \geq 2$, and it is

$$D(\mathbf{u}_0, \mathbf{w}_0) - D(\mathbf{u}_1, \mathbf{w}_0) = k + k\ln w_0$$

for $t = 1$ where $k = \sum_{i=1}^n u_i$ is the number of attributes in the target disjunction $\mathbf{u}$. Thus, in the non-shifting case the term $\delta Z^- - \gamma Z^+$ in the upper bounds of Lemmas 1 and 2 can be replaced by $k\ln\frac{1}{ew_0}$, which gives the theorem. □

7.1. Proofs for specific choices of the parameters

Proofs of Theorems 4 and 5. By Theorem 3 and Corollary 1 and simple calculations. □

Proof of Theorem 6. For $\alpha = 1+\epsilon$ and $w_0 = k/n$ we get from Theorem 3 that

$$\mathbf{E}M(\textsc{Swin}, S) \le (c+\epsilon)\frac{k\ln\frac{n}{ek} + A\ln(1+\epsilon) + k}{\ln(1+\epsilon)}$$

with $c = 1$ and

$$M(\textsc{Swin}, S) \le (c+\epsilon)\frac{k\ln\frac{n}{ek} + A\ln(1+\epsilon) + k}{\ln(1+\epsilon)}$$

with $c = 2$. Then

$$\begin{aligned}
&(c+\epsilon)\frac{k\ln\frac{n}{ek} + A\ln(1+\epsilon) + k}{\ln(1+\epsilon)}\\
&\quad\le\ (c+\epsilon)A + (c+\epsilon)\frac{k\ln\frac{n}{k}}{\epsilon - \epsilon^2/2}\\
&\quad\le\ (c+\epsilon)A + (c+\epsilon)\frac{1+\epsilon}{\epsilon}k\ln\frac{n}{k}\\
&\quad=\ cA + \left(\epsilon A + \frac{c}{\epsilon}k\ln\frac{n}{k}\right) + (c+1+\epsilon)k\ln\frac{n}{k}.
\end{aligned}$$

In the second inequality we used that $\epsilon \le 1$. Substituting the values for c and ϵ gives the statements of the theorem. □

Proof of Theorem 7. For $\alpha = 1+\epsilon$, $\beta = \frac{\epsilon}{\ln\epsilon^{-1}}$, and $w_0 = \frac{\beta}{n}$ we get from Corollary 1 that

$$\mathbf{E}M(\textsc{Swin}, S) \le (c+\epsilon)\cdot \cdot\frac{A\ln(1+\epsilon) + Z\ln\left(\frac{(1+\epsilon)n\ln\epsilon^{-1}}{\epsilon}\right)\Big/ 2 + \min\{n, Z\}\cdot\ln\left(\frac{n\ln\epsilon^{-1}}{e^2(1+\epsilon)\epsilon}\right)\Big/ 2 + \frac{\epsilon}{\ln\epsilon^{-1}}}{\ln(1+\epsilon) - \epsilon^2/\ln\epsilon^{-1}}$$

with $c = 1$ and

$$M(\textsc{Swin}, S) \le (c+\epsilon)\cdot \cdot\frac{A\ln(1+\epsilon) + Z\ln\left(\frac{(1+\epsilon)n\ln\epsilon^{-1}}{\epsilon}\right)\Big/ 2 + \min\{n, Z\}\cdot\ln\left(\frac{n\ln\epsilon^{-1}}{e^2(1+\epsilon)\epsilon}\right)\Big/ 2 + \frac{\epsilon}{\ln\epsilon^{-1}}}{\ln(1+\epsilon) - \epsilon^2/\ln\epsilon^{-1}}$$

with $c = 2$. Then for $\epsilon \le 1/10$

$$\begin{aligned}
&(c+\epsilon)\frac{A\ln(1+\epsilon) + Z\ln\left(\frac{(1+\epsilon)n\ln\epsilon^{-1}}{\epsilon}\right)\Big/ 2 + \min\{n, Z\}\cdot\ln\left(\frac{n\ln\epsilon^{-1}}{e^2(1+\epsilon)\epsilon}\right)\Big/ 2 + \frac{\epsilon}{\ln\epsilon^{-1}}}{\ln(1+\epsilon) - \epsilon^2/\ln\epsilon^{-1}}\\
&\quad\le\ (c+\epsilon)A\frac{\ln(1+\epsilon)}{\ln(1+\epsilon) - \epsilon^2/\ln\epsilon^{-1}} + (c+\epsilon)\frac{(Z+\min\{n,Z\})\ln\left(\frac{(1+\epsilon)n\ln\epsilon^{-1}}{\epsilon}\right)\Big/ 2}{\ln(1+\epsilon) - \epsilon^2/\ln\epsilon^{-1}}\\
&\quad\le\ (c+\epsilon)A\left(1 + \frac{2\epsilon}{\ln\epsilon^{-1}}\right) + (c+\epsilon)\frac{(Z+\min\{n,Z\})\ln\left(\frac{n}{\epsilon^2}\right)\Big/ 2}{\epsilon}(1+\epsilon)
\end{aligned}$$

$$
\begin{aligned}
&\leq cA + \left(\epsilon A + \frac{c}{2\epsilon}(Z + \min\{n, Z\}) \ln\left(\frac{n}{\epsilon^2}\right)\right) \\
&\quad + \frac{2(c+\epsilon)\epsilon A}{\ln \epsilon^{-1}} + (c+1+\epsilon)(Z + \min\{n, Z\}) \ln\left(\frac{n}{\epsilon^2}\right) \Big/ 2 \\
&\leq cA + \epsilon A \left(2 + \frac{2(c+\epsilon)}{\ln \epsilon^{-1}} + \frac{\epsilon(c+1+\epsilon)}{c}\right)
\end{aligned}
$$

for

$$\epsilon = \sqrt{\frac{c}{2} \frac{Z + \min\{n, Z\}}{A} \ln \frac{An}{Z}}.$$

This gives the bounds of the theorem. □

8. 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 $\mathcal{T} = \langle \mathbf{u}_1, \ldots, \mathbf{u}_T \rangle$ where each disjunction $\mathbf{u}_t$ consists of exactly one literal, i.e., $\mathbf{u}_t = \mathbf{e}_j$ for some j where $\mathbf{e}_j$ is the jth unit vector.

Our first lower bound shows that for any deterministic algorithm there is an adversarial example sequence in $\mathcal{S}(Z, A, n)$ such that the algorithm makes at least $2A + \Omega(Z \log n)$ mistakes. Related upper bounds are given in Theorems 4 and 7.

THEOREM 8 *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 $\mathbf{e}_j$.

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. Observe that a lower bound for deterministic algorithms like

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

implies the following lower bound on randomized algorithms:

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

This follows from the fact that any randomized learning algorithm can be turned into a deterministic learning algorithm which makes at most twice as many mistakes as the randomized algorithm makes in the average. This means that Theorem 8 implies for any randomized algorithm L that there are sequences $S \in \mathcal{S}(Z, A, n)$ with $\mathbf{E}M(L, S) \geq A + \left\lfloor \frac{Z}{4} \right\rfloor \lfloor \log_2 n \rfloor$.

Remark. As an open problem it remains to show lower bounds that have the same form as the upper bounds of Theorem 7 with the square root term.

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

LEMMA 4 (Littlestone & Warmuth, 1994) *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 (Cesa-Bianchi et al., 1997) gives

LEMMA 5 (Cesa-Bianchi et al., 1997) *There are functions $n(\eta)$ and $A(n, \eta)$ such that for any $\eta > 0$, any randomized 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. The corresponding upper bounds are given in Theorems 5 and 6.

THEOREM 9 *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 10 *There are functions $n(\eta)$ and $A(n,\eta)$ such that for any $\eta > 0$, any randomized 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 9 and 10. 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 $n_i \geq \left\lfloor \frac{n}{k} \right\rfloor$ attributes. Furthermore, we choose numbers $a_i \geq \left\lfloor \frac{A}{k} \right\rfloor$, $i = 1, \ldots, k$, with $\sum_{i=1}^{k} a_i = A$. For each group G_i we choose a sequence $S_i \in \mathcal{S}_0(1, a_i, n_i)$, according to Lemmas 4 and 5, 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), respectively. Hence

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

and

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

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

The last theorem shows that for randomized algorithms the constant of 1 before A in Theorem 6 is optimal and that the best constant before the square root term is in $[1,2]$.

9. Conclusion

We developed algorithm SWIN which is a variant of WINNOW for on-line learning of disjunctions subject to target shift. We proved worst case mistake bounds for SWIN which hold for any sequence of examples and any kind of target drift (where the amount of error in the examples and the amount of shifts is bounded). There is a deterministic and a randomized version of SWIN where the analysis of the randomized version is more involved and interesting in its own right. Lower bounds show that our worst case mistake bounds are close to optimal in some cases. Computer experiments highlight that an explicit mechanism *is* necessary to make the algorithm adaptive to target shifts.

Acknowledgments

We would like to thank Mark Herbster and Nick Littlestone for valuable discussions. We also thank the anonymous referees for their helpful comments. M. K. Warmuth acknowledges the support of the NSF grant CCR 9700201.

Notes

1. By expanding the dimension to $2n$, learning non-monotone disjunctions reduces to the monotone case.
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 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 up to $A+Z$ 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.
3. For this potential function the weights must be positive. Negative weights are handled via a reduction (Littlestone, 1988, 1989; Haussler et al., 1994).
4. In the worst case the randomized algorithm makes a mistake with probability 1/2 in each trial and the deterministic algorithm always breaks the tie in the wrong way such that it makes a mistake in each trial. Thus the number of mistakes of the deterministic algorithm is twice the expected number of mistakes of the randomized algorithm.
5. Formally let $r_1, r_2, \ldots \to r^*$ with $p(r^*) = 1$ and $p(r_j) < 1$. Such a sequence $\langle r_j \rangle$ exists if $p(\cdot)$ is not equal to 1 everywhere and if there is a value r with $p(r) = 1$. For functions $p(\cdot)$ not satisfying these conditions algorithm SWIN can be forced to make an unbounded number of mistakes even in the absence of attribute errors.

References

Auer, P. (1993). On-line learning of rectangles in noisy environments. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory* (pp. 253–261). New York, NYU: ACM Press.

Auer, P., & Warmuth, M. K. (1995). Tracking the best disjunction. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science* (pp. 312–321). IEEE Computer Society Press.

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., & Warmuth, M. K. (1997). How to use expert advice. *Journal of the ACM*. (To appear)

Cesa-Bianchi, N., Freund, Y., Helmbold, D. P., & Warmuth, M. K. (1996). On-line prediction and conversion strategies. *Machine Learning*, *25*, 71–110. (An extended abstract appeared in *Eurocolt '93*)

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

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* Wiley.
Haussler, D., Kivinen, J., & Warmuth, M. K. (1994). *Tight worst-case loss bounds for predicting with expert advice* (Tech. Rep. No. UCSC-CRL-94-36). University of California, Santa Cruz, Computer Research Laboratory. (An extended abstract appeared in Eurocolt 1995. To appear in *IEEE Transactions on Information Theory*.)
Haykin, S. (1994). *Neural networks: a comprehensive foundation.* New York, NY: Macmillan.
Helmbold, D. P., & Schapire, R. E. (1997). Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, *27*, 51-68.
Herbster, M., & Warmuth, M. (1998). Tracking the best expert. *Machine Learning*. (Appears in this special issue.)
Jumarie, G. (1990). *Relative information.* Springer-Verlag.
Kapur, J. N., & Kesavan, H. K. (1992). *Entropy optimization principles with applications.* Academic Press, Inc.
Kivinen, J., & Warmuth, M. K. (1997). Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, *132*(1), 1–64.
Kivinen, J., Warmuth, M. K., & Auer, P. (1997). The perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*. (To appear)
Littlestone, N. (1988). Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.
Littlestone, N. (1989). *Mistake bounds and logarithmic linear-threshold learning algorithms.* Unpublished doctoral dissertation, Technical Report UCSC-CRL-89-11, University of California Santa Cruz.
Littlestone, N. (1991). Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory* (pp. 147–156). San Mateo, CA: Morgan Kaufmann.
Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, *108*(2), 212–261.
Maass, W., & Warmuth, M. K. (1998). Efficient learning with virtual threshold gates. *Information and Computation*. (To appear)
Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, *65*, 386–407. (Reprinted in *Neurocomputing* (MIT Press, 1988).)
Vovk, V. (1990). Aggregating strategies. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory* (pp. 371–383). Morgan Kaufmann.

Received September 16, 1997
Accepted December 12, 1997
Final Manuscript February 10, 1998