Journal of Machine Learning Research 4 (2003) 819-838 Submitted 4/03; Published 10/03

# Tracking Linear-threshold Concepts with Winnow

**Chris Mesterharm** MESTERHA@CS.RUTGERS.EDU
*Department of Computer Science*
*Rutgers University*
*110 Frelinghuysen Road*
*Piscataway, NJ 08854, USA*

**Editors:** Ralf Herbrich and Thore Graepel

## Abstract

In this paper, we give a mistake-bound for learning arbitrary linear-threshold concepts that are allowed to change over time in the on-line model of learning. We use a variation of the Winnow algorithm and show that the bounds for learning shifting linear-threshold functions have many of the same advantages that the traditional Winnow algorithm has on fixed concepts. These benefits include a weak dependence on the number of irrelevant attributes, inexpensive runtime, and robust behavior against noise. In fact, we show that the bound for tracking Winnow has even better performance with respect to irrelevant attributes. Let $X \in [0,1]^n$ be an instance of the learning problem. In the previous bounds, the number of mistakes depends on $\ln n$. In this paper, the shifting concept bound depends on $\max \ln(\|X\|_1)$. We show that this behavior is a result of certain parameter choices in the tracking version of Winnow, and we show how to use related parameters to get a similar mistake bound for the traditional fixed concept version of Winnow.

**Keywords:** Winnow, Concept Tracking, On-line Learning

## 1. Introduction

In this paper, we give a mistake bound for tracking shifting concepts using a variation of the Winnow algorithm. We show that this version of Winnow can learn arbitrary linear-threshold functions that are allowed to change over time.

The mistake bound applies to the on-line model of learning (Littlestone, 1989). On-line learning is composed of trials where each trial is divided into three steps. First, in trial $t$, the algorithm receives an instance, $X^t \in [0,1]^n$. Next, the algorithm predicts the label of the instance, $\hat{y}^t \in \{0,1\}$. Last, the environment returns the correct classification, $y^t \in \{0,1\}$. Normally, the correct classification is assumed to come from some fixed target function. In this paper, we will assume the target function is allowed to shift over the course of the trials. The goal of the algorithm is to minimize the total number of prediction mistakes made during the trials.

Winnow is an on-line algorithm that predicts with a linear-threshold function. The algorithm uses the correct classification returned by the environment to update the weights of the linear-threshold function. The update procedure is similar to the Perceptron algorithm (Rosenblatt, 1962, Minsky and Papert, 1969) except that the updates are multiplicative instead of additive. This results in an algorithm that is relatively insensitive to the behavior of a large number of irrelevant attributes, inexpensive to run, and robust to noise (Littlestone, 1988). The version of Winnow that we will be dealing with in this paper is often called, in recent literature, the unnormalized version of Winnow to distinguish it from other similar variations.

Our tracking version of the Winnow algorithm uses two tricks. The first is a standard trick to force the attribute weights to stay above a given constant, $\varepsilon > 0$. Anytime a normal Winnow update attempts to lower a weight below $\varepsilon$, the weight is set to $\varepsilon$. Intuitively, this modification allows the algorithm to quickly learn a moving concept by not allowing the weights to get too small. When an attribute becomes relevant, its weight only needs to be increased from $\varepsilon$ to the new value. After proving a bound with the above algorithm, we refine the result by preprocessing the instances to guarantee that the attribute weights can not get too large. This is accomplished by shifting the attributes so they are not allowed to get arbitrarily close to zero. While it is not obvious, very small positive attribute values can lead to large weights in the Winnow algorithm. These large weights increase the mistake bound, so we need some type of guarantee that the weights can not get too large. Preprocessing the instances is one way to get this guarantee.

The minimum weight modification has been used with various Winnow type algorithms. In particular, it has been used to give mistake-bounds for learning changing experts (Littlestone and Warmuth, 1994, Herbster and Warmuth, 1998) and shifting disjunctions (Auer and Warmuth, 1998). This paper builds on those results by giving a mistake-bound for learning arbitrary linear-threshold functions that are allowed to change over the trials. These types of algorithms have been shown to be useful in practice when learning shifting concepts such as predicting disk idle times for mobile applications (Helmbold et al., 2000) and solving load balancing problems on a network of computers (Blum and Burch, 2000). The additional knowledge that some of these algorithms have good bounds when learning arbitrary shifting linear-threshold concepts may help justify applying these algorithms to a wider range of tracking problems.

There are other linear-threshold algorithms that can track concepts. In Herbster (2001) a variation of the Perceptron algorithm is given that allows tracking concepts. In Kivinen et al. (2002), various $p$-norm algorithms are given that allow concept tracking. The $p$-norm algorithms were originally developed by Grove et al. (2001) as a class of linear-threshold algorithms that can learn fixed concepts. Later we will compare the bounds in this paper to two types of tracking $p$-norm algorithms. Other related work includes Herbster and Warmuth (2001) which gives a technique to convert a wide range of algorithms that learn fixed linear functions to tracking regression problems.

Another contribution of this paper is to show that this tracking version of Winnow eliminates the dependence of the algorithm on the number of attributes. With an appropriate setting of parameters, instead of the algorithm depending on $\ln(n)$, as in the normal Winnow algorithm proof (Littlestone, 1989, Auer and Warmuth, 1998), the algorithm depends on the maximum value over all trials of $\ln(\|X^t\|_1)$. We will also show how these parameters can be used for the normal Winnow algorithm to achieve a similar mistake bound. Therefore these algorithms performs well with sparse instances in ways that are similar to infinite attribute algorithms (Blum et al., 1991).

The remainder of the paper is organized as follows. In Section 2, we give a formal statement of the concept tracking problem and the tracking Winnow algorithm. We then present the mistake bound. In Section 3, we give a proof for this mistake bound. In Section 4, we analyze the strengths and weaknesses of the bound and give the instance transformation that improves the bound when the attribute values become small. In Section 5, we compare the mistake bound to previous bounds on certain types of linear-threshold problems. Section 6 contains the conclusion and ideas for future work.

## 2. Problem Statement and Solution

In this section, we review the concept tracking Winnow algorithm and give give the notation that is needed to understand an upper-bound on the number of mistakes.

### 2.1 Algorithm

Here is the modified Winnow algorithm for learning shifting concepts (Littlestone and Warmuth, 1994, Auer and Warmuth, 1998). The only change, in this paper, is that we always set the initial weights to the value of the minimum weight, $\varepsilon$.

**Initialization**

Weight multiplier, $\alpha > 1$.
Minimum value of the weights, $\varepsilon > 0$.
Algorithm weights, $w_1^1, \ldots, w_n^1 = \varepsilon$.
Trial number, $t = 1$.

The algorithm proceeds in a series of trials composed of the following three steps.

**Instance at trial** $t$

$X^t = (x_1^t, \ldots, x_n^t) \in [0,1]^n$

**Prediction at trial** $t$

if $\sum_{i=1}^n w_i^t x_i^t \geq 1$
  predict $\hat{y}^t = 1$
else
  predict $\hat{y}^t = 0$

**Update at trial** $t$

correct label $= y^t$
if $y^t = \hat{y}^t$
  $\forall i \in \{1, \ldots, n\}$ $w_i^{t+1} = w_i^t$
else (mistake)
  if $y^t = 1$ (promotion step)
    $\forall i \in \{1, \ldots, n\}$ $w_i^{t+1} = \alpha^{x_i^t} w_i^t$
  else $y^t = 0$ (demotion step)
    $\forall i \in \{1, \ldots, n\}$ $w_i^{t+1} = \max(\varepsilon, \alpha^{-x_i^t} w_i^t)$
$t \leftarrow t + 1$

### 2.2 Concept

The model of on-line tracking we present in this paper is similar to models defined by Kuh et al. (1991) and Helmbold and Long (1991). The central element of the mistake bound is a sequence of concepts. Let $C = (C_1, \ldots, C_T)$ be a sequence of concepts with one concept per trial where $T$ is the final trial. The concept is allowed to change each trial, and the bound will depend on how often and how much the concept changes. Intuitively, a good bound should be possible if the concept makes small changes or if the concept makes infrequent large changes. Allowing the concept to vary each trial allows us to represent changes that occur every trial, however the concept might remain the same for large sequences of trials.

An adversary generates the instances, and the goal of the algorithm is to minimize the number of mistakes. The mistake bound will depend on the sequence of concepts and the amount of noise in the instances generated by the adversary.

We define concept $C_t$ by specifying the weights, $U^t$, of a linear-threshold function and a margin parameter $\delta^t$. The concept is 1 if $\sum_{i=1}^n u_i^t x_i^t \geq 1$ and 0 otherwise. The margin parameter specifies a distance from the linear separator such that any instance that occurs in this region is considered a noisy instance. This is a standard assumption that is needed to prove mistake bounds for these types of algorithms (Littlestone, 1988, 1991, Grove et al., 2001). All other noisy instances come from instances misclassified by the current concept.

**Concept $C_t$ parameters**

$u_1^t, \ldots, u_n^t \geq 0$ where $u_i^t \in \mathbb{R}$
$0 < \delta^t \leq 1$ where $\delta^t \in \mathbb{R}$

Let $\aleph^t$ be a measure of the noise in trial $t$. An instance with $\aleph^t = 0$ corresponds to an instance of the trial concept. An instance with $\aleph^t > 0$ corresponds to a noisy instance, and $\aleph^t$ is a measure of the error in the linear-threshold function. The following noise function is related to the hinge-loss (Gentile and Warmuth, 1999).

**Concept $C_t$ noise**

if $y^t = 1$
$\aleph^t = \max\left(0, 1 + \delta^t - \sum_{i=1}^n (u_i^t x_i^t)\right)$
if $y^t = 0$
$\aleph^t = \max\left(0, \sum_{i=1}^n (u_i^t x_i^t) - (1 - \delta^t)\right)$

If the adversary is allowed to generate an arbitrary number of noisy instances, the concept would be impossible to learn. We can restrict the amount of noisy by making assumptions on $\aleph^t$. For example, we can assume that $\sum_{t=1}^T \aleph^t \leq K$ where $K \in \mathbb{R}$. This is the noise model in Littlestone (1989) where only a finite amount of noise is allowed over all trials. Another possibility is to allow $\sum_{i=1}^t \aleph^t \leq Kt$ where $K \in \mathbb{R}$ and $t$ is the current trial number. This allows the noise to grow linearly with the trial number. We can even assume that the various $\aleph^t$ are random variables allowing something similar to the noise model of Littlestone (1991). In general, the bound will depend on $\sum_{t=1}^T \aleph^t$, and we can make assumptions to constrain the behavior of this value.

There are other ways to understand the mistake-bounds given in this paper. Instead of focusing on an adversary picking the concept, we can focus on the instances. For a given sequence of instances, any concept sequence can be used to generate the instances. Obviously some concept sequences will have more noise than others. All the concept sequences will generate an upper-bound on the mistakes; we are interested in the concept sequence that gives the smallest upper-bound.

### 2.3 Mistake-bound

The mistake bound uses the following notation. This notation will be explained and motivated in the proof section. For notational convenience, we assume that $T$ is the final trial, although our results can apply to a potentially infinite set of trials.

**Terms used in the main mistake bound**

Let $\lambda \geq \max_{t \in \{1,\ldots,T\}} \|X^t\|_1$.
Let $\zeta = \min_{t \in \{1,\ldots,T\},\ i \in \{1,\cdots,n\}} \{x_i^t \mid x_i^t > 0\}$.

Let $H(C) = \sum_{i=1}^{n}(u_i^T + \sum_{t=1}^{T-1} \max(0, u_i^t - u_i^{t+1}))$.
Let $\delta \leq \min_{t\in\{1,\ldots,T\}} \delta^t$.
Let $N = \sum_{t=1}^{T} \aleph^t$

**Theorem 1** *For instances generated by a concept sequence $C$, if $\alpha = 1+\delta$ and $\varepsilon = \frac{\delta}{50\lambda}$ then the number of mistakes is less than*

$$\frac{H(C)(2.05+\delta)\left(\zeta\delta + \ln\left(\frac{\lambda}{\delta\zeta}\right) + 3.92\right)}{\delta^2} + \frac{(2.05+\delta)N}{\delta+\delta^2} \quad .$$

We can use asymptotic notation to make this bound a little more digestible. Removing some lower order terms and using the fact that $0 < \zeta, \delta \leq 1$, we get

$$O\left(H(C)\ln\left(\frac{\lambda}{\delta\zeta}\right)/\delta^2 + N/\delta\right) \quad .$$

## 3. Proof of Mistake-bound

The proof in this section closely follows the style of proof given by Littlestone (1988). While most modern Winnow type proofs use a potential function to bound the number of mistakes (Littlestone, 1989, 1991, Auer and Warmuth, 1998, Grove et al., 2001), we have found it useful to go back to the old style of proof (Littlestone, 1988) to deal with tracking linear-threshold functions. However, potential functions do have certain advantages. In particular, they are useful for generalizing over a range of learning algorithm (Grove et al., 2001, Herbster and Warmuth, 2001, Kivinen et al., 2002). It is a topic for future research to understand how this type of proof relates to the potential function technique.

The purpose of the next four lemmas is to give an upper-bound on the number of demotions as a function of the number of promotions. Remember that a promotion is an update that increases the weights on a mistake where the correct label is 1, and a demotion is an update that decreases the weights on a mistake where the correct label is 0. Intuitively, a bound must exist as long as every demotion removes at least a fixed constant of weight. This is because the algorithm always has positive weights and the only way the weights can increase is through a promotion.

Let $\triangle(f)$ represent the change in function $f$ after a trial is completed. For example, $\triangle(\sum_{i=1}^{n} w_i^t) = \sum_{i=1}^{n} w_i^{t+1} - \sum_{i=1}^{n} w_i^t$.

**Lemma 2** *On a promotion, $\triangle(\sum_{i=1}^{n} w_i^t) < (\alpha - 1)$.*

**Proof** After a promotion,

$$\sum_{i=1}^{n} w_i^{t+1} = \sum_{i=1}^{n} w_i^t \alpha^{x_i^t} \quad .$$

Next, we use the fact that $\alpha^{x_i^t} \leq (\alpha-1)x_i^t + 1$ for all $x_i^t \in [0,1]$. (This is true because $\alpha^{x_i^t}$ is a convex function.) This shows that,

$$\sum_{i=1}^{n} w_i^t \alpha^{x_i^t} \leq \sum_{i=1}^{n} w_i^t[(\alpha-1)x_i^t + 1] = (\alpha-1)\sum_{i=1}^{n} w_i^t x_i^t + \sum_{i=1}^{n} w_i^t \quad .$$

Since we have a promotion, $\sum_{i=1}^{n} w_i^t x_i^t < 1$. This combined with the above facts and remembering that $\alpha > 1$ gives,

$$\sum_{i=1}^{n} w_i^{t+1} < (\alpha - 1) + \sum_{i=1}^{n} w_i^t \ .$$

Therefore,

$$\triangle\left(\sum_{i=1}^{n} w_i^t\right) = \sum_{i=1}^{n} w_i^{t+1} - \sum_{i=1}^{n} w_i^t < (\alpha - 1) \ .$$

■

Next we want to prove a similar result about demotions. However, because demotions have the added difficulty of not being able to lower the weight below $\varepsilon$, the proof is a little more complex. First we will prove a small lemma that will help with the more difficult proof.

We are going to consider two types of attributes that occur during a demotion. Let $A \subseteq \{1, \ldots, n\}$ be all $i$ such that $w_i \alpha^{-x_i^t} < \varepsilon$. These are the indexes of weights that are forced to $\varepsilon$ since $w_i^{t+1} = \max(\varepsilon, \alpha^{-x_i^t} w_i^t)$. Let $B \subseteq \{1, \ldots, n\}$ be the indexes of weights that have a normal demotion. These are the attributes such that $x_i^t > 0$ and $w_i^t \alpha^{-x_i^t} \geq \varepsilon$. All the other weights do not change.

**Lemma 3** *On a demotion, if* $\sum_{i \in B} w_i^t x_i^t \geq \theta$ *then* $\triangle\left(\sum_{i \in B} w_i^t\right) \leq \frac{1-\alpha}{\alpha}\theta$.

**Proof** After a demotion,

$$\sum_{i \in B} w_i^{t+1} = \sum_{i \in B} w_i^t \, \alpha^{-x_i^t} \ .$$

Next, we use the fact that $\alpha^{-x_i^t} \leq \left(\frac{1}{\alpha} - 1\right) x_i^t + 1$ for all $x_i^t \in [0, 1]$ (This fact is true because $\alpha^{-x_i^t}$ is a convex function.) to show that

$$\sum_{i \in B} w_i^t \, \alpha^{-x_i^t} \leq \sum_{i \in B} w_i^t \left[\left(\frac{1}{\alpha} - 1\right) x_i^t + 1\right] = \frac{1 - \alpha}{\alpha} \sum_{i \in B} w_i^t x_i^t + \sum_{i \in B} w_i^t \ .$$

Based on the assumption of the lemma, $\sum_{i \in B} w_i^t x_i^t \geq \theta$. This combined with the above facts and remembering that $\alpha > 1$ gives,

$$\sum_{i \in B} w_i^{t+1} \leq \frac{1 - \alpha}{\alpha}\theta + \sum_{i \in B} w_i^t \ .$$

Therefore,

$$\triangle\left(\sum_{i \in B} w_i^t\right) = \sum_{i \in B} w_i^{t+1} - \sum_{i \in B} w_i^t \leq \frac{1 - \alpha}{\alpha}\theta \ .$$

■

Here is the main lemma concerning demotions. This lemma gives an upper-bound on how much the weights decrease after a demotion. The amount of decrease will depend on various factors including $\lambda \geq \max_{t \in N} \sum_{i=1}^{n} x_i^t$. Also this lemma assumes that $\varepsilon < 1/\lambda$. This is a reasonable assumption that ensures the algorithm can lower weights during demotions. Otherwise the algorithm could be forced to always make a mistake on certain instances for many linear-threshold concepts. For example, assume there are $\lambda$ attributes that are not part of the target concept. An adversary can give these attributes value 1 on every instance and every other attribute value 0. This creates an instance with label 0. If $\varepsilon \geq 1/\lambda$ then the algorithm will always incorrectly predict 1 since

$$\sum_{i=1}^{n} w_i^t x_i^t \geq \sum_{i=1}^{n} \varepsilon x_i^t = \varepsilon \sum_{i=1}^{n} x_i^t = \varepsilon\lambda \geq 1.$$

**Lemma 4** *On a demotion, if* $\varepsilon < 1/\lambda$ *then* $\triangle\left(\sum_{i=1}^{n} w_i^t\right) < \frac{1-\alpha}{\alpha}(1-\varepsilon\lambda)$.

**Proof** We are going to use the set of attributes $A$ and $B$ defined earlier. We will break the proof into two cases. For the first case assume that there is at least one attribute in $B$. Since a demotion has occurred we know,

$$\sum_{i \in A} w_i^t x_i^t + \sum_{i \in B} w_i^t x_i^t = \sum_{i=1}^{n} w_i^t x_i^t \geq 1 \quad .$$

From this we can use Lemma 3 to derive that

$$\triangle\left(\sum_{i \in B} w_i^t\right) \leq \frac{1-\alpha}{\alpha}\left(1 - \sum_{i \in A} w_i^t x_i^t\right) \quad .$$

Now we want to get $\sum_{i \in A} w_i^t x_i^t$ into a more useful form. Let $v_i^t$ represent the amount $w_i^t$ is above $\varepsilon$.

$$\sum_{i \in A} w_i^t x_i^t = \sum_{i \in A} (\varepsilon + v_i^t) x_i^t = \varepsilon \sum_{i \in A} x_i^t + \sum_{i \in A} v_i^t x_i^t < \varepsilon\lambda + \sum_{i \in A} v_i^t \quad .$$

Being careful to keep track of what is negative, we can substitute this into the previous formula.

$$\triangle\left(\sum_{i \in B} w_i^t\right) < \frac{1-\alpha}{\alpha}\left(1 - \varepsilon\lambda - \sum_{i \in A} v_i^t\right) \quad .$$

Next, since all the weights with an index in $A$ get demoted to $\varepsilon$, we know that for all $i \in A$, the weight $v_i^t$ will be removed. Therefore $\triangle\left(\sum_{i \in A} w_i^t\right) = -\sum_{i \in A} v_i^t$. Combining this information proves the first case where $B$ has at least one element.

$$\triangle\left(\sum_{i=1}^{n} w_i^t\right) = \triangle\left(\sum_{i \in B} w_i^t\right) + \triangle\left(\sum_{i \in A} w_i^t\right)$$

$$< \frac{1-\alpha}{\alpha}\left(1 - \varepsilon\lambda - \sum_{i \in A} v_i^t\right) - \sum_{i \in A} v_i^t \leq \frac{1-\alpha}{\alpha}(1-\varepsilon\lambda) \quad .$$

The second case, where $B$ is empty, is similar. Using some of the same notation,

$$\triangle\left(\sum_{i=1}^{n} w_i^t\right) = \triangle\left(\sum_{i \in A} w_i^t\right) = -\sum_{i \in A} v_i^t \quad .$$

Since a demotion has occurred, we know that $\sum_{i=1}^{n} w_i^t x_i^t \geq 1$, but since the only active attributes are in $A$,

$$1 \leq \sum_{i=1}^{n} w_i^t x_i^t = \sum_{i \in A} (\varepsilon + v_i^t) x_i^t = \varepsilon \sum_{i \in A} x_i^t + \sum_{i \in A} v_i^t x_i^t \leq \varepsilon\lambda + \sum_{i \in A} v_i^t \quad .$$

We can use this to bound $\triangle\left(\sum_{i=1}^{n} w_i^t\right)$.

$$\triangle\left(\sum_{i=1}^{n} w_i^t\right) = -\sum_{i\in A} v_i^t \leq \varepsilon\lambda - 1 < \frac{1-\alpha}{\alpha}(1-\varepsilon\lambda) \quad .$$

The last step of the inequality is true since we assumed $\varepsilon < 1/\lambda$. ■

Now we can combine the lemmas to give an upper-bound on the number of demotions as a function of the number of promotions. Let $P = \{t \mid \text{promotion mistake on trial } t\}$, and let $D = \{t \mid \text{demotion mistake on trial } t\}$.

**Lemma 5** *If* $\varepsilon < 1/\lambda$ *then at any trial,* $|D| < \frac{\alpha}{1-\varepsilon\lambda}|P|$.

**Proof** We know $\sum_{i=1}^{n} w_i^t$ can never go below $\varepsilon n$, and we know $\sum_{i=1}^{n} w_i^t$ has a value of $\varepsilon n$ at the beginning of the algorithm. Since the weights can only change during demotions and promotions, we can relate the minimum weight to the current weight as follows.

$$\varepsilon n \leq \varepsilon n + \sum_{t\in P} \triangle\left(\sum_{i=1}^{n} w_i^t\right) + \sum_{t\in D} \triangle\left(\sum_{i=1}^{n} w_i^t\right) \quad .$$

Using the upper-bounds from Lemma 2 and Lemma 4,

$$\varepsilon n < \varepsilon n + (\alpha-1)|P| + \frac{1-\alpha}{\alpha}(1-\varepsilon\lambda)|D| \quad .$$

Noting that $1-\alpha$ is negative, we can rearrange the inequality to get the result that $|D| < \frac{\alpha}{1-\varepsilon\lambda}|P|$. ■

Our goal at this stage is to show how the relevant weights (weights, $w_i$, where the corresponding target weights have $u_i > 0$) increase in weight during the running of the algorithm. If we can show that they must eventually increase, and that they have a maximum value, we can derive a mistake bound on the algorithm. First we want to give a bound on the maximum value of a weight. Let $w_{max}$ be the maximum value of any weight, and $\zeta = \min_{t\in\{1,\ldots,T\},\ i\in\{1,\cdots,n\}}\{x_i^t \mid x_i^t > 0\}$.

**Lemma 6** *If* $\alpha < e$ *then* $w_{max} \leq \max\left(\varepsilon, \frac{\alpha^\zeta}{\zeta}\right)$.

**Proof** If a promotion never occurs the maximum weight is the initial weight $\varepsilon$. The only time a weight, $w_a^t$ can increase is after a promotion where $x_a^t > 0$. Promotions can only occur if $\sum_{i=1}^{n} w_i^t x_i^t < 1$. Therefore for any attribute $x_a^t > 0$,

$$w_a^t x_a^t \leq \sum_{i=1}^{n} w_i^t x_i^t < 1 \quad .$$

Since the new weight is $\alpha^{x_a^t} w_a^t$, we want to find $\max(\alpha^x w)$ over all $x$ and $w$ where $0 < \zeta \leq x \leq 1$ and $wx < 1$. Since any feasible solution to the above problem can be changed to increase the maximum by increasing either $w$ or $x$ until $wx = 1$, we can get an upper-bound on the maximum weight by setting $wx = 1$. This transforms the problem to $\max(\alpha^x/x)$ over all $x$ given that $0 < \zeta \leq x \leq 1$.

We can use calculus to show that the maximum must occur at $x = \zeta$. Let $f(x) = \alpha^x/x$. To find the critical points, we solve

$$f'(x) = \alpha^x \ln(\alpha)/x - \alpha^x/x^2 = 0$$

This gives a single critical point at $x_c = 1/\ln(\alpha)$. Since we are assuming that $\alpha < e$ then $x_c = 1/\ln(\alpha) > 1/\ln(e) = 1$. Therefore the maximum must occur at one of the endpoints since the critical point is past the right endpoint of $[\zeta, 1]$. Assume the maximum occurs at $x = 1$. Since $\alpha < e$,

$$f'(1) = \alpha \ln(\alpha) - \alpha < \alpha - \alpha = 0.$$

Therefore the slope of $f(x) = \alpha^x/x$ must be negative at $x = 1$. Since $f(x)$ is continuous, there must be a point $x_a < 1$ such that $f(x_a) > f(1)$ This is a contradiction since we assumed $f(1)$ was the maximum value in the interval. The maximum must occur at $x = \zeta$. This gives an upper-bound on any weight of $\max(\varepsilon, \alpha^\zeta/\zeta)$. ■

The next lemma deals with the effects of the demotions and promotions on a sequence of target concepts. Let $H(C) = \sum_{i=1}^n \left(u_i^T + \sum_{t=1}^{T-1} \max(0, u_i^t - u_i^{t+1})\right)$, and $\delta \leq \min_{t\in\{1,\ldots,T\}} \delta^t$.

**Lemma 7** *If $\alpha < e$ then*

$$\log_\alpha\left(\frac{w_{max}}{\varepsilon}\right) H(C) + N > (1+\delta)|P| - (1-\delta)|D| \;\; .$$

**Proof** Let $z_i^t = \log_\alpha(w_i^t/\varepsilon)$. This is just the amount weight $i$ has been increased by an $\alpha$ factor over the minimum value $\varepsilon$ before the update on trial $t$. Based on this formula, $z_i^1 = 0$. Assume that the last trial is $T$. Because an update could occur on this final trial, the last index of $z$ is $T+1$.

The value of $w_i^t$ is $\varepsilon$ multiplied by $\alpha^{x_i^t}$ on every promotion and effectively multiplied by a number as small as $\alpha^{-x_i^t}$ on every demotion. The multiplier on demotions may be larger since the weight value can not go below $\varepsilon$. For example, if the current weight value is $\varepsilon$, a demotion would not change the value, therefore, in this case, the effective multiplier is 1. Using the definition of $z$, gives $z_i^{t+1} - z_i^t = x_i^t$ on a promotion, $z_i^{t+1} - z_i^t \geq -x_i^t$ on a demotion, and $z_i^{t+1} - z_i^t = 0$ on a trial without an update.

Let $P$ be the set of promotion trials and let $D$ be the set of demotion trials. At this point, we want to weight the change in the $z$ values by $u$, the target weights. This will allow us to relate the $z$ values to the mistakes.

$$\sum_{i=1}^n \sum_{t=1}^T u_i^t (z_i^{t+1} - z_i^t) \geq \sum_{t\in P} \sum_{i=1}^n u_i^t x_i^t - \sum_{t\in D} \sum_{i=1}^n u_i^t x_i^t \;\; .$$

Let $\hat{P} = \{t \mid t \in P \text{ and } \aleph^t > 0\}$ and $\hat{D} = \{t \mid t \in D \text{ and } \aleph^t > 0\}$. These are the noisy promotion and demotion trials. Using this notation, we can break up the summations. The last formula is equal to

$$\sum_{t\in P-\hat{P}} \sum_{i=1}^n u_i^t x_i^t - \sum_{t\in D-\hat{D}} \sum_{i=1}^n u_i^t x_i^t + \sum_{t\in \hat{P}} \sum_{i=1}^n u_i^t x_i^t - \sum_{t\in \hat{D}} \sum_{i=1}^n u_i^t x_i^t \;\; .$$

Since for every non-noisy promotion $\sum_{i=1}^n u_i^t x_i^t \geq (1+\delta)$, and every non-noisy demotion $\sum_{i=1}^n u_i^t x_i^t \leq (1-\delta)$, then the last formula is greater or equal to

$$(1+\delta)|P-\hat{P}| - (1-\delta)|D-\hat{D}| + \sum_{t\in \hat{P}} \sum_{i=1}^n u_i^t x_i^t - \sum_{t\in \hat{D}} \sum_{i=1}^n u_i^t x_i^t$$

$$= (1+\delta)|P| - (1-\delta)|D| - \sum_{t \in \hat{P}} \left( (1+\delta) - \sum_{i=1}^{n} u_i^t x_i^t \right) - \sum_{t \in \hat{D}} \left( (1-\delta) + \sum_{i=1}^{n} u_i^t x_i^t \right) .$$

Using the definitions of noisy promotion and demotion trials along with the definition $N = \sum_{t=1}^{T} \aleph^t$, we can use the previous equations to conclude that

$$\sum_{i=1}^{n} \sum_{t=1}^{T} u_i^t (z_i^{t+1} - z_i^t) \geq (1+\delta)|P| - (1-\delta)|D| - N \ .$$

Based on the definition of $z$, $0 \leq z_i^t < \log_\alpha \left( \frac{w_{max}}{\varepsilon} \right)$. We can use this constraint to compute an upper-bound. Using the fact that $z_i^1 = 0$,

$$(1+\delta)|P| - (1-\delta)|D| - N \leq \max \left( \sum_{i=1}^{n} u_i^1 (z_i^2 - z_i^1) + \cdots + u_i^T (z_i^{T+1} - z_i^T) \right)$$

$$= \max \left( \sum_{i=1}^{n} z_i^2 (u_i^1 - u_i^2) + \cdots + z_i^T (u_i^{T-1} - u_i^T) + z_i^{T+1} (u_i^T) \right)$$

$$< \log_\alpha \left( \frac{w_{max}}{\varepsilon} \right) \sum_{i=1}^{n} \left( u_i^T + \sum_{t=1}^{T-1} \max(0, u_i^t - u_i^{t+1}) \right) \ .$$

Rearranging the terms proves the lemma. ■

In its current form, the above lemma is not very intuitive. However, there is another way to look at the bound. Instead of $h(i) = u_i^T + \sum_{t=1}^{T-1} \max(0, u_i^t - u_i^{t+1})$, one can look at the local maximum and minimum of the sequence of $u_i$. Define $u_i^0 = 0$ and $u_i^{T+1} = 0$. The value $h(i)$ is equivalent to summing the local maximums and subtracting the local minimums for the sequence of target weights. For example, if the sequence of a particular $u_i$ is $(0, .1, .3, .8, .8, .4, .2, .5, .6, 0)$ then there is a maxima of .8 followed by a minima of .2 and a final maxima of .6. This gives $h(i) = .8 - .2 + .6 = 1.2$. This works because the $u_i^t - u_i^{t+1}$ is only positive while the target weights are decreasing in value. The target weights decrease in value from a local maximum, $u_i^a$, to the next local minimum, $u_i^b$. The sum of these differences, as we go from $u_i^a$ to $u_i^b$ is just $u_i^a - u_i^b$.

This suggests how an adversary can maximize the number of mistakes. The adversary should increase a weight as much as possible during the maximum target value and decrease the weight to $\varepsilon$ during the minimum. This requires that the adversary knows the future behavior of the sequence of concepts in order to know if the current concept is a local maximum or a local minimum. For some practical problems, the adversary may not have this knowledge. However, the bounds in this paper assume the adversary knows the future sequence of concepts.

At this point, we want to prove the main theorem. We have attempted to set the parameters and arrange the inequalities to optimize the mistake bound for small $\delta$.

**Theorem 1** *For instances generated by a concept sequence $C$, if $\alpha = 1+\delta$ and $\varepsilon = \frac{\delta}{50\lambda}$ then the number of mistakes is less than*

$$\frac{H(C)(2.05+\delta)\left(\zeta\delta + \ln\left(\frac{\lambda}{\delta\zeta}\right) + 3.92\right)}{\delta^2} + \frac{(2.05+\delta)N}{\delta+\delta^2} \ .$$

**Proof** First we want to show that $w_{max} \leq \alpha^{\zeta}/\zeta$. Based on Theorem 6, we know that $w_{max} \leq \max(\varepsilon, \alpha^{\zeta}/\zeta)$. Therefore, we just need to show that $\varepsilon \leq \alpha^{\zeta}/\zeta$. Using the facts that $\zeta \leq \lambda$, $\delta \leq 1$, and that $\alpha > 1$,

$$\varepsilon = \frac{\delta}{50\lambda} \leq \frac{1}{50\zeta} < \frac{\alpha^{\zeta}}{\zeta} \ .$$

Next we want to substitute Lemma 5 into Lemma 7 to get an upper-bound on $|P|$. The lemma condition that $\alpha < e$ is satisfied since $\alpha = 1+\delta \leq 2 < e$. The condition that $\varepsilon < 1/\lambda$ is satisfied since $\varepsilon\lambda = \delta/50 < 1$. Now we can proceed with the substitution.

$$H(C)\log_{\alpha}\left(\frac{\alpha^{\zeta}}{\varepsilon\zeta}\right) + N > (1+\delta)|P| - (1-\delta)\frac{\alpha}{1-\varepsilon\lambda}|P| \ .$$

Solving for $|P|$ gives,

$$|P| < \frac{H(C)\log_{\alpha}\left(\frac{\alpha^{\zeta}}{\varepsilon\zeta}\right) + N}{(1+\delta) - (1-\delta)\frac{\alpha}{1-\varepsilon\lambda}} \ .$$

To get a mistake bound add $|P|$ to both sides of Lemma 5 and substitute in the previous result to get a bound on the number of mistakes.

$$|P| + |D| < \left(1 + \frac{\alpha}{1-\varepsilon\lambda}\right)|P| < \left(1 + \frac{\alpha}{1-\varepsilon\lambda}\right)\frac{H(C)\log_{\alpha}\left(\frac{\alpha^{\zeta}}{\varepsilon\zeta}\right) + N}{(1+\delta) - (1-\delta)\frac{\alpha}{1-\varepsilon\lambda}}$$

$$= \frac{1-\varepsilon\lambda+\alpha}{(1+\delta)(1-\varepsilon\lambda) - (1-\delta)\alpha}\left(\frac{H(C)(\zeta\ln(\alpha) + \ln(1/\zeta) + \ln(1/\varepsilon))}{\ln(\alpha)} + N\right) \ .$$

Substituting in the values $\alpha = 1+\delta$ and $\varepsilon = \frac{\delta}{50\lambda}$, the preceding equation is equal to

$$\frac{2+.98\delta}{.98\delta(1+\delta)}\left(\frac{H(C)\left(\zeta\ln(1+\delta) + \ln\left(\frac{\lambda}{\delta\zeta}\right) + \ln(50)\right)}{\ln(1+\delta)} + N\right) \ .$$

To make the bound more intuitive, we use the fact that $\delta - \delta^2/2 \leq \ln(1+\delta) \leq \delta$. (One can prove this using the Taylor formula with a fourth term remainder and a third term remainder.) Therefore the above equation is less than or equal to

$$\frac{2+.98\delta}{.98\delta(1+\delta)}\left(\frac{H(C)\left(\zeta\delta + \ln\left(\frac{\lambda}{\delta\zeta}\right) + \ln(50)\right)}{\delta - \delta^2/2} + N\right) \ .$$

$$< \frac{H(C)(2.05+\delta)\left(\zeta\delta + \ln\left(\frac{\lambda}{\delta\zeta}\right) + 3.92\right)}{\delta^2} + \frac{(2.05+\delta)N}{\delta+\delta^2} \ .$$

■

## 4. Analyzing the Mistake-bound

In this section, we are going to analyze the mistake bound. To make this exposition clear we will use an asymptotic form of the bound. The number of mistakes of tracking Winnow is equal to

$$O\left(H(C)\ln\left(\frac{\lambda}{\zeta\delta}\right)/\delta^2+N/\delta\right) \quad .$$

### 4.1 Advantages of the Algorithm

One of the advantages of the concept tracking version of Winnow is the relative insensitivity to the number of irrelevant attributes. This can be seen by looking at $\lambda \geq \max_{t\in\{1,\ldots,T\}} \|X^t\|_1$. Based on this definition we can always set $\lambda$ to the number of attributes. Imagine we have a learning problem with $n_1$ attributes. Let $U$ be the target concept that optimizes the mistake bound. Now assume that we increase the number of attributes by $n_2$. Since the target concept must still be valid, the only effect on the bound is the potential increase in the value of $\lambda$ from $n_1$ to $n_1+n_2$. Since the bound only contains $\lambda$ inside the logarithm function, this, at worst, causes a small increase in the bound. Of course, the bound may not increase at all. If the added attributes help create a better target concept then this new target concept may lower the bound. Also, the new attributes may not increase the value of $\lambda$; they can increase the value of $\lambda$ by anything from 0 to $n_2$.

A related benefit of the proof is that the mistake bound only depends on $\lambda$ as opposed to $n$. Most upper-bounds on mistakes, for algorithms with multiplicative update algorithms, depend on the number of attributes (Littlestone, 1989, Auer and Warmuth, 1998, Herbster and Warmuth, 1998, Grove et al., 2001). For sparse problems, with few active attributes ($x_i > 0$), this could have a large effect on the bound. The tracking version of Winnow gets improved performance on sparse instances because of the appropriate setting of the initial weights that are used by the algorithm. Later, we will show how to use this parameter to extend this benefit to the normal non-tracking version of Winnow.

There is one practical problem with sparse instances. Since Winnow algorithms have only positive weights, a trick is used to allow negative weights. For each attribute $x_i$, a new complemented attribute $1-x_i$ is added. A positive weight on $1-x_i$ can effectively allow a negative weight on $x_i$. However, these extra attributes will force $\lambda = n$. Therefore, the advantages of a small $\lambda$ are only possible if one can assume the target concept only has positive weights and does not need complemented attributes.

It is possible to get similar sparse instance benefits with Winnow type algorithms by using an infinite attribute algorithm (Blum et al., 1991). Infinite attribute algorithms take advantage of the fact that only attributes that are active ($x_i > 0$) during updates effect the algorithm. Therefore the number of attributes that are involved in updates is just the maximum number of attributes active per trial times the mistake bound. Combining this with the logarithmic nature of the Winnow bounds gives a mistake bound that only depends logarithmically on the maximum number of attributes active per trial. However there are certain advantages to the proofs in this paper. First $\|X\|_1$ may be small, yet the number of active attributes may be large. Second when noise is involved in on-line learning, there cannot be a finite mistake bound, and a large number of attributes could eventually be active during mistakes. Since the analysis in this paper does not depend on the total number of active attributes during mistakes, these problems do not occur.

### 4.2 Algorithm Problems and Solutions

A major problem with the proof is that the mistake bound is allowed to grow arbitrarily large when $\zeta = \min\{x_i^t \mid x_i^t > 0\}$ approaches zero. There are several ways to overcome this problem.

We do have an alternative proof that gives a finite mistake bound even when $\zeta$ is allowed to approach 0. The idea of the proof is to bound how large a weight can get even if it is increased on every promotion trial. We can show that the maximum value of a weight is less than $\alpha|P|$, and we can show that if $|P| < a\ln|P| + b$ where $a > e^3$ and $b > 0$ then $|P| < a(\ln a)^2 + b\ln a$. Using these ideas to modify the proof of Theorem 1 gives a bound of

$$O\left[\frac{H(C)}{\delta^2}\left(\ln\left(\frac{H(C)}{\delta}\right)\right)^2 + \frac{H(C)\ln(\lambda/\delta)}{\delta^2}\ln\left(\frac{H(C)}{\delta}\right) + \frac{N}{\delta}\ln\left(\frac{H(C)}{\delta}\right)\right] \ .$$

However, this bound does not seem very tight. A more practical approach that gives a better bound is to increase the value of $\zeta$ by transforming the instances.

**Instance transformation 1**

Let $\hat{X}^t = (\hat{x}_1^t, \ldots, \hat{x}_n^t) \in [0,1]^n$ be the transformed instance.
Let $m > 0$ be the new minimum.
if $x_i^t \in [0, m/2]$
  $\hat{x}_i^t = 0$
else if $x_i^t \in (m/2, m]$
  $\hat{x}_i^t = m$
else
  $\hat{x}_i^t = x_i^t$

We want to show how much these transformed instances change the learning problem. We need some extra notation to explain the results. Let $v = \max_{t\in\{1,\ldots,T\}} \sum_{i=1}^n u_i^t$. This is just the maximum sum of target weights on any single trial. The smaller $v$ the better the bound. This corresponds to the assumption that the concept has few relevant attributes. Even in problems with shifting concepts, while the total number of attributes involved might be large, the number in a single trial might be relatively small.

**Corollary 8** *If $m = \delta/v$ then instance transformation 1 applied to concept sequence $C$ makes a number of mistakes less than or equal to*

$$O\left(H(C)\ln\left(\frac{v\lambda}{\delta}\right)/\delta^2 + N/\delta\right) \ .$$

**Proof** We use the hat notation to refer to a parameter dealing with the transformed instance. The $H(C)$ function is the same since the target function is the same. We have $\hat{\zeta} \geq \delta/v$ because the transformation forces this new minimum value, and $\hat{\lambda} \leq 2\lambda$ since we are at most doubling the value of any single attribute. To compute $\hat{\delta}$, consider the case where the label is 1. A non-noisy instance has $\sum_{i=1}^n u_i^t x_i^t \geq 1 + \delta$.

$$\sum_{i=1}^n u_i^t \hat{x}_i^t \geq \sum_{i=1}^n u_i^t (x_i^t - m/2) \geq \sum_{i=1}^n u_i^t x_i^t - vm/2 \geq 1 + \delta - \delta/2 \ .$$

Therefore the transformed problem has $\hat{\hat{\delta}} = \delta/2$. The same result holds for 0 labels. Noise is handled in a similar way. Again consider the case where the label is 1. The noise is defined as $\aleph^t = \max\left(0, 1 + \delta^t - \sum_{i=1}^n (u_i^t x_i^t)\right)$. Therefore,

$$1 + \hat{\hat{\delta}} - \sum_{i=1}^{n} u_i^t \hat{x}_i^t \leq 1 + \hat{\hat{\delta}} - \sum_{i=1}^{n} u_i^t (x_i^t - m/2) \leq 1 + \hat{\hat{\delta}} - \sum_{i=1}^{n} u_i^t x_i^t + \nu m/2 = 1 + \delta - \sum_{i=1}^{n} u_i^t x_i^t \ .$$

This shows that the noise can only decrease on the transformed instances. A same result holds for 0 labels. Plugging all of these values into Theorem 1 gives the result. ■

We conjecture that the following attribute transformation will give better performance against an adversary. It takes advantage of random numbers to perform the shifting of attributes. Random numbers are a useful strategy against adversaries assuming the adversary does not know the future sequence of random numbers.

**Instance transformation 2**

Let $\hat{X}^t = (\hat{x}_1^t, \ldots, \hat{x}_n^t) \in [0,1]^n$ be the transformed instance.
Let $m > 0$ be the new minimum.

if $x_i^t \in [0, m]$
  With probability $x_i^t/m$ set $\hat{x}_i^t = m$
  Otherwise set $\hat{x}_i^t = 0$
else
  $\hat{x}_i^t = x_i^t$

Any analysis using the above transformation will need to change many of the theorems in this paper into probabilistic statements. The main difficulty with these probabilistic statements is calculating how the attributes are shifted on trials with mistakes. This specific subset of trials could have an effect on the distribution of the attributes shifts.

At this point we are unsure if it necessary to use these attribute transformations for practical problems. A prudent approach is to monitor the weight values of the attributes. If a weight value starts getting large then start shifting that particular attribute value. This should make it harder for an adversary to exploit large weight values and possibly reduce the mistake bounds dealing with attribute shifts.

Another problem with the transformations is the addition of extra algorithm parameters. Besides needing to know good values for $\delta$ and $\lambda$, we need a value for $\nu$. A standard technique to partially get around this problem is to run several copies of the algorithm with different choices for the parameters. These copies would be used as experts in an algorithm such as the Weighted Majority Algorithm (WMA) (Littlestone, 1988, Littlestone and Warmuth, 1994, Cesa-Bianchi et al., 1997). As long we don't run an extremely large number of copies, the number of mistakes made by WMA should be close to the performance of the algorithm with the best parameter values.

## 5. Bounds on Specific Problems

In this section, we will apply the mistake-bound on tracking arbitrary linear-threshold functions to problems that have published bounds and see how the new bound compares to previous results.

### 5.1 Fixed Concept

First we will look at the bound for a fixed concept $C^f$ with no noisy trials. To fix concept $C^f$ assume $\forall t_1 \in \{1,\ldots,T\}$ and $\forall t_2 \in \{1,\ldots,T\}$ that $u_i^{t_1} = u_i^{t_2}$.

**Corollary 9** *When instances are generated from $C^f$, if $\alpha = 1+\delta$ and $\varepsilon = \frac{\delta}{50\lambda}$, then the number of mistakes of tracking Winnow is less than*

$$\frac{(2.05+\delta)\left(\zeta\delta + \ln\left(\frac{\lambda}{\delta\zeta}\right) + 3.92\right)\sum_{i=1}^n u_i^1}{\delta^2} .$$

**Proof** We just use Theorem 1 with the fact that $H(C) = \sum_{i=1}^n u_i^1$ and $N = 0$. ■

Comparing this bound with the general concept tracking bound, it is clear the main difference is the value of the $H(C)$ function. The $H(C)$ function encodes the extra difficulty of tracking a changing concept.

For the standard Winnow algorithm (Littlestone, 1988),[1] when $\alpha = 1+\delta$ and the starting weights are $\varepsilon$, the number of mistakes is less than or equal to

$$\frac{(2+\delta)\left(n\varepsilon + \sum_{i=1}^n u_i^1 \ln u_i^1 + \sum_{i=1}^n u_i^1 \ln(1/\varepsilon) - \sum_{i=1}^n u_i^1\right)}{\delta^2} .$$

While $\varepsilon = \sum_{i=1}^n u_i/n$ optimizes this bound, setting $\varepsilon = 1/n$ gives a good result without requiring the algorithm to know $\sum_{i=1}^n u_i$ in advance. Using $\varepsilon = 1/n$, the number of mistakes is less than or equal to

$$\frac{(2+\delta)\left(1 + \sum_{i=1}^n u_i^1 \ln u_i^1 + \ln n \sum_{i=1}^n u_i^1 - \sum_{i=1}^n u_i^1\right)}{\delta^2} .$$

Looking at just the main terms in the bounds, several differences are evident. The term $\ln(1/\zeta)$ in tracking Winnow is a result of the possible large weight values. There is no corresponding term in normal Winnow bound since the weights of the algorithm must eventually get close to the target weight values. Intuitively, if the weights do not get close then the adversary can make more mistakes exploiting this difference. However, this is not the best strategy for a shifting concept. If the adversary waits until a relevant attribute $x_i$ becomes irrelevant then it can use all the weight in this newly irrelevant attribute to help perform demotions on any currently relevant attributes. The adversary can cause more demotions when $x_i$ becomes irrelevant than when it is relevant. Therefore, the adversary tries to build up a large weight on $x_i$ while it is relevant and then uses this weight for demotions when $x_i$ becomes irrelevant. More precisely, as can be seen in Lemma 6, the adversary should maximize the weight values when $u_i$ reaches a global maximum and minimize the weight value when $u_i$ reaches a global minimum.

Another difference is how the standard Winnow bound depends on $\ln n$. In the target tracking proof, the bound depends on $\ln(\lambda/\delta)$, where $\lambda$ is equal to $\max \|X\|_1$. This difference is largely an effect of the starting weight values used by the algorithms. If we use the same starting weight value as the tracking version of Winnow and use a slightly different representation of the concept, we can give a bound for the normal version of Winnow that performs well on sparse instances. The proof

1. The actual bounds are small refinements of Littlestone (1988); they can be found at the authors webpage *http://paul.rutgers.edu/~mesterha/*.

exploits the fact that any set of concept weights that correctly classifies the instances gives a mistake bound. We will transform the concept weights to a form that does well for sparse instances. This new concept sets several of the target weights to the starting weight value.

**Theorem 10** *When instances are generated from $C^f$ where at most $k$ attributes are relevant, if $\alpha = 1+\delta$ and $\varepsilon = \frac{\delta}{50\lambda}$ then, without loss of generality, we can assume the first $k$ attributes are relevant and that the number of mistakes for the normal Winnow algorithm is less than*

$$\frac{(2.09+1.05\delta)\left(\left(\ln\left(\frac{\lambda}{\delta}\right)+3.92\right)\sum_{i=1}^{k}u_i^1+\sum_{i=1}^{k}u_i^1\ln u_i^1\right)}{\delta^2}\ .$$

**Proof** For every target weight value $u_i^1$ that is less than $\varepsilon$, shift its value to $\varepsilon$. Every other target weight keeps the same value. Let $\hat{u}_i$ be the new target weights and let $k$ be the number of new target weights that do not equal $\varepsilon$.

Since we have increased the target weight values, the value of $\delta$ when the label is 0 may decrease. Because we have increased a relevant weight by at most $\varepsilon$ and $\lambda$ is an upper-bound for the one-norm of an instance then when the label is 0

$$\sum_{i=1}^{n}\hat{u}_i x_i^t \le \sum_{i=1}^{n}u_i^1 x_i^t+\varepsilon\lambda \le 1-\delta+\varepsilon\lambda\ .$$

Let $\hat{\delta} = \delta-\varepsilon\lambda$. Without loss of generality, assume the first $k$ attributes are the relevant attributes. Substituting the new target weight values and $\hat{\delta}$ into the Winnow bound gives

$$\frac{(2+\hat{\delta})\left(n\varepsilon+\sum_{i=1}^{k}u_i^1\ln\left(u_i^1/\varepsilon\right)-\sum_{i=1}^{k}u_i^1+\sum_{i=k+1}^{n}\varepsilon\ln 1-\sum_{i=k+1}^{n}\varepsilon\right)}{\hat{\delta}^2}\ .$$

$$=\frac{(2+\hat{\delta})\left(k\varepsilon+\sum_{i=1}^{k}u_i^1\ln\left(u_i^1/\varepsilon\right)-\sum_{i=1}^{k}u_i^1\right)}{\hat{\delta}^2}\ .$$

$$<\frac{(2.09+1.05\delta)\left(\left(\ln\left(\frac{\lambda}{\delta}\right)+3.92\right)\sum_{i=1}^{k}u_i^1+\sum_{i=1}^{k}u_i^1\ln u_i^1\right)}{\delta^2}\ .$$

■

In asymptotic form, this bound is $O\left(H(C)\ln(\lambda/\delta)/\delta^2\right)$. This bound is identical to the tracking Winnow bound, minus the noise, except that it does not have a $\zeta$ parameter. The similar bound for the two algorithms shows that the key to getting a good bound on sparse instances for Winnow algorithms is to properly set the starting weight of the attributes. However, the tracking version uses this starting weight value for two purposes. One is to set the minimum weight to allow concept tracking, and as a side effect this gives good performance on sparse instances. The normal version of Winnow has more freedom. One should set the starting weight values to whatever gives the minimum number of mistakes. When $50\lambda/\delta < n$ then $\varepsilon = \frac{\delta}{50\lambda}$ is a good choice. However when $50\lambda/\delta > n$ then $\varepsilon = 1/n$ is a better choice.

### 5.2 Tracking Disjunctions

As a second example, we give a bound for shifting disjunctions with boolean attributes. Look at a concept sequence $C^d$ such that each $u_i^j \in \{0,2\}$ and $H(C^d) = 2Z^+$. This corresponds to a sequence where $Z^+$ is the number of disjunction literals that are added to the concept either at the start of the algorithm or during the trials.

**Corollary 11** *When instances $X \in \{0,1\}^n$ are generated from concept $C^d$ with noise $N$, if $\alpha = 2$ and $\varepsilon = \frac{2}{50\lambda}$, then the number of mistakes is less than*

$$6.1Z^+(\ln\lambda + 4.92) + 1.53N \ .$$

**Proof** Concept sequence $C^d$ has $H(C^d) = 2Z^+$, $\zeta = 1$ and $\delta = 1$. Substituting these values into Theorem 1 gives the result. ■

To compare this with the bound given by Auer and Warmuth (1998), let $Z^-$ equal to the number of disjunction literals that are removed from the concept and let $Z = Z^+ + Z^-$. For the deterministic disjunction tracking algorithm given in Theorem 4 of Auer and Warmuth (1998) the number of mistakes is less than or equal to

$$4.32Z(\ln n + 3.89) + 4.32\min(n,Z)(\ln n + 1.34) + 0.232 + 1.2N \ .$$

The bounds we give are similar and have a distinct advantage for sparse instances, but our bounds can be slightly larger for certain problems, however, our bounds have been optimized for small $\delta$. Looking over Theorem 1, we can remove a few of the approximations in the proof and change the parameter choices. Letting $\alpha = 1.35$ and $\varepsilon = \frac{1}{35\lambda}$, we can derive a mistake bound of

$$3.98Z^+(\ln\lambda + 3.86) + 1.2N \ .$$

The lower $\alpha$ value is used to improve the performance of the algorithm on noise. In general, there is a trade-off with the $\alpha$ parameter. There is a choice for $\alpha$ that will optimize the first portion of the bound that does not deal with noise, but a smaller $\alpha$ value lowers the constant on the noise term.

We can also give lower bounds for learning disjunctions by slightly modifying the results of Auer and Warmuth (1998) to handle sparse instances. In Auer and Warmuth (1998), they prove that any deterministic learning algorithm must generate at least

$$\lfloor (Z+1)/2 \rfloor \lfloor \log_2(n) \rfloor + N$$

mistakes in the worst case when $n \geq 2$. It is not difficult to generalize that proof to handle sparse binary instances. Assuming $\lambda$ is the maximum number of attributes active in a binary instance, we can derive a lower bound of

$$\lfloor (Z+1)/2 \rfloor \lfloor \log_2(\lambda) \rfloor + N \ .$$

This shows that our bound for disjunctions with sparse instances is within a constant of the lower bound. We can set $\alpha = 2.94$ to get a bound where the constants on the dominate terms are within a factor of 2 of the lower bound when $Z^+ = Z^-$.

### 5.3 Tracking Linear-threshold Functions

To get a better feel for the strength of the concept tracking Winnow bound, we will compare it the standard Winnow bound on a modified on-line tracking problem. Assume the environment gives additional information to the learning algorithm by revealing when the concept is going to change. A straightforward use of the standard Winnow algorithm is to reset the weights on every concept change. The new worst case bound is just the sum of the previous bounds. Assuming the same $\delta$ for the various concepts, the new bound is primarily the same as the old except the target weight sums include the target weights from all concepts.

Now consider the target tracking version of Winnow without the extra information about concept shifts. If each concept change shares no relevant attributes from the previous concept the $H(C)$ function is just the sum of the target weights from all the concepts. If consecutive concepts share relevant attributes then the bound will be lower. Therefore, the concept tracking version of Winnow has the potential to do better even without using any advanced knowledge of concept shifts.

A related concept tracking algorithm is the Approximate Large Margin algorithm also known as ALMA. The ALMA algorithm was presented by Gentile (2001), and mistake bounds for concept tracking for a simplified form of ALMA are given in Kivinen et al. (2002). ALMA is a $p$-norm algorithm (Grove et al., 2001) which changes behavior when the value of the parameter $p$ is changed. The algorithm has Perceptron like bounds when $p = 2$ and Winnow like bounds when $p = \log n$. Let $v = \max_{t\in\{1,\ldots,T\}} \sum_{i=1}^{n} u_i^t$. The bound for ALMA when $p = \log n$ is

$$O\left(\frac{vH(C)\ln n}{\delta^2} + \frac{N}{\delta}\right) \ .$$

When $\delta$ and $v$ are small, this compares favorably to the Winnow bound in Corollary 8 of

$$O\left(\frac{H(C)\ln(v\lambda/\delta)}{\delta^2} + \frac{N}{\delta}\right) \ .$$

However, when $\delta$ is small both bounds may become too large to be useful because of the common $1/\delta^2$ factor. Let $X_{max} = \max_t ||X^t||_2$, $U_{max} = \max_t ||U^t||_2$, and $H_2(C) = \sum_{t=1}^{T} ||U^t - U^{t+1}||_2$. The bound for ALMA when $p = 2$ is

$$O\left(\frac{X_{max}^2 U_{max} H_2(C)}{\delta^2} + \frac{N}{\delta}\right) \ .$$

Comparing against the Winnow bound, this bound might be better for sparse instances especially when the target concept has negative weights. This is because $\lambda$ needs to be set to $n$ when complemented attributes are used. For these types of problems the $X_{max}^2$ term in ALMA might be smaller than the $\ln(vn/\delta)$ term in Winnow. It may also do better for large concept shifts since the $H_2(C)$ function involves the 2-norm instead of the 1-norm. These advantage may be offset by the extra term $U_{max}$ if the maximum concept size is large.

Of course, it remains to be seen how tight the various bounds are and how the algorithms perform in practice, particularly since most real world problems do not involve adversaries. Also, parameter selection is an important problem, and it remains to be seen how robust the various algorithms are when given suboptimal parameters. As we mentioned before, expert algorithms such as WMA can be used to help pick good parameter values, but these techniques can have problems when the parameters are not robust.

## 6. Conclusions

In this paper, we give a proof for the concept tracking version of Winnow that shows that the bounds for learning shifting linear-threshold functions have many of the same advantages that the traditional Winnow algorithm has on fixed concepts. These benefits include a weak dependence on the number of irrelevant attributes, inexpensive runtime, and robust behavior against noise. We also show how the performance of this algorithm does not depend on the number of attributes and instead depends on $\max \ln(\|X\|_1)$. This is similar to the infinite attribute model but has advantages when dealing with real world constraints such as noise.

To get Winnow to track concepts, we use the standard technique of setting a minimum value for the weights. However, given our proof technique, this gives a mistake bound that is allowed to grow arbitrarily large when $\zeta = \min\{x_i^t \mid x_i^t > 0\}$ approaches zero. One solution is to transform the instances such that $\zeta$ is not allowed to get arbitrarily small. By making the appropriate assumptions, it is possible to allow the effects of these small shifts in the value of attributes to be characterized in the $\delta$ parameter of the learning problem. These transformations allow us to improve the worst-case mistake bounds. At this point, we are unsure if these transformations are needed in practice, and if needed, which are the best techniques. We plan to do experiments in the future.

Another area to explore is the development of lower bounds for the tracking version of Winnow when learning certain types of shifting concepts, such as concepts where all the $\delta^t$ are equal. Cases where the $\delta^t$ change for the various concepts will most likely be more difficult, but at a minimum, tighter upper-bounds can be made for these types of problems. Another variation to take into account is the number of trials an adversary has between changes in the concept. Again this will complicate the bounds, but we believe there should be a concise way to express the results.

Recently there has been a lot of work on tracking linear-threshold functions. In this paper, we compare the algorithm to other algorithms and show that our bound compares favorably for certain types of problems. In particular, we compare our bound to the best existing Winnow bounds for fixed concepts and shifting disjunctions, and we compare to tracking bounds for ALMA (Kivinen et al., 2002).

## Acknowledgments

We would like to thank Haym Hirsh, Sofus Macskassy, and the anonymous reviewers for reading this paper and providing valuable comments and corrections. We would also like to thank Nick Littlestone for stimulating much of this work.

## References

P. Auer and M. K. Warmuth. Tracking the best disjunction. *Machine Learning*, 32(2):127–150, 1998.

A. Blum and C. Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, 2000.

A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. In *Proceedings of the Third Annual Conference on Computational Learning Theory*, pages 157–166, 1991.

N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the Association for Computing Machinery*, 44(3):427–485, 1997.

C. Gentile. A new approximate maximal margin classification algorithm. *Machine Learning*, 2: 213–242, 2001.

C. Gentile and M. K. Warmuth. Linear hinge loss and average margin. In *Neural Information Processing Systems 11*, pages 225–231. MIT Press, 1999.

A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(2):173–210, 2001.

D. P. Helmbold, D. D. Long, T. L. Sconyers, and B. Sherrod. Adaptive disk spin-down for mobile computers. *Mobile Networks and Applications*, 5(4):285–297, 2000.

D. P. Helmbold and P. M. Long. Tracking drifting concepts using random examples. In *Proceedings of the Third Annual Conference on Computational Learning Theory*, pages 13–23, 1991.

M. Herbster. Learning additive models online with fast evaluating kernels. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 444–460. Springer, 2001.

M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Machine Learning*, 1:281–309, 2001.

J. Kivinen, A. J. Smola, and R. C. Williamson. Large margin classification for moving targets. In Nicolò Cesa-Bianchi, Masayuki Numao, and Rüdiger Reischuk, editors, *Algorithmic Learning Theory, 13th International Conference*, volume 2533 of *Lecture Notes in Computer Science*, pages 113–127. Springer, 2002. ISBN 3-540-00170-0.

A. Kuh, T. Petsche, and R. L. Rivest. Learning time-varying concepts. In *Neural Information Processing Systems 3*, pages 183–189. Morgan Kaufmann Publishers, Inc., 1991.

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

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

N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. In *Proceedings of the Third Annual Conference on Computational Learning Theory*, pages 147–156, 1991.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

M. L. Minsky and S. A. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, DC, 1962.