Tracking Linear-threshold Concepts with Winnow

Chris Mesterharm

Rutgers Computer Science Department
110 Frelinghuysen Road
Piscataway, NJ 08854
mesterha@paul.rutgers.edu

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 standard 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 the tracking version of 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 traditional algorithm, the bound depends on $\ln n$. In this paper, the shifting concept bound depends approximately on $\max \ln \left(\|X\|_1 \right)$.

1 Introduction

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

The mistake bound applies to the on-line model of learning [1]. 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[2, 3] 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 versus noise[4].

The only change in the tracking version of Winnow is that the attribute weights are not allowed to go below a given constant, $\epsilon > 0$. Anytime a normal Winnow update attempts to lower a weight below ϵ, the weight is set to ϵ. 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 ϵ to the new value. Therefore the algorithm is able to track moving concepts while still keeping the advantages of the traditional Winnow algorithm.

This weight minimum modification has been used with various Winnow type algorithms. In particular, it has been used to give mistake-bounds for learning changing experts [5, 6] and drifting disjunctions [7]. This paper builds on those results by giving a mistake-bound for arbitrary linear-threshold functions that are allowed to shift. These types of algorithms have been shown to be useful in practice, learning shifting concepts such as predicting disk idle times for mobile applications [8] and solving load balancing problems on a network of computers [9]. 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.

Another contribution of this paper is to show that the 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, the algorithm depends approximately on the maximum value over all trials of $\ln(\|X^t\|_1)$.[2] Therefore the algorithm performs well with sparse instances in ways that are similar to infinite attribute algorithms[10].

Other related work includes [11] which gives a technique to convert a wide range of algorithms that learn fixed linear functions to shifting problems. While learning linear functions is similar to learning linear-threshold functions, at this point, these results have only been extended to cover disjunctions and not arbitrary linear-threshold functions.

The remainder of the paper is organized as follows. In Sect. 2, we give a formal statement of the concept tracking problem and the tracking Winnow algorithm. We then present the mistake bound. In Sect. 3, we give a proof for this mistake bound. In Sect. 4, we compare the mistake bound to previous bounds on certain types of linear-threshold problems. Section 5 contains the conclusion and ideas for future work.

[2] These benefits can most likely be extended to the normal, fixed concept version of Winnow by appropriately setting the parameters.

2 Problem Statement and Solution

In this section, we review the concept tracking Winnow algorithm and 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[5, 7]. The only change in the algorithm, in this paper, is that we force the initial weights to be set to the value of the minimum weight, ϵ.

Initialization

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

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)
        $\forall i \in \{1, \ldots, n\}$ $w_i^{t+1} = \max(\epsilon, \alpha^{-x_i^t} w_i^t)$
$t \leftarrow t + 1$

2.2 Concept

The model of concept on-line tracking we present in this paper is similar to a model defined in [12]. The central element of the mistake bound is a sequence of concepts. Let $C = (C_1, \ldots, C_k)$ be a sequence of concepts where $k \in N$.

An adversary generates the instances and is allowed two operations. The goal of the algorithm is to minimize the number of mistakes for any possible sequence of operations.

Adversary operations

1. The adversary can generate instance (X^t, y^t) if the noise predicate Ω is true.
2. The adversary can step forward in the concept sequence.

The predicate Ω is necessary to limit the amount of noisy instances the adversary can generate. If the adversary was allowed to generate an arbitrary number of noisy instances, the concept would be impossible to learn. For example, we could restrict the adversary so that it is only allowed to generate a fixed amount of noise over all the trials. In this case, $\Omega = \left(\sum_{j=1}^{t} \aleph^j \leq K\right)$ where $K \in R$ and $\aleph^j$ is a measure of the noise in trial j. This is similar to the noise model in [1], yet by changing the Ω predicate, it should be possible to extended the results to a more complex noise model such as that found in [13].

We define concept C_j by specifying the weights, $\mathbf{u^j}$, of a linear-threshold function and a separation parameter δ^j. Notice that are using a slightly different notation than that used for the algorithm weights. For the algorithm weights the superscript refers to the trial number; for the target weights the superscript refers to the concept number. The separation parameter specifies a distance from the concept 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 [4, 13, 14]. All other noisy instances come from an instance that is misclassified by the current concept C_j.

Concept C_j parameters

$u_1^j, \ldots, u_n^j \geq 0$ where $u_i^j \in R$
$0 < \delta^j \leq 1$ where $\delta^j \in R$

An instance with $\aleph^t = 0$ corresponds an instance of the current 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.

Concept C_j noise

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

2.3 Mistake-bound

The mistake bound uses the following notation. This notation will be explained and motivated in the proof section.

Terms used in the main mistake bound

Let $\lambda \geq \max\left(6.5\delta, \max_{t \in N} \|X^t\|_1\right)$.
Let $\zeta = \min_{t \in N,\ i \in \{1,\ldots,n\}}\{x_i^t \mid x_i^t > 0\}$.
Let $H(C) = \sum_{i=1}^{n}(u_i^k + \sum_{j=1}^{k-1} \max(0, u_i^j - u_i^{j+1}))$.
Let $\delta \leq \min_{j \in \{1,\ldots,k\}} \delta^j$.

Theorem 1. *For instances generated by a concept sequence C with $\sum_{t\in N} \aleph^t$ noise, if $\alpha = 1+\delta$ and $\epsilon = \frac{\delta}{\lambda \ln(\lambda/\delta)}$ then the number of mistakes is less than*

$$\frac{H(C)(2+\delta)\left(\zeta\delta + \ln\left(\frac{1}{\zeta}\right) + \ln\left(\frac{\lambda}{\delta}\right) + \ln\ln\left(\frac{\lambda}{\delta}\right)\right)}{\delta^2\left(1-\frac{1}{\ln(\lambda/\delta)}\right)} + \frac{(2+\delta)\sum_{t\in N}\aleph^t}{(\delta+\delta^2)\left(1-\frac{1}{\ln(\lambda/\delta)}\right)} \quad .$$

Using the fact that $\lambda \geq 6.5\delta$ and $0 < \zeta, \delta \leq 1$, this bound is an element of

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

3 Proof of Mistake-bound

The proof in this section closely follows the style of proof given in [4]. While most modern Winnow type proofs use a potential function to bound the number of mistakes[1, 13, 14, 7], the potential function style of proof has been difficult to convert to tracking arbitrary linear-threshold concepts. While the potential function proof has benefits,[3] we have found it fruitful to go back to the old style of proof to deal with tracking linear-threshold functions. It is a topic for future research to understand how this type of proof relates to the potential function techniques.

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

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

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.) to show that,

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

[3] It is useful for generalization[14] and has slightly better bounds for fixed concepts.

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 ϵ, 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 $w_i \alpha^{-x_i^t} < \epsilon$. These are the indexes of weights that are forced to ϵ since $w_i^{t+1} = \max(\epsilon, \alpha^{-x_i^t} w_i^t)$. 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 $w_i^t \alpha^{-x_i^t} \geq \epsilon$. All the other weights do not change.

Lemma 2. *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 $\alpha^{-x_i^t} \leq \left(\frac{1}{\alpha} - 1 \right) x + 1$ 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 $\lambda \geq \max(6.5\delta, \max_{t \in N} \sum_{i=1}^{n} x_i^t)$. (The reason for this condition will become clear latter in the proof.) Also this lemma assumes that $\epsilon \leq 1/\lambda$. 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 some linear-threshold concepts.

Lemma 3. *On a demotion, if $\epsilon < 1/\lambda$ then $\triangle\left(\sum_{i=1}^{n} w_i^t\right) < \frac{1-\alpha}{\alpha}(1 - \epsilon\lambda)$.*

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 2 to derive that

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

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 ϵ.

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

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

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

Next, since all the weights with an index in A get demoted to ϵ, 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.

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

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

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,

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

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

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

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

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

Proof. We know $\sum_{i=1}^{n} w_i^t$ can never go below ϵn, and we know $\sum_{i=1}^{n} w_i^t$ has a value of ϵn at the beginning of the algorithm. Since the weights can only change during demotions and promotions.

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

Using the upper-bounds from lemma 1 and lemma 3,

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

Rearranging this inequality proves the lemma. □

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. To do this, we will use the parameter $\zeta = \min_{t \in N,\ i \in \{1,\cdots,n\}}\{x_i^t \mid x_i^t > 0\}$.

Lemma 5. *if $\alpha < e$ then $\forall i \in \{1, \ldots, n\}$ $\forall j \in N \quad w_i^j < \frac{\alpha^\zeta}{\zeta}$.*

Proof. If a promotion never occurs the maximum weight is ϵ. Otherwise, we need to look at weights that are increased by a promotion, because a weight must reach its maximum value after a promotion. The only time a weight, w_a^j can increase is after a promotion where the attribute $x_a^j > 0$. Promotions can only occur if $\sum_{i=1}^{n} w_i^j x_i^j < 1$. Therefore for any attribute $x_a^j > 0$,

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

Since the new weight is $\alpha^{x_a^j} w_a^j$, 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 $0 < \zeta \leq x \leq 1$. This can be solved with calculus to show that the maximum must occur at one of the ends of the interval. This gives an upper-bound on any weight of $\max(\epsilon, \alpha, \alpha^\zeta/\zeta)$. Using the assumption that $\epsilon < 1$ and that $\alpha > 1$ shows that $\max(\epsilon, \alpha, \alpha^\zeta/\zeta) = \max(\alpha, \alpha^\zeta/\zeta)$.

To show that $\max(\alpha, \alpha^\zeta/\zeta) = \alpha^\zeta/\zeta$, we need the assumption that $\alpha < e$. Let $f(x) = \alpha x$ and $g(x) = \alpha^x$. Taking derivatives, $f'(x) = \alpha$ and $g'(x) = \ln(\alpha)\,\alpha^x$. Therefore when $x \in (0, 1]$ and $1 < \alpha < e$

$$g'(x) = \ln(\alpha)\,\alpha^x < \alpha = f'(x) \ .$$

Since the slope of $g(x)$ is always less than the slope of $f(x)$ in the $(0, 1]$ interval, the functions can intersect at most once in the interval. (One can prove this with the mean value theorem.) The point they intersect is at $x = 1$. Since $g(0) > f(0)$, it is straightforward to show that for $x \in (0, 1]$ and $1 < \alpha < e$ that $g(x) \geq f(x)$, and therefore $\alpha^x/x \geq \alpha$. This shows that $\max(\alpha, \alpha^\zeta/\zeta) = \alpha^\zeta/\zeta$ when $\alpha < e$. □

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

Lemma 6. *If $\alpha < e$ then*

$$\log_\alpha \left(\frac{\alpha^\zeta}{\epsilon\zeta} \right) H(C) + \sum_{t \in N} \aleph^t > (1+\delta)|P| - (1-\delta)|D| \ .$$

Proof. First, we define $s(j) \in N$ as the first trial that has an instance generated from concept j. Let $s(k+1)$ be the final trial in the sequence of instances[4]. The adversary will determine the values of the $s(j)$.

Let $z_i^j = \log_\alpha \left(w_i^{s(j+1)}/\epsilon \right)$ and $z_i^0 = 0$. This is just the amount weight i has been increased by an α factor over the minimum value ϵ after the last trial of concept j. The value of $w_i^{s(j)}$ is ϵ 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. Let $P_j = \{t \mid \text{promotion mistake on trial } t \text{ during concept } j\}$. Let $D_j = \{t \mid \text{demotion mistake on trial } t \text{ during concept } j\}$.

$$w_i^{s(j)} \geq \epsilon \exp\left[\ln(\alpha) \left(\sum_{t \in P_1 \cup \cdots \cup P_j} x_i^t \quad - \sum_{t \in D_1 \cup \cdots \cup D_j} x_i^t \right) \right] \ .$$

[4] The analysis still works for a potentially infinite number of trials, since the proof does not depend on when this final trial occurs.

Using the definition of z, this gives

$$z_i^j \geq \left(\sum_{t \in P_1 \cup \cdots \cup P_j} x_i^t \quad - \sum_{t \in D_1 \cup \cdots \cup D_j} x_i^t \right) \ .$$

The value $z_i^j - z_i^{j-1}$ is the change in z_i during concept j. Again on every promotion, we add x_i^t, and on every demotion, we subtract at most x_i^t. Therefore,

$$z_i^j - z_i^{j-1} \geq \left(\sum_{t \in P_j} x_i^t - \sum_{t \in D_j} x_i^t \right) \ .$$

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 on non-noisy instances.

$$\sum_{j=1}^{k} \sum_{i=1}^{n} u_i^j (z_i^j - z_i^{j-1}) \geq \sum_{j=1}^{k} \sum_{i=1}^{n} u_i^j \left(\sum_{t \in P_j} x_i^t - \sum_{t \in D_j} x_i^t \right) \ .$$

Let $c(t) = j$ where j is the concept that is in effect during trial t. The preceding formula is equal to

$$\sum_{t \in P} \sum_{i=1}^{n} u_i^{c(t)} x_i^t - \sum_{t \in D} \sum_{i=1}^{n} u_i^{c(t)} x_i^t \ .$$

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 preceding formula is equal to

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

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

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

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

Using the definitions of noisy promotion and demotion trials, we can use the previous equations to conclude that

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

Based on lemma 5, when $\alpha < e$ the maximum value of any weight is less than α^ζ/ζ, therefore, $0 \leq z_i^j < \log_\alpha\left(\frac{\alpha^\zeta}{\epsilon\zeta}\right)$. We can use this constraint to compute an upper bound. Therefore,

$$(1+\delta)|P| - (1-\delta)|D| - \sum_{t\in N} \aleph^t \leq \max\left(\sum_{i=1}^{n} u_i^1(z_i^1 - z_i^0) + \cdots + u_i^k(z_i^k - z_i^{k-1})\right)$$

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

$$< \log_\alpha\left(\frac{\alpha^\zeta}{\epsilon\zeta}\right) \sum_{i=1}^{n}\left(u_i^k + \sum_{j=1}^{k-1} \max(0, u_i^j - u_i^{j+1})\right) \quad .$$

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 $h(i) = u_i^k + \sum_{j=1}^{k-1} \max(0, u_i^j - u_i^{j+1})$, one can look at the local maxima and minima of the sequence of u_i. The value $h(i)$ is equivalent to summing the local maximums and subtracting the local minimums. For example, if the sequence of a particular u_i is $(.1, .3, .5, .5, .2, .1, .4, .2)$ then $h(i) = .5 - .1 + .4 = .8$.

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

Theorem 1. *For instances generated by a concept sequence C with $\sum_{t\in N} \aleph^t$ noise, if $\alpha = 1+\delta$ and $\epsilon = \frac{\delta}{\lambda \ln(\lambda/\delta)}$ then the number of mistakes is less than*

$$\frac{H(C)(2+\delta)\left(\zeta\delta + \ln\left(\frac{1}{\zeta}\right) + \ln\left(\frac{\lambda}{\delta}\right) + \ln\ln\left(\frac{\lambda}{\delta}\right)\right)}{\delta^2\left(1 - \frac{1}{\ln(\lambda/\delta)}\right)} + \frac{(2+\delta)\sum_{t\in N}\aleph^t}{(\delta+\delta^2)\left(1 - \frac{1}{\ln(\lambda/\delta)}\right)} \quad .$$

Proof. First we want to show that the algorithm condition $\epsilon < 1$ is satisfied. Using the fact that $\lambda \geq 6.5\delta$, (This is part of the reason we define $\lambda \geq \max\left(6.5\delta, \max_{t\in N} \|X^t\|_1\right)$. Another reason is to minimize the mistake-bound.)

$$\epsilon = \frac{\delta}{\lambda \ln(\lambda/\delta)} \leq \frac{\delta}{6.5\delta \ln(6.5)} = \frac{1}{6.5 \ln(6.5)} < 1 \quad .$$

Next, we want to substitute lemma 4 into lemma 6 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 $\epsilon < 1/\lambda$ is satisfied since

$$\epsilon\lambda = \frac{\delta}{\ln(\lambda/\delta)} \leq \frac{\delta}{\ln(6.5)} < 1 \quad .$$

Now we can proceed with the substitution.

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

Solving for $|P|$ gives,

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

Now we will add $|P|$ to both sides of lemma 4 and substitute in the previous result to get a bound on the number of mistakes.

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

$$=\frac{1-\epsilon\lambda+\alpha}{(1+\delta)(1-\epsilon\lambda)-(1-\delta)\alpha}\left(\frac{H(C)(\zeta\ln(\alpha)+\ln(1/\zeta)+\ln(1/\epsilon))}{\ln(\alpha)}+\sum_{t\in N}\aleph^t\right).$$

Substituting in the values $\alpha=1+\delta$ and $\epsilon=\frac{\delta}{\lambda\ln(\lambda/\delta)}$, the preceding equation is equal to

$$\frac{2+\left(1-\frac{1}{\ln(\lambda/\delta)}\right)\delta}{(\delta+\delta^2)\left(1-\frac{1}{\ln(\lambda/\delta)}\right)}\left(\frac{H(C)\left(\zeta\ln(1+\delta)+\ln\left(\frac{1}{\zeta}\right)+\ln\left(\frac{\lambda}{\delta}\right)+\ln\ln\left(\frac{\lambda}{\delta}\right)\right)}{\ln(1+\delta)}+\sum_{t\in N}\aleph^t\right).$$

To make the bound more intuitive use the fact that $\delta-\delta^2/2\le\ln(1+\delta)\le\delta$. (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

$$\frac{2+\left(1-\frac{1}{\ln(\lambda/\delta)}\right)\delta}{(\delta+\delta^2)\left(1-\frac{1}{\ln(\lambda/\delta)}\right)}\left(\frac{H(C)\left(\zeta\delta+\ln\left(\frac{1}{\zeta}\right)+\ln\left(\frac{\lambda}{\delta}\right)+\ln\ln\left(\frac{\lambda}{\delta}\right)\right)}{\delta-\delta^2/2}+\sum_{t\in N}\aleph^t\right)$$

$$<\frac{H(C)(2+\delta)\left(\zeta\delta+\ln\left(\frac{1}{\zeta}\right)+\ln\left(\frac{\lambda}{\delta}\right)+\ln\ln\left(\frac{\lambda}{\delta}\right)\right)}{\delta^2\left(1-\frac{1}{\ln(\lambda/\delta)}\right)}+\frac{(2+\delta)\sum_{t\in N}\aleph^t}{(\delta+\delta^2)\left(1-\frac{1}{\ln(\lambda/\delta)}\right)} \ .$$

□

4 Bounds on specific problems

In this section of the paper, 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 the previous results.

4.1 Fixed concept

First we will look at the bound for a fixed concept with no noisy trials. In this case, $C = (C_1)$.

Corollary 1. *When instances are generated from concept C_1, if $\alpha = 1 + \delta$ and $\epsilon = \frac{\delta}{\lambda \ln(\lambda/\delta)}$, then the number of mistakes is less than*

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

Proof. We just use theorem 1 with the fact that $H(C_1) = \sum_{i=1}^{n} u_i^1$ and $\aleph = 0$. □

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 normal Winnow algorithm[15], when $\alpha = 1 + \delta$ and the threshold parameter is set to n, the number of attributes in an instance, the number of mistakes is less than or equal to

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

While the two bounds are similar, notice how the preceding Winnow bound depends on $\ln n$. In the target tracking proof, the bound depends on $\ln(\lambda/\delta)$, where λ is roughly equal to $\max \|X\|_1$.

It is possible to get similar benefits with Winnow type algorithms by using an infinite attribute algorithm[10]. 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 the algorithm 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 result that only depends logarithmically on the maximum number of attributes active per trial. However there are certain advantages to the algorithm 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 for the concept tracking version of Winnow in this paper.

4.2 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

$\sum_{i=1}^{n} \sum_{i=1}^{k} u_i^j = 2Z^+$. 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 2. *When instances $X \in \{0,1\}^n$ are generated from concept C^d with noise $\sum_{t \in N} \aleph^t$, if $\alpha = 2$ and $\epsilon = \frac{1}{\lambda \ln \lambda}$, then the number of mistakes is less than*

$$\frac{6Z^+ (1 + \ln(\lambda) + \ln\ln(\lambda))}{\left(1 - \frac{1}{\ln(\lambda)}\right)} + \frac{3\sum_{t \in N} \aleph^t}{2\left(1 - \frac{1}{\ln(\lambda)}\right)} \ .$$

Proof. Concept sequence C^d has $H(C^d) = 2Z^+$ and $\delta = 1$. Substituting these values into theorem 1 gives the result. □

To compare this with the bound given in [7], 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 [7] the number of mistakes is less than or equal to

$$4.32Z(\ln n + 3.89) + 4.32 \min(n, Z)(\ln n + 1.34) + 0.232 + 1.2 \sum_{t \in N} \aleph^t \ .$$

Using the fact that for disjunctions $6.5 \leq \lambda \leq n$, these bounds show that the more general results in this paper compare favorably to the previous bound that deals with the limited case of tracking shifting disjunctions.

5 Conclusions

In this paper, we give a proof for the concept tracking version of Winnow that shows it still has good bounds when tracking arbitrary linear-threshold functions. In particular, we compared the new bounds to the best existing Winnow bounds for fixed concepts and shifting disjunctions, showing that the bounds compare favorably. We also show how the performance of this algorithm does not depend on the number of attributes and instead depends approximately on $\max \|X\|_1$. This is similar to the infinite attribute model but has advantages when dealing with real world constraints such as noise.

One problem with the techniques in this paper is that the mistake bound 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 ζ is not allowed to get arbitrarily small. By making the appropriate assumptions, it is possible to allow the effects of small shifts in the value of attributes to be characterized in the δ and $\sum_{t \in N} \aleph^t$ parameters of the learning problem. However this is not completely satisfactory as the issue with ζ is partially an artifact of the proof technique. An interesting question for future research is to eliminate or mitigate the effect of ζ on the bound.

Additional future work is to extend these results to other Winnow algorithms such as the normalized version of Winnow and Balanced Winnow[1]. Normalized Winnow has various advantages such as simple techniques to generalize

from binary prediction to multi-class prediction[16]. This will allow the efficient tracking of multi-class linear-threshold concepts. 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 δ_j are equal. Cases where the δ_j 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.

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 Mark Herbster for explaining how the current work on tracking linear functions relates to tracking linear-threshold functions.

References

[1] Littlestone, N.: Mistake bounds and linear-threshold learning algorithms. PhD thesis, University of California, Santa Cruz (1989) Technical Report UCSC-CRL-89-11.
[2] Rosenblatt, F.: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington, DC (1962)
[3] Minsky, M.L., Papert, S.A.: Perceptrons. MIT Press, Cambridge, MA (1969)
[4] Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning **2** (1988) 285–318
[5] Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Information and Computation **108** (1994) 212–261
[6] Herbster, M., Warmuth, M.K.: Tracking the best expert. Machine Learning **32** (1998) 151–178
[7] Auer, P., Warmuth, M.K.: Tracking the best disjunction. Machine Learning **32** (1998) 127–150
[8] Helmbold, D.P., Long, D.D., Sconyers, T.L., Sherrod, B.: Adaptive disk spin-down for mobile computers. Mobile Networks and Applications **5** (2000) 285–297
[9] Blum, A., Burch, C.: On-line learning and the metrical task system problem. Machine Learning **39** (2000) 35–58
[10] Blum, A., Hellerstein, L., Littlestone, N.: Learning in the presence of finitely or infinitely many irrelevant attributes. In: COLT-91. (1991) 157–166
[11] Herbster, M., Warmuth, M.K.: Tracking the best linear predictor. Journal of Machine Learning Research **1** (2001) 281–309
[12] Kuh, A., Petsche, T., Rivest, R.L.: Learning time-varying concepts. In: NIPS-3, Morgan Kaufmann Publishers, Inc. (1991) 183–189
[13] Littlestone, N.: Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. In: COLT-91. (1991) 147–156
[14] Grove, A.J., Littlestone, N., Schuurmans, D.: General convergence results for linear discriminant updates. In: COLT-97. (1997) 171–183
[15] Littlestone, N.: (1998) Unpublished research that generalizes Winnow algorithm.
[16] Mesterharm, C.: A multi-class linear learning algorithm related to winnow. In: NIPS-12, MIT Press (2000) 519–525