**CS 4540/CS 8803: Algorithmic Theory of Intelligence** Fall 2023

# Lecture 2: Winnow and Weighted Majority

*Instructor: Santosh Vempala* *Lecture date: 8/28,8/30*

We recall the Perceptron algorithm discussed in the last lecture. For normal vector $\|w^*\|_2 \leq 1$, samples $\|x\|_2 \leq 1$, and margin $\gamma = \min_x |\langle w^*, x\rangle|$, we showed that the number of mistakes by the Perceptron Algorithm is at most $1/\gamma^2$.

More generally, the bound we have is

$$\text{\# of mistakes } \leq \frac{\|w^*\|_2^2 \max_x \|x\|_2^2}{\gamma^2}.$$

Viewing $x = (x_1, \cdots, x_n)$ as a feature vector. $\|x\|_2^2$ can grow with the dimension. In many natural settings the unknown hypothesis vector could be sparse, e.g., only $k$ out of $n$ features are relevant. A natural question is whether we can improve the bound in this setting. In this lecture, we will discuss another algorithm that works well for sparse concepts.

## 1 Winnow Algorithm

### 1.1 Learn disjunction of $r$ relevant variables

Consider labeled data $(x, y)$, where $x = (x_1, \cdots, x_n) \in \{0,1\}^n$. There are $r$ relevant variables out of $n$, denoted as $S = \{x_{i_1}, \cdots, x_{i_r}\} \subset \{x_1, \cdots, x_n\}$, the labeling function is

$$y = l(x) = x_{i_1} \vee x_{i_2} \cdots \vee x_{ir}$$

For example, let $x = (x_1, \cdots x_5), S = (x_2, x_4)$ are the relevant variables. Then

$$l(1,0,1,0,0) = 0, \quad l(1,1,0,0,0) = 1, \quad l(0,0,0,1,1) = 1, \cdots$$

The labeling function can also be written as $\mathbf{1}(\sum_{x \in S} x \geq 1)$, where $\mathbf{1}$ is the indicator function.

**Algorithm 1** Winnow Algorithm to learn a disjunction of $r$ variables

```
Start with $w_i = 1, 1 \leq i \leq n$.
for each input $x$ do
    Predict + if $\sum_{i=1}^n w_i x_i \geq n$ and $-$ otherwise.
    For a mistake on a positive example, for all $i$ with $x_i = 1$, set $w_i \leftarrow 2w_i$ .
    For a mistake on a negative example, for all $i$ with $x_i = 1$, set $w_i \leftarrow w_i/2$ .
end for
```

**Theorem 1.** *The number of mistakes made by Winnow on any possible sequence is at most* $3r \log_2 n + 1$.

*Proof.* Denote $M_+$ as # of mistakes on positive examples, and $M_-$ as # of mistakes on negative examples. Each time the algorithm makes a mistake on a positive example, we will double $w_i$. However, $w_i$ cannot exceed $n$. So we can bound $M_+$ as

$$M_+ \leq r \log_2 n.$$

On a mistake of a positive example, the total weight increases by at most $n$. On a mistake of a negative example, the total weight decreases by at least $n/2$. Since the sum of weights remains nonnegative, we have

$$M_- \leq 2M_+ + 1$$

where the +1 is to account for the fact that the weights start out at a total of $n$. By combining all mistakes, we have

$$M_- + M_+ \le 3r \log_2 n + 1.$$

□

## 1.2 Learn $k$-out-of-$r$ function

We here generalize the hypothesis class. Let $x_i \in \{0,1\}, 1 \le i \le n$. We label the data $x = (x_1, \cdots, x_n)$ as

$$l(x) = \mathbf{1}(x_1 + \cdots + x_r \ge k).$$

**Algorithm 2** Winnow Algorithm to learn $k$-out-of-$r$ function

```
Start with $w_i = 1, 1 \le i \le n$.
for each input $x$ do
    Predict + if $\sum_{i=1}^n w_i x_i \ge n$ and − otherwise.
    For a mistake on a positive example, for all $i$ with $x_i = 1$, set $w_i \leftarrow w_i(1+\epsilon)$ .
    For a mistake on a negative example, for all $i$ with $x_i = 1$, set $w_i \leftarrow w_i/(1+\epsilon)$ .
end for
```

The algorithm is a slight generalization of Algorithm 1.1. By choosing $\epsilon = 1$, the algorithm is the same as Algorithm 1.1. It has the following guarantee.

**Theorem 2.** *The number of mistakes made by Winnow is $O(rk \log n)$.*

*Proof.* Denote $M_+$ as # of mistakes on positive examples, and $M_-$ as # of mistakes on negative examples. We note that the total increase in weight on a mistake of positive example $\le \epsilon n$. After $M_+$ mistakes on positive examples, the total increase in weight on a mistake of positive examples $\le \epsilon n M_+$. Similarly, after $M_-$ mistakes on negative examples, the total decrease in weight on a mistake of negative examples is at least $(n - \frac{n}{1+\epsilon})M_- = \frac{\epsilon n}{1+\epsilon} M_-$. Since the total weight is initialized at $n$, and stays non-negative, we have

$$n + \epsilon n M_+ \ge \frac{\epsilon n}{1+\epsilon} M_-$$

This implies

$$M_- \le \frac{1+\epsilon}{\epsilon} + (1+\epsilon) M_+$$

For a relevant variable $x_i$, it is involved in $a_i$ positive mistakes and $b_i$ negative mistakes. Since $w_i$ cannot exceed $(1+\epsilon)n$, we know

$$a_i - b_i \le \log_{1+\epsilon}(1+\epsilon)n = 1 + \log_{1+\epsilon} n.$$

Summing up over $i$, and we have

$$\sum_{i=1}^r a_i - \sum_{i=1}^r b_i \le r(1 + \log_{1+\epsilon} n).$$

On a mistake on a positive example, the weights of at least $k$ relevant variables increase by $(1+\epsilon)$ factor. On a mistake on a negative example, the weights of at most $k-1$ relevant variables decrease by a $(1+\epsilon)$ factor. This indicates that

$$\sum_{i=1}^r a_i - \sum_{i=1}^r b_i \ge kM_+ - (k-1)M_-$$

This implies that

$$kM_+ - (k-1)M_- \le r(1 + \log_{1+\epsilon} n)$$

Substituting the bound on $M_-$, we have

$$M_+(k-(k-1)(1+\epsilon)) \leq r(1+\log_{1+\epsilon})n + (k-1)\frac{1+\epsilon}{\epsilon}$$

By choosing $\epsilon = \frac{1}{2(k-1)}$, we get

$$\frac{1}{2}M_+ \leq r(1+\log_{1+\frac{1}{2(k-1)}} n) + 2(k-1)^2(1+\frac{1}{2(k-1)})$$

So

$$M_+ = O(rk\log n),$$

and the total number of mistakes can be bounded

$$M_+ + M_- = O(rk\log n).$$

□

## 1.3 Learning halfspaces

Here we consider learning the halfspace $w_1^*x_1 + \cdots + w_n^*x_n \geq w_0^*$. We first do the following three pre-processing steps. Since we do not know $w^*$ in advance, the third step is for analysis only.

1. Scale and shift such that $x_i \in [0,1], w_i^* \in \mathbb{Z}$.
2. If some $w_i^* < 0$, we can use $y_i = 1 - x_i$, and the corresponding term

$$w_i^*x_i = w_i^*(1-y_i) = w_i^* - w_i^*y_i$$

   This step ensures that $w_i^* \geq 0$.
3. For each term $i$, we make $w_i^*$ copies of $x_i$. Define $W^* = \sum_{i=1}^n w_i^*$.

This reduces the problem to learning a $w_0^*$-out-of-$W^*$ function, and we can apply Algorithm 1.2. We give the guarantee as follows.

**Theorem 3.** *The number of mistakes made by the Winnow Algorithm is at most* $O(\|w^*\|_1^2\log(n\|w^*\|_1))$.

Since we assume that the parameters are integers by doing the pre-processing steps, the margin here was 1. More generally, we get the following bound.

$$\#\text{ of mistakes } = O\left(\frac{\|w^*\|_1^2\|x\|_\infty^2\log(n\|w^*\|_1)}{\gamma^2}\right), \text{ where } \gamma := \min_x |\langle w^*, x\rangle|.$$

We can compare it to the bound for the Perceptron algorithm, where the number of mistakes is upper bounded by $O\left(\frac{\|w^*\|_2^2\max_x\|x\|_2^2}{\gamma^2}\right)$ for the same definition of $\gamma$.

# 2 Weighted Majority

**Predicting from Expert advice**. In this setting, there are $n$ experts, and each expert makes 0/1 predictions. The algorithm makes a prediction based on the predictions of the experts (“advice") and then the outcome is revealed.

For example, for the question "will it rain today?" on day $i$, experts $E_j, j \in [n]$, each have predictions in $\{0,1\}$, indicating whether or not it will rain. The algorithm makes predictions taking into account the predictions of the experts, with the goal of making as few mistakes as possible, as compared to the “best" expert, i.e., the minimum number of mistakes made by any expert, for any number of rounds, i.e., days.

**Goal**: $\#$ of mistakes of the algorithm after $T$ days $\lesssim$ minimum $\#$ of mistakes made by any expert after $T$ days.

### 2.1 Follow the Leader

One natural strategy is to follow the leader of the experts. In other word, the algorithm predicts according to the best experts. However, in some bad cases, it can actually make $n$ times as many mistakes as the best expert. The adversarial case is when the leader we choose, which performs the best in the past, make a mistake in the current round.

### 2.2 Weighted Majority

A natural approach is to follow the majority. However, this algorithm could be bad, when there exist few experts with perfect predictions while the majority of the experts stick to the incorrect predictions. An effective variant is to use the weighted majority algorithm. The idea is to keep a weight $w_i$ associated with each expert $E_i$, and the algorithm predicts according to the weighted majority.

**Algorithm 3** Weighted Majority Algorithm

```
Start with $w_i = 1, 1 \leq i \leq n$.
for $t = 1, 2, \cdots, T$ do
    Predict 1 if $\sum_{i:E_i=1} w_i \geq \sum_{i:E_i=0} w_i$ and 0 otherwise.
    On a mistake, for every expert $E_i$ that predicts incorrectly, set $w_i \leftarrow w_i/2$.
end for
```

Define

$$M_T : \text{\# of mistakes by weighted majority after } T \text{ rounds}$$

$$m_T : \text{\# of mistakes by the best expert after } T \text{ rounds}$$

We can bound $M_T$ by the following theorem.

**Theorem 4.** *$\forall T$, the number of mistakes by weighted majority after $T$ rounds satisfies*

$$M_T \leq 2.5 m_T + 2.5 \log_2 n$$

*Proof.* After each mistake, the incorrect weights will be halved. Since these incorrect total weights are higher than the correct weights, after a step of update, the total weights will decrease at least 1/4. So after $M_T$ mistakes, the total weights $\leq n(3/4)^{M_T}$.

Denote the best experts at time $T$ as $E_j$. Since it updates at most $m_T$ times, we have $w_j \geq (1/2)^{m_T}$. Combining the two inequalities, we have

$$(\frac{1}{2})^{m_T} \leq w_j \leq \sum_i w_i \leq n(\frac{3}{4})^{M_T}$$

This leads to

$$(\frac{4}{3})^{M_T} \leq n \cdot 2^{m_T}$$

Taking log, and we have

$$M_T \log_2 \frac{4}{3} \leq \log_2 n + m_T$$

This implies

$$M_T \leq \frac{1}{\log_2 \frac{4}{3}}(m_T + \log_2 n) \leq 2.5 m_T + 2.5 \log_2 n$$

□

**Algorithm 4** Randomized Weighted Majority Algorithm

```
Start with $w_i = 1, 1 \leq i \leq n$.
for $t = 1, 2, \cdots, T$ do
    Predict according to $E_i$ with probability $w_i / \sum_j w_j$.
    On a mistake, for every expert $E_i$ that predicts incorrectly, set $w_i \leftarrow (1-\epsilon)w_i$.
end for
```

## 2.3 Randomized Weighted Majority

We note that we can generalize the update step by setting $w_i \leftarrow (1-\epsilon)w_i$ with a flexible variable $\epsilon$. We also add some randomness. The detailed algorithm is given below.

**Theorem 5.** *For any number of rounds $T > 0$, the expectated number of mistakes of Randomized Weighted Majority satisfies*

$$\mathbb{E}(M_T) \leq (1+\epsilon)m_T + \frac{\log n}{\epsilon}$$

*Proof.* Let $f_t$ be the weight fraction of experts that make a mistake at time $t$. Let $M^t$ be the number of mistakes made by randomized weighted majority at time $t$ for $1 \leq t \leq T$. Since we pick each expert with probability proportional to its weight,

$$\mathbb{E}(M^t) = f_t \text{ and } \mathbb{E}(M_T) = \sum_{t=1}^{T} \mathbb{E}(M^t) = \sum_{t=1}^{T} f_t.$$

In expectation, the total weight after $T$ rounds satisfies

$$\mathbb{E}(\text{total weight}) \leq n \prod_{t=1}^{T} (f_t(1-\epsilon) + (1-f_t)) = n \prod_{t=1}^{T} (1-\epsilon f_t).$$

Let $m_T$ be the minimum number of mistakes after $T$ rounds, made by some expert $E_j$. By our update rule.

$$w_j \geq (1-\epsilon)^{m_T}.$$

Combining the two inequalities, and we have

$$(1-\epsilon)^{m_T} \leq n \prod_{t=1}^{T} (1-\epsilon f_t).$$

By taking log on both sides,

$$m_T \log(1-\epsilon) \leq \log n + \sum \log(1-\epsilon f_t) \leq \log n - \epsilon \sum_{t=1}^{T} f_t = \log n - \epsilon \mathbb{E}(M_T).$$

This implies

$$\mathrm{E}[M_T] \leq \frac{1}{\epsilon}(m_T \log \frac{1}{1-\epsilon} + \log n) \leq \frac{1}{\epsilon}(m_T(\epsilon + \epsilon^2) + \log n) = m_T(1+\epsilon) + \frac{\log n}{\epsilon}.$$

Here we used the fact that $(1+x) \geq e^{x-x^2}$ for $x < 1$. □

**Corollary 6.** *By choosing $\epsilon = \sqrt{\frac{\log n}{m_T}}$, we achieve the best bound for Randomized Weighted Majority,*

$$\mathrm{E}[M_T] \leq m_T + 2\sqrt{m_T \log n}.$$

*The average mistakes made per round is*

$$\frac{\mathrm{E}[M_T]}{T} \leq \frac{m_T}{T} + 2\sqrt{\frac{\log n}{T}}$$

*Proof.* We choose $\epsilon$ so as to minimize $m_T(1+\epsilon)+\frac{\log n}{\epsilon}$. Since

$$m_T\epsilon+\frac{\log n}{\epsilon}\geq 2\sqrt{m_T\log n},$$

and the equality holds when $\epsilon=\sqrt{\frac{\log n}{m_T}}$, we know

$$\mathrm{E}[M_T]\leq m_T+2\sqrt{m_T\log n}$$

Consider the average mistakes made per round, we have

$$\frac{\mathrm{E}[M_T]}{T}\leq\frac{m_T}{T}+2\sqrt{\frac{m_T\log n}{T^2}}\leq\frac{m_T}{T}+2\sqrt{\frac{\log n}{T}}$$

The last inequality comes from $m_T\leq T$. When $T\to\infty$, the second term $\to 0$. So the number of mistakes per round made by the algorithm goes to that of the best expert. □