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

Lecture 2: Weighted Majority and Winnow

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

We recall the Perceptron algorithm discussed in the last lecture. For normal vector $\|w^*\|_2 \le 1$, samples $\|x\|_2 \le 1$, 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

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

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 $\mathbf{1}(\sum_{x\in S} x \ge 1)$, where $\mathbf{1}$ is the indicator function.

Algorithm 1 Winnow Algorithm to learn a disjunction of r variables

Start with $w_i = 1, 1 \le i \le n$.
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 $w_i \leftarrow 2w_i$.
  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

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

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

$$M_- \le 2M_+ + 1$$

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

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

$\square$

1.2 Learn k-out-of-r function

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

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

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

> 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 $w_i \leftarrow w_i(1 + \epsilon)$.
>   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 $\leq \epsilon n$. After M_+ mistakes on positive examples, the total increase in weight on a mistake of positive examples $\leq \epsilon n M_+$. 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

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

This implies

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

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. Then by considering the total change of the relevant variables in terms of the number of factors of $(1 + \epsilon)$, we have

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

Substituting the bound on M_-, we have

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

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

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

So

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

and the total number of mistakes can be bounded

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

$\square$

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

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

   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.

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

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