CS639: Introduction to Computational Learning Theory	**Scribe:** Changho Shin
Lecture 4: Winnow and Perceptron Algorithms	**Lecturer:** Ilias Diakonikolas

4.1 Winnow 1 Algorithm[1]

- Online learning algorithm for monotone disjunctions
- Much better performance than elimination algorithm when $k \ll n$, where k is the number of variables that appear in unknown function and n is the total number of variables.
- Attribute efficiency / sparsity
- Hypotheses: LTF

$$h(x) = \begin{cases} 1, \text{ if} \sum w_i x_i \geq 0 \\ 0, \text{ otherwise} \end{cases}$$

, where $X = \{0,1\}^n$

4.1.1 Winnow 1 Algorithm Procedure

- Initial hypothesis: $w_i = 1$, for all $1 \leq i \leq n$,

$$h_0(x) = \begin{cases} 1, \text{ if} \sum_i^n x_i \geq n \\ 0, \text{ otherwise} \end{cases}$$

- Update rule: given $x \in \{0,1\}^n$
  - If $h(x) = c(x)$, don't change h.
  - If $h(x) = 1$ and $c(x) = 0$, for all i s.t. $x_i = 1$, set $w_i = 0$. (Demotion step)
  - If $h(x) = 0$ and $c(x) = 1$, for all i s.t. $x_i = 1$, set $w_i = 2w_i$. (Promotion step)

4.1.2 Analysis on Winnow 1

Theorem 4.1.1 *Winnow 1 has the mistake bound on a disjunction of length k over $\{0,1\}^n$ of at most $O(k \log n)$.*

Lemma 4.1.2 *Throughout the execution of Winnow 1 algorithm, we have $w_i \geq 0$ for all i.*

Lemma 4.1.3 *In each promotion step, some variable that is contained in c gets its w_i doubled.*

Proof: Given x s.t. $c(x) = 1$ and $h(x) = 0$, it means

$$c(x) = \vee_{j \in S} x_j, \text{ where } c(x) = 1 \rightarrow \exists i \in S : x_i = 1, |S| \leq k$$

Winnow 1 algorithm doubles the weights of all coordinates i that are set to 1 in $x(x_i = 1)$. ■

Lemma 4.1.4 *Throughout algorithm, we always have* $w_i \leq 2n$

Proof: Since no weights < 0, if some w_j is $> n$ and we are given x s.t. $x_j = 1$, then $h(x) = 1$ ($\sum_{i=1}^{n} w_i x_i \geq w_j > n$). So, no weight w_j which is $> n$ ever doubled. Thus, no weight is ever $> 2n$. ■

Lemma 4.1.5 *If c is a disjunction of k variables, $p(\doteq$=the number of promotion steps$) \leq k \log_2(2n)$.*

Proof: No variable that is contained in c is ever demoted. Any variable in c is promoted $\leq \log_2(2n)$ times. We have k variables in c. So $p \leq k \log_2(2n)$. ■

Lemma 4.1.6 *Let d be the number of demotion steps. Then, $d \leq p + 1$.*

Proof: We are going to use $w = \sum_{i=1}^{n} w_i$, which is the total weight of current hypothesis. This quantity is n initially because of the definition of our initial values, and changes during the execution of the algorithm.
Fact 1: $w \geq 0$
Fact 2: At each demotion step w decrease by $\geq n$. Note the condition the demotion occurs: x s.t. $\sum_{i=1}^{n} w_i x_i \geq n$. In the demotion step, find all i s.t. $x_i = 1$ and set all corresponding $w_i = 0$. Note that $\sum_{i=1}^{n} w_i x_i = \sum_{i:x_i=1} w_i$. So wehave that before the demotion step happened $\sum_{i:x_i=1} \geq n$. And, we reduce w by this amount.
Fact 3: At each promotion step w increases by $< n$. Note that the promotion means that given x s.t. $\sum_{i=1}^{n} w_i x_i < n$, w increases by this amount. As a result, we get the following inequality.

$$0 \leq w \leq n - dn + pn$$

, which leads to $d \leq p + 1$. ■

Proof of [: 4.1.1] By the lemma 4.1.6, the total number of mistakes $= p + d \leq 2p + 1 \leq O(k \log n)$ ■

Remarks

- This mistake bound is optimal.
- Computationally efficient.
- Bad with noisy data
- Open problem: can we get computationally efficient algorithm for length-k decision list with the mistake bound $O(k \log n)$?
  - There exists algorithm with this mistake bound, but it runs in exponential time.
  - There exists a computationally efficient algorithm with the mistake bound $O(kn)$

Say a function $c : \{0,1\}^n \to \{0,1\}$ is "$\delta-$separable" if there exist weights $w_1, \cdots, w_n$ such that

$$c(x) = 1 \iff \sum_{i=1}^{n} w_i x_i \geq 1$$

$$c(x) = 0 \iff \sum_{i=1}^{n} w_i x_i < 1 - \delta$$

Notion of margin.

Exercise 1. Any disjunction is $\frac{1}{2}$ separable.
Exercise 2. Say $c(x) = 1 \iff x_1 + ... + x_k \geq \gamma$ (r-of-k threshold function).

$$\Rightarrow \frac{1}{\gamma} x_1 + ... + \frac{1}{\gamma} x_k \geq 1$$

Note that if $\frac{1}{\gamma} x_1 + ... + \frac{1}{\gamma} x_k < 1$, then in fact it is $\leq 1 - \frac{1}{\gamma}$ because $x_i = 0$ or 1.
$\Rightarrow c$ is $\frac{1}{\gamma}$-separable.

4.2 Winnow 2 Algorithm

Winnow 2 algorithm takes parameters $\theta > 0$, $a > 1$. It predicts using $h(x) = \begin{cases} 1, \text{ if } \sum_i^n w_i x_i \geq \theta \\ 0, \text{ otherwise} \end{cases}$

Winnow 2 Algorithm Procedure

- Initialize $w_i = 1$ for all i.
- Update
  - Promotion ($h(x) = 0, c(x) = 1$): Multiply each w_i s.t. x_i in x by a
  - Demotion ($h(x) = 1, c(x) = 0$): Multiply each w_i s.t. $x_i = 1$ in x by $1/a$

Theorem 4.2.1 *Suppose target concept c is $\delta-$separable, $0 < \delta < 1$. Let $w_1, ..., w_n$ be weights for c. If $a = 1 + \delta/2$ and $\theta \geq 1$, then winnow 2 makes $\leq \frac{8n}{\delta^2 \theta} + \left(\frac{5}{\delta} + \frac{14 \log \theta}{\delta^2}\right) \sum_{i=1}^{n} w_i$, which is very roughly $\frac{1}{\delta^2} \log n$.*

Excercise. If c is an $r-$out of k function, $\delta = \frac{1}{\gamma}$, $a = 1 + \frac{1}{2r}$ and $\theta = n$, it has the mistake bound $O(r^2 + kr \log n)$.

Fact. The decision list is $\delta-$separable for $\delta = \frac{1}{2^{O(n)}}$.

So, Winnow 2 for this particular function does not give good mistake bounds.

References

[1] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.