CS639: Introduction to Computational Learning Theory	**Scribe:** Changho Shin
Lecture 3: Online Learning of Disjunctions and Decision Lists	**Lecturer:** Ilias Diakonikolas

3.1 Learning Decision Lists

3.1.1 Decision Lists

A decision list (DL) is a boolean functions of the form "If l_1 then output b_1, else if l_2 then output b_2, ..., else b_{k+1}", where l_i is the literal (a variable or its negation). The length of decision list is the number of literals that are tested in the worst-case.

Obviously, every conjunction/disjunction can be expressed as a decision list.

Without loss of generality, any decision list has length $\leq n$. Having x_i and $\bar{x_i}$ simultaneously is not meaningful in decision list.

3.1.2 Online Learning Algorithm for Decision Lists

We will consider length $r(1 \leq r \leq n)$ decision lists over $x_1, ..., x_n$. Our learning algorithm uses "generalized decision lists", where each level can have multiple "if l then output b" rules (even conflicting ones). In each step/trial, go through levels 1,2,... until we find a level, with some rule (l, b), where l is satisfied by x, output b. If two or more conflicting rules apply, pick arbitrarily. For the intial h, put all possible rules in level 1. There are total $4n+2$ possible rules.($l \rightarrow 2n, b \rightarrow 2, 0/1$ (the else cases) $\rightarrow$). And, update the hypothesis h given example x.

- If $h(x) = c(x)$, do nothing.
- If $h(x) \neq c(x)$, take the rule that we used to predict x, and move it to the next level.

Theorem 3.1.1 *The described algorithm above makes $O(nr)$ mistakes on any sequence of examples if true concept c is a length-r decision list over n boolean variable.*

Proof: Suppose that true target concept $c = (l_1, b_1), ..., (l_r, b_r)$. First rule (l_1, b_1) is never moved to level 2 in the hypothesis maintained by our algorithm. If l_1 holds, then the output of c is b_1 by definition. By induction on i for $2 \leq i \leq r$, the rule (l_i, b_i) never gets sent to level $j > i$. Then, no rule in level $r + 2$ is ever consulted, so no rule is ever sent to level $r + 3$. Every mistake moves a rule on level. The total number of levels moved by all rules is less than $(4n + 2)(r + 1)$, which leads to the mistake bound $O(nr)$. Note that computation time per example/trial is $O(n^2)$, which is also good. ■

3.2 Learning Sparse Disjunctions with Winnow Algorithms

Consider learning sparse disjunctions, for example, $n = 10^6$, but $c(x) = x_{7145} \vee x_{11146} \vee x_{218342}$. Elimination algorithm takes mistake bound n. We will show winnow algorithm learns length-k

disjunctions with mistake bound $O(k \log n)$.

Winnow algorithm uses linear threshold functions (LTFs) as hypotheses, i.e. $(w_1, ..., w_n, \theta)$ and

$$h(x) = \begin{cases} 1, \text{ if } \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0, \text{ otherwise} \end{cases}$$

Note that LTFs can express disjunctions and conjunctions as following examples.

1) Disjunction example
$x_1 \vee ... \vee x_k$
$\Rightarrow w_1 = w_2 = ... = w_k = 1, \theta = 1/2$
$\Rightarrow x_1 + ... + x_k \geq 1/2$

2) Conjunction example
$x_1 \wedge ... \wedge x_k$
$\Rightarrow x_1 + ... + x_k \geq k - 1/2$

Theorem 3.2.1 *Any decision list can be expressed as a linear threshold function.*