Lecturer: Siu On Chan *Based on Rocco Servedio's notes*

Notes 2: Online Mistake Bound Model

1. Online mistake bound model

A sequence of trials/rounds, each being:

(1) An unlabeled example $x \in X$ arrives
(2) Algorithm maintains hypothesis $h : X \to \{0,1\}$ and outputs $h(x)$
(3) Algorithm is told the correct value of $c(x)$
(4) Algorithm may update its hypothesis

Goal: minimize number of mistakes (i.e. $h(x) \neq c(x)$) on the worst sequence of examples and $c \in \mathcal{C}$

Trivial mistake bounds:
If X finite, #mistakes $\leqslant |X|$ (memorize $c(x)$)
If $\mathcal{C}$ finite, #mistakes $\leqslant |\mathcal{C}| - 1$ (try all $c \in \mathcal{C}$)

---

2. Monotone conjuctions

A conjunction is **monotone** if all its literals are positive, e.g. $c(x) = x_2 \wedge x_4 \wedge x_5$

Elimination Algorithm

Initialize: $h(x) =$ conjunction of all literals $= x_1 \wedge x_2 \wedge \cdots \wedge x_n$
False negative ($h(x) = 0, c(x) = 1$): remove all literals that are false in x
False positive ($h(x) = 1, c(x) = 0$): output FAIL

else no need to update h

Invariant: h always contains all literals in c
Corollary: Algorithm never fails
#Mistakes $\leqslant n$: Each mistake removes at least one literal from h
We will see later that this bound is tight!

Variant 1: Monotone disjunction — same idea
Variant 2: non-monotone conjuction
Initial hypothesis begins with $2n$ literials $h(x) = x_1 \wedge \overline{x}_1 \wedge x_2 \wedge \overline{x}_2 \wedge \cdots \wedge x_n \wedge \overline{x}_n$
First mistake removes n literials, then at most n more mistakes ($n+1$ total)
Variant 3: k-DNF for fixed constant k — same elimination idea

---

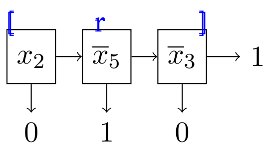
3. Decision lists

A 1-decision list (1-DL) has the form

$$\begin{gathered}\text{if } y_1 \text{ then output } b_1\\ \text{else if } y_2 \text{ then output } b_2\\ \vdots\\ \text{else if } y_r \text{ then output } b_r\\ \text{else output } b_{r+1}\end{gathered}$$

example:
if email title contains products, then spam
if email ....

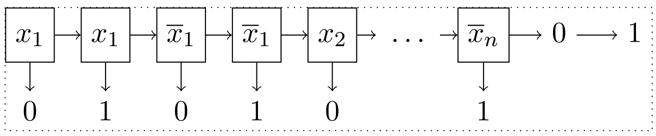
where y_i are literals, $b_i \in \{0,1\}$ are bits
e.g.



disjunction and conjunction can be reduced as decision is 1-DL of length 3

Every 1-DNF is 1-DL, so is every 1-CNF
Can assume no variable appears twice in 1-DL $\Rightarrow$ length at most n
How many 1-DL of length r are there? about $(4n+2)^r$
$(4n+2)$ rules: $4n$ "$y_i \to b_i$" and two "$\to b_i$"

Algorithm to learn 1-DL of length r with $O(nr)$ mistakes:
Hypothesis has several "levels". It has all $4n+2$ rules, each belonging to one of the levels
Rules of the same level are ordered arbitrarily, say lexicographically
Initially all rules are at level 1 eg:0010 ≈x1 is mis-classify



level 1

All rules of lower level come before rules of higher level
On every sample x:
    hypothesis h classifies x by the first rule whose condition is satisfied by x
    if h misclassifies x (i.e. $h(x) \neq c(x)$), move that rule to the next level

e.g. if $x = 101$, $c(x) = 1$, initial hypothesis misclassifies x due to "$x_1 \to 0$"
    Move this rule to level 2 after the mistake

Claim 3.1. *This algorithm makes* $\leqslant (4n+2)(r+1) = O(nr)$ *mistakes on any* 1-*DL of length* r

Observation: 1st rule in c (call it r_1) is never moved above level 1
Reason: if h classifies x based on r_1, h agrees with c since c also classifies x based on r_1
Observation: 2nd rule in c (call it r_2) is never moved above level 2
Reason: if h classifies x based on r_2 while r_2 is at level 2, r_1 must remain at level 1 by previous observation, thus x violates r_1's condition, and h agrees with c since they both classify x based on r_2

Inductively, ith rule in c is never moved above level i
Conclusion: no rule is moved above level $r+2$, because the last rule in c (which is unconditional) stays within level $r+1$ in h, and h never classifies samples using any rule at level $r+2$
Each rule is moved at most $r+1$ times, proving the claim

---

k-decision list (k-DL): like a decision list, but each condition y_i is a conjuction of at most k literals
Algorithm to learn k-DL of length r — same idea