CS639: Introduction to Computational Learning Theory	**Scribe:** Changho Shin
Lecture 2: Introduction to Online Learning	**Lecturer:** Ilias Diakonikolas

2.1 Online Mistake Bound Learning Model (OMBLM or OL)

- A learning session proceeds in a sequence of trials/rounds
- Through the session, learner (=learning algorithm) maintains a hypothesis (belief of the learner about the true function) $h : X \to \{0, 1\}$
- In each trial,
  - Learner is given data point $x \in X$ without label.
  - Learner uses its current hypothesis h for prediction, $h(x)$
  - Learner is told/given the true label of the concept, $c(x)$
  - If $h(x) \neq c(x)$, the algorithm is charged with a mistake
  - Learner is allowed to update its hypothesis h. Update rule is the learning algorithm.
  - Performance measure: The number of mistakes (Assumption: it is bounded).

Definition 2.1.1 (Mistake bound) *A learning algorithm $\mathcal{A}$ has **mistake bound** M for a concept class C if for any target concept $c \in C$ and any sequence of examples from X, $\mathcal{A}$ makes $\leq M$ mistakes.*

Observations

1. If X is bounded, $\mathcal{A}$ can always achieve mistake bound of $|X|$.
2. If C is finite, can always achieve, can always achieve $|C| - 1$. (Strategy: when h meets the wrong case, it changes h into another c)

Example
Let $X = \{0, 1, 2, ..., 2^n - 1\}$ and C is the class of all "initial intervals" (e.g. $C = \{0, 1, ..., 17\}$). In this case, $|C| = 2^n + 1$, where 1 comes from empty interval.

Both observations above imply that we can learn C in the OMBLM with $\leq 2^n$ mistakes. We will show that there is a better algorithm that in fact can learn C with the mistake bound $\leq n$.

Idea: Use "binary search"

Example: $n = 10, X = \{0, 1, ..., 1023\}$.
Initial hypothesis h that our algorithm will be $\{0, 1, ..., 512\}$. Suppose that the final example we are given to predict is $x = 400$. Our initial h will predict $h(x) = 1$. If $c(x) = 1$, then it makes no

mistake. If $c(x) = 0$, then we know that the true interval defining $c(x)$ is a subset of $[0, 399]$. Rule out points 400 and larger as endpoints. And, update h to be $\{0, ..., 200\}$.

Claim: For each mistake we value the size of the active domain at least by a factor of 2. It leads the mistake bound $\log 2^n$, i.e. n.

Food for thought
Consider $X = [0, 1]$, C is the all initial intervals $c = [0, a]$, $a \in [0, 1]$. Then, there exists no finite mistake bound algorithm because examples can be arbitrarily close to boundary.

2.2 Learning (monotone) Disjunctions

Let C is all monotone disjunctions, where monotone measn "no negator" and disjunctions a class for OR of variables. Define the domain $X = \{\pm 1\}^n$, $x = (x_1, ..., x_n)$, $x_k \in \{\pm 1\}$. (e.g. $c(x) = x_1 \vee x_4 \vee x_5$)

2.2.1 Online Learning Algorithm for Monotone Disjunctions

- Initial hypothesis is $h(x) = x_1 \vee ... \vee x_n$
- Hypotheses are updated as follows. Given $x, h(x), c(x)$
  - If $h(x) = 1$ and $c(x) = 0$ (false positive case), then look at the coordinates i of vector $x = (x_1, ..., x_n)$ s.t. $x_i = 1$ and remove variables x_i from $h(x)$.
  - If $h(x) = 0$ and $c(x) = 1$ (false negative case), stop and output "FAIL" (Note: it cannot happen unless it has noisy labels)
  - If $h(x) = c(x)$, do not change h
- Assumption: we only update h in the case of mistake. Actually, we can always convert the algorithm that updates even when h does not make a mistake into the one that update only h in the case of mistake.

Example
$n = 5$, initial $h(x) = \vee_{i=1}^{5} x_i$. Get $x = 10010$. Initial prediction $h(x) = 1$. Suppose $c(x)$. Then update h to $h(x) = x_2 \vee x_3 \vee x_5$.

Theorem 2.2.1 *This online algorithm has the mistake bound n.*

Claim 2.2.2 *No variable that appears in c is ever removed from h.*

Proof: Var x_i is removed from h only if x_i in an example x such that $c(x)$. But such an x_i cannot be in c since if it was, it would cause $c(x)$ to be 1, which is a contradiction. ■

Claim 2.2.3 *Algorithm only makes false positive mistakes.*

Proof: By claim 2.2.2, any x_i in c remains in h. False negative means $c(x) = 1$ and $h(x) = 0$, but this can't happen since the satisfied var in c is also in h.

$$c(x) = \vee_{i \in S_c} x_i$$

$$h(x) = \vee_{i \in S_h} x_i$$

Since always $S_c \subseteq S_h$, $\forall x : c(x) = 1 \rightarrow h(x) = 1$. ■

Proof of 2.2.1: By claim 2.2.3, algorithm does not output "FAIL". Variables in h are always superset of vars in c. Start with n variables in h. Each mistake causes h to eliminate at least one variable. So we can have $\leq n$ mistakes. ■

More about Elimination Algorithm

- Initial hypothesis: $h(x) = x_1 \vee x_2 \vee ... \vee x_n$.
- On false positive ($h(x) = 1, c(x) = 0$), eliminate vars x_i s.t. $x_i = 1$ in x from h
- Mistake bound $O(n)$ and computational requirements $O(n)$ time per examples are good.
- Noisy tolerant? $\rightarrow$ No. For example, the case that $x = 1...1$ but given noisy label 0, it will eliminate all variables and cannot recover from it.
- General (including non-monotone disjunctionsm, e.g. $x_3 \vee \bar{x_5} \vee x_7$) can be solved with the variant of this algorithm, which use augmented initialization:

  $$h(x) = x_1 \vee \bar{x_1} \vee x_2 \vee \bar{x_2} \vee ... \vee x_n \vee \bar{x_n}$$

  Use "literal" for "variable" in previous elimination algorithm

Example
If $n = 4$, suppose that we get $x = 1101$ at first and $h(x) = 1$. If $c(x) = 0 \rightarrow h(x) = \bar{x_1} \vee \bar{x_2} \vee x_3 \vee \bar{x_4}$. In this case, the mistake bound is $n + 1$. Note that it is not $2n$. First example removes n of $2n$ literals.

Remark. The elimination algorithm can also be used to learn conjunctions using de morgan's law. First we can learn the negation of conjunctions, which is disjunctions. And then we can take the negation. Consider the following example.

$$c(x) = x_1 \wedge \bar{x_2} \wedge x_3$$

$$\bar{c(x)} = \bar{x_1} \vee x_2 \vee \bar{x_3}$$

Or, we can work a dual elimination algorithm for conjunctions, which updates its hypothesis on false negative example.