

# Computational Learning Theory

Agam Goyal

January 27, 2023

## Lec 01: Introduction

**Computational learning theory** studies machine learning from a theoretical computer science point of view. **Machine Learning:** Programs that enable automatic extraction of useful information from “raw” data, and which improve their performance via interaction with this data. Examples: Classification, Clustering, Prediction, etc.

### Overview of Computational Learning Theory

**A Learning Model should specify the following:**

- (1) “Who” is learning?
  - A computer program, usually restricted
    - (i) Polynomial run-time
    - (ii) Small sample complexity
  - Particular format for output hypothesis (for the interpretability of the model)
- (2) “What” are we learning?
  - Skills and environment
  - “Classification” rule: Boolean function
- (3) “How” does the learner get information?
  - Learner is given examples as point-label pairs:  $(x, f(x))$ 
    - (i) “Passive” learning - only observes the information provided by the environment without influencing it
    - (ii) Assumptions about  $X$ :
      - Randomly chosen
      - Maliciously chosen (Adversary models)
      - Provided by teacher (Adversarial teacher model – worst-case scenario)
  - Learner makes queries
    - (i) “Active” learning - interacts with the environment at training time, by posing queries or performing experiments

### Is the data ever noisy or incomplete?

The data is usually almost always noisy and/or incomplete, and it is useful to study the robustness of learning models to these datasets.

### What “prior rule” does the learner have?

The typical assumption is that there is an a priori representational scheme for the function being learned known as the **target concept**.

### Performance Criterion

- How to measure accuracy?
- Online vs Offline/Batch Learning Methods

## Lecture 02: Online Mistake-Bound learning Models

### Setup

$\mathbf{X}$  : instance space

$C$  : concept (a Boolean function on  $X$ ) —  $C : \mathbf{X} \rightarrow \{0, 1\}$

$\mathcal{C}$  : Concept Class (set of all Boolean functions) —  $C \in \mathcal{C}$

Typically,  $\mathbf{X} = \{0, 1\}^n$  or  $\mathbf{X} = \mathbb{R}^n$  and the learner knows  $\mathcal{C}$  and  $\mathbf{X}$ , but doesn't know the individual concept target  $C$  which is a specific function.

### Example #1

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

$\mathcal{C} =$  All **monotone conjunctions** over  $x_1, \dots, x_n$

A monotone conjunction is an AND of variables such that no variable in the conjunction is negated.

$$|\mathcal{C}| = 2^n \rightarrow \text{All possible subsets}$$

Example of a concept:  $C(\mathbf{x}) = x_3 \wedge x_5 \wedge x_6$

### Example #2

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

$\mathcal{C} =$  All conjunctions over  $x_1, \dots, x_n$

$$|\mathcal{C}| = 3^n \rightarrow \text{Each literal can exist as itself, its conjugate, or not exist}$$

Example of a concept:  $C(\mathbf{x}) = \bar{x}_2 \wedge \bar{x}_4 \wedge x_7 \wedge \bar{x}_8$

### Example #3

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

$\mathcal{C} =$  All **DNFs** over  $x_1, \dots, x_n$  with  $\leq n^2$  terms

A Disjunctive Normal Form (DNF) is an OR of ANDs of literals.

$$|\mathcal{C}| = \text{Number of terms in the DNF formula}$$

Example of a concept:  $C(\mathbf{x}) = (x_2 x_3) \vee (x_1 \bar{x}_2 x_4) \vee (\bar{x}_2 \bar{x}_4 \bar{x}_5 x_7) \vee x_7 \bar{x}_9$

We can extend this idea to a **k-DNF**: a DNF with  $\leq k$  variables.

### Example #4

$$\mathbf{X} = \mathbb{R}^n$$

$\mathcal{C} =$  All Linear Threshold Functions (LTFs / Halfspaces)

Example of a concept:  $C : \mathbb{R}^n \rightarrow \{0, 1\}$

$$\exists n \text{ real weights } w_1, w_2, \dots, w_n \in \mathbb{R} \text{ and } \theta \in \mathbb{R}, \text{ s.t. } \forall \mathbf{x} \in \mathbb{R}^n, C(\mathbf{x}) = \begin{cases} 1, & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

where  $\mathbf{w}$  is the vector of weights  $w_1, w_2, \dots, w_n$ , and  $\mathbf{x} \in \mathbf{X}$ .

## Online Mistake-Bound Learning Models

---

**Algorithm 1** Online Mistake-Bound Learning Model

---

```
1: A learning session proceeds in a sequence of trials
2: Throughout the session, learner maintains hypothesis  $h : \mathbf{X} \rightarrow \{0, 1\}$ 
3: while In a trial do:
4:   Learner is given some  $\mathbf{x} \in \mathbf{X}$ 
5:   Learner outputs  $h(\mathbf{x})$  as prediction
6:   Learner is told the true  $c(\mathbf{x})$ 
7:   if  $h(\mathbf{x}) \neq c(\mathbf{x})$  then
8:     Learner is charged for the mistake
9:   Learner may update  $h$ 
10:
```

---

<b>Performance Measure = Number of Mistakes</b>
---

### Definition: Mistake Bound (M)

A learning algorithm  $A$  has a Mistake Bound  $M$  for a concept class  $\mathcal{C}$  if

$(\forall c \in \mathcal{C}) (\forall \text{ sequence of examples from } \mathbf{X}) \quad A \text{ makes } \leq M \text{ mistakes.}$

### Observations:

- If  $\mathbf{X}$  is finite, the learner can always achieve  $M = |\mathbf{X}|$ .
- If  $\mathcal{C}$  is finite, the learner can always achieve  $M = |\mathcal{C}| - 1$ .

### Example #1

$\mathbf{X} = \{0, 1, \dots, 2^n - 1\}$

$\mathcal{C} =$  All initial intervals  $[0, \alpha)$  where  $\alpha \in \mathbb{Z}^+ \cup 0$

$|\mathcal{C}| = 2^n + 1 \rightarrow$  Includes every possible integer from 0 to  $2^n - 1$  and also the interval that even excludes 0.

Note that we can use **binary search** to learn  $\mathcal{C}$  with  $M = \log_2(2^n) \implies \boxed{M = n}$ . Our initial hypothesis  $h$  in this case would just be the initial half interval  $\{0, 1, \dots, 2^{n-1}\}$ . And then if the learner makes a mistake for some input  $\mathbf{x}$ , we can update the interval accordingly. So essentially, each mistake eliminates  $\geq$  Half of the remaining possibilities.

### Example #2

$\mathbf{X} = [0, 1]$

$\mathcal{C} =$  All initial intervals  $[0, \alpha)$  where  $\alpha \in [0, 1]$

In this case, the mistake bound  $\boxed{M = \infty}$  because of examples arbitrarily close to the boundaries we choose. So we may never really get to the *exact* answer we desire. However, in reality we would often just want to achieve a result that would be a *good enough* approximation.