Machine Learning - B
Lectures 5
Lecture 5: Computational Learning Theory

Amartya Sanyal

University of Copenhagen

Computational Learning Theory

- So far our questions were of the form: Given m samples i.i.d. from a distribution D, can we bound the error $\varepsilon = O(m^{-\alpha})$?
- Now, we will treat the learning algorithm as a computational process.
- We must specify: interaction with D, prior information, runtime, and hypothesis representation.
- Goal: Determine what can be learned under computational restrictions.

Binary Classification Setting

- Instance space X (e.g. $\mathbb{R}^d$, $\{0,1\}^d$).
- Label space $Y = \{+1, -1\}$.
- Concept class $\mathcal{C}$ and hypothesis class $\mathcal{H}$: subsets of functions $X \to Y$.
- Data distribution D over X.
- Example oracle $EX(c, D)$ returns $(x, c(x))$ for $x \sim D$.
- Target concept $c \in \mathcal{C}$ is unknown.

Learning Algorithm Model

- Learning algorithm A for concept class $\mathcal{C}$ using hypotheses from $\mathcal{H}$.
- Queries $EX(c, D)$ multiple times at unit cost.
- Returns hypothesis $h \in \mathcal{H}$.
- Randomness from sampling and internal coin flips.
- Overall probability denoted by Pr.
- Population Error of h is $\mathrm{err}_D(h) = \mathbb{E}_{(x,y)\sim D}[h(x) \neq y]$

PAC Learnability: Definition 1

Definition

Definition: A concept class $\mathcal{C}$ over domain X is PAC learnable with $\mathcal{H}$ if

- there exists an algorithm A and polynomial p there exists an algorithm A and polynomial p
- such that for any target $c \in \mathcal{C}$ and distribution D, such that for any target $c \in \mathcal{C}$ and distribution D,
- given $m \geq p(1/\epsilon, 1/\delta)$ i.i.d. samples from $EX(c, D)$, given $m \geq p(1/\epsilon, 1/\delta)$ i.i.d. samples from $EX(c, D)$,
- the algorithm outputs $h \in \mathcal{H}$ satisfying $\Pr[\mathrm{err}_D(h) \leq \epsilon] \geq 1 - \delta$ the algorithm outputs $h \in \mathcal{H}$ satisfying $\Pr[\mathrm{err}_D(h) \leq \epsilon] \geq 1 - \delta$

Understanding the PAC Definition

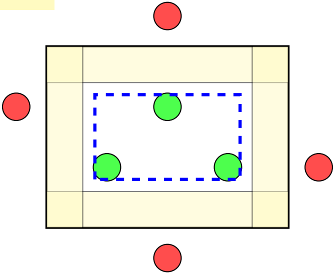
- **Efficiency/Computational Complexity**: time measured in calls to $EX(c, D)$ and runtime of algorithm.
- **Proper vs. improper learning**: whether $\mathcal{H} = \mathcal{C}$.
- **Known**: Concept Class $\mathcal{C}$, error ε, and failure probability δ; Unknown: target c and distribution D.
- **Oracle variants**: noisy labels, positive-only, membership queries, statistical queries.

Problem 1: Learning Axis-Aligned Rectangles

Target Concept: True rectangle unknown to learner.

Data Sampling: Draw $m = \frac{4}{\varepsilon} \ln \frac{4}{\delta}$ samples; mark positives (green) and negatives (red).

Algorithm: Output minimal rectangle covering all positive samples (blue).



Proof

Step 1: Split true rectangle boundary into 4 $\epsilon/4$-mass strips (yellow shading).

Step 2: Each strip empty w.p. $\leq e^{-m\varepsilon/4}$.

Step 3: Over 4 strips gives failure $\leq 4e^{-m\varepsilon/4} \leq \delta$.

Empty with probability $(1 - \frac{\epsilon}{4})^m \leq e^{(-m\epsilon/4)}$.

Apply union bound for overall failure.

Theorem: PAC Learnability of Axis-Aligned Rectangles

Theorem

Let $\mathcal{C}_{rect}$ be the class of axis-aligned rectangles in $\mathbb{R}^2$. For any $0 < \epsilon, \delta < 1$, sampling

$$m = \frac{4}{\epsilon} \ln \frac{4}{\delta}$$

i.i.d. examples from $EX(c, \mathcal{D})$ suffices for a learner to output a hypothesis rectangle h such that

$$\Pr_{\mathcal{D}}[h(x) \neq c(x)] \leq \epsilon$$

with probability at least $1 - \delta$. Hence $\mathcal{C}_{rect}$ is PAC learnable.

Proof Sketch: Follows directly from the earlier analysis: the algorithm uses $m = O\big(1/\epsilon \ln(1/\delta)\big)$ samples and with high probability achieves error at most ϵ, matching the PAC definition.

Representation and Complexity (PAC Learnability - Definition 2)

- **Representation Scheme.** A mapping $\rho : (\Sigma \cup \mathbb{R})^* \to \mathcal{C}$ where each concept $c \in \mathcal{C}$ is encoded by a string σ of length $|\sigma| = \mathrm{size}(c)$. E.g., axis-aligned rectangle specified by its four real-valued corners, or a conjunction specified by the list of its literals.
- **Instance and Concept Size.** Let X_d be all inputs of "size" d and $\mathcal{C}_d$ the corresponding concept class. Typically, $\mathrm{size}(c) = O(d)$.
- **Efficient PAC Learning.** A concept class $\mathcal{C}$ is efficiently PAC learnable if there exists an algorithm A that, on input $\epsilon, \delta, d, \sigma$ runs in time $\mathrm{poly}(d, \mathrm{size}(c), 1/\epsilon, 1/\delta)$

We will next talk about a learning task where the size of the hypothesis class is important for learnability

PAC Learnability: Definition II

Definition

A concept class $\mathcal{C} = \bigcup_{d \geq 1} \mathcal{C}_d$ over instance spaces $\mathcal{X} = \bigcup_{d \geq 1} \mathcal{X}_d$ is *efficiently PAC learnable* by hypothesis class $\mathcal{H} = \bigcup_{d \geq 1} \mathcal{H}_d$ if

- there exists an algorithm A and polynomial p there exists an algorithm A and polynomial p
- such that $\forall d \geq 1$, $c \in \mathcal{C}_d$, distributions D on $\mathcal{X}_d$, and $0 < \epsilon, \delta < 1$, such that $\forall d \geq 1$, $c \in \mathcal{C}_d$, distributions D on $\mathcal{X}_d$, and $0 < \epsilon, \delta < 1$,
- given $m \geq p\big(d, \mathrm{size}(c), 1/\epsilon, 1/\delta\big)$ samples from $\mathrm{EX}(c, D)$, given $m \geq p\big(d, \mathrm{size}(c), 1/\epsilon, 1/\delta\big)$ samples from $\mathrm{EX}(c, D)$,
- the algorithm A outputs $h \in \mathcal{H}$ satisfying $\Pr[\mathrm{err}_D(h) \leq \epsilon] \geq -\delta$, the algorithm A outputs $h \in \mathcal{H}$ satisfying $\Pr[\mathrm{err}_D(h) \leq \epsilon] \geq 1 - \delta$,
- and A runs in time at most $p\big(d, \mathrm{size}(c), 1/\epsilon, 1/\delta\big)$.

Problem 2: PAC Learning Conjunctions

Problem 2: Instance space $X = \{0,1\}^d$,
Concept class $\mathcal{C}_\wedge = \{\, c_{P,N} : P, N \subseteq [d]\}$, where

$$c_{P,N}(x) \;=\; \bigwedge_{i \in P} x_i \;\;\wedge\;\; \bigwedge_{j \in N} (1 - x_j).$$

Representation Size: Each conjunction is encoded by its active literal sets P, N, so $\mathrm{size}(c) = |P| + |N| = O(d)$.

Algorithm: Start with all $2d$ literals in the hypothesis. For each positive example x, drop any literal ℓ for which $\ell(x) = 0$. Use

$$m = \tfrac{2d}{\epsilon} \ln \tfrac{2d}{\delta} \text{ samples.}$$

Problem 2: PAC Learning CONJUNCTIONS

Theorem

Let $\mathcal{C}_\wedge$ be the class of conjunctions of d literals over $X = \{0,1\}^d$. For any $0 < \epsilon, \delta < 1$, sampling $m = \frac{2d}{\epsilon} \ln \frac{2d}{\delta}$ i.i.d. examples from $EX(c, \mathcal{D})$ suffices for a learner to output $h \in \mathcal{C}_\wedge$ such that

$$\Pr\left\{ \Pr_{\mathcal{D}}[h(x) \neq c(x)] \;\leq\; \epsilon \right\} \geq 1 - \delta$$

Key idea: The only way our hypothesis can err is by retaining a literal that should have been dropped. Call such a literal "bad," and then show that every bad literal is eliminated with high probability over our m samples.

Proof sketch: Learning Conjunctions

- **Setup:** Let $c^* \in \mathcal{C}_\wedge$ be the target conjunction. We write $\ell(x) = 1$ if the assignment x makes the the literal ℓ true and vice-versa.
- **Definition of Bad Literal:** A literal ℓ is *bad* if

$$\Pr_{x \sim D}\left[c^*(x) = 1 \wedge \ell(x) = 0\right] \geq \frac{\epsilon}{2d}.$$

- **Bounding Survival Probability:** Any fixed bad literal ℓ survives m positive samples with probability

$$\left(1 - \tfrac{\epsilon}{2d}\right)^m \leq \exp\left(-\tfrac{m\epsilon}{2d}\right).$$

- **Union Bound Over Literals:** There are at most $2d$ literals, so the probability any bad literal remains is

$$2d \exp\left(-\tfrac{m\epsilon}{2d}\right) \leq \delta$$

for $m = \frac{2d}{\epsilon} \ln \frac{2d}{\delta}$. Hence, with probability $1 - \delta$, no bad literals survive and the learned conjunction has error $\leq \epsilon$.

Consistent Learner

Definition

An algorithm A is a *consistent learner* for a concept class $\mathcal{C}$ using $\mathcal{H}$ if, for every target concept $c \in \mathcal{C}_d$, and every finite sample of labeled examples

$$(x_1, c(x_1)),\ (x_2, c(x_2)),\ \ldots,\ (x_m, c(x_m)),$$

the algorithm A, when given this sample, outputs $h \in \mathcal{H}_d$ satisfying

$$\forall\, i \in \{1, \ldots, m\} : \quad h(x_i) = c(x_i).$$

- If A runs in time polynomial in d, m, and the representation size of c, it is an *efficient* consistent learner.
- Consistent learners coincide with Empirical Risk Minimisers under the 0–1 loss.
- By uniform convergence (finite VC-dimension), any efficient consistent learner yields an efficient PAC learner.

Sample Complexity: Upper and Lower Bounds

Upper bound (via VC-dimension). Let a hypothesis class have VC-dimension d_{VC}. Any consistent learner for this class is a PAC learner once the sample size satisfies

$$m \;\geq\; \kappa \left(\frac{1}{\epsilon} \ln \frac{1}{\delta} \;+\; \frac{d_{\mathrm{VC}}}{\epsilon} \ln \frac{1}{\epsilon} \right),$$

for some universal constant $\kappa > 0$.

Lower bound. If the VC-dimension $d_{\mathrm{VC}} > 25$, then any PAC learning algorithm for the class must use at least

$$m \;\geq\; \frac{d_{\mathrm{VC}} - 1}{32\,\epsilon}$$

examples.

Consequences:

- Finite VC-dimension $\iff$ statistical PAC learnability.
- Even with finite VC-dimension, efficient (polynomial-time) PAC learning may fail (e.g. for 3-DNF).

Problem 3: Learning 3-DNFs

- **Instance space:** $X = \{0,1\}^d$, labels $Y = \{0,1\}$.
- **Concept class:**

$$\mathcal{C}_{\mathrm{3DNF}} = \Big\{ T_1 \vee T_2 \vee T_3 : T_i \in \{\text{conjunctions over } d \text{ vars}\} \Big\}.$$

- **Representation size:** Each T_i has $\leq 2d$ literals, so any 3-DNF has size $O(d)$.
- **VC dimension:** $\mathrm{VCdim}(\mathcal{C}_{\mathrm{3DNF}}) \leq 6d$, so information-theoretically PAC learnable, but...

Computational Hardness of Learning 3-DNFs

Theorem (Kearns–Valiant)

Assuming $\mathrm{RP} \neq \mathrm{NP}$, *there is no randomized polynomial-time algorithm that PAC learns 3-DNF.*

Complexity Classes: NP vs. RP

- **NP** (nondeterministic polynomial time): A language L is in NP if there is a poly-time verifier $V(x, w)$ such that

$$x \in L \iff \exists\, w\ (|w| = \mathrm{poly}(|x|) \text{ and } V(x, w) = 1).$$

  Intuition: "yes"-instances have short certificates we can check efficiently.

- **RP** (randomized poly time): A language L is in RP if there is a randomized poly-time algorithm $A(x)$ such that

$$\begin{aligned} x \in L &\implies \Pr[A(x) \text{ accepts}] \geq \tfrac{1}{2}, \\ x \notin L &\implies \Pr[A(x) \text{ accepts}] = 0. \end{aligned}$$

- It is widely believed $\mathrm{NP} \neq \mathrm{RP}$.

Computational Hardness of Learning 3-DNFs

Theorem (Kearns–Valiant)

Assuming $\mathrm{RP} \neq \mathrm{NP}$, *there is no randomized polynomial-time algorithm that PAC learns 3-DNF.*

Proof strategy: Reduce an NP-complete problem (3-COLOUR) to 3-DNF learning:

1. Construct a training set $\mathcal{D}$ from the NP instance so that a consistent 3-DNF exists *iff* the instance is in the language.
2. Show that any efficient PAC learner for 3-DNF would then decide that language in RP time, forcing $\mathrm{RP} = \mathrm{NP}$.

Proof Part I: From NP to PAC Learning 3-Term DNF

In more detail,

- Suppose L is NP-complete. Given input π, we build a finite sample $S = S^+ \cup S^-$ of labeled Boolean vectors.
- We use the uniform distribution D over S, and set PAC parameters $\epsilon = \frac{1}{2|S|}$, $\delta = \frac{1}{2}$.
- If $\pi \in L$, our construction ensures $\exists$ a 3-term DNF ϕ consistent with all of S. Simulating $EX(\phi, D)$ is trivial: just return random $(x, y) \in S$.
- A PAC learner for 3-TERM-DNF, with prob.$\geq 1 - \delta = \frac{1}{2}$, outputs h with $\mathrm{err}(h) \leq \epsilon$. But $\epsilon < 1/|S|$ forces h to be perfect on S, so h is fully consistent.
- Hence if a poly-time PAC learner existed, we'd have an RP procedure for L. Contradiction unless $\mathrm{NP} = \mathrm{RP}$.

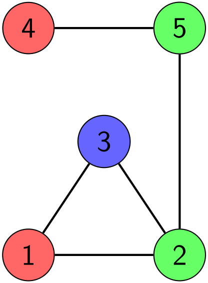
Reduction Setup: 3-COLOURABLE $\rightarrow$ 3-TERM-DNF

A graph $G = (V, E)$ on n vertices is 3-COLOURABLE if $\exists$ mapping $\chi : V \rightarrow \{r, g, b\}$ with $\chi(u) \neq \chi(v)$ for every edge (u, v).

Build a labelled sample $S = S^+ \cup S^- \subseteq \{0,1\}^n \times \{0,1\}$ such that $G \in$ 3-COLOURABLE $\iff \exists\, \phi \in$ 3-TERM-DNF consistent with S.

- **Build S$^+$** For each vertex $i \in V$, define the vector $v(i) \in \{0,1\}^d$
  $v(i)_k = \begin{cases} 0, & \text{if } k = i, \\ 1, & \text{if } k \neq i. \end{cases}$ Label it positive (label 1).
- **Build S$^-$** For each edge $\{i, j\} \in E$, define the vector $e(i,j) \in \{0,1\}^d$
  $e(i,j)_k = \begin{cases} 0, & \text{if } k = i \text{ or } k = j, \\ 1, & \text{otherwise.} \end{cases}$ Label it negative (label 0).

Example: 3-Colouring $\to$ 3-TERM-DNF



Colour classes:
$V_r = \{1, 4\},\ V_g = \{2, 5\},\ V_b = \{3\}$

Positive examples S^+:

$v(i)$	1	2	3	4	5
$v(1)$	0	1	1	1	1
$v(2)$	1	0	1	1	1
$v(3)$	1	1	0	1	1
$v(4)$	1	1	1	0	1
$v(5)$	1	1	1	1	0

Negative examples S^-:

$e(i, j)$	1	2	3	4	5
$e(1, 2)$	0	0	1	1	1
$e(2, 3)$	1	0	0	1	1
$e(3, 1)$	0	1	0	1	1
$e(4, 5)$	1	1	1	0	0
$e(2, 5)$	1	0	1	1	0

The induced 3-term DNF is

$$T_r = x_2 \wedge x_3 \wedge x_5, \quad T_g = x_1 \wedge x_3 \wedge x_4, \quad T_b = x_1 \wedge x_2 \wedge x_4 \wedge x_5, \quad \phi = T_r \vee T_g \vee T_b.$$

If G 3-Colourable, a Consistent 3-TERM-DNF Exists

Claim: If G is 3-colourable, then there exists a 3-term DNF formula consistent with T.

- Suppose $\chi : V \to \{r, g, b\}$ is a valid colouring. Let V_r, V_g, V_b be the vertices of each colour class.
- Define three conjunctions:

  $$T_c = \bigwedge_{i \notin V_c} x_i, \qquad \phi = T_r \vee T_g \vee T_b.$$

  Intuition: T_c accepts exactly those vectors that "omit" only positions in V_c.
- *Check positives:* For any $v(i)$, since $i \in V_c$ for some colour c, all bits except position i are 1, so $T_c(v(i)) = 1$. Hence $\phi(v(i)) = 1$.
- *Check negatives:* Consider edge $e(i,j)$. In each term T_c, at least one of $i, j \notin V_c$, so that coordinate is forced 0 in $e(i,j)$, making $T_c(e(i,j)) = 0$. Hence $\phi(e(i,j)) = 0$.

If a Consistent 3-TERM-DNF Exists, G is 3-Colourable

Claim: If there is a 3-term DNF formula ϕ consistent with T, then G is 3-colourable.

- Let $\phi = T_r \vee T_g \vee T_b$ be any 3-term DNF consistent with S.
- For each vertex i, $v(i) \in S^+$ must satisfy ϕ, so it satisfies at least one term T_c. Assign colour c to vertex i.
- If an edge (i, j) had $\chi(i) = \chi(j) = c$, then both $v(i)$ and $v(j)$ satisfy T_c. But consistency demands $T_c(e(i, j)) = 0$. That forces both x_i and x_j to appear as positive literals in T_c, yet in $e(i, j)$ both bits are 0, so $T_c(e(i, j)) = 1$—a contradiction.
- Hence no edge's endpoints share a colour, so χ is a valid 3-colouring of G.

Statistical Computational Trade-off

- 3-term DNF has no poly-time PAC learner (unless NP=RP).
- Any 3-term DNF $\phi = T_1 \vee T_2 \vee T_3$ can be converted to an equivalent 3-CNF, but this may require $\Theta(d^3)$ clauses.
- The VC-dimension of 3-CNF over d vars is $\Theta(d^3)$, so PAC requires

$$m = O\left(\frac{d^3 + \ln(1/\delta)}{\epsilon}\right) \text{ examples.}$$

- **Homework:** Prove that, with m as above, there is a polynomial-time PAC learner for 3-CNF.
- **Trade-off:** Converting a 3-term DNF to an equivalent 3-CNF trades computational hardness for statistical cost. In particular, by collecting

  $$m = \Theta\left(d^3/\epsilon\right)$$

  examples, you can PAC learn 3-CNFs in polynomial time, thus "beating" the NP$\neq$ RP barrier at the expense of large statistical complexity.