Lecture 10

* HW1 due on Wednesday

RECAP

Theorem: 3-term DNF formulae are not efficiently PAC learnable unless RP=NP.

However a 3-term DNF can be expressed as a 3-CNF.

Theorem: The class of 3-CNF formulae is efficiently PAC learnable.

Important takeaway The choice of representation / hypothesis can make the difference between efficient algorithms & intractability! Going to more expressive hypothesis class (3 term DNF → 3 CNF) makes learning efficient! Statistically, learning over a richer hypothesis class can never help if you know your target hypothesis lies in smaller class, but computationally the picture is very different!

To account for this we distinguish between the

Concept class: The class $\mathcal{C}$ from which the target hypothesis is chosen.

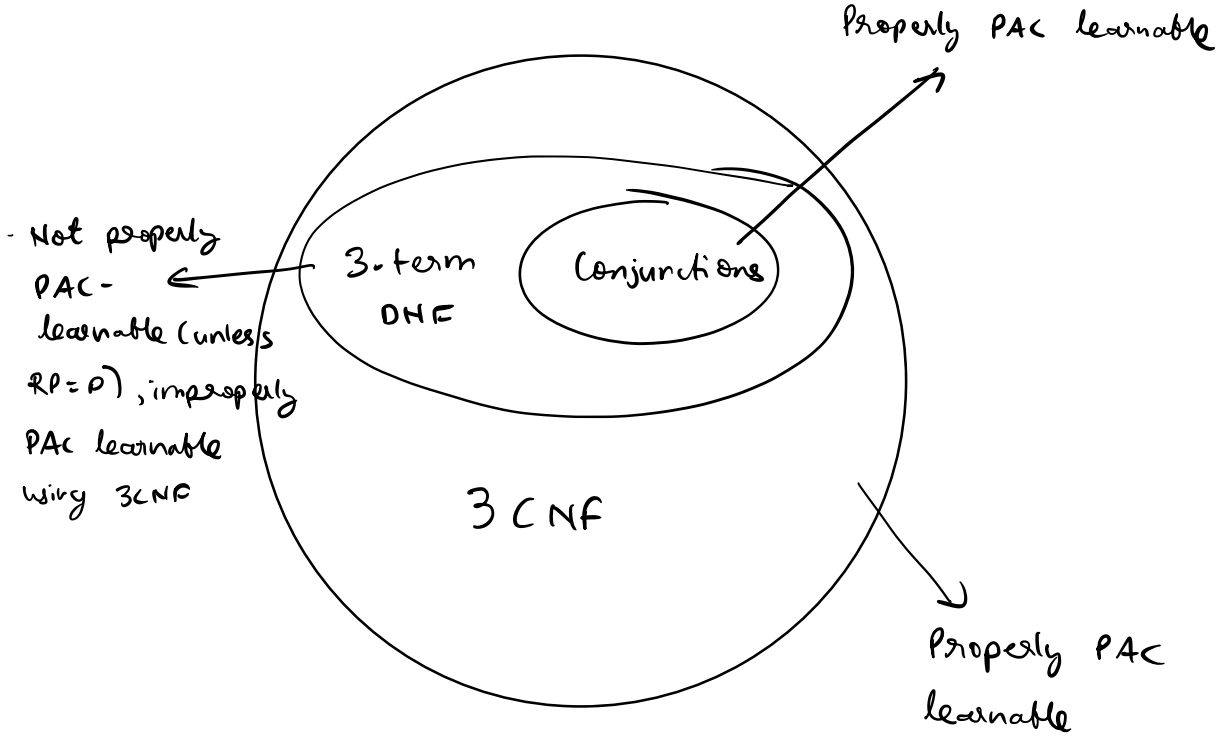
Hypothesis class: The class $\mathcal{H}$ from which the learner chooses its hypothesis.

Revised definition of PAC learning:

If $\mathcal{C}$ is a concept class over the instance space X^d and $\mathcal{H}$ is a hypothesis class over X^d, we say that $\mathcal{C}$ is (efficiently) PAC learnable using $\mathcal{H}$ if our basic definition of PAC learning is met by an algorithm which is allowed to output a hypothesis from $\mathcal{H}$. Here we implicitly assume that $\mathcal{H}$ is atleast as expressive as $\mathcal{C}$ ($\mathcal{C}$ so there is a representation in $\mathcal{H}$ for every function in $\mathcal{C}$).

If $\mathcal{C} = \mathcal{H}$: **proper** learning algorithm

If $\mathcal{C} \subset \mathcal{H}$: **improper** learning algorithm.

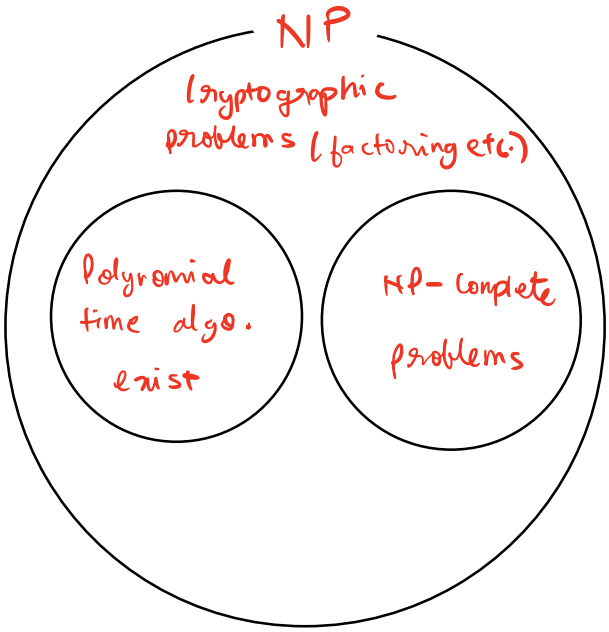


Are there learning problems which are hard, even improperly?

Representation-independent hardness results for learning

Intuitively, results about cryptographic security of some system translate to unlearnability of some corresponding learning tasks (and vice-versa too!)

However, we currently have no way of ensuring that a cryptographic protocol is not breakable, even if we assume $P \neq NP$. Rely on stronger average-case assumptions.



Sketch of idea

Definition (one-way function) A one-way function is a function $f: \{0,1\}^d \to \{0,1\}^d$ that is easy to compute, but hard to invert. More formally, f can be computed in time poly(d) but for any (randomized) polynomial time algorithm A, and for any polynomial $p(\cdot)$,

$$\Pr[f(A(f(x))) = f(x)] < \frac{1}{p(d)},$$

where the probability is taken over x drawn uniformly from $\{0,1\}^d$, and randomness in A.

Definition (trap-door one-way function) A one-way function f is called a trap door one-way function if for some polynomial $p(\cdot)$, $\exists$ a bit string s (called a secret key) of length $(s) \leq p(d)$, such that there is a polynomial time algorithm that for all $x \in \{0,1\}^d$ on input $(f(x), s)$ outputs x.

Candidate trap-door one-way function: Discrete cube root

Let $N = p \cdot q$ be a product of primes p, q (of roughly equal length). Let $f_N(x) = x^3 \bmod N$.

If you know p & q, easy to invert.
But, widely believed to be hard to invert without knowing p, q. Forms basis of RSA cryptosystem (Discrete Cube Root Assumption (DCRA))

For fixed d, let $\mathcal{F}$ be family of all functions

$\mathcal{F} = \{ f_N(x) : N = p \cdot q$, primes p, q, length (p), length $(q) \leq p(d)\}$
(for some polynomial $p(\cdot)$)

Under DCRA, given N & $y = f_N(x)$ for some random $x \in \{0,1\}^d$, hard to compute x (cannot do in polynomial time).

Can we convert this to learning problem s.t. a successful learner can invert $f_N(x)$?

$$\mathcal{C} = \{ f_N^{-1}(y) : N = p \cdot q, \text{ primes } p, q, \text{ length}(p, q) \leq p(d) \}$$

Can we generate examples

$$(y_1, f_N^{-1}(y_1))$$

(y_1: input; $f_N^{-1}(y_1)$: output / label)

$$(y_2, f_N^{-1}(y_2))$$

$$\vdots$$

$$(y_n, f_N^{-1}(y_n)) \; ?$$

Yes! Generate examples by sampling $x_i \in \{0,1\}^d$ and obtaining labelled examples $(f_N(x_i), x_i)$ [this is easy to do, since computing $f_N(x_i)$ for any x_i is easy]

We can show that $f_N(x)$ is a bijection, $\therefore$ above sampling process has same distribution as choosing y_i at random & then assigning the label $f_N^{-1}(y_i)$.

Note following discussion in class:

As per our one-way function definition, it's not necessary to ensure the distribution over y_i is uniform, as long as the distribution over x_i is uniform. However, the above analysis shows the problem is hard even if the learning algo. says it only works for the uniform distribution over inputs y_i.

If f exists an algorithm for solving the learning problem to error $\leq \varepsilon$, then that algorithm can be used to invert y drawn uniformly at random from $\{0,1\}^d$ with failure probability $\leq \varepsilon$, therefore violating DCRA.

$\therefore$ learning is impossible (even, improperly).

Also, note that statistically we're fine. The concept class C is parameterized by the secret key p, q which have length $p(d)$.

$$\therefore |C| \leq 2^{2p(d)}$$

$\therefore$ learnable with $O(\log(|C|) = O(p(d))$ samples.

$$f: \{0,1\}^d \to \{0,1\}^d \qquad X \to Y$$

f is a bijection

$\Rightarrow$ uniform over $X \Rightarrow$ uniform over Y.

Some modern results on hardness of learning.

Recent results which show representation-independent hardness for hypothesis encountered more frequently in practice.

Consider the class of halfspaces / linear threshold functions (LTFs)

$$\mathcal{H} : \{ x \to \text{sign}(w^T x) : w \in \mathbb{R}^d \}$$

Thm The class of halfspaces is efficiently PAC learnable.

However agnostically learning halfspaces, even improperly, is hard (under complexity assumption.

Informal (Theorem) (Daniely '16)

Under "hardness of refuting random k-XOR",
for any constant C, there is no poly-time algo. that given a sample of d^C points in $\{-1, 1\}^d$ can distinguish whp,

(a) case where labels are uniformly random coin flips.
(b) the case that $\exists$ a LTF with error at most 10%. (can replace 10% by any constant > 0).

"Refuting random k-XOR"

k-XOR : AND of XOR (Parity)

Parity / XOR : outputs 1 iff the number of 1s in input is odd.

2-XOR : $(x_1 \oplus x_2) \wedge \neg(x_2 \oplus x_3) \wedge (x_4 \oplus x_5) \wedge \ldots$

($x_1 \oplus x_2$: XOR of 2 literals; n formulae)

Claim : If a satisfying assignment exists, easy to find using Gaussian elimination.

Proof

$$\begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 \cdots & 0 \\ 0 & 0 & 0 & 1 & 1 \cdots 0 \\ & & \vdots & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_d \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{pmatrix}$$

linear system over $\mathbb{F}_2$. ∎

However, if no satisfying assignment exists, finding an assignment that satisfies as many XORs as possible is NP-Hard (Hastad '01).

Stronger assumption

Given $n \leq d^{\sqrt{k}\log k}$ terms, it is hard to distinguish

a) the case of a random k-XOR formulae with n constraints.

b) a formula of k-XOR constraints that has a satisfying assignment satisfying at least $\frac{1}{2}$ of the constraints.

Note: a random formula with $n \geq \frac{d}{\varepsilon^2}$ k-XOR constraints will whp have the property that no assignment satisfies more than $\left(\frac{1}{2} + \varepsilon\right)$ fraction.

Why? Fix any assignment to x and then draw n constraints at random (select k variables at random, then randomly choose to negate or not). Each constraint is satisfied w.p. $1/2$.

By Hoeffdings, $\Pr[x \text{ satisfies} \geq \frac{1}{2} + \varepsilon \text{ fraction}]$

$$\leq e^{-2n\varepsilon^2}$$

$\therefore$ By union bound over all 2^d possible assignments, prob that any assignment satisfies $\geq \frac{1}{2} + \varepsilon$ is $\leq 2^d \cdot e^{-2n\varepsilon^2} \leq e^{-d}$ (for $n = \frac{d}{\varepsilon^2}$).

This shows that agnostic learning gets hard pretty quickly. Agnostic learning is a very strong requirement.

Relaxations

→ Learn over natural distributions ! Real-world distributions are maybe nice, not worst-case.

For e.g.

Thm. (Kalai-Klivans-Mansour-Servedio '05) The class of halfspaces is efficiently agnostically learnable under the uniform distribution $\{\pm 1\}^d$, or unit sphere, or the Gaussian distribution over $\mathbb{R}^d$.

→ Agnostic model: the points which are incorrectly labelled by the target $c \in \mathcal{C}$ could have arbitary labels (worst-case noise).

Relaxation: Consider random noise, instead of worst-case noise.

Thm: (Blum-Frieze-Kannan-Vempala '96)
Halfspaces are efficiently PAC learnable under random noise.

Major challenge: reconcile practical success of neural networks / gradient based approaches with worst-case hardness. What about these problems makes them easy?