# Lecture 9

* HW1 due in 1 week.

## RECAP

**Definition (Efficient PAC learning)**

A hypothesis class $\mathcal{H}$ is PAC-learnable if $\exists$ a learning algorithm $A$ with the following property: for every $\epsilon, \delta \in (0,1)$, every distribution $D$ over $\{X\}^d$ (where $X$ is typically $\{0,1\}$ or $\mathbb{R}$), and every $h^* \in \mathcal{H}$, if $A$ is given access to example oracle $EX(h^*, D)$ and $\epsilon, \delta$, with probability $1-\delta$, it outputs a hypothesis $h \in \mathcal{H}$ with $R(h) \leq \epsilon$.

$\mathcal{H}$ is <u>efficiently</u> PAC learnable if running time of $A$ is polynomial in $d, \frac{1}{\epsilon}, \frac{1}{\delta}$.

**Theorem** The class of conjunctions on Boolean literals is efficiently PAC learnable.

# Intractability of learning 3-Term DNF

What is a 3-term DNF (Disjunctive Normal Form)?

3-Term-DNF$_d$ = $\{ T_1 \vee T_2 \vee T_3 \mid T_i$ is a conjunction on $\{x_1, \ldots, x_d\}\}$

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

NP: Problems whose solutions are easily verifiable

$$3SAT_d = (x_1 \vee \bar{x_2} \vee x_5) \wedge (x_5 \vee x_6 \vee \bar{x_7}) \wedge \vdots \wedge (x_{d-2} \vee \overline{x_{d-1}} \vee x_d)$$

RP: C is in RP if $\exists$ poly-time algo. with one-sided error.

3SAT instance is satisfiable, output TRUE w.p. $\geq 2/3$.
" not satisfiable, always output FALSE.

NP-Complete: C is in NP, all problems in NP reducible to P in polynomial time.

---

# Proof of hardness of learning 3-term DNF.

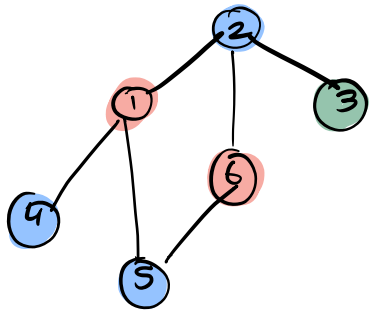
**Intuition:** Reduce a NP-complete problem to the problem of learning 3-term DNF. The key property we want from mapping is that the answer to decision problem is "yes" if and only if a set of labelled examples is consistent with some hypothesis $h \in \mathcal{H}$.

**Definition:** Let $U = \{(a_1, y_1), \ldots, (a_n, y_n)\}$ be labelled set of instances. Let $h$ be any hypothesis. We say that $h$ is consistent with $U$ if $\forall\, i \in [n]$, $h(a_i) = y_i$.

This recipe of showing hardness is useful quite generally. We now state the NP-complete problem we use.

**PROBLEM:** Graph 3-COLORING

Given an undirected graph $G = (V, E)$ with vertex set $V = \{1, \ldots, d\}$, is there an assignment from every vertex $v \to \{R, B, G\}$ s.t. for every edge $e \in E$, the endpoints of $e$ are assigned different colors?




Graph 3-COLORING is NP-complete.

## Reduction

Graph $G$ $\xrightarrow{\text{polynomial time}}$ Set $U_G$ of +ve & -ve labelled examples

We will show that given $G$, we can construct $U_G$ s.t. $U_G$ is consistent with some $h \in \mathcal{H}$ if and only if $G$ is 3-colorable.

Let's see why this is sufficient.

Define :

$D$ : uniform on $U_G$

$EX(h^*, D)$ : Pick a point uniformly at random from $U_G$.

$\delta : 1/3$

$\varepsilon = \frac{1}{2|U_G|}$

## ALGORITHM.

1. Given instance of 3-COLOR, construct set $U_G$.
2. Use PAC-learning algorithm $A$ for 3-term DNF with $Ex(h^*, D)$, $\delta = 1/3$, $\varepsilon = \frac{1}{2|U_G|}$.
3. Let $h$ be the 3-term DNF returned by $A$.
4. If $h$ is consistent with $U_G$
5. return "yes"
6. else
7. return "no"

Note that $D$ is uniform over set of size $|U_G|$.

$\therefore \varepsilon = \frac{1}{2|U_G|}$ & $R(h) < \varepsilon$

$\Rightarrow h$ is consisent with set $U_G$.

If A succeeds, it must return consistent hypothesis.

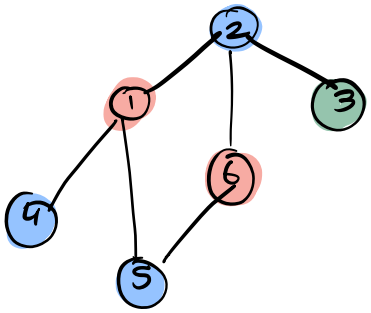
Therefore, all that remains is to show we can construct $U_G$ s.t. there is some consistent hypothesis $h \in H$ if and only if $G$ is 3 colorable.

## Constructing $U_g$

$U_g = U_g^+ \cup U_g^-$

$U_g^+$ : +ve examples

$U_g^-$ : -ve examples




$U_g^+$, $|U_g^+| = |V|$

$(v(1), +1) = ((0,1,1,1,1,1), +1)$

$(v(2), +1) = ((1,0,1,1,1,1), +1)$

$\vdots$

$(v(6), +1) = ((1,1,1,1,1,0), +1)$

$U_g^-$, $|U_g^-| = |E|$

$(e(12), -1) = ((0,0,1,1,1,1), -1)$

$(e(14), -1) = ((0,1,1,0,1,1), -1)$

$\vdots$

$(e(56), -1) = ((1,1,1,1,0,0), -1)$

**Part 1:** 3-colorable $\Rightarrow \exists$ a consistent 3-term DNF

3-term DNF $\phi = T_R \vee T_B \vee T_G$

R: set of all vertices colored Red.
B: } similarly
G: }

$T_R$: conjunctions of all variables whose index <u>does not</u> appear in R.

$$T_R = x_2 \wedge x_3 \wedge x_4 \wedge x_5$$

Similarly for $T_B$ & $T_G$,

$$T_B = x_1 \wedge x_3 \wedge x_6$$

$$T_G = x_1 \wedge x_2 \wedge x_4 \wedge x_5 \wedge x_6$$

* For each $i \in R$
Example $v(i)$ must satisfy $T_R$ because $x_i$ does not appear in $T_R$.

* Further no $e(i,j) \in U_y^-$ can satisfy $T_R$.
→ both $i$ & $j$ cannot be colored red, one if $x_i$ or $x_j$ must appear in $T_R$.
But $e(i,j)$ has 0 values for both $x_i$ & $x_j$.
∴ $T_R$ cannot be satisfied by $e(i,j)$.

∴ $\phi = T_R \vee T_B \vee T_G$ is consistent.

**Part 2:** 3-term DNF => 3-colorable

$\phi = T_R \vee T_B \vee T_y$

for a vertex $i$ : if $v(i)$ satisfies $T_R$, color $i$ Red
" $T_B$, " Blue
" $T_G$, " Green

(Break any ties arbitrarily)

Since formula is consistent, every $v(i)$ must satisfy at least one of $T_R, T_B, T_y$

$\therefore$ every vertex is assigned a color.

**Claim:** coloring is valid 3-coloring.

**Proof:** If $i$ & $j$ ($i \neq j$) are assigned the same color (say Red), both $v(i)$ & $v(j)$ satisfy $T_R$.

ith coordinate

$v(i) = (1, \ldots, 0, \ldots 1)$

$v(j) = (1, \ldots, 1, \ldots, 0, \ldots 1)$

$\therefore$ Since ith bit of $v(i)$ is 0
& ith bit of $v(j)$ is 1
$\Rightarrow$ neither $x_i$ nor $\bar{x}_i$ appears in $T_R$.

$e(i,j) = (1, \ldots 0, \ldots, 0, \ldots 1)$

$e(i,j)$ & $v(j)$ differ only in ith coordinate

$\therefore$ if $v(j)$ satisfies TR, so does $e(i,j)$

$e(i,j)$ should be labelled +ve

$\Rightarrow$ $e(i,j) \notin U_{\bar{G}}$

$\therefore$ $(i,j) \notin E$.

■

## Using 3-CNF formulae to avoid intractability

So far, we've restricted the learning algorithm to output a hypothesis from the same class it was learning.

What if we allow the algorithm to output a hypothesis from a different, more expressive class?

Distributive law:

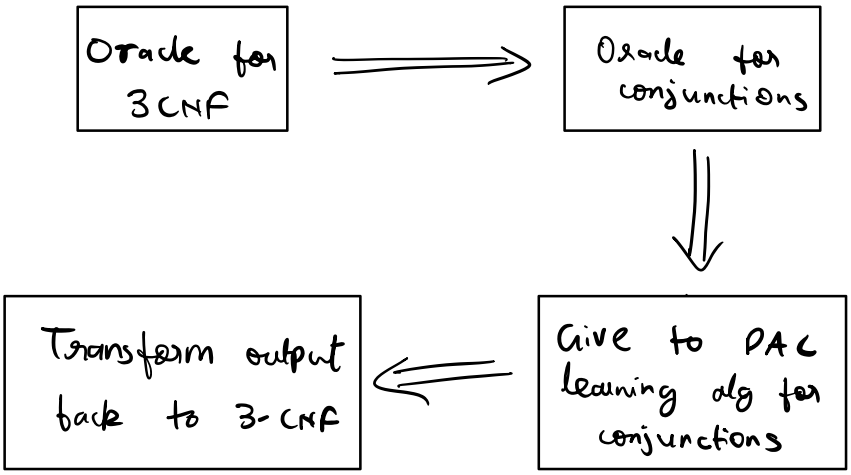
$$(u \wedge v) \vee (w \wedge x) = (u \vee w) \wedge (u \vee x) \wedge (v \vee w) \wedge (v \vee x)$$

We can represent any 3-term DNFd $\phi = T_1 \vee T_2 \vee T_3$ by a 3-CNFd $\psi = \bigwedge_{\substack{u \in T_1 \\ v \in T_2 \\ w \in T_3}} (u \vee v \vee w)$

This is a 3 CNF (Conjunctive Normal Form).

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

**Proof**: We will reduce the problem of PAC learning 3-CNF formulae to the problem of PAC learning conjunctions.




**Idea**: Regard 3CNF formulae as a conjunction over a new & larger variable set.

**Transformation**: For every triple of literals $u, v, w$ over the original variable set $\{x_1, \dots, x_d\}$ the new variable set contains a variable $y_{u,v,w} = u \vee v \vee w$. When $u = v = w$ $y_{u,v,w} = u$, so all original variables are in new set.

$$\#\text{ of variables} = (2d)^3, \quad O(d^3)$$

**Transforming oracles**: For any assignment $a \in \{0,1\}^d$ to original variable, we can in $O(d^3)$ time compute assignment to new variable set $\{y_{u,v,w}\}$.

Note that any 3-CNF over $x_1, \ldots, x_d$

$\equiv$ a simple conjunction over $\{y_{u,v,w}\}$.

(replacing any clause $u \vee v \vee vw$ by $y_{u,v,w}$)

$\therefore$ we can run algorithm for conjunctions.
We can transform the output $h'$ of the algorithm back to a 3-CNF $h$, by expanding any occurrence of $y_{u,v,w}$ by $(u \vee v \vee vw)$.

**Claim** If $h^*$ and $D$ are the target 3-CNF formula and the distribution over $\{0,1\}^d$, and $h^{*\prime}$ and $D'$ are the corresponding conjunction over $y_{u,v,w}$ and the corresponding distribution over $y_{u,v,w}$, then if $h'$ has error $\leq \varepsilon$ with respect to $h^{*\prime}$ and $D'$, then $h$ has error $\leq \varepsilon$ with respect to $h^*$ and $D$.

**Proof** Our transformation of instances is one $\to$ one

if $a_1 \to a_1'$ & $a_2 \to a_2'$

$a_1 \neq a_2 \Rightarrow a_1' \neq a_2'$

$\therefore$ our reduction is done

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!