A General Model

- Input/instance space X (e.g. $\mathbb{R}^2$)
- Target concept/function
  $c \subseteq X$ (positive ex's)
  $c : X \to \{0,1\}$ or $\{+,-\}$
- Concept class $\mathcal{C}$, (quite) restricted
  E.g. rectangles in $\mathbb{R}^2$

- We assume $c \in \mathcal{C}$:

  $\mathcal{C}$ known to learner

  c unknown

- Input distribution P over X

  unknown and arbitrary

- Learner given access to labeled samples

  $\langle x, c(x) \rangle$, $x \sim P$

Definition $\mathcal{C}$ is **PAC learnable**

if $\exists$ learning algo L s.t.

$\forall c \in \mathcal{C}$ (target)

$\forall P$ over X (dist.)

$\forall \varepsilon, \delta > 0$:

- With prob. $\geq 1-\delta$, L outputs $h \in \mathcal{C}$ s.t. $\varepsilon(h) \leq \varepsilon$

  $\left(\varepsilon(h) \triangleq \Pr_{x \sim P}[h(x) \neq c(x)]\right)$

- Sample size & runtime of L are "efficient", e.g. polynomial in $1/\varepsilon$, $1/\delta$ and...

- ... "complexity" of X and C:

- e.g. $\mathbb{R}^2$ vs $\mathbb{R}^d$, must depend on d, ideally polynomially or better
- e.g. "size" of c:
  - # nodes in decision tree
  - # weights in neural net
  - ⋮
- We'll be precise as needed

- What classes $\mathcal{C}$ are PAC-learnable?
- What classes are (provably) not PAC, and why?
- What are general algo tools/reductions?
- What are interesting variations on model?

<u>Theorem</u> The class $\mathcal{C}$ of axis-aligned rectangles in $\mathbb{R}^2$ is PAC learnable.

Let's look at another e:

conjunctions of
Boolean features.

- Domain $X = X_n = \{0,1\}^n$

- conjunctions: e.g.

$c(x) = x_1 \neg x_3 x_4 \quad n = 6$

$c(110100) = 1$

$c(111111) = 0$

- generalize & specialize rectangles in $\mathbb{R}^2$:

$2 \to n$

$x_i \in [a,b] \to x_i = 0, 1, *$

Let C be class of conjunctions over $\{0,1\}^n$.

- What is $|C|$?
- Is C PAC learnable in time polynomial in $1/\varepsilon$, $1/\delta$, and n?
- Algorithm?

- Initial hypothesis:

$h \leftarrow x_1 \neg x_1 x_2 \neg x_2 \cdots x_n \neg x_n$

- Given $\langle x, y \rangle$, $x \sim P$:

  $y = 0 \rightarrow$ ignore

  $y = 1 \rightarrow$ delete contradictions from h

- E.g. on $1\ 1\ 0\ 1 \cdots$, $y = 1$:

  delete $\neg x_1$ $\neg x_2$ x_3 $\cdots$

- Every deletion proven
  $\Rightarrow$ most specific h
  $\Rightarrow$ consistent with data so far
- Only mistake: fail to delete some $x_i, \neg x_i$ that is harmful ("bad event")
- Let's analyze for some $z \notin c$ ($z = x_i$ or $\neg x_i$)

- Define

$$q(z) = \Pr_{x \sim P}[c(x)=1 \ \& \ z=0 \text{ in } x]$$

$=$ deletion prob. of z

- $\varepsilon(h) \le \sum_{z \in h} q(z)$

- call z bad if $q(z) \ge \varepsilon/2n$

- h has no bad $z \Rightarrow \varepsilon(h) \le \varepsilon$

- For fixed bad z:

  prob. z not deleted in m $x \sim P$

  $$\leq (1-\varepsilon/2n)^m \quad \text{indep.}$$

- Prob. some/any bad z not deleted

  $$\underbrace{\leq 2n(1-\varepsilon/2n)^m}_{\text{Set } \leq \delta \text{, solve for } m} \quad \text{union bound}$$

Algo is PAC for

$$m \geq \frac{2n}{\varepsilon}\left(\ln(2n) + \ln(1/\delta)\right)$$

Running time $O(m \cdot n)$

Q: Even stronger property of algo?

A (slight?) generalization:

$\mathcal{C}$ = 3-term DNF

- Now target $c = T_1 \vee T_2 \vee T_3$
- Each T_i a conjunction over $\{0,1\}^n$,

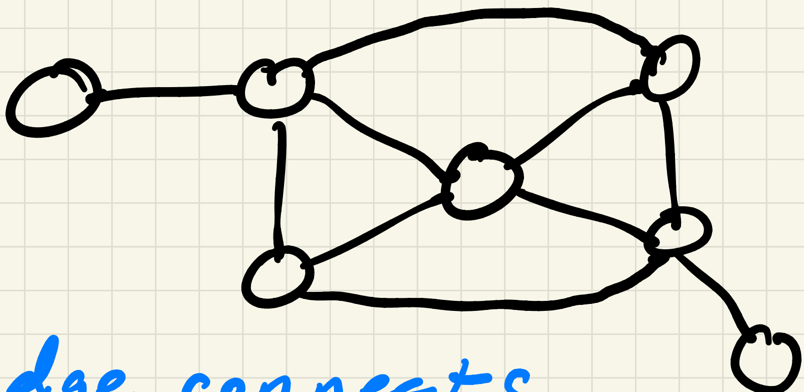
e.g. $c = \underbrace{x_1 \neg x_5}_{(T_1)} \vee \underbrace{x_1 x_2 x_7}_{(T_2)} \vee \underbrace{\neg x_1 \neg x_2}_{(T_3)}$

$$c(x) = T_1(x) \vee T_2(x) \vee T_3(x)$$

Claim: If 3-term DNF is PAC learnable, then NP = RP.

The Graph 3-Coloring Problem

<u>Input</u>: undirected graph/network G: e.g.

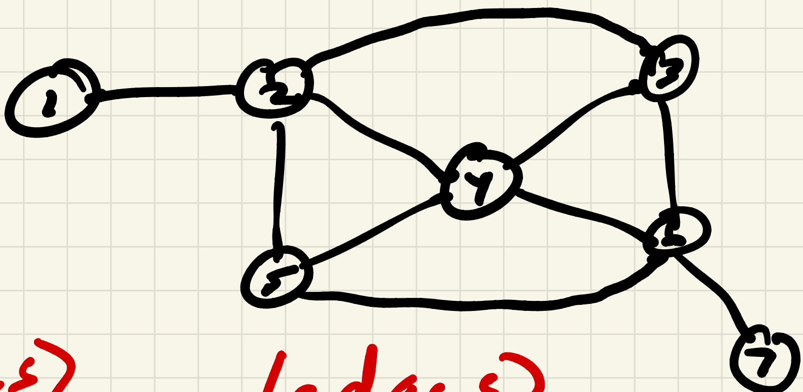


every edge connects **different** colors

<u>Output</u>: "yes" if G 3-colorable, "no" else.

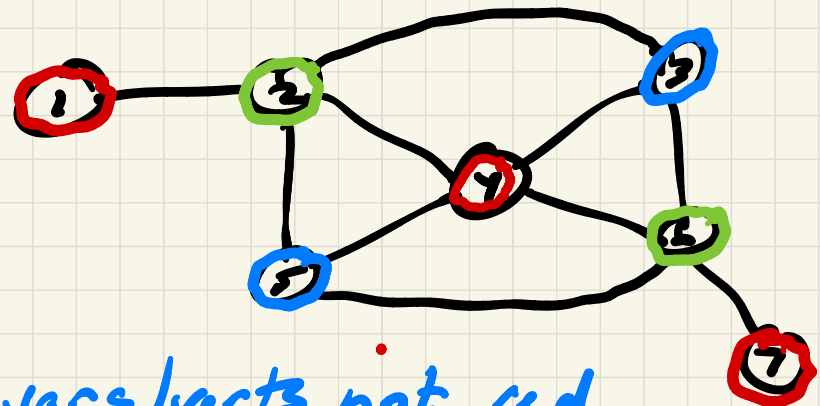
An **NP-complete** problem.

Encode as 3-term DNF learning problem: create labeled sample S.



(vertices) + ex's	(edges) − ex's
0111111, +	0011111, −
1011111, +	1001111, −
1101111, +	1010111, −
⋮	1011011, −
1111110, +	⋮
	1111100, −

Suppose G is 3-colorable:



T_R = all vars/verts not red
$= x_2 x_3 x_5 x_6$

$T_B = x_1 x_2 x_4 x_6 x_7$

$T_G = x_1 x_3 x_4 x_5 x_7$

<u>Claim</u>: $T_R \vee T_B \vee T_G$ consistent with S.

Now suppose some
$T_R \vee T_B \vee T_G$ consistent with S.

- Define color of var/vertex i to be the T that satisfies $\langle 11 \cdots 1 \underset{i}{0} 1 \cdots 1, + \rangle$
- If i & j both R and $(i,j) \in G$:

$$\left.\begin{array}{l} — \overset{i}{0} — \overset{j}{1} — ,+ \\ — 1 — 0 — ,+ \end{array}\right\} \text{sat. } T_R$$

$\Rightarrow$ none of $x_i, \neg x_i, x_j, \neg x_j \in T_R$

$\Rightarrow$ $— 0 — 0 —$ sats T_R

$\Rightarrow\Leftarrow$ with consistency!

$\therefore$ G is 3-colorable.

So G is 3-colorable

$$\Longleftrightarrow$$

$S = S(G)$ is consistent with some 3-term DNF.

$$\Longrightarrow$$

So what?

What does this have to do with PAC?

Where are our friends P, ε, δ?

Need to simulate them.

- Let A be a black-box PAC algo.

- Given $G \to S = S(G)$
- Let P be uniform over S

  $|S| = \#\text{vertices} + \#\text{edges}$

  $= \text{size of } G$

- Choose $\varepsilon < 1/|S|$

  and any small $\delta > 0$

- Run A on P, ε, δ

  $\to$ test output $T_1 \vee T_2 \vee T_3$ for consistency

- G 3-colorable $\Rightarrow$ w.p. $\geq 1-\delta$, A output consistent hypothesis.
- G not 3-colorable $\Rightarrow$ w.p. 1, A fails to output consistent hypo.

$\therefore$ PAC learning 3-term DNF $\Rightarrow NP = RP$.

Moral: As mysterious & powerful as ML can sometimes seem, it obeys same "computation laws" as any other algorithmic problem/framework.

But now let's weasel out of this result.

A little Boolean algebra.

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{\substack{u \in T_1 \\ v \in T_2 \\ w \in T_3}} (u \vee v \vee w)$$

(3-term DNF) (3CNF)

- e.g. $T_1 = 1 \Rightarrow$ each $u \in T = 1$
  $\Rightarrow RHS = 1$
- $LHS = 0 \Rightarrow$ some $u, v, w = 0$
  $\Rightarrow RHS = 0$

$\therefore$ 3-term DNF $\subseteq$ 3CNF

$(\neq)$

Create meta-features:
("linearization")

- $z(u,v,w) \triangleq u \vee v \vee w$

- # meta-features $\sim \binom{2n}{3} = O(n^3)$

- RHS on last page is a conjunction over z's

- given $x \in \{0,1\}^n$, expand:

$$\underset{n}{x} \rightarrow \underset{\sim n^3}{z(x)}$$

So: 3-term DNF

is PAC-learnable

... "by" 3CNF.

We have circumvented the hardness result by enlarging our hypothesis class.

Notes:

- We are using $\forall P$ part of PAC defn!
- E.g. P uniform over $\{0,1\}^n \not\Rightarrow P'$ uniform over z's
- Output of conjuncts algo may not be = any 3CNF
- Hypo. representation matters
- "Overcompleteness"

Definition $\mathcal{C}$ is PAC learnable by $\mathcal{H}$

if $\exists$ learning alg. L:

$\forall c \in \mathcal{C}$ (target)

$\forall P$ over X (dist.)

$\forall \varepsilon, \delta > 0$:

- With prob. $\geq 1-\delta$, L outputs $h \in \mathcal{H}$ s.t. $\varepsilon(h) \leq \varepsilon$

$$\varepsilon(h) \triangleq \Pr_{x \sim P}[h(x) \neq c(x)]$$

- Sample size & runtime of L are "efficient", e.g. polynomial in $1/\varepsilon$, $1/\delta$ and...

Recap:

- PAC learning 3-term DNF by 3-term DNF is NP-hard.
- 3-term DNF is PAC learnable by 3CNF

Q: Can we ever be sure any C is truly hard to learn?