A General Model

- Input/instance space $X$ (e.g. $\mathbb{R}^2$)

- Target concept/function $c \subseteq X$ (positive ex's)
  $C : X \rightarrow \{0, 1, 3 \} \cup \{3^+, -3\}$

- Concept class $C$, (quite) restricted
  E.g. rectangles in $\mathbb{R}^2$
• We assume $C \in C$:
  - Known to learner
  - Unknown

• Input distribution $P_{over\ X}$
  - Unknown and arbitrary

• Learner given access to labeled samples $\{(x, c(x)) : x \sim P\}$
Definition: \( C \) is PAC learnable if there exists a learning algo \( L \) s.t.

\[
\forall c \in C \quad \text{(target)}
\]

\[
\forall P \text{ over } X \quad \text{(dist.)}
\]

\[
\forall \varepsilon, \delta > 0:
\]

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

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

- Sample size and runtime of \( L \) are "efficient", e.g., polynomial in \( \frac{1}{\varepsilon}, \frac{1}{\delta} \) and...
...“complexity” of X and c:

- e.g. $R^2$ vs $R^d$, 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 $C$ are PAC-learnable?
• What classes are (provably) not PAC, and why?
• What are general algo tools/reductions?
• What are interesting variations on model?
Theorem: The class $C$ of axis-aligned rectangles in $\mathbb{R}^2$ is PAC learnable.
Let's look at another example:

- **Conjunctions of Boolean features.**

- **Domain** \( X = X_n = \{0, 1\}^n \)

- **Conjunctions:** e.g.,
  \[
  c(x) = x_1 \land x_3 \land x_4 \quad n = 6
  \]
  \[
  c(110100) = 1
  \]
  \[
  c(111111) = 0
  \]

- **Generalize & specialize rectangles in \( \mathbb{R}^2 \):**
  \[
  2 \rightarrow n
  \]
  \[
  x_i \in [a, b] \rightarrow x_i = 0, 1, \ast
  \]
Let $C$ be class of conjunctions over $\{0,1\}^n$.

- What is $|C|$?
- Is $C$ PAC learnable in time polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}$, and $n$?
- Algorithm?
- Initial hypothesis:
  \[ h \leftarrow x_1 x_1 x_2 x_2 \cdots x_n x_n \]

- Given \( (x, y), x \in P \):
  \[ y = 0 \rightarrow \text{ignore} \]
  \[ y = 1 \rightarrow \text{delete contradictions from } h \]

- E.g., on \( 1101 \cdots \), \( y = 1 \):
  delete \( x_1, x_2, x_3 \cdots \)
• Every deletion proven
  → most specific h
  → consistent with data so far
• Only mistake: fail to delete some \( x_i, \bar{x}_i \) that is harmful ("bad event")
• Let's analyze for some \( z \in C \)
  \( (z = x_i \text{ or } \bar{x}_i) \)
Define
\[ q(z) = \Pr_{x \sim p} \left[ c(x) = 1 \land z = 0 \text{ in } x \right] \]
= deletion prob. of \( z \)

\[ e(h) \leq q(z) \leq q(h) \]
\[ z \in h \]

call \( z \) bad if \( q(z) \geq \frac{1}{2n} \)

\[ h \text{ has no bad } z \Rightarrow e(h) \leq \frac{1}{3} \]
For fixed bad $z$:

- prob. $z$ not deleted
  \[
  \Pr[E \not\subseteq \delta] \leq (1 - \frac{\epsilon}{2n})^m \text{ indep.}
  \]

- Prob. some/any bad
  $z$ not deleted

\[
\leq 2n \left(1 - \frac{\epsilon}{2n}\right)^m \text{ union bound}
\]

Set $\delta = \delta$, solve for $m$
Algo is PAC for

\[ m \geq \frac{2^n}{\varepsilon} \left( \ln(2n) + \ln(\frac{1}{\varepsilon}) \right) \]

Running time \( O(m \cdot n) \)

Q: Even stronger property of algo?
A (slight?) generalization: $C = 3$-term DNF

Now target $C = T_1 \lor T_2 \lor T_3$

Each $T_i$: a conjunction over $X_i \subseteq \{0,1\}^n$

e.g. $C = X_1 \lor X_5 \lor X_1 X_2 X_7 \lor \neg X_1 \neg X_2$

\(C(x) = T_1(x) \lor T_2(x) \lor T_3(x)\)

Claim: If 3-term DNF is PAC learnable, then $NP = RP$. 
The Graph 3-Coloring Problem

Input: undirected graph/network \( G \): e.g.

every edge connects different colors

Output: "yes" if \( G \) 3-colorable, "no" else.

An NP-complete problem.
Encode as 3-term DNF learning problem:
create labeled samples $S$.

\[
\begin{align*}
\text{(vertices)} & \quad + \, \text{ex's} \\
0111111, & + \\
1011111, & + \\
1101111, & + \\
\vdots & \\
1111110, & + \\
\end{align*}
\]

\[
\begin{align*}
\text{(edges)} & \quad - \, \text{ex's} \\
0011111, & - \\
1001111, & - \\
1010111, & - \\
1011011, & - \\
\vdots & \\
1111100, & - \\
\end{align*}
\]
Suppose $G$ is 3-colorable:

$T_R = \text{all vars/verts not red} = x_2 x_3 x_5 x_6$

$T_B = x_1 x_2 x_4 x_6 x_7$

$T_G = x_1 x_3 x_4 x_5 x_7$

Claim: $T_R \lor T_B \lor T_G$ consistent with $S$. 
Now suppose some $TR \vee Tb \vee Tc$ consistent with $S$.

- Define color of vertex $i$ to be the $T$ that satisfies $<11\ldots 01\ldots 1>^i$.
- If $i \not= j$ both $R$ and $(i,j) \in G$:

$$
\begin{align*}
\neg b - i^- \rightarrow & >^+ \\
\rightarrow \neg i - o^- \rightarrow & >^+ \\
\Rightarrow \text{none of } x_{i,j}, x_{i,i}, x_{i,j}, x_{j,j} \in Tr \\
\Rightarrow \neg 0 - 0 - \text{ sats } Tr \\
\Rightarrow & \text{ with consistency!} \\
\therefore \ G \text{ is 3-colorable.}
\end{align*}
$$
So $G$ is 3-colorable if and only if

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

So what?
What does this have to do with PAC?
Where are our friends $P, E, \delta$?
Need to simulate them.
• Let $A$ be a black-box PAC algo.
• Given $G \rightarrow S = S(G)$
• Let $P$ be uniform over $S$
  \( |S| = \#\text{vertices} + \#\text{edges} \)
  \( = \text{size of } G \)
• Choose $\varepsilon \leq 1/|S|$
  and any small $\delta > 0$
• Run $A$ on $P, \varepsilon, \delta$
  \( \rightarrow \text{test output } T_1, T_2, T_3 \)
  \( \rightarrow \text{consistency} \)
• $G$ 3-colorable $\Rightarrow$ w.p. $\geq 1 - 8$, $A$ output consistent hypothesis.

• $G$ not 3-colorable $\Rightarrow$ w.p. 1, $A$ fails to output consistent hypoth.

∴ 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.

\[ T_1 \lor T_2 \lor T_3 = \bigwedge (u \lor v \lor w) \]

\((3\text{-term DNF}) u \in T_1 \quad (3\text{CNF}) v \in T_2 \quad w \in T_3\)

\cdot e.g. \( T_1 = 1 \Rightarrow \text{each } u \in T = 1 \Rightarrow \text{RHS} = 1\)

\cdot \text{LHS} = 0 \Rightarrow \text{some } u, v, w = 0 \Rightarrow \text{RHS} = 0\)

\therefore 3\text{-term DNF} \preceq 3\text{CNF} \]

\(\nabla\)
Create meta-features: ("linearization")

- $\exists (u, v, w) \triangleq u \lor v \land w$
- \#meta-features
- $\sim (2^n) = O(n^3)$
- RHS on last page is a conjunction over $\exists$'s
- given $x \in \exists_0, 1, 3$,
  expand:
  $x \rightarrow \exists(x)
  \lor n \sim n^3$
So: 3-term DNF is PAC-learnable
...“by” 3CNF.

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

- We are using HP part of PAC dim.
- E.g. $P$ uniform over $\{0,1\}^n \Rightarrow P'$ uniform over $\pm 1$'s
- Output of conjuncts algo may not be $=\text{any 3cnf}$
- Hypo. representation matters
- "Overcompleteness"
Definition: $C$ is PAC learnable by $H$ if there exist learning algorithms $A$ such that for all $h \in C$, for all $P$ over $X \times \{\text{dist.}\}$, for all $\varepsilon, \delta > 0$:

- With probability $\geq 1 - \delta$, $h$ outputs $h' \in H$ such that $3(\varepsilon) = \exists z(x) = \Pr_{x \sim P}[h(x) \neq c(x)]$

- Sample size and runtime of $L$ are "efficient", e.g., polynomial in $\frac{1}{\varepsilon}, \frac{1}{\delta}$ 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 task is truly hard to learn?