CSCI699: Theory of Machine Learning Fall 2021

Lecture 9: Intractability of learning 3-DNF

Instructor: Vatsal Sharan Scribe: Jiahao Wen

1 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 1. *Let $U = \{(a_1, y_1), ..., (a_n, y_n)\}$ 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 quite generally useful. 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 any assignment from every vertex $v \to \{R, B, G\}$, such that for every edge $e \in E$, the endpoints of e are assigned different colors?

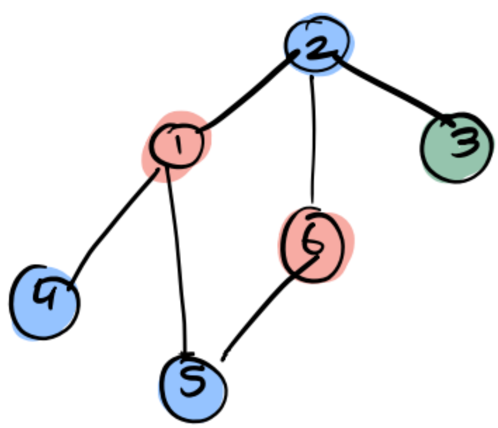
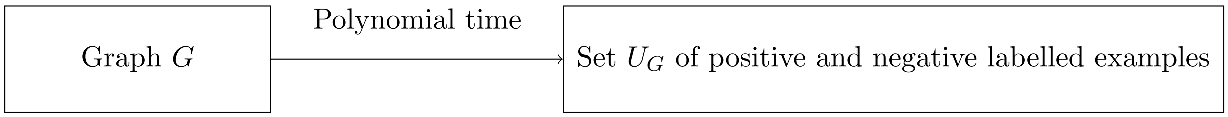


Figure 1: An example of graph 3-Coloring.

It should be noted that graph 3-Coloring is NP-complete.

Reduction



We will show that given G, we can construct U_G, such that 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.
- $E_x(h^*, D)$: pick a point uniformly at random from U_G.
- δ: $\frac{1}{3}$.
- ϵ: $\frac{1}{2|U_G|}$.

Algorithm 1

Given instance of 3-Color, construct set U_G.
Use PAC-learning algorithm A for 3-term DNF with $E_x(h^*, D), \delta = \frac{1}{3}, \epsilon = \frac{1}{2|U_G|}$.
Let h be the 3-term DNF returned by A.
if h is consistent with U_G **then**
  return "Yes".
else
  return "No".
end if

Note that D is uniform over set of size $|U_G|$, $\epsilon = \frac{1}{2|U_G|}$ and $R(h) < \epsilon$, so we can infer that h is consistent with set U_G.

If A succeeds, it must return consistent hypothesis. Therefore, all that remains is to show we can construct U_G such that there is some consistent hypothesis $h \in \mathcal{H}$ if and only if G is 3-Colorable.

Constructing U_G

$$U_G = U_G^+ \cup U_G^-$$
$$U_G^+\text{: positive examples}$$
$$U_G^-\text{: negative examples}$$

To construct U_G^+, for every vertex i in the graph, we create a positively labelled example which is 0 at the index i and 1 everywhere else. For the example shown in fig. 1, we construct U_G^+ as follows:

$$
\begin{aligned}
&|U_G^+| = |V| \\
&(v(1), +1) = ((0, 1, 1, 1, 1, 1), +1) \\
&(v(2), +1) = ((1, 0, 1, 1, 1, 1), +1) \\
&\vdots \\
&(v(6), +1) = ((1, 1, 1, 1, 1, 0), +1)
\end{aligned}
$$

To construct U_G^-, for every edge (i, j) in the graph, we create a negatively labelled example which is 0 at the coordinates i and j and 1 everywhere else. For fig. 1, we have

$$
\begin{aligned}
&|U_G^-| = |E| \\
&(e(1, 2), -1) = ((0, 0, 1, 1, 1, 1), -1) \\
&(e(1, 4), -1) = ((0, 1, 1, 0, 1, 1), -1) \\
&\vdots \\
&(e(5, 6), -1) = ((1, 1, 1, 1, 0, 0), -1)
\end{aligned}
$$

Part I: 3-Colorable → there exists a consistent 3-term DNF

3-term DNF: $\phi = T_R \cup T_B \cup T_G$

- R: set of all vertices colored red.
- B: set of all vertices colored blue.
- G: set of all vertices colored green.

T_R: conjunction of all variables whose index doesn't appear in R. $\rightarrow T_R = x_2 \cap x_3 \cap x_4 \cap x_5$. Similarly, we can get $T_B = x_1 \cap x_3 \cap x_6$ and $T_G = x_1 \cap x_2 \cap x_4 \cap x_5 \cap x_6$.

For each $i \in R$, example $v(i)$ must satisfy T_R because x_i doesn't appear in T_R.

Further, no $e(i, j) \in U_G^-$ can satisfy T_R. Both i and j cannot be colored red at the same time, one of x_i or x_j must appear in T_R. But $e(i, j)$ has 0 values for both x_i and x_j. So, T_R cannot be satisfied by $e(i, j)$. The same argument follows for T_B and T_G, and we have therefore shown that $\phi = T_R \cup T_B \cup T_G$ is consistent.

Part II: 3-term DNF → 3-colorable

$\phi = T_R \cup T_B \cup T_G$

For a vertex i, if $v(i)$ satisfies T_R, color i red. Similar with T_B and T_G. (Break any ties arbitrarily.)

Since formula is consistent, every $v(i)$ must satisfy at least one of T_R, T_G, T_B. So every vertex is assigned a color.

Claim 2. *The coloring is valid 3-Coloring.*

Proof. If i and $j(i \neq j)$ are assigned the same colors (say red), both $v(i)$ and $v(j)$ satisfy T_R.

$$v(i) = (1, \dots, 0, \dots, 1, \dots, 1)$$
$$v(j) = (1, \dots, 1, \dots, 0, \dots, 1)$$
$$e(i, j) = (1, \dots, 0, \dots, 0, \dots, 1)$$

Since i-th bit of $v(i)$ is 0 and i-th bit of $v(j)$ is 1, we can infer that neither x_i nor $\bar{x}_i$ appears in T_R. We can see $e(i, j)$ and $v(j)$ only differs in i-th coordinate. If $v(j)$ satisfies T_R, so does $e(i, j)$. Then $e(i, j)$ should be labelled positive. So $e(i, j) \notin U_G^-$ and $(i, j) \notin E$. □

2 Using 3-CNF formulae to avoid intractability

So far, we 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?

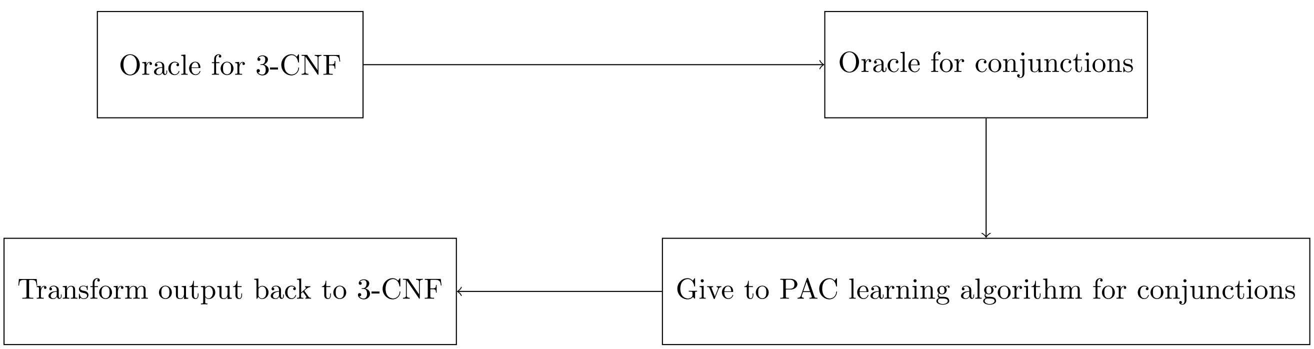
Distributive law:

$$(u \cap v) \cup (w \cap x) = (u \cup w) \cap (u \cup x) \cap (v \cup w) \cap (v \cup x)$$

We can represent any 3-term DNF_d $\phi = T_1 \cup T_2 \cup T_3$ by a $3 - CNF_d$ $\phi = \cap_{u \in T_1, v \in T_2, w \in T_3} (u \cup v \cup w)$. This is a 3-CNF (conjunctive normal form).

Theorem 3. *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 3-CNF formulae as a conjunction over a new and 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 $y_{u,v,w} = u \cup v \cup w$. When $u = v = w$, $y_{u,v,w} = u$. So all original variable are in the new set. The number of variables should be $(2d)^3$, $O(d^3)$.

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 $x_1, \dots, x_d$ is equivalent to a conjunction over $\{y_{u,v,w}\}$. (Replacing any clause $u \cup v \cup w$ by $y_{u,v,w}$.)

Then 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 $(u \cup v \cup w)$.

Claim 4. *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 errors $\leq \epsilon$ with respect to $h^{*\prime}$ and D', then h has errors $\leq \epsilon$ with respect to h^* and D.*

Proof. We simply note that our transformation of instances is one $\to$ one.

$$\text{If } a_1 \to a_1' \text{ and } a_2 \to a_2'$$
$$a_1 \neq a_2 \implies a_1' \neq a_2'.$$

Therefore, our transformation preserves the error with respect to the original and transformed instances. □

This completes our polynomial time reduction. □

Important takeaway: The choice of representation/hypothesis can make the difference between efficient algorithms and intractability. Going to more expressive hypothesis class (3-term DNF $\to$ 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!