CS6781 - Theoretical Foundations of Machine Learning

Lecture 9: Hardness of Learning

February 18, 2020

Lecturer: Nika Haghtalab *Readings: The Design and Analysis of Algorithms, D. Kozen*
Scribe: Yurong You and David Low

1 Overview

For the first 4 weeks, we studied definitions of learnability, focusing on the *statistical*, i.e., the sample complexity, aspect of learning. Today, we focus on the *computation*, i.e. run time, aspect of learning. We can divide the computational aspect of learning into two categories: representation *dependent* and representation *independent*.

- In representation dependent learning, we fix a class $\mathcal{C}$ and our algorithm outputs $h \in \mathcal{C}$. This is "proper learning" because we can draw h from a known class $\mathcal{C}$. It is easier to prove results in representation dependent learning, so we will focus on this category.
- In representation independent learning, we allow our algorithm to output $h \notin \mathcal{C}$. This is "improper learning" because h is not constrained to be from $\mathcal{C}$. The community has only recently proven results of this form; we will only state results from this category.

2 Hardness and Reductions

Definition 2.1. Let L be a decision problem. We say that

1. L is NP-hard when for every problem H in NP, there is a polynomial time reduction from H to L.
2. L is NP-complete if $L \in \mathrm{NP}$ and L is NP-hard. by an oracle machine with an oracle for L.

These problems are only *conjectured* to be hard; if a polynomial time algorithm for solving any NP-complete problem is found, then $\mathrm{NP} = \mathrm{P}$ and all NP-complete problems can be solved in polynomial time. Examples of NP-complete problems are max independent set, 3-CNF SAT, subset sum, and hypergraph coloring.

To show a problem B is hard, take A from the set of NP-complete problems and show that you could solve any instance of problem A in polynomial time if you are given an algorithm for problem B. You can write this reduction as

$$A \leq_p B.$$

We need to map *instances* of problems A to *instances* of problems B. To show this, you need to prove that for any instance (input) of problem A,

1. converting an instance of problem A to an instance of B needs at most polynomial time;
2. the solution to the converted instance of B gives the solution to the instance of A with work that takes at most a polynomial amount of time.

3 Representation Dependent Hardness

In the representation-dependent setting, we consider the hardness of *proper* learning, where the algorithm is required to return a hypothesis that belongs to a predetermined hypothesis class $\mathcal{C}$.

That is, assuming that there exists a $c^* \in \mathcal{C}$ such that $\mathrm{err}_{\mathcal{D}}(c^*) = 0$ (realizable setting), how hard is it to find a hypothesis $h \in \mathcal{C}$ such that $\mathrm{err}_{\mathcal{D}}(h) \leq \varepsilon$? More generally, how hard is it to find a hypothesis $h \in \mathcal{C}$ even in the non-reailzable setting such that

$$\operatorname*{err}_{\mathcal{D}}(h) \leq \varepsilon + \alpha \min_{c \in \mathcal{C}} \operatorname*{err}_{\mathcal{D}}(c),$$

for some values of α and ϵ.

3.1 Intersection of two Halfspaces

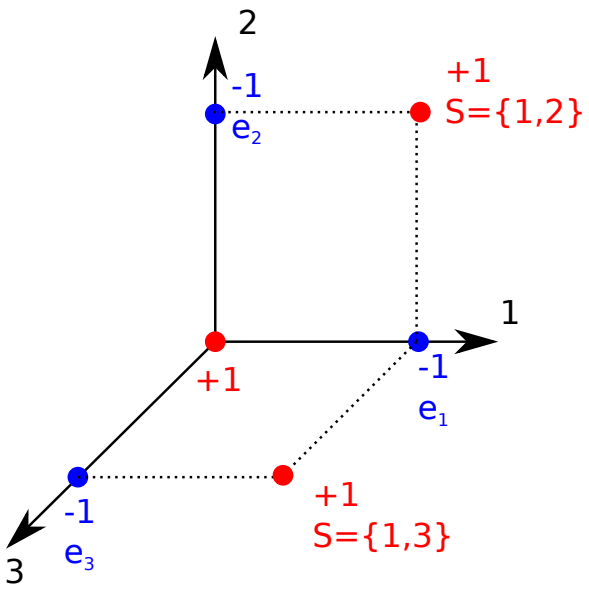
We know from previous lectures that when $\mathcal{C}$ is the class of halfspaces, there is an LP that solves the consistency problem: That is, given $\mathcal{S}$, find a halfspace h such that $h(x_i) = y_i$ for all $(x_i, y_i) \in \mathcal{S}$, or establishes that none exists. The same linear program can be used to efficiently learn halfspaces in the realizable setting. In this section, we see that when $\mathcal{C}$ represent the class of intersections of two halfspaces, this this learning task becomes hard. We first show that solving the consistency problem in this setting is hard.

Theorem 3.1. *Learning intersections of two halfspaces in $\mathbb{R}^d$ in the consistency model is hard.*

Definition 3.2 (Hypergraph Coloring Problem)**.** Given a universe $[n]$ and m subsets $S_1, S_2, \cdots, S_m \subseteq [n]$, decide whether one can color each element of $[n]$ with either the color red or blue such that no S_i is monochromatic. This is an NP-Complete Problem.

We are going to show that any instance of the hypergraph coloring problem can be mapped onto an instance of the intersection of two halfspaces problem. Then if we can solve the intersection of two halfspaces in the consistency model, we can solve the NP-Complete hypergraph coloring problem.

We are given a universe $[n]$ and subsets $S_1, \cdots, S_m$. One way to map the hypergraph coloring problem onto the intersection of halfspaces in $\mathbb{R}^d$ i.e., to form a sample space $\mathcal{S}$ from the universe $[n]$ and subsets $S_1, \cdots, S_m$, is as follows:



1. Take $d = n$.

   The number of elements in the universe $[n]$ is the dimensionality of the intersection of halfspace problem.

2. $(\vec{0}, +1)$.

   Label the origin as $+1$.

3. $(\vec{e_i}, -1)$ for all $i \in [n]$.

   Each standard basis vector $\vec{e_i} \in \mathbb{R}^n$ is labeled -1. The jth element of the standard basis vector $\vec{e_i}$ is defined as $\vec{e_i}[j] = \delta_{ij}$ where δ_{ij} is the Kronecker delta defined as 1 if $i = j$ and 0 otherwise.

4. $(\vec{S_i}, +1)$ for all $i \in [n]$.

   Create, and label as $+1$, $\vec{S_i}$ the indicator vector for S_i. We can write $\vec{S_i}$ as

$$\vec{S_i}[j] = \begin{cases} 1 & \text{if } j \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently, $\vec{S_i} = \sum_{j \in S_i} \vec{e_j}$.

Claim 3.3. *$\mathcal{S}$ is consistent with an intersection of two halfspaces if and only if there is a legal 2-coloring of $S_1, S_2, \cdots, S_m \subseteq [n]$.*

Proof. $\Rightarrow$ direction: given two halfpsaces (h_1, h_2) whose intersection is consistent with $\mathcal{S}$, we want to convert it into a legal 2-coloring for $S_1, S_2, \cdots, S_m \subseteq [n]$.

Suppose

$$h_1(\vec{x}) = \text{sign}(w_1 x_1 + \cdots + w_n x_n \geq w_0),$$
$$h_2(\vec{x}) = \text{sign}(w_1' x_1 + \cdots + w_n' x_n \geq w_0').$$

We color points in $[n]$ as follows. For all $i \in [n]$,

$$\text{color}(i) = \begin{cases} \text{red} & \text{if } h_1(\vec{e_i}) = -1, \\ \text{blue} & \text{if } h_2(\vec{e_i}) = -1, \\ \text{either color} & \text{otherwise} \end{cases}$$

We show that this is a legal 2-coloring by contradiction. Assume S_i that is monochromatic. Without loss of generality, we assume the nodes in S_i are colored all blue. Then,

1. $w_0' \leq 0$:
   By construction, $(\vec{0}, +1) \in \mathcal{S}$. So, we must have that $h_2(\mathbf{0}) = +1$ This implies that $0 \geq w_0'$.
2. $w_j' < w_0'$:
   For all $j \in S_i$, $(\vec{e_j}, -1) \in \mathcal{S}$ colored blue. Thus, $h_2(\vec{e_j}) = -1 \implies w_j' < w_0'$.

Thus

$$\vec{w'} \cdot \vec{S_i} = \sum_{j \in S_i} w_j' < |S_j| \cdot w_0' \leq w_0' \implies h_2(\vec{S_i}) = -1,$$

which contradicts with the fact that $h_1(\vec{S_i}) \wedge h_2(\vec{S_i}) = +1$.

By the symmetry of the coloring procedure, the proof for when all of the elements of S_i are all red follows similarly.

$\Leftarrow$ direction: Given a legal 2-coloring of $S_1, S_2, \cdots, S_m \subseteq [n]$, we need to construct two halfspaces h_1 and h_2 whose intersection is consistent with $\mathcal{S}$.

We construct the $h_1(\vec{x}) = \text{sign}(w_1 x_1 + \cdots + w_n x_n \geq w_0)$ and $h_2(\vec{x}) = \text{sign}(w_1' x_1 + \cdots + w_n' x_n \geq w_0')$ as follows,

$$\forall j \in [n], w_j = \begin{cases} -1 & \text{if } j \text{ is colored red}, \\ +n & \text{if } j \text{ is colored blue}, \end{cases}$$
$$w_0 = -\frac{1}{2},$$

and

$$\forall j \in [n], w'_j = \begin{cases} -1 & \text{if } j \text{ is colored blue,} \\ +n & \text{if } j \text{ is colored red,} \end{cases}$$
$$w'_0 = -\frac{1}{2}.$$

We check the classifications are consistent:

- For $\vec{0}$:

$$\vec{w} \cdot \vec{0} = \vec{w'} \cdot \vec{0} = 0 \geq -1/2 \implies h_1(\vec{0}) = h_2(\vec{0}) = +1 \implies h_1(\vec{0}) \wedge h_2(\vec{0}) = +1$$

- For all $\vec{e_j}$ that were colored red:

$$h_1(\vec{e_j}) = \text{sign}(w_j \geq -1/2) = \text{sign}(-1 \geq -1/2) = -1,$$
$$h_2(\vec{e_j}) = \text{sign}(w'_j \geq -1/2) = \text{sign}(+n \geq -1/2) = +1,$$

  which implies $h_1(\vec{e_j}) \wedge h_2(\vec{e_j}) = -1$.

- For all $\vec{e_j}$ that were colored blue:

$$h_1(\vec{e_j}) = \text{sign}(w_j \geq -1/2) = \text{sign}(+n \geq -1/2) = +1,$$
$$h_2(\vec{e_j}) = \text{sign}(w'_j \geq -1/2) = \text{sign}(-1 \geq -1/2) = -1,$$

  which implies $h_1(\vec{e_j}) \wedge h_2(\vec{e_j}) = -1$.

- For S_j, since it is not monochromatic, there must at least one $k \in S_j$ such that w_k (and $w'_{k'}$) set as $+n$, thus

$$\vec{w} \cdot \vec{S_j} \geq n + (n-1)(-1) = 1 \geq -1/2 \implies h_1(\vec{S_j}) = +1,$$
$$\vec{w'} \cdot \vec{S_j} \geq n + (n-1)(-1) = 1 \geq -1/2 \implies h_2(\vec{S_j}) = +1$$

  which implies $h_1(\vec{S_j}) \wedge h_2(\vec{S_j}) = +1$.

So, $h_1 \wedge h_2$ is consistent with $\mathcal{S}$. We proved both directions are valid. □

3.2 NP vs RP

Definition 3.4. Randomized polynomial time (RP) is the complexity class of problems for which a probabilistic Turing machine exists such that:

- The algorithm runs in polynomial time in the input size
- If the correct answer is NO, it always returns NO.
- If the correct answer is YES, then it returns YES with probability at least $1/2$.

Theorem 3.5. *Proper PAC learning of intersections of two halfspaces in the realizable setting is hard unless* $\mathrm{NP} = \mathrm{RP}$.

In other words, assume there exists an algorithm with runtime in $\mathrm{poly}(n, 1/\varepsilon, 1/\delta)$ that properly PAC learns in this setting to accuracy ε and confidence $1 - \delta$. Then, this is is an algorithm in RP that solves an NP complete problem.

Proof. To prove this, repeat the same construction as before to get the sample set $\mathcal{S}$ that was used in the consistency model. Define $\mathcal{D}$ to be the uniform distribution over $\mathcal{S}$ and define $\varepsilon = 1/(2|\mathcal{S}|)$ and $\delta = 1/100$. The actual instance $\mathcal{S}' \sim \mathcal{D}^m$ is taken for m required for the algorithm and the algorithm takes this $\mathcal{S}'$ as input returns an intersection of two halfspaces. We say YES if the PAC learning algorithm learns a hypothesis $h_1 \wedge h_2$ that corresponds to a legal 2-coloring. Otherwise, say NO.

It is easy to see that if the instance has no legal 2-coloring, then by definition our algorithm returns NO. Note that because $\varepsilon \leq 1/(2|\mathcal{S}|)$, then $\mathrm{err}_{\mathcal{D}}(h_1 \wedge h_2) \leq \epsilon$ if and only if $h_1 \wedge h_2$ is consistent with $\mathcal{S}$. Therefore, when the original instance has a legal 2-coloring, then with probability $1 - \delta$ the PAC learning algorithm returns $h_1 \wedge h_2$ that leads to this legal 2 coloring. □