Lecture 15: Hardness of Learning

Han Shao

1 Computational Hardness

Our sample-complexity bounds describe how much data is sufficient for learning. They do not, by themselves, bound the time needed to find a suitable hypothesis. In this lecture, we study the computational barriers to learning.
In the realizable setting, the goal of *proper learning* is to find a hypothesis $h_S \in \mathcal{H}$ with $\mathrm{err}_{\mathcal{D}}(h_S) \le \varepsilon$. More generally, one can ask for a proper hypothesis satisfying

$$\mathrm{err}_{\mathcal{D}}(h_S) \le \inf_{h \in \mathcal{H}} \mathrm{err}_{\mathcal{D}}(h) + \varepsilon.$$

The hardness result below already applies in the realizable setting.
A *decision problem* has a YES or NO answer on each input. The class P consists of decision problems solvable in deterministic polynomial time. The class NP consists of decision problems for which an input is a YES instance if and only if it has a polynomial-length certificate accepted by a polynomial-time verifier.

Definition 1 (Polynomial-time reduction)**.** *For decision problems A and B, write $A \le_p B$ if there is a polynomial-time computable map r such that, for every input z,*

$$z \text{ is a YES instance of } A \quad \Longleftrightarrow \quad r(z) \text{ is a YES instance of } B.$$

Thus, an efficient algorithm for B would give an efficient algorithm for A: first compute $r(z)$ and then solve the resulting instance.

Definition 2 (NP-hardness and NP-completeness)**.** *A decision problem B is* NP-hard *if $A \le_p B$ for every $A \in$ NP. It is* NP-complete *if it is NP-hard and also belongs to* NP*.*

To prove a new problem B NP-hard, it is enough to reduce a known NP-complete problem to B. We must verify both that the construction takes polynomial time and that it preserves YES and NO answers. NP-hardness is a conditional computational barrier: a polynomial-time algorithm for an NP-complete problem would imply P = NP.

2 Proper Learning of Intersections of Two Halfspaces Is Hard

Recall that a halfspace has the form

$$h_{w,b}(x) = \mathrm{sign}(w^\top x + b), \qquad w \in \mathbb{R}^d, \quad b \in \mathbb{R},$$

where $\mathrm{sign}(u) = +1$ for $u \ge 0$ and -1 otherwise. For a single halfspace, consistency on a finite sample can be decided by linear programming. We now consider the class

$$\mathcal{H}_d^{\cap 2} = \{h_1 \wedge h_2 : h_1, h_2 \text{ are halfspaces in } \mathbb{R}^d\},$$

where the signed-label conjunction is defined by

$$(h_1 \wedge h_2)(x) = \begin{cases} +1 & \text{if } h_1(x) = h_2(x) = +1, \\ -1 & \text{otherwise.} \end{cases}$$

The class of intersections of two affine halfspaces in $\mathbb{R}^d$ has VC dimension $O(d)$. Indeed, its growth function is at most the square of the growth function of affine halfspaces, and Sauer's lemma gives the bound.

The *consistency problem* asks whether a given labeled sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$ admits some $h \in \mathcal{H}_d^{\cap 2}$ with $\mathrm{err}_S(h) = 0$. Here the dimension d is part of the input, not a fixed constant.

Theorem 1 (Hardness of consistency)**.** *Deciding consistency with an intersection of two halfspaces is NP-hard. This holds even when every instance in the sample belongs to* $\{0,1\}^d$.

We prove theorem 1 by reducing hypergraph two-coloring to the consistency problem.

Definition 3 (Hypergraph two-coloring)**.** *Given a vertex set* $[n]$ *and nonempty subsets* $E_1, \ldots, E_q \subseteq [n]$*, decide whether the vertices can be colored red or blue so that every* E_j *contains both colors. Such a coloring is called a* legal two-coloring*.*

We use the NP-completeness of this problem as our starting point; its NP-completeness proof is not part of this lecture. The sets E_j are the *hyperedges*. They may have more than two vertices, so this is not the ordinary graph bipartiteness problem.

Constructing the labeled sample. Given a hypergraph on $[n]$, set $d = n$. For each hyperedge, let

$$u_j = \sum_{i \in E_j} e_i \in \{0,1\}^n, \qquad j \in [q],$$

where e_i is the ith standard basis vector. Construct the sample

$$S = \big((0, +1), (e_1, -1), \ldots, (e_n, -1), (u_1, +1), \ldots, (u_q, +1)\big). \tag{1}$$

Thus $m = n + q + 1$: the origin and hyperedge indicators are positive, while all standard basis vectors are negative. The construction takes polynomial time and uses only coordinates in $\{0,1\}$.

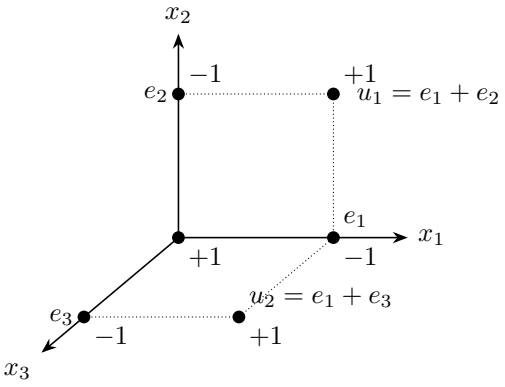


Figure 1: The construction for $n = 3$ with hyperedges $E_1 = \{1, 2\}$ and $E_2 = \{1, 3\}$. The labels are written next to the points; dotted lines indicate the vector sums.

Lemma 1 (Correctness of the reduction)**.** *The sample in* (1) *is consistent with a member of* $\mathcal{H}_n^{\cap 2}$ *if and only if the hypergraph has a legal two-coloring.*

Proof. We prove the two directions separately.

From consistent halfspaces to a coloring. Suppose $h_1 \wedge h_2$ is consistent, where

$$h_r(x) = \operatorname{sign}\big((w^{(r)})^\intercal x + b_r\big), \qquad r \in \{1, 2\}.$$

Since the origin is positive, both halfspaces accept it, so $b_1, b_2 \geq 0$. Since every e_i is negative, at least one of the two halfspaces rejects it. Assign colors by

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

If a vertex is blue, then $h_1(e_i) = +1$ and hence $h_2(e_i) = -1$. If both halfspaces reject e_i, the rule assigns red; only one rejecting halfspace is needed to choose its color.
Suppose, for a contradiction, that some hyperedge E_j is all red. For each $i \in E_j$, rejection by h_1 implies

$$w_i^{(1)} + b_1 < 0, \qquad \text{so} \qquad w_i^{(1)} < -b_1.$$

Because E_j is nonempty and $b_1 \geq 0$,

$$\begin{aligned} (w^{(1)})^\intercal u_j + b_1 &= \sum_{i \in E_j} w_i^{(1)} + b_1 \\ &< -|E_j| b_1 + b_1 \\ &= (1 - |E_j|) b_1 \leq 0. \end{aligned}$$

Thus $h_1(u_j) = -1$, contradicting the positive label of u_j. If E_j is all blue, the same argument using $w^{(2)}$ and b_2 gives $h_2(u_j) = -1$, again a contradiction. Every hyperedge therefore contains both colors.

From a coloring to consistent halfspaces. Conversely, suppose the hypergraph has a legal two-coloring. Set $b_1 = b_2 = 1/2$ and, for each $i \in [n]$, define

$$w_i^{(1)} = \begin{cases} -1 & \text{if } i \text{ is red,} \\ n & \text{if } i \text{ is blue,} \end{cases} \qquad w_i^{(2)} = \begin{cases} n & \text{if } i \text{ is red,} \\ -1 & \text{if } i \text{ is blue.} \end{cases}$$

Let $h_r(x) = \operatorname{sign}((w^{(r)})^\intercal x + 1/2)$. We check all three types of examples.

1. **The origin.** Both scores equal $1/2 > 0$, so $(h_1 \wedge h_2)(0) = +1$.

2. **The basis vectors.** If i is red, then $w_i^{(1)} + 1/2 = -1/2 < 0$, so $h_1(e_i) = -1$. If i is blue, then $w_i^{(2)} + 1/2 = -1/2 < 0$, so $h_2(e_i) = -1$. Therefore the intersection labels every e_i negative.

3. **The hyperedge indicators.** Each E_j contains a blue vertex, which contributes n to $(w^{(1)})^\intercal u_j$. All other contributions are at least -1, and there are at most $n - 1$ of them. Hence

   $$(w^{(1)})^\intercal u_j \geq n - (n - 1) = 1.$$

   Because E_j also contains a red vertex, the same reasoning gives $(w^{(2)})^\intercal u_j \geq 1$. Both halfspaces label u_j positive.

The intersection is therefore consistent with S. □

Proof of theorem 1. The sample construction is polynomial-time, and lemma 1 shows that it preserves YES and NO answers. Thus hypergraph two-coloring reduces to consistency with an intersection of two halfspaces. The latter problem is NP-hard. □

3 From Consistency Hardness to PAC-Learning Hardness

A PAC learner is not a deterministic consistency algorithm: it sees an i.i.d. sample and may fail with small probability. To turn the preceding reduction into a learning-hardness result, we use a randomized complexity class.

Definition 4 (Randomized polynomial time)**.** *A decision problem belongs to* RP *if there is a randomized polynomial-time algorithm such that*

- *on every NO instance, the algorithm always returns NO;*
- *on every YES instance, it returns YES with probability at least* $1/2$*.*

Theorem 2 (Hardness of proper PAC learning)**.** *Unless* NP = RP*, there is no randomized polynomial-time algorithm that properly PAC learns* $\mathcal{H}_d^{\cap 2}$ *in the realizable setting, where the dimension* d *is part of the input.*

Here efficiency means polynomial sample complexity and running time in the dimension, the required numerical representation lengths, $1/\varepsilon$, and $1/\delta$. A proper output is an explicitly represented pair of halfspaces whose predictions can be evaluated in polynomial time.

Proof. Suppose an efficient proper PAC learner $\mathcal{A}$ exists. We use it to solve hypergraph two-coloring with one-sided randomized error. Given a hypergraph, form the sample S in (1) and write $m = |S| = n + q + 1$.
Let $\mathcal{D} = \mathrm{Unif}(S)$ be the joint distribution obtained by choosing one of the m labeled sample occurrences uniformly. Run $\mathcal{A}$ with

$$\varepsilon = \frac{1}{2m}, \qquad \delta = \frac{1}{100},$$

on a fresh sample $S' \sim \mathcal{D}^M$, where M is sufficient for its PAC guarantee. Since the learner is efficient, M and its running time are polynomial in the hypergraph input size for these parameters.
Let $h_{S'} = \mathcal{A}(S')$. Return YES only if the output is a valid pair of halfspaces whose intersection agrees with *every* labeled occurrence in S; otherwise return NO. The output can be checked on all m occurrences in polynomial time. If the learner's running-time guarantee is stated only for realizable inputs, stop it after the promised polynomial bound and return NO on a timeout.

NO instances are always rejected. If the hypergraph is not two-colorable, lemma 1 shows that no intersection of two halfspaces is consistent with S. Thus no output can pass the check.

YES instances are accepted with high probability. If the hypergraph is two-colorable, the same lemma shows that $\mathcal{D}$ is realizable by $\mathcal{H}_n^{\cap 2}$. The PAC guarantee therefore gives

$$\Pr_{S',\mathcal{A}}\left[\mathrm{err}_{\mathcal{D}}(h_{S'}) \le \frac{1}{2m}\right] \ge \frac{99}{100}.$$

The probability includes both the fresh sample and the internal randomness of $\mathcal{A}$. Each indexed occurrence in S is selected with probability $1/m$, so making even one mistake on S would imply

$$\mathrm{err}_{\mathcal{D}}(h_{S'}) \ge \frac{1}{m} > \frac{1}{2m}.$$

Consequently, whenever the PAC guarantee holds, $h_{S'}$ is consistent with all of S, and the algorithm returns YES.

We have a polynomial-time algorithm that never accepts a NO instance and accepts each YES instance with probability at least 99/100. Thus hypergraph two-coloring belongs to RP. Its NP-completeness gives NP $\subseteq$ RP, while RP $\subseteq$ NP: an accepting random-bit sequence serves as a polynomially checkable certificate. Therefore NP = RP. □

Hardness of agnostically learning halfspaces. Although a single halfspace can be learned efficiently in the realizable setting, the agnostic setting is computationally harder. Guruswami and Raghavendra (2009) showed that, for arbitrarily small constants $\eta, \alpha > 0$, even when a halfspace with empirical error at most η exists, finding a halfspace with empirical error at most $1/2 - \alpha$ is NP-hard. This yields hardness of proper agnostic PAC learning of halfspaces. Allowing improper hypotheses does not necessarily remove this obstacle: under an additional average-case hardness assumption concerning refutation of random K-XOR formulas, Daniely (2016) proved that no polynomial-time learner can guarantee an efficiently evaluable classifier h_S satisfying

$$\mathrm{err}_{\mathcal{D}}(h_S) \le \frac{1}{2} - \frac{1}{d^c},$$

for any fixed constant $c > 1$, even when some halfspace has error at most an arbitrarily small constant $\eta > 0$. Here d is the input dimension, and h_S may have any representation, not necessarily a halfspace. Thus, under the respective hardness assumptions, even beating random guessing can be computationally hard in the agnostic setting, for both proper and improper learners.