**Theory of Machine Learning** **Fall 2021**

# Lecture 10: Representation Independent Hardness of Learning

*Lecturer: Vatsal Sharan* *Scribe: Chandra Sekhar Mukherjee*

This lecture deals with the problem of computational intractability of learning problem, their relation to cryptography, some more modern results on representation independent learnability and finally relaxation of problem definitions that make them learnable. First we review the major takeaway of the last lecture.

> The choice of representation hypothesis can make the difference between efficient algorithms and intractability. Going to more expressive (richer) hypothesis classes (for example, from 3-Term DNF to 3-Term CNF) can make learning efficient.
>
> Statistically, learning over a richer hypothesis class can never help if you know the target class is in a smaller hypothesis, but when computational efficiency is brought into the picture, the scenario is very different.

To account for this, we make the following distinction:

**Definition 1** (Concept and Hypothesis Class)**.**

- *The concept class $\mathcal{C}$ is the class from which the hypothesis originally belongs.*
- *The hypothesis class $\mathcal{H}$ is the class from which the learner chooses its hypothesis.*

Against this backdrop, we have a revised definition for efficient PAC learning.

**Definition 2** (Proper and Improper PAC Learning)**.** *If $\mathcal{C}$ is a concept class over the instance space $X^d$ and $\mathcal{H}$ is a hypothesis class over $X^d$, we say that $\mathcal{C}$ is (efficiently) PAC learnable using $\mathcal{H}$ if our basic definition of PAC learning is met by an algorithm which is allowed to output a hypothesis form $\mathcal{H}$. Here we implicitly assume that $\mathcal{H}$ is at least as expressive as $\mathcal{C}$ (So that thee is a representation in $\mathcal{H}$ for every function in $\mathcal{C}$.*

- *If $\mathcal{C} = \mathcal{H}$ then the algorithm is called a proper learning algorithm.*
- *If $\mathcal{C} \subset \mathcal{H}$ then the algorithm is called an imporper learning algorithm.*

Question: Are there learning problems which are hard even improperly?

To answer this we look into representation independent hardness results for learning, through the connection between learning theory and cryptography!

## 1 Representation Independent Hardness Results for Learning

Intuitively, results about cryptographic security of some systems translate to unlearnability of some corresponding learning tasks, and vice versa!

However, we currently have no way of ensuring that a cryptographic protocol is not breakable, even if we assume $P \neq NP$. Therefore must rely on stronger average-case assumptions.

## Sketch of Idea:

**Definition 3** (One Way Functions)**.** *A one-way function $f : \{0,1\}^d \to \{0,1\}^d$ is one that is easy to compute, but hard to invert.*

*More formally, $f$ can be computed in time poly$(d)$ but for any randomized polynomial algorithm $A$ and for any polynomial $p()$, we have*

$$Pr\left[f(A(f(*)) = f(x)\right] \leq \frac{1}{p(d)}$$

*where the probability is taken over $x$ drawn uniformly from $\{0,1\}^d$, and randomness in $A$.*

Next we define a trapdoor one way functions.

**Definition 4** (Trapdoor One Way Functions)**.** *A one-way function $f$ is called a trapdoor oneway function, if for some polynomial $p()$ there exists a bit string $s$ (called a secret key) of length $\leq p(d)$, such that there is a polynomial time algorithm that for all $x \in \{0,1\}^d$ on input $f(x), s$ outputs $x$.*

Proving the existence of one-way functions is one of the long standing open problems in computer science, although they are widely conjectured to be true. In this regard we discuss a candidate one-way trapdoor function.

## Discrete Cube Root- A candidate Trapdoor One-way Function

Let $N = p \cdot q$ b a product of two primes of roughly equal length. Let $f_N(x) = x^3 \bmod N$.

If one knows $p$ and $q$, this function is easy to invert. However, it is widely believed that this function is hard to invert without the knowledge of $p$ and $q$. This also forms the basis of the famous RSA cryptosystem (Discrete Cube Root Assumption (DCRA)).

For a fixed $L$, let $F$ be the family of all functions

$$F = \{f_N(x), N = p \cdot q, p \text{ and } q \text{ are primes}, length(p), length(q) \leq p(d) \text{ for some polynomial } p()\}$$

Under DCRA, given $N$ and $y = f_N(x)$ for some random $x \in \{0,1\}^d$, it is hard to compute $x$ in polynomial time.

Now we reduce the hardness of this problem to the unlearnability of a learning problem, such that a successful learning algorithm will be able to invert $f_N(x)$.

## A Learning Problem from DCRA

We define the concept class: $\mathcal{C} : \{{f_N}^{-1}(x) : N = p \cdot q, length(p), length(q) \leq p(d)\}$.

For this to be learning problem in the PAC learning model, the learning algorithm needs to be sent i.i.d samples from the distribution of the form $(y, {f_N}^{-1}(y))$. Now, one may think obtaining this set of examples requires inverting $f_N$, but there is a simple alternative.

We i.i.d sample $x$, obtain $f_N(x)$, and then pass the ordered tuples $f_N(x), x$ as samples to the learning algorithm in question.

This works because $f_N$ is a bijection, and thus the pairs $(f_N(x), x)$ and $(y, {f_N}^{-1})$ are distributed identically indifferent of whether $x$ or $y$ is sampled i.i.d.

Note: This is not requirement for a PAC learning (as in PAC learning one should be able to learn from any input distribution), but rather is a stronger result that says that learning under just the uniform distribution of inputs is as hard as the DCRA assumption.

**Statistical Viewpoint**

Here one should note that the statistical viewpoint and tractability of learning $\mathcal{C}$ is not affected by the DCRA assumption. The concept class $\mathcal{C}$ is parameterized by the secret key of length $p(d)$ and thus $|\mathcal{C}| \leq 2^{2p(d)}$, and is thus learnable with $\mathcal{O}(p(d))$ samples.

## 2 Some Modern Results on Hardness Of Learning

Let us now discuss some recent results show which show representation independent hardness for hypothesis encountered more frequently in practice.

Consider the class of halfspaces/ linear threshold functions (LTF)

$$\mathcal{H} : \{x \rightarrow sign(w^T x), w \in \mathbb{R}^d\}$$

**Theorem 5.** *The class of halfspaces is efficiently PAC learnable.*

However, agnostically learning halfspaces, even improperly is hard under complexity assumptions.

**Theorem 6** (Informal, [1])**.** *Under "hardness of refuting random $K = k$-XOR", for any constant c, there is no poly-time algorithm that given a sample of $d^c$ points $\{-1, 1\}^d$ can distinguish whp between the cases:*

1. *Labels are uniformly random coin flips.*
2. *The labels represent a LTF with error at most 10%.*

Let us now look into the "Refuting random $k$-XOR " problem.

### Refuting Random $k$-XOR

The $k$-XOR function is the AND functions of literals where each literal is a parity of at most $k$ variables.

Now we discuss a result about solving the $k$-xor problem, which then gives rise to the stronger "random $k$-XOR" assumption.

**Proposition 7.** *In relation to the random $k$-XOR problem we have the following results:*

1. *If a satisfying assignment exists, it is easy to find using Gaussian elimination.*
2. *However, if no satisfying assignment exists, then finding an assignment that satisfies the maximal number of XOR literals is an NP hard problem [2].*

In this direction, we have the following stronger, assumption.

**Claim 8.** *Given $n \leq d^{\sqrt{k}\log k}$ terms, it is hard to distinguish between*

1. *The case of a random $k$-XOR formula with $n$ constraints.*
2. *A formula of $k$-XOR constraints that has a satisfying assignment satisfying at least 90% of the constraints.*

We also note that with high probability, a random $k$-XOR formula with enough constraints has the property that any assignment can only satisfy about half the constraints.

**Proposition 9.** *A random formula with $n \geq \frac{d}{\epsilon^2}$ many $k$-XOR constraints will w.h.p have the property that no assignment satisfies more than $1/2 + \epsilon$ fraction of the constraints.*

*Proof.* Fix any $x \in \{0, 1\}$ at random to be the assignment. Then draw $n$ constraints at random, where each draw consists of selecting $k$ variables at random and taking their parity and then randomly negating it.

Each constraint is satisfied with probability $\frac{1}{2}$ , because of the properties of parity functions.

Furthermore, by Hoeffding's

$$Pr[x \text{ satisfies } \geq \frac{1}{2} \text{ fraction } \leq 2^{-2n\epsilon^2}$$

This happens for any random assignment. There can be $2^d$ possible assignments, and thus the total success probability is upper bounded by $2^{-2n\epsilon^2} 2^d \leq e^d$ for $n = \frac{d}{\epsilon^2}$. □

Thus even though the two cases of the assumption are far apart, they can not be agnostically learned.

**This shows that agnostic learning gets hard pretty quickly.**

Let us then discuss some relaxations that are more reflective of real world scenario. In real world distributions are generally nice, and do not operate in the worst case scenario. We have the following examples:

**Theorem 10.** *[3] The class of halfspaces is efficiently agnostically learnable under the uniform distribution, of $\{-1, 1\}^d$, or the unit sphere, or the Gaussian distribution over $\mathbb{R}^d$.*

## 2.1 Random Classification Noise

Finally we discuss a more natural relaxation of the agnostic model. In the agnostic model the points are incorrectly labeled by the target could in reality have arbitrary labels. This can be viewed as an adversarial choice designed to hinder the learning algorithm. Against this backdrop, one can consider the case of random noise on the labeling, instead of the worst-case scenario. Then we have the following result.

**Theorem 11.** *[4] Halfspaces are efficiently PAC learnable under random noise.*

Having discussed both the computational intractability of certain problems and how relaxations on the problem definition can make said problems efficiently learnable, we end this lecture with the following challenge.

**Major Challenge:** How can one reconcile the practical success of neural networks/gradient descent based approaches with worst case hardness? What about real life scenarios do make these problems tractable?

# References

[1] Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 105–117, 2016.

[2] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[3] Adam Tauman Kalai, Adam R Klivans, Yishay Mansour, and Rocco A Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008.

[4] Avrim Blum, Alan Frieze, Ravi Kannan, and Santosh Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1):35–52, 1998.