CS 354: Unfulfilled Algorithmic Fantasies May 8, 2019

Lecture 14: Hardness of PAC Learning & Random k-XOR

Lecturer: Aviad Rubinstein *Scribe: Reese Pathak*

1 Introduction

Today we are discussing PAC Learning and Random k-XOR. We will explain how the hardness of random k-XOR implies that PAC learning is hard.

2 PAC Learning

The notion of probably approximately correct (PAC) learning was introduced in [Val84]. The idea is that as input we are given examples $(x, h(x))$, where x are drawn from distribution $\mathcal{D}$, and h is a label function lying in hypothesis class $\mathcal{H}$. As output, we desire a function f, that with probability (at least) $1-\delta$, over the samples x drawn from $\mathcal{D}$, we have that

$$\mathbf{P}_{Z\sim\mathcal{D}}\{f(Z) \neq g(Z)\} \geq 1-\varepsilon.$$

The probability $1-\delta$ is the notion of "probably" correct, and the error $1-\varepsilon$ quantifies the "approximately" correct part.

There are two further desiderata one can consider from a learning algorithm:

Proper vs improper In proper learning, we require f is required to lie inside of $\mathcal{H}$. With improper learning, the algorithm is allowed to return any function that approximates the true labels.

Agnostic vs realizable Realizable learning corresponds to the case we described above, where the samples are actually of the form $(x, h(x))$ for some $\mathcal{H}$. In agnostic learning we have access to samples (x, y), where $y = h(x)$ for *most* x.

The canonical example of a class that is PAC learnable in the realizable case is *halfspaces*, namely

$$\mathcal{H} := \left\{h_w(x) = \text{sign}(\sum w_i x_i)\right\}_{w\in\mathbb{R}^n}.$$

For proper, agnostic learning, it is NP-hard [Hås01]. But arguably the most interesting question is: can improper, agnostic PAC learning (in particular halfspaces) be computationally efficient?

3 Maximum k-XOR

For maximum k-XOR, we now have inputs that are clauses $C_1, \ldots, C_m$, which take the form

$$C_j = \prod_{i\in I_j} b_i x_i, \qquad \text{for } j = 1, \ldots, m.$$

Here, b_i and x_i are $\{\pm 1\}$-valued, and I_j is a k-subset of $[n]$. As a warm up, let us remark that checking if a series of clauses is simultaneously satisfiable is actually easy. To do so, let us give an example. Consider the clauses

$$C_1 = x_1 x_2 x_3 \quad \text{and} \quad C_2 = x_2 x_3 x_4.$$

Above, we are assuming that the literals $x_1, \ldots, x_4$ are $\{\pm 1\}$ valued, but just as easily, they could be $\{0,1\}$-valued. In particular, let us map $\{\pm 1\}$ to $\{0,1\}$, and let us interchange multiplication with addition, modulo 2.[1] The clauses above are equivalent to the following clauses when literals take values with representation $\{0,1\}$:

$$C_1 = x_1 \oplus x_2 \oplus x_3 \quad \text{and} \quad C_2 = x_2 \oplus x_3 \oplus x_4.$$

Checking if there is an assignment of the variables $\{x_1, x_2, x_3, x_4\}$ that makes these two clauses satisfiable is equivalent to checking if the equations

$$x_1 \oplus x_2 \oplus x_3 = 1 \quad \text{and} \quad x_2 \oplus x_3 \oplus x_4 = 1,$$

are simultaneously satisfiable over $\mathbf{F}_2$. To answer these kinds of questions we can use Gaussian elimination (over $\mathbf{F}_2$). This general strategy makes checking satisfiability of a sequence of k XOR clauses easy.

3.1 Random k-XOR

By a *random* instance of k-XOR, we mean the following. Fix k (number of terms per clause), m (number of clauses), and n (the number of variables). Then a random k$-$XOR instance, is $C_1, \ldots, C_m$, where $I_1, \ldots, I_m$ are random k-subsets of $[n]$, and b_i are drawn randomly from $\{\pm 1\}$ for $i \in I_j$ and $j = 1, \ldots, m$.

For random k-XOR, we know that if $m \ll n$, then there exists a satisfying assignment with high probability, and if $m \gg n$, then we know there will not exist a satisfying assignment, with high probability. Here is some intuition: the probability that an assignment satisfies all m constraints is 2^{-m}, but the number of possible assignments is 2^n, so when $m < n$, it is likely we will have a satisfying assignment (unless there are pathologies, like contradicting clauses).

3.2 Refuting k-XOR

Given a random instance such that $m \gtrsim n^{k/2}$, w.h.p. there exists a short certificate that the instance does *not* have a satisfying assignment (this called *refuting* the instance). Furthermore, in this regime this certificate can be found efficiently (again, w.h.p.) [BM16].

It is conjectured that for k-XOR when $m \ll n^{k/2}$, distinguishing between a random instance and a $(1-\varepsilon)$-satisfiable instance is computationally hard. In the next section we describe how this implies that PAC, impromper, agnostic learning of halfspaces is hard, assuming this conjectue.

4 Learning halfspaces is hard

We will now describe a result of Daniely that demonstrates halfspaces are not efficiently agnostically PAC learnable, via the hardness of refuting random k-XOR.

[1] To differentiate this addition, we use the notation $\oplus$

Theorem 4.1 ([Dan16])**.** *Let $0 < \varepsilon < 1$ and positive integer n be given. Then, under the assumption of hardness for random k-XOR (see below), there is no poly-time, poly-sample PAC agnostic improper learning algorithm for the class of halfspaces.*

Above, we use the following hardness assumption.

Assumption 4.2. *Suppose that $m = n^{O(\sqrt{k})}$. Then it is hard to distinguish between random k-XOR formulae with m constraints and and formulae of m k-XOR constraints which are $(1-\varepsilon)$-satisfiable.*

Why is hardness for random k-XOR important for ruling out improper learning? At a conceptual level, the idea is to map $(1-\varepsilon)$-satisfiable instances of k-XOR to YES instances of the halfspaces problem (*i.e.*, $y \approxeq h(x)$), and random instances of k-XOR to NO instances with (uniform, i.i.d.) random labels. Now suppose by contradiction that we can improperly learn the YES instances, i.e. come up with a function $z \notin \mathcal{H}$ that approximates y. Then we could predict the label of the next sample with probability close to 1. This would distinguish those samples from ones with random labels, where no function can predict the label of the next sample.

Now, k-XOR and halfspaces are quite different functions, which brings us to the technical part of the reduction. Consider for example the following clause:

$$\bigoplus_i a_i x_i.$$

Thinking of a_i's and x_i's as taking values in $\{-1, 1\}$, we observe that whether this clause is satisfied or not depends only on the (weighted) sum $\sum a_i x_i$. We can represent this clause using the function $f : [-k, k] \to \{0, 1\}$ which satisfies

$$f(\sum a_i x_i) = \bigoplus_i a_i x_i. \tag{1}$$

(f is basically the parity function.) Eq. (1) determines the value of f on $k+1$ values, so we can extend it to a degree-k polynomial from the reals to the reals.

Once we have a degree-k polynomial, we can represent it as an affine function over n^k variables (where each variable represents a monomial). We can now try to learn the halfspace defined by this affine function, but our learning algorithm expects to see $\mathsf{poly}(n^k) \gg n^{\sqrt{k}}$ samples, which is too much to refute Assumption 4.2.

The trick is to observe that while in order to represent f exactly we really need degree k, we can approximate it with degree $O(\sqrt{k})$. Specifically, for a random $x \in \{-1, 1\}^k$, we have by Hoeffding's inequality that $\sum a_i x_i \in [-O(\sqrt{k}), O(\sqrt{k})]$ with probability $1-\varepsilon$. (Actually x may not be random because we particularly care about the value of f for the optimal assignment, but fixing any x and the values of all the clauses, we can choose $a_2, \ldots, a_k$ at random.) Hence, it suffices to fix the values of f only in the interval $[-O(\sqrt{k}), O(\sqrt{k})]$. This gives a degree $O(\sqrt{k})$-polynomial, and in turn an affine function over $n^{O(\sqrt{k})}$ variables. Now our learning algorithm expects $\mathsf{poly}(n^{O(\sqrt{k})}) = n^{O(\sqrt{k})}$ samples, which by Assumption 4.2 is not enough to get an efficient algorithm.

References

[BM16] Boaz Barak and Ankur Moitra. Noisy tensor completion via the sum-of-squares hierarchy. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 417–445, 2016.

[Dan16] Amit Daniely. Complexity theoretic limitations on learning halfspaces. pages 105–117, 2016.

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.