CSCI699: Machine Learning Theory Fall 2023

Lecture 6: Learning with Noise, and Statistical Queries

Instructor: Vatsal Sharan

These lecture notes are based on scribe notes by Chandra Sekhar Mukherjee, Fatih Erdem Kizilkaya, and Neel Patel.

1 Representation Independent Hardness Results for Learning

We were exploring a connection between cryptography and learning last time. As we discussed last time, cryptography must rely on stronger average-case assumptions than worst-case assumptions such as $P \neq NP$. One common assumption is that of *one way functions.*

Definition 1 (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 polynomial time but for any randomized polynomial time algorithm A and for any polynomial $p()$, we have*

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

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

A candidate one-way function is the Discrete Cube Root function. Let $N = p \cdot q$ b a product of two primes of roughly equal length. The Discrete Cube Root Let $f_N(x)$ is defined as $f_N(x) = x^3 \bmod N$.

If we know p and q, it can be shown that this function is invertible in polynomial time (using Euclid's GCD algorithm). However, it is widely believed that this function is hard to invert without knowledge of p and q. This forms the basis of the famous RSA cryptosystem.

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

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

The Discrete Cube Root Assumption (DCRA) states that given N and $y = f_N(x)$ for some random $x \in \{0,1\}^d$, it is hard to compute x in polynomial time. We will now derive a learning problem from the Discrete Cube Root function.

1.1 A Learning Problem where Outputs are Indistinguishable from Random

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

We would like to argue that if there is a learning algorithm for the concept class $\mathcal{C}$, then we can use that algorithm to invert the Discrete Cube Root function for random inputs. Suppose there is such an algorithm A. For A to learn, it needs access to labelled examples for training. One way to get labelled examples would be to sample x at random from $\{0,1\}^d$, and have its label be $f_N^{-1}(x)$.

But we cannot actually create labeled examples this way, since we do not know a way of inverting $f_N(x)$.

There's a simple workaround which relies on the fact that the one-way function is simple to evaluate even though it is hard to invert. We can sample x at random from $\{0,1\}^d$, compute $f_N(x)$, and have the labeled datapoint be $(f_N(x), x)$ where the input is $f_N(x)$ and the desired output is x. We can actually show that if x is sampled at random from $\{0,1\}^d$, then $f_N(x)$ is also uniformly distributed on $\{0,1\}^d$ (though this is unnecessary for the proof).

If this algorithm is a valid PAC learner then we can train it to obtain error at most ϵ on examples $(f_N(x), x)$ for x drawn at random. The error guarantee of the algorithm implies that when it is given $f_N(x)$ for a uniformly randomly drawn x, then it can find the inverse with probability $1 - \epsilon$. This violates DCRA, and hence no such algorithm can exist.

Therefore no efficient PAC learning (proper or improper) exists for learning $\mathcal{C}$. Note that the problem is statistically rather easy, in fact only $O(d)$ samples suffice for learning. To verify note that the concept class $\mathcal{C}$is parameterized by two numbers p and q, each having lenght at most d. Therefore, $|\mathcal{C}| \leq 2^{2d}$, and is thus learnable with $\mathcal{O}(d)$ samples.

1.2 Halfspaces: Hardness and Learnability

The above hardness result shows that a concept class based on the discrete cube root function is hard to learn. Though the discrete cube root finds many applications in cryptography, it is arguably not a function we commonly expect to see when solving practical lerning problems. Are there more natural concept classes which are also hard to learn, even improperly?

Some recent work has shown that it is in fact possible to show hardness of improper learning under certain complexity assumptions.

Consider the class of halfspaces, also called linear threshold functions (LTFs)

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

You will prove the following result in HW2.

Theorem 2. *The class of halfspaces is efficiently PAC learnable.*

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

Theorem 3 (Informal, see [1])**.** *Under "hardness of refuting random 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 between the following two cases with non-negligible probability:*

1. *The labels are uniformly random coin flips.*
2. *There is some LTF which gets error at most* 10% *on the labels.*

Since it is not possible to even tell if the error rate is 50% or 10%, agnostic learning is also hard. You can read more about the hardness assumption below if you like.

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.

We wefirst discuss the k-XOR problem, which then gives rise to the stronger "random k-XOR" assumption.

Proposition 4. *The following is true for random k-XOR:*

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].*

The "refuting random k-XOR" assumption asserts the following.

Claim 5. *Given $n \leq d^{\sqrt{k}\log k}$ terms, it is hard to distinguish between*

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

We will explore parities further next lecture, and if you like you can come back to this problem after that.

Therefore even this seemingly simple class of linear predictors is most likely hard to learn agnostically.

Given the fact that in practice various algorithms for learning halfspaces seem quite successful, it is natural to ask if we can loosen the requirement of agnostic learning to make learning computationally tractable, even in theory.

Agnostic learning in its glorious generality places no assumptions on the margin distribution of the instances x, or on the conditional distribution of the labels y given x. One relaxation is to require the learner to only work on some more natural and benign classes of distributions over the datapoints, which are perhaps more reflective of what might be seen in practice.

Theorem 6 ([3])**.** *The class of halfspaces is efficiently agnostically learnable if the datapoints are drawn from the uniform distribution over $\{-1,1\}^d$ or the unit sphere, or the Gaussian distribution over $\mathbb{R}^d$.*

We can also relax the assumption that the labels of the datapoints could be arbitrary. Recall that with the realizability assumption we said that the labels of all datpoints must exactly be given by some underlying concept from the concept class. With agnostic learning, the points incorrectly

labeled by the target could have arbitrary labels, and the labels could be adversarially chosen to trick the learning algorithm. We can consider something in between, where the labels have some random noise.

Theorem 7 ([4])**.** *Halfspaces are efficiently PAC learnable under random noise.*

Our next topic of study will be to understand learning under random classification noise.

This story we saw with halfspaces here is part of a more general effort in learning theory: *How can one reconcile the practical success of various algorithmic techniques (more recently neural networks and gradient-based approaches) with worst case hardness? What about real-world instances makes these problems tractable?*

2 Learning with Random Classification Noise (RCN)

As we discussed before, agnostic learning quickly gets hard, here is another hardness result even for simple classes such as rectangles and conjunctions (this time only for proper learning though).

Theorem 8 ([5])**.** *There is no efficient proper learning algorithm for agnostically learning rectangles in $\mathbb{R}^d$ or conjunctions, unless* $\mathsf{RP} = \mathsf{P}$.

We will now consider the random classification noise model which is not as worst-case, and we will show that both rectangles and conjunctions can actually be learned in this model.

Given a set of examples $\mathcal{X}$ with corresponding binary labels $\mathcal{Y}$, suppose that each label is flipped with probability η (this is also known as the white-noise model because the noise level is the same for every datapoint). Also note that only the labels are noisy, there is no noise in the datapoints. We can model this setting using the following oracle:

Noisy Example Oracle: Given a hypothesis $c(x) : \mathcal{X} \to \mathcal{Y}$ and a distribution D over $\mathcal{X}$, the oracle $\mathsf{EX}^{\eta}(c, D)$ is defined as follows:

- Draw an example $x \sim D$ from $\mathcal{X}$
- With probability $1 - \eta$, return $(x, c(x))$
- With probability η, return $(x, 1 - c(x))$

Criterion for Success: With probability $1 - \delta$, we want to find a hypothesis h such that:

$$error(h; c, D) = \Pr_{x \sim D}[c(x) \neq h(x] \leq \varepsilon$$

where c is the hypothesis which determines the true labels.

We also restrict η to lie in $[0, 1/2)$ since learning is impossible when $\eta = 1/2$. As η approaches $1/2$ learning becomes harder, so the guarantees for our algorithms will depend polynomially in $1/(1 - 2\eta)$.

Definition 9 (PAC Learning with RCN). *Let $\mathcal{C}$ be a concept class and $\mathcal{H}$ be a hypothesis class over $\mathcal{X}^d$. We say that $\mathcal{C}$ is PAC learnable with RCN if there exists a learning algorithm A such that for all $c \in \mathcal{C}$, and for all distributions D over $\mathcal{X}^d$, and for all $\varepsilon, \delta \in (0, 1/2)$ if A is given access to $\mathsf{EX}^\eta(c, D)$ and inputs ε, δ and $\eta_0 \in (\eta, 1/2)$, then A outputs h such that $error(h; c, D) \leq \varepsilon$ with probability $1 - \delta$ (probability includes randomness in noisy labels).*

We say that C is efficiently PAC learnable with RCN if A also runcs in time $\text{poly}(d, 1/\varepsilon, 1/\delta, 1/(1-2\eta_0))$.

Algorithm for Learning Conjunctions

The algorithm we saw in last lecture for learning conjunctions fails miserably under random noise. This is because the algorithm is very sensitive to every single datapoint (verify that given enough datapoints in the presence of noise, that algorithm will eventually drop every literal and predict 1 on every datapoint). We now give a robust algorithm that relies on aggregate statistics instead of a single example.

Suppose that l is a literal ($\bar{x_i}$ or x_i). We define

- $P_0(l) = \Pr_{a \sim D}[l \text{ is 0 in assignment } a]$,
- $P_{bad}(l) = \Pr_{a \sim D}[l \text{ is 0 in assignment } a \text{ and the true label of } a \text{ is } c(a) = 1]$.

Note that if l is in conjunction $c(x)$, then $P_{bad}(l) = 0$. We say that

- l is *significant* if $P_0(l) \geq \varepsilon/8d$,
- l is *harmful* if $P_{bad}(l) \geq \varepsilon/8d$.

where d is the number of variables.

Claim: If h is conjunction of all significant literals that are not harmful, then $error(h; c, D) \leq \varepsilon/2$.

Proof. We can decompose $error(h; c, D)$ into (i) and (ii) as follows:

$$error(h; c, D) = \underbrace{\Pr_{a \sim D}[h(a) = 1 \wedge c(a) = 0]}_{\text{(i)}} + \underbrace{\Pr_{a \sim D}[h(a) = 0 \wedge c(a) = 1]}_{\text{(ii)}}$$

(i) This probability corresponds to the events where $c(x)$ has a literal l that $h(x)$ does not have (and l is set to 0 in assignment a). Note that l cannot be a harmful literal by definition. Moreover, since h is conjunction of all significant literals that are not harmful, l must be insignificant. Then, since we have $2d$ literals (including negations), we get (i) $\leq 2d \cdot \varepsilon/8d = \varepsilon/4$.

(ii) This probability corresponds to the events where $h(x)$ has a literal l that $c(x)$ does not have (and l is set to 0 in assignment a). By definition, l is a significant literal that is not harmful. So similarly, we get (ii) $\leq 2d \cdot \varepsilon/8d = \varepsilon/4$.

Thus, $error(h; c, D) =$ (i) + (ii) $\leq \varepsilon/2$. ■

This algorithm requires access to the statistics $P_0(l)$ and $P_{bad}(l)$ for every literal l. We can first verify that when we are in the noiseless case, then we can estimate these by taking a large set of samples and getting concentration with Chernoff's bound (we bound $error(h; c, d)$ by $\varepsilon/2$ instead of ε to allow for this additional slack). We can also estimate these statistics using noisy examples, as we shall see soon.

3 Statistical Query (SQ) Learning

Motivated by the previous example where the algorithm only relied on aggregate statistics and hence had hope of working in the presence of noise, we define the following oracle.

Statistical Query Oracle: Given a hypothesis $c(x) : \mathcal{X} \to \mathcal{Y}$ and a distribution D over $\mathcal{X}$, the oracle $\mathsf{STAT}(c, D)$ is defined as follows:

- A query to $\mathsf{STAT}(c, D)$ is a pair (ϕ, τ) where $\phi : \mathcal{X} \times \{0, 1\} \to \{0, 1\}$ and $\tau \in (0, 1)$.
- Let $P_\phi = \Pr_{x \sim D}[\phi(x, c(x)) = 1]$
- $\mathsf{STAT}(c, D)$ returns $\hat{P}_\phi$ satisfying $\hat{P}_\phi \in [P_\phi - \tau, P_\phi + \tau]$.

Examples: The statistical query oracle responds with an approximation of the fraction of examples which satisfy the query made by the algorithm. Many natural queries can be written as statistical queries:

1. Consider a spam classification task. The following might be a statistical query that we ask to the oracle:

   "What fraction of e-mails labelled as spam (label is 1) have the words `Urgent` *and* `Free` *but not the word* `USC`*, and the numbers of words is less than or equal to 20?"*

2. We can also estimate $P_0(l)$ from our conjunction algorithm by query ϕ_l where

$$\phi_l(x, y) = \begin{cases} 1 & \text{if } l \text{ is 0 in } a \\ 0 & \text{otherwise} \end{cases}$$

   since $P_{\phi_l} = \mathop{\mathbb{E}}_{x \sim D}[\phi_l(x, c(x))] = P_0(l)$.

   Similarly, we can estimate $P_{bad}(l)$ from our conjunction algorithm by query ϕ'_l where

$$\phi'_l(x, y) = \begin{cases} 1 & \text{if } l \text{ is 0 in } a \text{ and } y = 1 \\ 0 & \text{otherwise} \end{cases}$$

   since $P_{\phi'_l} = \mathop{\mathbb{E}}_{x \sim D}[\phi'_l(x, c(x))] = P_{bad}(l)$.

We can now define SQ learning.

Definition 10 (SQ Learning). *Let $\mathcal{C}$ be a concept class and $\mathcal{H}$ be a hypothesis class. We say that $\mathcal{C}$ is efficiently PAC learnable from statistical queries using $\mathcal{H}$, if there exists an algorithm A, and polynomials $p(\cdot, \cdot)$, $q(\cdot, \cdot)$ and $r(\cdot, \cdot)$ such that for all $c \in \mathcal{C}$, and for all distributions D over $\mathcal{X}^d$, and for all $\varepsilon \in (0, 1/2)$, if algorithm A is given access to $\mathsf{STAT}(c, D)$ and input ε then*

- *For each query (ϕ, θ) made by algorithm A, the predicate ϕ can be evaluated in time $q(1/\varepsilon)$ and $1/\tau$ is bounded by $r(1/\varepsilon, d)$.*
- *Algorithm A runs in time $p(1/\varepsilon, d)$.*

– *Algorithm A outputs hypothesis $h \in \mathcal{H}$ such that $error(h; c, D) \leq \varepsilon$.*

Remark 11. *Notice that there is no confidence parameter δ in definition of SQ learning. We used δ before to take care of the probability of seeing a set of "bad examples". However, the example oracle $\mathsf{EX}(c, D)$ has been replaced by statistical query oracle $\mathsf{STAT}(c, D)$, which is deterministic.*

We can show that SQ learnability implies PAC learnability.

Theorem 12. *If a concept class $\mathcal{C}$ is efficiently SQ learnable using a hypothesis class $\mathcal{H}$, then $\mathcal{C}$ is efficiently PAC learnable using $\mathcal{H}$.*

Proof. *Exercise: Implement $\mathsf{STAT}(c, D)$ using $\mathsf{EX}(c, D)$.* ■

4 SQ Learning $\implies$ PAC Learning in presence of RCN

In this section, we will show that if a concept class is efficiently SQ learnable then it is also PAC learnable.

Theorem 13. *If a concept class $\mathcal{C}$ is efficiently SQ learnable then $\mathcal{C}$ is PAC learnable in the presence of random classification noise.*

Proof. In order to prove the theorem, we need to show that given access to EX^η, we can simulate $\mathrm{STAT}(c, D)$ oracle with bounded (low) failure probability. It is helpful to map the labels from $\{0, 1\}$ to $\{\pm 1\}$. Similarly any SQ $\phi(x, c(x))$ also maps to $\{\pm 1\}$, $\phi(x, c(x)) : \mathcal{X} \times \{\pm 1\} \to \{\pm 1\}$.

Now we can decompose any SQ $\phi(x, c(x))$ as:

$$\begin{aligned}
\underset{x\sim D}{E}[\phi(x, c(x))] &= \underset{x\sim D}{E}[\phi(x, 1) \cdot \mathbb{1}(c(x) = 1)] + \underset{x\sim D}{E}[\phi(x, -1) \cdot \mathbb{1}(c(x) = -1)] \\
&= \underset{x\sim D}{E}\left[\phi(x, 1) \cdot \left(\frac{1 + c(x)}{2}\right)\right] + \underset{x\sim D}{E}\left[\phi(x, 1) \cdot \left(\frac{1 - c(x)}{2}\right)\right] \\
&= \underbrace{\frac{1}{2}\left(\underset{x\sim D}{E}[\phi(x, 1)] + \underset{x\sim D}{E}[\phi(x, -1)]\right)}_{\text{“target-independent query”}} \\
&\quad + \underbrace{\frac{1}{2}\left(\underset{x\sim D}{E}[\phi(x, 1) \cdot c(x)] + \underset{x\sim D}{E}[\phi(x, -1) \cdot c(x)]\right)}_{\text{“correlational query”}}.
\end{aligned}$$

Target-independent queries do not depend on the underlying concept $c(x)$. Therefore, it is straightforward to estimate them using noisy data. Using concentration bounds, we can show that our estimate has error $O(\tau)$ with probability $1 - \delta'/2$ with $O\left(\frac{1}{\tau^2}\log(2/\delta')\right)$ queries to EX^η (where we will ignore the labels in the respones of the oracle).

We will see correlational queries more later too, so it is helpful to define a Correlational Statistical Query (CSQ) oracle:

Correlational Statistical Query Oracle: Given a hypothesis $c(x) : \mathcal{X} \to \mathcal{Y}$ and a distribution D over $\mathcal{X}$, the oracle $\mathsf{CSQ}(c, D)$ is defined as follows:

– A query to $\mathsf{CSQ}(c, D)$ is a pair (ψ, τ) where $\psi : \mathcal{X} \to \{\pm 1\}$ and $\tau \in (0, 1)$.

– Let $P_\psi = E_{x\sim D}[\Psi(x) \cdot c(x)]$

– $\mathsf{CSQ}(c, D)$ returns $\hat{P}_\psi$ satisfying $\hat{P}_\psi \in [P_\psi - \tau, P_\psi + \tau]$.

We now show that the oracle $\mathsf{CSQ}(c, D)$ can be simulated with access to noisy examples. For simplicity, we begin with the case where the noise level η is already known. Define a $\{\pm 1\}$ random variable Z which is 1 w.p. $1 - \eta$ and -1 w.p η. Note that for the noisy example oracle, $y = c(x)Z$. Let the distribution of which has distribution $B(\eta)$.

$$\begin{aligned} E_{(x,y)\sim EX^\eta(c,D)}[\Psi(x) \cdot y] &= E_{x\sim D}[E_Z[\Psi(x) \cdot c(x) \cdot Z]] \\ &= E_{x\sim D}[\Psi(x) \cdot c(x) E_Z[Z]] \\ &= (1 - 2\eta) E_{x\sim D}[\Psi(x) \cdot c(x)] \\ \implies E_{x\sim D}[\Psi(x) \cdot c(x)] &= \frac{1}{1 - 2\eta} \cdot E_{(x,y)\sim EX^\eta(c,D)}[\Psi(x) \cdot y]. \end{aligned}$$

Since we have access to the oracle $EX^\eta(c, D)$, we can estimate $E_{(x,y)\sim EX^\eta(c,D)}[\Psi(x) \cdot y]$. Note that some error τ' in the estimate of $E_{(x,y)\sim EX^\eta(c,D)}[\Psi(x) \cdot y]$ will translate to an error $\tau/(1 - 2\eta)$ in the estimation of $E_{x\sim D}[\Psi(x) \cdot c(x)]$. Therefore to approximate $E_{x\sim D}[\Psi(x) \cdot c(x)]$ to some error $O(\tau)$ with probability $1 - \delta'/2$ we need to take $O\left(\frac{\log(2/\delta')}{(1-2\eta)^2\tau^2}\right)$ samples from the oracle $EX^\eta(c, D)$.

Therefore for any query ϕ made by the algorithm, with probability $1 - \delta'$ we can return $\hat{P}_\phi \in [P_\phi - \tau, P_\phi + \tau]$ with $O\left(\frac{\log(2/\delta')}{(1-2\eta)^2\tau^2}\right)$ calls to the noisy example oracle.

We can repeat this for all SQ queries made by the SQ learning algorithm. If the algorithm makes m queries, we can set $\delta' = \delta/m$ and by union bound, we can bound the overall failure probability by δ.

The only thing that remains is to get rid of our earlier assumption that we know η. In the learning with RCN setup, we know some $\eta_0 \geq \eta$ (where $\eta_0 < 1/2$). Now, consider a small $\Delta > 0$, and construct Δ-net for all possible values of η, i.e.

$$\Gamma = \left\{ i \cdot \Delta : 0 \leq i \leq \left\lfloor \frac{\eta_0}{\Delta} \right\rfloor \right\}$$

Try all values of $\hat{\eta} \in \Gamma$. Let h_i be the hypothesis produced by the algorithm for $\hat{\eta} = i\Delta$. You should verify that the previous simulation which assumes knowledge of η still works if η is known up to some error Δ, for some $\Delta = O(\tau)$. Therefore at least one of the h_i must be such that it has small error with respect to the underlying concept $c(x)$ on distribution D. We can simply check the error of all the h_i on a fresh sufficiently large set of samples, and pick the best one. Since at least one h_i gets error less than the desired error rate ϵ, the best h_i will also get error at most ϵ.

If the algorithm makes m queries to the SQ oracle each of which has tolerance at most τ, then the overall number of queries made to the noisy example oracle to ensure that with probability $1 - \delta$ the response to every query is accurate up to tolerate τ is $O\left(\frac{\log(m/\delta)}{(1-2\eta)^2\tau^3}\right)$. Since the algorithm is an efficient SQ algorithm $m = \text{poly}(d)$ and $1/\tau = \text{poly}(d)$, and hence the overall number of queries made to the noisy example oracle is still polynomial in d.

■

Using this result, we can now say that the class of conjunctions is efficiently learnable with RCN.

Theorem 14. *The class of conjunctions is efficiently learnable in the SQ model. Therefore, it is efficiently learnable with RCN.*

Proof. Our conjunction learning algorithm can be implemented in the SQ model. ■

5 Privacy, SQ Learnability and SQ Dimension

The statistical query oracle is useful more generally, beyond showing learnability with random classification noise. One nice connection is to the notion of *differential privacy*.

Definition 15. *Consider some randomized algorithm A which takes some dataset S as input, and computes some output $A(S)$ (this could be some hypothesis built from the data). Consider some datasets S and S' which differ in just one example. We say that A satisfies ϵ-differential privacy if for any set T in the range of $A(\cdot)$ and for any such datasets S and S', we have that*

$$\frac{P(A(S) \in T)}{P(A(S') \in T)} \leq e^{\epsilon}.$$

If A outputs some hypothesis based on the training set S, then the set T is just some subset of hypothesis. The definition requires for any subset of hypothesis, the probability of a hypothesis from that subset being output is similar for datasets which differ in exactly one datapoint. This means that we cannot figure out if any one specific individual was in the training dataset or not by looking at the output of the algorithm. Since we cannot even figure out if some individual was in the dataset or not based on the algorithm's output, we also cannot infer any particular individual's attributes based on the algorithm's output.

SQ learnability implies learnability with aggregate statistics, without looking at individual datapoints, which is intuitively desirable for privacy. It can in fact be shown that if SQ learnability implies learnability with differential privacy.

Theorem 16 (Informal, [6, 7])**.** *If a concept class $\mathcal{C}$ is efficiently SQ learnable, then $\mathcal{C}$ is efficiently PAC learnable with differential privacy.*

The proof of this result is not complicated after developing some understanding of differential privacy. The basic idea is to add some noise to the labels to preserve privacy, which as we showed in the previous section also does not hurt SQ learnability.

It turns out that most algorithms can be implemented in the SQ model as well, for example gradient descent based algorithms for optimization. In fact, for most problems where we do not have algorithms which work in the SQ models, we often do not have any efficient algorithms at all for solving the problem. Therefore the SQ model can be a useful tool for understanding when a certain problem can be solved computationally efficiently.

Another major reason for why the SQ model is useful for understanding when there are efficient algorithms is that it is usually much more tractable to show that there are no efficient SQ algorithms for a certain problems. In fact, there is a notion of dimension which characterizes learnability in the SQ model. To define this notion, we first define uncorrelated functions:

Definition 17. *Two functions f, g defined on the same domain are uncorrelated if*

$$\Pr_{x \sim D}[f(x) = g(x)].$$

Now, we are ready to define SQ dimension. Earlier in the class we saw VC dimension, which characterizes how many samples are needed for learning. We will show that SQ dimension closely captures the runtime needed for learning within the SQ framework. As we discussed the SQ framework encompasses many known classes of algorithms, so such a characterization can be quite useful.

Definition 18. *The SQ-dimension of a class $\mathcal{C}$ w.r.t. a distribution D over $\mathcal{X}$ is the size of the largest subset $\mathcal{C}' \subset \mathcal{C}$ such that for all $f, g \in \mathcal{C}'$*

$$\left| \Pr_{x \sim D}[f(x) = g(x)] - 1/2 \right| < 1/|C'|.$$

Let us develop some intuition for this definition. The SQ dimension of some concept class is large if there is a large set of almost uncorrelated functions in that class. Such a class is harder to learn because any successful algorithm must be able to distinguish between every pair of functions in this large set of uncorrelated functions (since every pair makes very different predictions), but distinguishing between every pair in a large set could take a lot of statistical queries.

Remark 19. *What is the relationship between VC dimension and SQ dimension? Intuitively, it feels like if the VC dimension is large then the SQ dimension must also be large. This is because the VC dimension being large implies that there are functions within the class which can make completely different predictions on a set of samples, which should imply that the SQ dimension is also large. This can be shown formally (Section 3.1.3 in [8]). As we shall see next week, the SQ dimension can also be exponentially larger than the VC dimension.*

SQ dimension closely characterizes learnability in the SQ model. Before we show that, we need to define a notion of *weak learning*.

Definition 20 (Weak Learning)**.** *An algorithm A is a weak learner with advantage γ for class $\mathcal{C}$ if: for any dist. D and any target $c \in \mathcal{C}$, given access to $EX(c, D)$, w.p. $(1-\delta)$, produces a hypotheses with error$(h; c, D) \leq \gamma$.*

Later in the course, we will show that weak learning actually implies the usual (strong) PAC learning. We are now ready to show upper and lower bounds for learnability using SQ dimension. The following results, which we will prove in the next lecture, show that learnability in the SQ model is closely captured by the SQ dimension of the hypothesis class.

Theorem 21. *If SQ-DIM$_D$ = poly(d), then we can efficiently "weak learn" $\mathcal{C}$ over D (get error rate $\leq 1/2 - 1/$poly(d)) using a SQ-learning algorithm.*

Theorem 22. *If SQ-DIM$_D$ > poly(d) then we cannot efficiently learn $\mathcal{C}$ over D by any SQ-algorithm.*

6 Further Reading

You can read Chapter 6 of Kearns-Vazirani [9] (available using your USC account here) for more on how cryptographic assumptions give hard learning problems, including more discussion for the Discrete Cube Root problem. Chapter 5 is a good reference for learning with RCN. The survey [8] is a good reference for further reading about the SQ model.

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*, 22:35–52, 1998.

[5] Shai Ben-David, Nadav Eiron, and Philip M Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

[6] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, 2005.

[7] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[8] Lev Reyzin. Statistical queries and statistical algorithms: Foundations and applications. *arXiv preprint arXiv:2004.00557*, 2020.

[9] Michael J Kearns and Umesh V Vazirani. Computational learning theory. *ACM SIGACT News*, 26(1):43–45, 1995.