A Basic Overview of Private PAC learning

Satchit Sivakumar

April 2023

1 Definition of PAC Learning

Today, we'll be motivated by trying to theoretically model classification problems.

For example, imagine that you want to build a predictor to figure out if a CT scan image indicates the presence of a tumor or not. Our goal is to build a predictor that can take an image as an input and output whether or not the image contains a tumor.

We can model this scenario as follows: suppose that there's an underlying function operating on the pixel values of the image that determines whether the image contains a tumor or not. Unfortunately, such a function may be complicated, and we apriori do not know exactly what it is. However, we may have access to prior CT scans, and can have them annotated by medical experts to classify them into positive for presence of a tumor and negative for absence of a tumor. We can then set up a 'learning' problem, where an algorithm, on seeing this prior data (often called the 'training data'), tries to learn the underlying function.

This was what the PAC learning model tries to capture. I'll first define the model, and then explain the choices made.

Definition 1.1. *Fix a data domain $\mathcal{X}$ and a set of functions H that map $\mathcal{X} \to \{0,1\}$. Then, an algorithm A is said to PAC learn H with n samples if:*
for all distributions D over $\mathcal{X}$,
for all functions $f \in H$,
with probability at least $3/4$ over the randomness of the data and that used by the algorithm, the algorithm on seeing a dataset $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$ (where $x_1, \ldots, x_n \sim D^n$), outputs a function $g \in H$ such that

$$\mathsf{err}(g, f) = \Pr_{x \sim D}(f(x) \neq g(x)) \leq \frac{1}{4}.$$

Let's unpack the definition- what is this set of functions H (often called the 'hypothesis class'?). This is often used to model prior information that might be known about the function- for example, based on things domain experts tell us, or experimental evidence. For example, we may know or guess that the function can be represented by a complicated neural network. Such prior knowledge is encoded into the set of functions H- which for e.g., may be the set of all functions computed by neural networks with depth d and ReLu activation.

Next, this model is often called 'distribution free'. That is, we want it to work regardless of the underlying distribution. This is a reasonable requirement, because often we don't know the exact distribution that patients taking CT scans make come from for e.g., so asking that the PAC learning algorithm work for all distributions (as long as it sees examples drawn from that distribution) makes sense.

Next, we want it to work for any target function in the class (since to the best of our knowledge, the function could be any of them).

Next, the output of the algorithm will be another function. However, the function may not be exactly the function f, but close to it, so we want some error metric to verify its output by. Since we often want to predict on another patient from the same distribution for e.g. (maybe a new patient who visited the hospital), one natural thing to ask is that for such a new patient, the function output by the algorithm generally predicts correctly, i.e. $\Pr_{x \sim D}(f(x) \neq g(x))$ is small.

Finally, since the samples are randomly drawn, maybe we get a bad sample that doesn't give us enough information to learn things (for example, maybe we always sample one person). To account for this, we allow the algorithm to fail (i.e., output a bad function) with some probability.

Note that the choice of $1/4$ for failure probability is arbitrary, we chose a constant to simplify presentation, but in general, this is not necessary.

Additionally, for people who are familiar with PAC learning, the setting I am referring to is actually the realizable proper PAC learning setting under the 0-1 loss; but we will just call that PAC learning for simplicity.

1.1 Notation:

- H: hypothesis class
- f: unknown function we are trying to learn
- D: Unknown distribution over data domain $\mathcal{X}$.
- $\mathsf{err}_D(h) : \Pr_{x \in D}(f(x) \neq h(x))$. We will call this the 'true' error of the function.
- We will use $S \in \mathcal{X}^n$ to represent datasets containing examples.
- We will often consider the number of mislabeled examples on the training dataset S, and will represent this by $err_S(h) = \frac{1}{n}\sum_{i=1}^{n} 1[f(x) \neq h(x)]$. We call this the 'empirical error' of a hypothesis.

2 PAC Learning of Finite Hypothesis Classes

Next, we will design PAC learning algorithms to understand this definition a bit better. The common question of interest is how many data points does an algorithm need to see to learn a hypothesis class i.e., how large does n need to be such that there exists a PAC learner with n samples for the hypothesis class?

First, consider any finite hypothesis class H. We will show that such a class is PAC learnable with $O(\log |H|)$ samples.

The algorithm A on seeing a dataset S will simply consider an ordering of the hypothesis class and output the first function h such that $\mathsf{err}_S(h) = 0$. This strategy is frequency called 'empirical risk minimization', since it involves minimizing the empirical error.

Theorem 2.1. *Fix any finite hypothesis class H. Algorithm A when given a dataset of size $O(\log |H|)$ is a PAC learner for hypothesis class H.*

Proof. Fix a distribution D. Let the underlying true function be $h^* \in H$. We start by defining some useful sets. Let

$$H_{bad} = \Big\{h \in H \mid \mathsf{err}_D(h) > \frac{1}{4}\Big\},$$

and

$$S_{bad} = \Big\{S \in \mathcal{X}^n \mid \exists h \in H_{bad} \text{ such that } \mathsf{err}_S(h) = 0\Big\}.$$

Note that $\Pr_{S\sim D^n}(err_D(A(S)) > \frac{1}{4}) = Pr_{S\sim D^n}(A(S) \in H_{bad}) \leq Pr_{S\sim D^n}(S \in S_{bad})$, because the algorithm always outputs a hypothesis that is consistent with the training data, and the bad case is when the hypothesis it outputs does not generalize. Now, we analyze the latter probability.

$$\begin{aligned} Pr_{S\sim D^n}(S \in S_{bad}) &= \Pr[\exists h \in H_{bad} \text{ such that } \mathsf{err}_S(h) = 0] \\ &\leq \sum_{h \in H_{bad}} \Pr[\mathsf{err}_S(h) = 0] = \sum_{h \in H_{bad}} \prod_{i=1}^{n} \Pr_{x\sim D}[h(x) = h^*(x)] \\ &\leq \sum_{h \in H_{bad}} (1 - \frac{1}{4})^n \leq |H| e^{-\frac{n}{4}}. \end{aligned}$$

A simple calculation shows that if $n = O(\log |H|)$, this probability is less than 0.25, completing the proof. ■

Next, we consider a simple but infinite class of functions. Consider the real interval $[0, 1]$. Each function is parametrized by a value in this interval, and will label all points below that value with 0 and all points above that value with 1. This class of functions is commonly called the class of 'thresholds'.

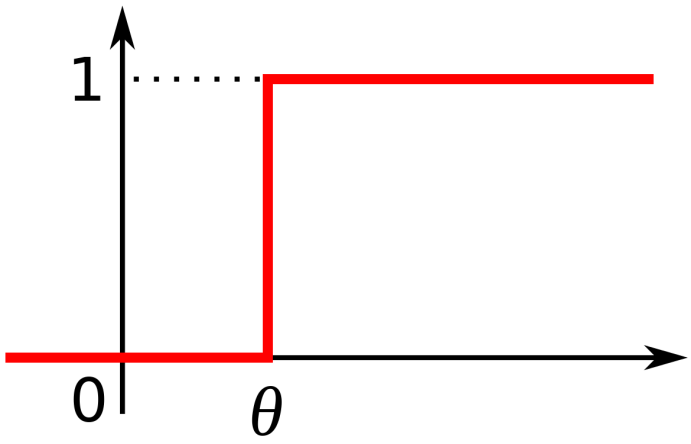

Figure 1: A threshold function parametrized by θ, where the X-axis represents the data values and the Y-axis represents the function value.

Clearly, since there are infinitely many real numbers between 0 and 1, the prior strategy does not provably give us any PAC learner with a finite number of samples. However, it turns out that via a different proof, one can show that a similar empirical risk minimization algorithm A is a PAC learner with only $O(1)$ samples (roughly 10)! While there are proofs of this that are not very complicated, we omit it since we want to move to discuss the relatonship to differential privacy.

Theorem 2.2. *Let H be the class of threshold functions over $[0, 1]$. Algorithm A when given a dataset of size $O(1)$ is a PAC learner for H.*

Finally, before moving on, I will mention that for any hypothesis class, there is a combinatorial dimension of the hypothesis called its VC dimension that exactly characterizes the sample complexity of learning it in the PAC model. See for example, the book by Shai Shalev-Shwartz and Shai Ben-David for more details [9].

3 Private PAC learning

Next, we get to the main point of this talk. The classification tasks that we are interested in involve learning from data. What if this data is sensitive, and we want to ensure that an adversary doesn't learn anything about individuals from the output of a learning algorithm? This is particularly important in the CT scan example we considered, for example.

Hence, it is natural to ask whether PAC learning can be performed by algorithms that are differentially private with respect to their training data. Formally, we can define PAC learning in this model as follows (this was first investigated by Kasiviswanathan et al. [7]):

Definition 3.1. *Fix a data domain $\mathcal{X}$ and a set of functions H that map $\mathcal{X} \to \{0, 1\}$. Then, an algorithm A is said to (ε, δ)- privately PAC learn H with n samples if:*

1. *For any $S \in \mathcal{X}^n$ fed to the algorithm A, it is (ε, δ)-differentially private with respect to S.*

2. A is a PAC learner for hypothesis class H.

Note that we want to be differentially private REGARDLESS of what the training data is (i.e. regardless of whether it's actually drawn from a distribution or not). On the other hand, accuracy as defined earlier is defined probabilistically.

A natural question is whether private PAC learning is possible, and if so how many more samples are necessary to do it when compared with non-private PAC learning? We start addressing this question by considering finite hypothesis classes again.

Recall the learner without privacy which simply output a function that was consistent with the training data. However, if the training data is changed by one example, this could completely change the function output by the learner. Hence, we need a different strategy. It turns out that a clever use of the exponential mechanism solves this problem.

Let $\mathsf{score}(S, h) = -n \times \mathsf{err}_S(h)$, that is, we measure the performance of a hypothesis on a dataset by the number of misclassified examples (and apply a negative sign to ensure that a higher number of misclassified examples corresponds to a lower score). Algorithm A' applies the exponential mechanism over output space H to sample a hypothesis with probability proportional to $e^{-\frac{\varepsilon \mathsf{score}_S(h)}{2}}$.

Recall the accuracy guarantee of the exponential mechanism.

Lemma 3.2 (Exponential Mechanism [8])**.** *Let H be a set of outputs and $\mathsf{score} : H \times \mathcal{X}^n \to \mathbb{R}$ be a function that measures the quality of each output on a dataset S. Assume that for every $f \in H$, the function $\mathsf{score}(h)$ has ℓ_1-sensitivity at most Δ. Then, for all $\varepsilon, n > 0$ and for all datasets $S \in \mathcal{X}^n$, there exists an $(\varepsilon, 0)$-DP mechanism that outputs an element $h \in H$ such that, for all $a > 0$, we have*

$$\Pr[\max_{h^* \in [H]} \mathsf{score}(S, h^*) - \mathsf{score}(S, h) \geq 2\Delta \frac{(\ln |H| + a)}{\varepsilon}] \leq e^{-a}.$$

Theorem 3.3 (Private PAC learner for finite classes [7])**.** *Fix any finite hypothesis class $H, 0 < \varepsilon \leq 1$. Algorithm A' when given a sample of size $O(\frac{\log(H)}{\varepsilon})$ is an ε-DP private PAC learner for hypothesis class H.*

Proof. Recall that the exponential mechanism is differentially private whenever it's applied to a function with sensitivity 1. It is easy to verify that the score function defined has sensitivity 1 (it corresponds to the number of misclassified points in the dataset). Hence A' is ε-DP.

To prove utility, we argue two facts. Firstly, similar to the non-private case, we argue that with high probability, any hypothesis with true error larger than $1/4$ has training error larger than $1/8$. Then, we argue that for sufficiently large sample size, the exponential mechanism with high probability outputs a hypothesis with training error at most $\frac{1}{16}$.

Fix a distribution D. Let the true function be h^*. We start by defining some useful sets. Let

$$H_{bad} = \Big\{ h \in H \mid \mathsf{err}_D(h) > \frac{1}{4} \Big\},$$

and

$$S_{bad} = \Big\{ S \in \mathcal{X}^n \mid \exists h \in H_{bad} \text{ such that } \mathsf{err}_S(h) \leq \frac{1}{8} \Big\}.$$

Now, we analyze the probability of getting a bad sample. The second to last inequality in the following sequence is by a Chernoff bound.

$$\begin{aligned} Pr_{S \sim D^n}(S \in S_{bad}) &= \Pr[\exists h \in H_{bad} \text{ such that } \mathsf{err}_S(h) \leq \frac{1}{8}] \\ &\leq \sum_{h \in H_{bad}} \Pr[\mathsf{err}_S(h) < \frac{1}{8}] \leq \sum_{h \in H_{bad}} \Pr[err_S(h) \leq \frac{1}{2}\mathsf{err}_D(h)] \\ &\leq \sum_{h \in H_{bad}} e^{-\frac{n}{8}} \leq |H| e^{-\frac{n}{8}}. \end{aligned}$$

Hence, if $n = O(\log |H|)$, we get that with probability at least 7/8, any hypothesis with true error larger than 1/4 has training error larger than 1/8, i.e. every hypothesis with training error smaller than 1/8 has true error smaller than 1/4.

Next, we invoke the accuracy guarantee of the exponential mechanism, which tells us that with probability at least 7/8, the output has training error at most 1/8. (We leave this simple calculation using Lemma 3.2 to the reader).

Hence, using a union bound, we get that with probability at least 3/4, the algorithm outputs a hypothesis with true error at most 1/4, completing the proof. ■

This tells us that for finite classes, and reasonable ε, the cost of privacy for learning finite hypothesis classes is not significant. A natural conjecture to hence make is that this extends to infinite classes, that is, any hypothesis class that can be PAC learned with a finite number of samples can also be privately PAC learned with a finite number of samples.

We will now see that this is not true, by considering again the class of threshold functions. Recall that non-privately, this was learnable with only $O(1)$ samples. On the other hand, we will see that this class is not privately learnable with finitely many samples!

Theorem 3.4. *Let H be the hypothesis class of threshold functions. Then, for all $n > 0$, there exists no $(1, o(1/n))$-DP PAC learner for H with n samples.*

3.1 Other work on Private PAC learning

Before going into the proof of this, I'll briefly describe the history of study on private PAC learning. Ever since the work by Kasviswanathan et al. in 2007, there has been a flurry of other work on private PAC learning. In 2010, Beimel et al. [2] gave a separation between proper and improper differentially private PAC learning (for the $(\varepsilon, 0)$ case) of an important infinite class of functions called point functions (proper means outputting a function in the hypothesis class, improper means the algorithm can output any function). Their improper private PAC learner had sample complexity matching the non-private sample complexity, but in contrast, they showed that there is no proper $(\varepsilon, 0)$-differentially private PAC learner for this class. In 2013, Beimel et al. [3] showed that there is a combinatorial dimension associated with any hypothesis class called its Representation Dimension that exactly characterizes the number of samples needed to PAC learn a hypothesis class under $(\varepsilon, 0)$-DP. In 2014, Feldman and Xiao [6] showed that Representation dimension of a hypothesis class was equivalent to a complexity theoretic measure called the randomized one-way communication complexity of the hypothesis class, and showed more dramatic separations between $(\varepsilon, 0)$-DP learning and non-private learning using it. They also showed a separation between $(\varepsilon, 0)$-DP learning and (ε, δ)-DP learning by exhibiting that there were concept classes with arbitrarily large Representation Dimension, but which could be learned by (ε, δ)-DP learners.

At this point, it was open whether (ε, δ)-DP learning was equivalent to non-private learning. In 2015, Bun et al. [5] showed the first separation between (ε, δ)-DP learning and non-private learning. For the class of threshold functions defined earlier, they showed that there was no proper (ε, δ)-DP learner. This was extended by Alon et al. [1] to apply to improper learners as well. Additionally, Alon et al. [1] used this result to show that all hypothesis classes that could be privately PAC learned (in the (ε, δ) sense), could also be online learned with a finite mistake bound (see these notes for a formal definition of online learning). Conversely, Bun et al. [4] proved the opposite direction, that any hypothesis class that could be online learned with a finite mistake bound could be privately PAC learned with a finite number of samples. It turns out that the parameter underlying online learning is a combinatorial dimension called the Littlestone dimension, so these works imply that a hypothesis class is approximate privately PAC learnable if and only if it has finite Littlestone dimension. However, while we have come a long way, this characterization does not come close to giving us an accurate quantitative pinning of the exact sample complexity required for approximate private PAC learning, which remains a crucial open question.

3.2 Ramsey Theory Lower Bound

Now, we are ready to prove Theorem 3.4. The proof will use concepts from graph theory, specifically, Ramsey's theorem! Our presentation loosely follows that in Section 4.2 of Mark Bun's thesis.

We will need one theorem about PAC learning that we will state without proof.

Theorem 3.5. *Let H be a hypothesis class and let $f \in H$ be an unknown function. Let $n > 0$, and let S be a dataset from the same data domain as H of size $9n$. Then, if there exists an (ε, δ)-DP PAC learner achieving $\frac{1}{4}$-accuracy on H with n samples, there exists an (ε, δ)-DP algorithm that takes in S and with probability at least $\frac{3}{4}$ only over the randomness of the algorithm, outputs a hypothesis g that mislabels at most a quarter of the samples in S.*

We call the task referenced in the second part of the above theorem the 'empirical' version of the learning task. Now, because of this theorem, it is sufficient to show that there is no private learning algorithm A that empirically learns the class of threshold functions in order to prove Theorem 3.4.

First, we will recall some graph theory concepts. Let $G = (V, E)$ be a hypergraph, where V represents the set of vertices and E the set of hyperedges. Note that in regular graphs, edges are between two vertices. However, in hypergraphs, edges constitute a subset of vertices (which might be more than 2).

An n-uniform hypergraph is one where every edge connects exactly n vertices. Regular graphs (without self loops) are 2-uniform.

A graph coloring involves assigning a color to every edge of the hypergraph. A clique in the hypergraph G is a set of $m \geq n$ vertices, such that for every subset of n of these m vertices, there is an edge in G that connects them.

A complete hypergraph is one where there is an edge between all subsets of n vertices in V.

Ramsey's theorem says that for any sufficiently large hypergraph and coloring c of that hypergraph with k colors, there exists a large clique in the graph such that every edge in the clique has the same color. Quantitatively, the version we care about is as follows.

Theorem 3.6. *Let $G = (V, E)$ be a complete n-uniform hypergraph, such that there exists a coloring of G with n colors, such that there is no clique of size $15n$ that is monochromatic. Then, $|V| \leq n^{n^{\cdots^{n}}}$ (i.e, the graph has a bounded number of vertices- the bound is an exponential tower of n with length n), or equivalently $n \geq \log^* |V|$ (i.e. the number of colors grows with the size of the graph).*

Now, we will use this theorem to prove our lower bound.

Proof of Theorem 3.4. Let $V = [0, 1]$, i.e. let every vertex correspond to a real number. Fix finite $n > 0$. Let E be the set of all n-hyper-edges in the graph (i.e. subsets of n real numbers). These will correspond to ordered, 'balanced' datasets, where the first $n/2$ elements in the ordering are labeled 0 and the next $n/2$ elements are labeled 1. Let edge $e \in E$ correspond to dataset S_e.

We will consider a coloring of the hypergraph as follows. Assume we are given a $(1, o(1/n))$-DP, empirical learning algorithm A that given any dataset S of size n, with probability at least $3/4$, outputs a real number in $[0, 1]$ such that at least $3/4$th of the dataset is correctly classified by the threshold function parametrized by that real number.

We will define a coloring from such an algorithm as follows. Let the color of edge e be the index i such that $\Pr_A(A(S) \in [x_i, x_{i+1}])$ is maximum. Note that this defines a deterministic coloring of the hypergraph with at most n colors.

First, assume by way of contradiction that $n \leq \log^* |V|$. Now, consider any subset of vertices of size $15n$. We will argue that there is a pair of edges within the corresponding induced subgraph that do not have the same color in this coloring. This would then imply that there is no monochromatic clique of size $15n$, thereby contradicting Ramsey's theorem (or more specifically, the contrapositive of the theorem). This would complete the proof.

This argument is a twist on a classic 'packing' argument in differential privacy. Let the subset of vertices correspond to values $0 \leq x_0 < x_1 < \cdots < x_{15n-1} \leq 1$. Suppose for contradiction that every edge is assigned color i. Let S be the dataset $x_0, x_1, \ldots, x_{i-1} x_i, x_{i+13n}, \ldots, x_{14n-i-1}$. Let $S_1, \ldots, S_{13n}$ be datasets where

S_j is $(x_2, \ldots, x_{i-1}, x_{i-1+j}, X_{(i-1)+j+1}, x_{i+3n}, \ldots, x_{14n-i-2}$ (each dataset corresponds to an edge). Note that the Hamming distance between S and S_j is 2 for all j.

Now, by accuracy, we have that for all j, $\Pr[A(S_j) \in] \geq \frac{3}{4}$ because an accurate threshold function needs to belong to the dataset. Hence, we have by an averaging argument that $\max_a \Pr_A[A(S_j) \in [x_a, x_{a+1}]] \geq \frac{3}{4n}$. Now, since every dataset is colored i, we have that the maximum lies in interval $[x_{i-1+j}, x_{(i-1)+j+1}]$ for dataset S_j, i.e.

$$\Pr_A\left[A(S_j) \in [x_{i-1+j}, x_{(i-1)+j+1}]\right] \geq \frac{3}{4n}.$$

By group privacy (for groups of size 2), we get that

$$\Pr_A\left[A(S) \in [x_{i-1+j}, x_{(i-1)+j+1}]\right] \geq e^{-2}[\frac{3}{4n}] - \frac{\delta}{e^2} - \frac{\delta}{e} > \frac{1}{13n}$$

Hence, since there are $13n$ different datasets, we can write

$$\sum_j \Pr_A[A(S_j) \in x_{i-1+j}, x_{(i-1)+j+1}] > \frac{1}{13n} \times 13n = 1.$$

However, the intervals $[x_{i-1+j}, x_{(i-1)+j+1}]$ are disjoint for distinct j, which means this probability sum should be < 1, which gives a contradiction, completing the proof. ■

References

[1] Noga Alon, Roi Livni, Maryanthe Malliaris, and Shay Moran. Private pac learning implies finite littlestone dimension. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 852–860, New York, NY, USA, 2019. Association for Computing Machinery.

[2] Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 437–454. Springer, 2010.

[3] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Characterizing the sample complexity of pure private learners. *J. Mach. Learn. Res.*, 20:146:1–146:33, 2019.

[4] Mark Bun, Roi Livni, and Shay Moran. An equivalence between private classification and online prediction. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 389–402. IEEE, 2020.

[5] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 634–649. IEEE Computer Society, 2015.

[6] Vitaly Feldman and David Xiao. Sample complexity bounds on differentially private learning via communication complexity. In Maria Florina Balcan, Vitaly Feldman, and Csaba Szepesvári, editors, *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *Proceedings of Machine Learning Research*, pages 1000–1019, Barcelona, Spain, 13–15 Jun 2014. PMLR.

[7] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, 2008.

[8] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103, 2007.

[9] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.