Yale University CPSC 683
Stability in Machine Learning: Generalization, Privacy & Replicability

Lecture 5: PAC Learning with Pure DP and Online Learning

Instructor: Alkis Kalavasis, alkis.kalavasis@yale.edu

Lectures 2–4 explored various techniques for controlling generalization error, emphasizing the stability of learning algorithms as a key factor. In Lecture 2, we introduced the concept of stability for learning algorithms (Bousquet and Elisseeff, 2002) and observed that it ensures a small generalization error (i.e., the training loss closely approximates the population loss). Lecture 3 demonstrated that stochastic gradient descent (SGD) on parametric models is uniformly stable.

In this lecture, we ask: what concept classes $H \subseteq \{0,1\}^{\mathcal{X}}$ can be learned by an algorithm whose output does not depend too heavily on any one input or specific training example?

Here, we investigate learning algorithms that satisfy a particular notion of stability, namely *differential privacy*, a notion that provides strong confidentiality guarantees in contexts where aggregate information is released about a dataset containing sensitive information about individuals.

Notation We will focus on binary classification tasks, let $A : (\mathcal{X} \times \{0,1\}))^n \to \{0,1\}^{\mathcal{X}}$ be a randomized learning rule that takes as input a dataset S of size n and maps it to a classifier in a randomized manner. That is for a fixed S, $A(S)$ is a distribution over classifiers and the realized classifier is determined by the internal randomness of the algorithm A. Equivalently, we can think of deterministic learning rules $A : (\mathcal{X} \times \{0,1\})^n \times \mathcal{R} \to \{0,1\}^{\mathcal{X}}$ that takes both a dataset S and a random string $r \sim \mathcal{R}$ and returns a classifier $A(S,r)$. Thus, $A(S)$ corresponds to a random variable but $A(S,r)$ is a deterministic object.

1 Differential Privacy as a form of Stability

A (randomized) algorithm (in our context, this will usually be a learning algorithm) is private if neighboring databases induce nearby distributions on its outcomes:

Definition 1: Pure Differential Privacy

A randomized algorithm A is ϵ-differentially private if for all neighboring databases S, S', and for all sets Y of outputs of A, it holds that

$$\mathop{\mathbf{Pr}}_{h\sim A(S)}[h \in Y] \leq \exp(\epsilon) \cdot \mathop{\mathbf{Pr}}_{h\sim A(S')}[h \in Y].$$

The probability is taken over the internal randomness (i.e., the random coins) of A.

Typical values for ϵ are of order 0.1. Another way to see the above is that the DP requirement corresponds to the *max-divergence*:

$$D_\infty(A(S) \parallel A(S')) = \sup_{Y \in \text{range(A)}} \log\left(\mathop{\mathbf{Pr}}_{A(S)}[h = Y] / \mathop{\mathbf{Pr}}_{A(S')}[h = Y]\right).$$

Max-divergence can be seen as a worst-case analogue of KL divergence, which takes a weighted average over the possible outputs of the algorithm, similar to the way that min-entropy is a worst-case analog of Shannon entropy.

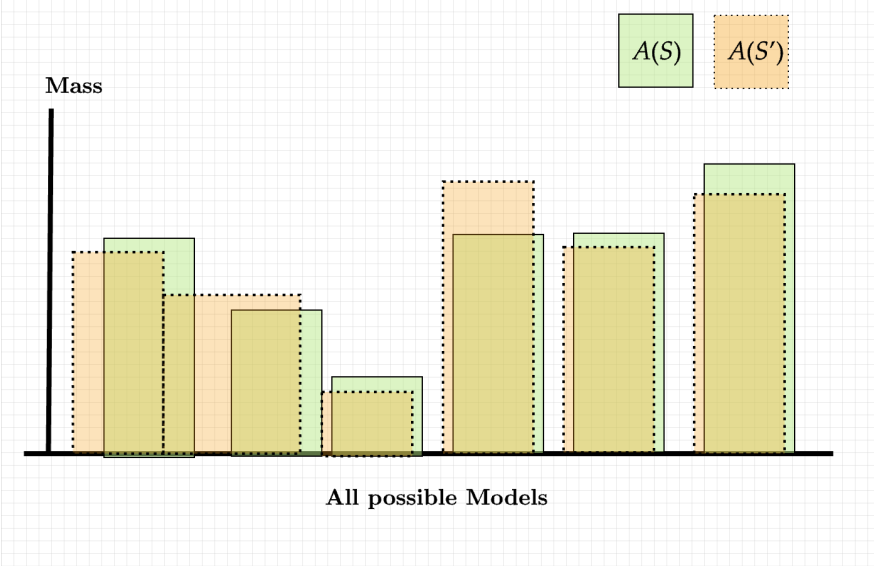

Figure 1: Illustration of how the p.m.f. changes when training the algorithm with S' instead of S.

Definition 2: Approximate Differential Privacy

A randomized algorithm A is (ϵ, δ)-differentially private if for all neighboring databases S, S', and for all sets Y of outputs of A, it holds that

$$\Pr_{h \sim A(S)}[h \in Y] \leq \exp(\epsilon) \cdot \Pr_{h \sim A(S')}[h \in Y] + \delta\,.$$

The probability is taken over the internal randomness (i.e., the random coins) of A.

Typical values for ϵ are of order 0.1 and for $\delta \ll 1/n$.

At first glance, differential privacy looks very similar to the notion of 'algorithmic stability' that we saw in previous lectures. The most closely related notion is the change-one error stability (say uniform stability), which measures how much the generalization error changes when an input is changed.

In contrast, differential privacy measures how the distribution over the entire output changes, which is a more complex measure of stability (differential privacy implies change-one error stability).

2 Which concept classes can we learn privately?

Recall that a concept is a function from examples to labels, and a class of concepts is learnable if for any distribution $\mathcal{D}$ on examples, one can, given limited access to examples sampled from $\mathcal{D}$ labeled according to some target concept c, find a hypothesis which predicts c's labels with high probability over future examples taken from the same distribution. In the PAC model, a learning algorithm can access a polynomial number of labeled examples in the following sense.

Let $d \in \mathbb{N}$ be a parameter that measures the size of the examples in X_d and consider the ensemble of concept classes $\mathcal{C} = \{C_d\}_d$. For instance, d could be dimension and C_d could be the class of $d-$dimensional halfspaces.

We first study PAC learning under pure DP.

Definition 3: Agnostic PAC Learning with Pure DP

Let d as above. Concept class $\mathcal{C}$ is privately PAC learnable using a hypothesis class $\mathcal{H}$ if there exists an algorithm that takes inputs d, privacy ϵ, accuracy α, confidence β, and uses a dataset of size $n = \text{poly}(d, 1/\epsilon, 1/\alpha, \log(1/\beta))$ and satisfies

- Privacy: For all $\epsilon > 0$, algorithm $\mathcal{A}(\cdot, \epsilon, \cdot, \cdot)$ is ϵ-differentially private.
- Accuracy: $\mathcal{A}$ agnostically PAC learns $\mathcal{C}$ using H.

Note that the privacy guarantee is worst-case but accuracy is on average over a set of examples drawn i.i.d. from a distribution $\mathcal{D}$.

3 Differential Private Learning and Finite Classes

We present a generic private learning algorithm A, introduced in Kasiviswanathan et al. (2011), which guarantees that for any concept class $\mathcal{C}$, algorithm A is a distribution-free differentially private agnostic PAC learner for $\mathcal{C}$ that uses a number of samples proportional to $\log |\mathcal{C}|$. Hence, the algorithm comes with useful guarantees when $\mathcal{C}$ is finite.

This is a private analogue of the cardinality version of Occam's razor, a basic sample complexity bound from (non-private) learning theory. The generic private learner is based on the *exponential mechanism*.

Consider a function q that take a dataset S of size n and some hypothesis $h \in \mathcal{H}_d$ (this is the class that the algorithm of our definition uses) and assigns it a score

$$q(S, h) = -|\{i : y_i \neq h(x_i)\}| .$$

Hypotheses will low score are not good predictors for S. The classic Occam's razor argument assumes a learner that selects a hypothesis with maximum score (that is, minimum empirical error). Instead, our private learner $A_{\epsilon,q}$ is defined to sample a random hypothesis with probability dependent on its score:

$$\mathop{\mathbf{Pr}}_{h \sim A_{\epsilon,q}(S)}[h] \propto \exp(\epsilon \cdot q(S, h)/2) .$$

Since the score ranges from $-n$ to 0, hypotheses with low empirical error are exponentially more likely to be selected than ones with high error. Hence, the above corresponds to a "smoothed" version of the classical ERM algorithm.

Note that even if $|\mathcal{H}_d|$ is polynomial, sampling from the above distribution may be computationally hard.

Observe that changing one example of S, changes the score by at most 1. Hence the algorithm is clearly ϵ-differentially private. It remains to show that the drawn classifier is also a good PAC learner.

The guarantee of the above algorithm is the following: for all $d \in \mathbb{N}$, any concept class $\mathcal{C}_d$ whose cardinality is at most $\exp(\text{poly}(d))$ is privately agnostically PAC learnable using $\mathcal{H}_d = \mathcal{C}_d$.

Theorem 1

For all $d \in \mathbb{N}$, any concept class $\mathcal{C}_d$ whose cardinality is at most $\exp(\text{poly}(d))$ is privately agnostically PAC learnable using $\mathcal{H}_d = \mathcal{C}_d$. The learner uses $n = O(\log |\mathcal{H}_d| + \log 1/\beta) \max\{\frac{1}{\epsilon\alpha}, 1/\alpha^2\}$ samples.

Proof. The privacy guarantee follows by the exponential mechanism. Let us show that the utility condition is also satisfied.

Consider the bad event

$$E = \{A_{\epsilon,q}(S) = h \text{ with } \text{err}(h) > \alpha + \text{OPT}\} .$$

This is an event over the dataset S and the randomness of the algorithm. We will show that $\mathbf{Pr}[E] \leq \beta$.

Define the training error of h by $\mathrm{err}_S(h) = -q(S,h)/n$. For a fixed hypothesis, this random variable concentrates:

$$\Pr_S[|\mathrm{err}(h) - \mathrm{err}_S(h)| > t] \leq 2\exp\Big(-2t^2 n\Big)$$

Hence the probability that there exists a hypothesis in the finite class $\mathcal{H}_d$ that violates the inequality $|\mathrm{err}(h) - \mathrm{err}_S(h)| \leq t$ is at most $2|\mathcal{H}_d|\exp\big(-2t^2 n\big)$.

Let us condition on this event. We now analyze the error of $A_{\epsilon,q}(S)$. For any $h \in \mathcal{H}_d$, the probability that $A_{\epsilon,q}(S)$ draws h is

$$\propto \exp(\epsilon q(S,h)/2)$$

and so it is equal to

$$\frac{\exp(-n\cdot\epsilon\cdot\mathrm{err}_S(h)/2)}{\sum_{h'\in\mathcal{H}_d}\exp(-n\cdot\epsilon\cdot\mathrm{err}_S(h')/2)} \leq \frac{\exp(-n\cdot\epsilon\cdot\mathrm{err}_S(h)/2)}{\max_{h'\in\mathcal{H}_d}\exp(-n\cdot\epsilon\cdot\mathrm{err}_S(h')/2)}$$

But the h' that achieves the max should be the one that minimizes the training error. So, the RHS is equal to

$$\exp\left(-(\epsilon/2)\cdot n\cdot(\mathrm{err}_S(h) - \min_{h'\in\mathcal{H}_d}\mathrm{err}_S(h'))\right)$$

Since we have conditioned on the event that any h' concentrates, we can upper bound the above by

$$\exp(-(\epsilon/2)\cdot n\cdot(\mathrm{err}_S(h) - (\mathrm{OPT}+t)))$$

This allows us to control the probability that the algorithm chooses a sub-optimal hypothesis: the probability that the algorithm picks any concept h with population loss at least $\mathrm{OPT}+t$ is at most

$$|\mathcal{H}_d|e^{-\epsilon n t/2}.$$

Let us set $t = \alpha/3$. Then $\mathbf{Pr}[E] \leq |\mathcal{H}_d|(e^{-\epsilon n\alpha/6} + 2e^{-2n\alpha^2/9})$. This gives the desired sample complexity. □

> **Insight 1**
>
> Pure DP is characterized by the probabilistic representation dimension (Beimel et al., 2013), which roughly speaking allows one to reduce the problem to an analysis similar to the above: discretize the class and run the exponential mechanism.

4 Online Learning

In the next lectures, we will focus on PAC learning with *approximate differential privacy.* For this, we need to first understand the concept of *online learnability,* going back to the work of Littlestone (1988).

We introduce the online learning game. In this game, there are two players, the adversary P_A and the learner P_L. They play in rounds: first, P_A chooses a feature vector and reveals it to P_L. Then the learner tries to guess a label for the given example. Finally, P_A reveals the true label of the example.

1. The adversary picks a point $x_t \in \mathcal{X}$.
2. The learner guesses a value $\widehat{y}_t \in \{0,1\}$
3. The adversary chooses the value y_t as true label so that $y_t = h(x_t)$ for some $h \in \mathcal{H}$ that is consistent with the previous examples (x_p, y_p) for any $p \leq t$.

The learner makes a mistake in round t whenever the guess $\widehat{y}_t$ differs from the true label y_t. The goal of the learner is to minimize her loss and the adversary's intention is to provoke many errors to the learner.

We will say that a class $\mathcal{H}$ is online learnable with d mistakes if there is a (deterministic) learning algorithm that makes at most d mistakes against any adversary. (In some sense, online learnability is another quite strong notion of stability.)

Our goal is to characterize online learnability. First, we will need some notation. For a sequence $y = (y_1, y_2, ...)$, we denote $y_{\leq k} = (y_1, ..., y_k)$. We may also usually identify elements of $\{0,1\}^d$ with strings or a prefix of a sequence of length d.

Before going to the general result, it is natural to ask whether VC dimension is useful for understanding online learnability. Unfortunately, there is a class with VC dimension 1 but the adversary can force infinitely many mistakes to any learning algorithm.

This class corresponds to thresholds in $[0,1]$. Inspired by this example, we can introduce the following combinatorial measure.

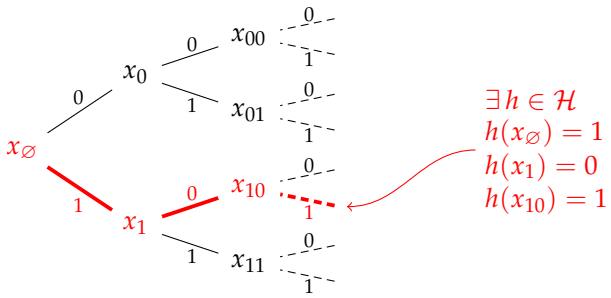

Figure 2: A Littlestone tree of depth 3. Every branch is consistent with a concept $h \in \mathcal{H}$. This is illustrated here for one of the branches. **Taken from Bousquet et al. (2021).**

Definition 4: Littlestone Tree

A Littlestone tree for $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ is a complete binary tree of depth $d \leq \infty$ whose internal nodes are labeled by $\mathcal{X}$, and any left (resp. right) edge connecting a node to its left (resp. right) child is labeled 0 (resp. 1), such that every path of length at most d emanating from the root is consistent with a concept $h \in \mathcal{H}$. Typically, a Littlestone tree is a collection

$$\bigcup_{0 \leq \ell < d} \left\{ x_u : u \in \{0,1\}^\ell \right\} = \{x_\varnothing\} \cup \{x_0, x_1\} \cup \{x_{00}, x_{01}, x_{10}, x_{11}\} \cup ...$$

such that for every path $y \in \{0,1\}^d$ and finite $n < d$, there exists $h \in \mathcal{H}$ so that $h(x_{y_{\leq \ell}}) = s_{y_{\leq \ell+1}}$ for $0 \leq \ell \leq n$, where $s_{y_{\leq \ell+1}}$ is the label of the edge connecting the nodes $x_{y_{\leq \ell}}$ and $x_{y_{\leq \ell+1}}$.

Let us first see some examples.

Example 1

For a finite class $\mathcal{H}$, the maximal depth of a Littlestone tree is $d \leq \log |H|$.

Example 2

Thresholds on $[0,1]$ admit an infinite Littlestone tree, i.e., a Littlestone tree that satisfies the above definition with $d = \infty$. This is due to the density of the reals on $[0,1]$.

Definition 5: Littlestone Dimension

The Littlestone dimension of a class $\mathcal{H}$, denoted by $\text{Ldim}(\mathcal{H})$ is the maximal depth of a Littlestone tree shattered by $\mathcal{H}$.

Observe that for any concept class $\mathcal{H}$, it holds that $\text{VC}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$ since VC corresponds to Littlestone trees where at each level, the adversary has to use the *same* example.

Observe that the Littlestone dimension puts a *uniform* upper bound on the number of mistakes an adversary can cause to the learner. We will revisit this point later in this course. We are now ready to provide the main result of online learning.

Theorem 2: Online Learnability

A class $\mathcal{H}$ is online learnable with d mistakes if and only if $d = \text{Ldim}(\mathcal{H})$.

To see why the Littlestone dimension is a lower bound, the adversary's strategy is clear. Whenever the learner predicts a label, she must choose a different label that will cause the learner to make a mistake in that round; a Littlestone tree is exactly the combinatorial structure the adversary is looking for.

On the other hand, how should the learner play? Since there is no Littlestone tree of depth $\text{Ldim}(\mathcal{H}) + 1$, the learner's predictions should lead her to a leaf of the Littlestone tree that is defined by her interaction with the learner. However, to reach to that level fast, the learner should play in a smart way.

If $\mathcal{H}$ is finite, then halving is the most natural approach: at each round, predict the label that is consistent with the majority vote. Then, either the learner is correct or the adversary has to reduce multiplicatively by $1/2$ the set of consistent hypotheses. Hence, for finite classes, this approach makes at most $\log |\mathcal{H}|$ mistakes.

It turns out that there is a natural extension of this approarch, called Standard Optimal Algorithm (SOA), which guarantees that the learner will make at most $\text{Ldim}(\mathcal{H})$ mistakes.

To gain some intuition, it is instructive to consider the first step of her algorithm. The learner is presented with a point x_1 and she picks the label $y_1 = \text{argmax}_{y \in \{0,1\}} \text{Ldim}(\mathcal{H}_{x_1,y})$. Here, $\mathcal{H}_{x_1,y}$ is the subset of $\mathcal{H}$ consistent with the example (x_1, y). It is easy to see that there is at most one y such that $\text{Ldim}(\mathcal{H}_{x_1,y}) = d$; if there were two of them then $\text{Ldim}(\mathcal{H}) = d + 1$. Thus, by picking the one that induces the subset of the hypothesis class with the largest Littlestone dimension, the learner either does not make a mistake or gets closer to a leaf of the tree.

Definition 6: SOA

Let $V_1 = \mathcal{H}$. Let $t \geq 1$:

1. Get x_t
2. For any y, let $V_t^y = \{h \in V_t : h(x_t) = y\}$.
3. Predict $\text{argmax}_y \text{Ldim}(V_t^y)$.
4. Get true label y_t.
5. Update $V_{t+1} = \{h \in V_t : h(x_t) = y_t\}$.

Lemma 1

It holds that SOA makes at most $\text{Ldim}(\mathcal{H})$ mistakes.

Proof. It suffices to prove that whenever the algorithm makes a mistake, we have that the Littlestone di-

mension of the updated class V_{t+1} decreases, i.e., $\text{Ldim}(V_{t+1}) \leq \text{Ldim}(V_t) - 1$. Assume, for contradiction, that $\text{Ldim}(V_{t+1}) = \text{Ldim}(V_t)$. This means that $\text{Ldim}(V_t^y) = \text{Ldim}(V_t)$ for $y \in \{0,1\}$. Then we can construct a tree of depth $\text{Ldim}(V_t) + 1$ for the class V_t, which leads to the desired contradiction, since $\text{Ldim}(V_t)$ corresponds to the maximal tree depth. □

The immediate issue with this approach is how one could actually implement Step 3. While we will not focus on that, since we care only about statistical efficiency and assume access to an oracle that given a class, computes its Littlestone dimension, we mention that there is some recent work on design more natural online learning algorithms (e.g., based on ERM oracles (Assos et al., 2023)).

For the agnostic setting, we refer to Shalev-Shwartz and Ben-David (2014) and Ben-David et al. (2009).

Examples. Can you online learn thresholds over a finite domain? Over $\mathbb{N}$? Over $\mathbb{R}$? When can you learn with a finite number of mistakes?

Spoiler and Future Reference. We say $\mathcal{H}$ has an *infinite Littlestone tree* if there is a Littlestone tree for $\mathcal{H}$ of depth $d = \infty$.

The above notion is closely related to the *Littlestone dimension*. As we saw, a concept class $\mathcal{H}$ has Littlestone dimension d if it has a Littlestone tree of depth d but not of depth $d + 1$. Classical online learning theory yields a learning algorithm (SOA) that makes at most d mistakes in classifying any *adversarial*.

We will prove an extension of the above later in the course: The non-existence of an infinite Littlestone tree characterizes the existence of an algorithm that guarantees a *finite* (but not necessarily uniformly bounded) number of mistakes for every realizable sequence of examples (Bousquet et al., 2021).

It is crucial to observe that that having having an unbounded Littlestone dimension does not imply that the class has an infinite Littlestone tree.

$\text{Ldim}(\mathcal{H}) = \infty$ due to existence of finite Littlestone trees of *arbitrarily large depth*, which does not imply the existence of any single tree of infinite depth.

To see an example, consider the class of all threshold functions $h_t(x) = \mathbf{1}_{x \geq t}$ where $t \in \mathbb{N}$. This is a VC class (its VC dimension is 1), and it does not have an infinite Littlestone tree. Note, however, that this class has unbounded Littlestone dimension (it shatters Littlestone trees of arbitrary finite depths), so that it does not admit an online learning algorithm that makes a uniformly bounded number of mistakes. (Fixing the root of the Littlestone tree limits the adversary to cause a finite number of mistakes, but the value at the root can be an arbitrarily large integer).

Beyond Worst Case. The Littlestone dimension characterization is essentially a negative result since most interesting classes are not online learnable. This negative result has inspired interesting models of online learning based on smoothed analysis (Spielman and Teng, 2004). For details, see e.g., (Haghtalab et al., 2020, 2024; Block et al., 2022).

References

Assos, A., Attias, I., Dagan, Y., Daskalakis, C., and Fishelson, M. K. (2023). Online learning and solving infinite games with an erm oracle. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 274–324. PMLR.

Beimel, A., Nissim, K., and Stemmer, U. (2013). Characterizing the sample complexity of private learners. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 97–110.

Ben-David, S., Pál, D., and Shalev-Shwartz, S. (2009). Agnostic online learning. In *COLT*, volume 3, page 1.

Block, A., Dagan, Y., Golowich, N., and Rakhlin, A. (2022). Smoothed online learning is as easy as statistical learning. In *Conference on Learning Theory*, pages 1716–1786. PMLR.

Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526.

Bousquet, O., Hanneke, S., Moran, S., van Handel, R., and Yehudayoff, A. (2021). A theory of universal learning. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 532–541, New York, NY, USA. Association for Computing Machinery.

Haghtalab, N., Roughgarden, T., and Shetty, A. (2020). Smoothed analysis of online and differentially private learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9203–9215. Curran Associates, Inc.

Haghtalab, N., Roughgarden, T., and Shetty, A. (2024). Smoothed analysis with adaptive adversaries. *Journal of the ACM*, 71(3):1–34.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011). What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Spielman, D. A. and Teng, S.-H. (2004). Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463.