Notes on Stability in Machine Learning: Generalization, Privacy, and Replicability

Alkis Kalavasis
Yale University

August 11, 2025

Abstract

These notes are based on the author's lectures on statistical learning theory for the course "Stability in Machine Learning: Generalization, Privacy, and Replicability" (Yale CPSC 683), taught at Yale University in Spring 2025. The course examines the generalization and stability of machine learning systems, exploring the many ways in which a learning algorithm's stability can be defined. A standard approach, inspired by sensitivity analysis, measures how variations in input affect a system's output. This perspective leads to several important notions of stability — including uniform stability, differential privacy, and replicability — which we study in detail. We investigate these concepts, their implications for learning theory, and the surprising connections between them.

Contents

Stability in Machine Learning: Generalization, Privacy & Replicability Course Syllabus[1]

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

Lecture	Context	References
1	VC Theory and Uniform Convergence	Vapnik (2013); Shalev-Shwartz and Ben-David (2014)
2	Generalization Bounds and Algorithmic Stability	Bousquet and Elisseeff (2002)
3	Stability of SGD and Empirical Observations	Hardt et al. (2016), Zhang et al. (2021)
4	Failure of Uniform Convergence and Domain Adaptation	Nagarajan and Kolter (2019)
5	Differentially Private PAC Learning	Kasiviswanathan et al. (2011)
6	Littlestone dimension is necessary for DP PAC learning	Alon et al. (2019)
7	Littlestone dimension is sufficient for DP PAC learning	Bun et al. (2020)
8	TV Stability, Replicability, and connections to Differential Privacy	Impagliazzo et al. (2022), Bun et al. (2023), Kalavasis et al. (2023)
9	Memorization and Generalization	Feldman (2020), Brown et al. (2021)
10	A Theory of Learning Curves	Bousquet et al. (2021)
11	Language Identification and Generation	Gold (1967), Angluin (1979), Kleinberg and Mullainathan (2024)
12	Diffusion Models	Chen et al. (2022); Chewi et al. (2025)
13	Presentations	

[1] Version 1.0 (April 30, 2025)

Lecture 1: Generalization in Machine Learning

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

In this first Lecture, we will cover the standard PAC learning framework for binary classification and discuss about the course plan for the upcoming Lectures.

1 What is Generalization in Machine Learning?

Generalization in statistical prediction is almost second nature to human reasoning. The work of Zhang et al. (2021) provides the following beautiful summary of what is generalization: Scientists, policymakers, and economists have capitalized on the empirical observation that *patterns in collected data often carry over when predicting future events or explaining unobserved phenomena*. This means that uncertain outcomes often follow patterns observed in past data. We call the process of identifying and leveraging these patterns *generalization*.

Towards formalizing and explaining generalization, Machine Learning theory is grounded on a statistical baseline. We assume that observations come from a fixed data generating process. Having collected the training data, we fit a model to the training set. Next, during testing, we evaluate the model by how well it performs on unseen generated data from the same process. This setting is extremely simple but it already raises interesting questions, which we are going to discuss today.

Consider an input (or feature) space $\mathcal{X}$ and a label space $\mathcal{Y}$. For this introduction, we will focus on the following classical learning problem. A binary classification problem is defined by a distribution $\mathcal{D}$ over labelled examples $(x, y) \in \mathcal{X} \times \{0, 1\}$.

We will need to important notions, that of a classifier and that of a learning algorithm.

Definition 1: Classifier and Learning Algorithm

- A classifier h is a mapping from $\mathcal{X}$ to $\mathcal{Y}$.

For the following we fix a training set size $n \in \mathbb{N}$:

- A deterministic learning algorithm (or learner) $A = A_n$ is a mapping from $(\mathcal{X} \times \mathcal{Y})^n$ to $\mathcal{Y}^{\mathcal{X}}$, which maps a training set S of size n to a classifier $A(S)$ from $\mathcal{X}$ to $\mathcal{Y}$.
- A randomized learning algorithm (or learner) $A = A_n$ is a mapping from $(\mathcal{X} \times \mathcal{Y})^n$ to $\Delta(\mathcal{Y}^{\mathcal{X}})$, which maps a training set S of size n to a distribution $A(S)$ over classifiers. [a] [b] [c]

[a] Here $\Delta(\cdot)$ denotes the probability simplex.

[b] In general, the training set S (which will contain labeled examples) will be drawn from some distribution $\mathcal{D}$. Hence, the randomness of the output of a deterministic learner is only due to the randomness of S. On the other side, a randomized learner has also internal randomness, which corresponds to eventually sampling a classifier from $\Delta(\mathcal{Y}^{\mathcal{X}})$.

[c] More formally, a learning algorithm A is a sequences of mappings $(A_n)_{n \in \mathbb{N}}$, where n is the training set size.

The learning algorithm does not know $\mathcal{D}$, but is able to collect a sample of n i.i.d. examples from $\mathcal{D}$. Then, during the training phase, the learner uses these n examples to build a classifier $\widehat{h}_n : \mathcal{X} \to \{0, 1\}$. The objective of the learner is to achieve small generalization error

$$R(\widehat{h}_n) = \mathop{\mathbf{Pr}}_{(x,y)\sim\mathcal{D}} \left[\widehat{h}_n(x) \neq y\right] .$$

Observe that this quantity is a random variable depending on the randomness of the training set, used to obtain $\widehat{h}_n$. Making this quantity small intuitively means that the learning algorithm has extracted from the training data all the useful patterns hidden in $\mathcal{D}$ and can explain unobserved samples coming from $\mathcal{D}$.

2 VC Theory and Uniform Convergence

Since the data distribution $\mathcal{D}$ is unknown to the learner, our theory of learning must be expressed in terms of some restrictions on $\mathcal{D}$. To this end, the Probably Approximately Correct (PAC) model of learning (Vapnik, 2013; Valiant, 1984) introduces a concept class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ (a class of functions from $\mathcal{X}$ to $\{0,1\}$) and aims to understand learnability with respect to the this (known) class. Introducing this concept class is crucial since now we can state assumptions about $\mathcal{D}$.

The simplest and most standard assumption is that $\mathcal{D}$ is realizable with respect to $\mathcal{H}$, in the sense that

$$\inf_{h \in \mathcal{H}} R(h) = 0.$$

In words, $\mathcal{H}$ contains hypotheses with arbitrarily small generalization error[2]. We denote the class of realizable distributions by $\mathrm{RE}(\mathcal{H})$.

The fundamental result of PAC learning theory (Blumer et al., 1989; Vapnik, 2013; Vapnik and Chervonenkis, 1971) states that:

$$\inf_{\widehat{h}_n} \sup_{\mathcal{D} \in \mathrm{RE}(\mathcal{H})} \mathop{\mathbf{E}}_{S \sim \mathcal{D}^n} \left[R(\widehat{h}_n) \right] = \Theta \left(\min \left\{ \frac{\mathrm{VC}(\mathcal{H})}{n}, 1 \right\} \right) \text{ for all } n \in \mathbb{N}, \tag{1}$$

where the expectation is over the training set of size n drawn i.i.d. from $\mathcal{D}$ and the classifier $\widehat{h}_n$ is the output of the learning algorithm trained on S. Observe that the PAC model is truly worst-case since it quantifies the expected generalization of an algorithm against the worst-case possible distribution (which is allowed to change with the sample size). Recall that $\mathrm{VC}(\mathcal{H})$ is corresponds to the Vapnik-Chervonenkis (VC) dimension, which is defined as follows.

Definition 2: Shattering

Given a class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ and a set of points $S = \{x_1, ..., x_n\} \subseteq \mathcal{X}$, we define the projection of $\mathcal{H}$ on S as

$$\mathcal{H}_S = \{(h(x_1), ..., h(x_n)) : h \in \mathcal{H}\} \subseteq \{0,1\}^n.$$

If $|\mathcal{H}_S| = 2^n$, we say that S is shattered by $\mathcal{H}$.

If a set S is shattered by $\mathcal{H}$, it means that the concept class has the expressive power to create all possible $2^{|S|}$ labelings of size $|S|$. Roughly speaking, being able to shattering very 'large' sets, it intuitively means that the concept class should be 'harder' to learn. The largest size that a class can shatter corresponds to its VC dimension.

Definition 3: VC Dimension

Given a class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, the VC dimension of $\mathcal{H}$ is the largest set size d that can be shattered by $\mathcal{H}$. If $\mathcal{H}$ can shatter sets of any size, then its VC dimension is ∞.

To give some examples, the VC dimension of thresholds over $\mathbb{R}$ is 1 but the VC dimension of the function class $x \mapsto \sin(\theta x)$ for $\theta \in \mathbb{R}$ is infinite. Going back to (1), the minimax formulation corresponds to how

[2]The notion of realizability is a little bit subtle. For instance, a stronger notion is that there exists an $h \in \mathcal{H}$ that realizes all the observed labels and achieves $R(h) = 0$. The infimum definition is weaker in the sense that it guarantees that for any $\epsilon > 0$, there exists some hypothesis h_ϵ such that $R(h_\epsilon) < \epsilon$. During the course, we will specify when each definition is used.

well the best classifier $\widehat{h}_n$ can perform against the worst-case possible realizable distribution $\mathcal{D}$. Note that $\widehat{h}_n$ does not have to belong to $\mathcal{H}$; if it does, it is called a *proper* learner, otherwise it is called *improper*. In fact, the algorithm that gets the optimal sample complexity is improper (Hanneke, 2016) and we know precisely when optimal proper learning is possible (Bousquet et al., 2020).

Hence, the classical theory of PAC learning gives a beautiful dichotomy on how fast the minimax generalization error goes to 0 with the number of samples n: *every concept class $\mathcal{H}$ has a rate that is either linear c/n or bounded away from zero, depending on the finiteness of the combinatorial parameter* $\mathrm{VC}(\mathcal{H})$.

It is important to mention that we will mostly discuss about statistical learning theory ignoring *computational* aspects (such as what is the computational complexity of finding a hypothesis that generalizes well).

There is a crucial reason why this theory is satisfying. The idea is that learnability comes with a learning algorithm and that this algorithm is the most natural and actually applied algorithm, namely *Empirical Risk Minimization* (ERM). That is, the fact that every VC class is PAC learnable is witnessed by any algorithm that outputs a concept $h \in \mathcal{H}$ that is consistent with the input sample (i.e., it minimizes the training error).

This follows from the celebrated *uniform convergence theorem* of Vapnik and Chervonenkis (1971). Moreover, any ERM algorithm achieves the optimal uniform learning rate, up to lower order factors. We also mention that this landscape extends (perhaps surprisingly) to the agnostic setting. In the agnostic setting, no assumptions are made on $\mathcal{D}$ but the learning algorithm competes with the best predictor in the fixed class $\mathcal{H}$, i.e., the goal is to make the difference $R(\widehat{h}_n) - \inf_{h\in\mathcal{H}} R(h)$ as small as possible. For that, the linear rate of the realizable case is replaced by the slower rate of $\Theta(\min\{\sqrt{\mathrm{VC}(\mathcal{H})/n}, 1\})$.

Uniform convergence refers to the phenomenon where, given a sufficiently large sample from a population, the empirical losses of *all* concepts $h \in \mathcal{H}$ approximate their true population losses. The crucial point is that this happens *uniformly* over all functions in $\mathcal{H}$. This immediately implies an algorithmic principle: pick a concept in the class that minimizes empirical loss, which for a given training set S of size n, is defined as

$$\widehat{R}_S(h) = \frac{1}{n} \sum_{(x,y)\in S} 1\{h(x) \neq y\}\,.$$

Definition 4: Uniform Convergence

A concept class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ has the uniform convergence (UC) property (or is a uniform Glivenko-Cantelli class[a]) if there is a function $n_{UC} : (0,1)^2 \to \mathbb{N}$ such that for any $\epsilon, \delta \in (0,1)$ and any $\mathcal{D}$, if S is a training set of size $n \geq n_{UC}(\epsilon, \delta)$ drawn i.i.d. from $\mathcal{D}$, then with probability $1-\delta$, it holds

$$\sup_{h\in\mathcal{H}} \left|\widehat{R}_S(h) - R(h)\right| \leq \epsilon\,.$$

[a] The name comes from the famous Glivenko-Cantelli theorem in probability theory where the empirical CDF F_n of a CDF F satisfies $\|F_n - F\|_\infty = \sup_x |F_n(x) - F(x)| \to 0$ almost surely.

Some remarks are in order. Observe that, in the above definition, we do not need to assume that $\mathcal{D} \in \mathrm{RE}(\mathcal{H})$. In fact, the standard proof of the fundamental theorem of PAC learning goes by showing that UC $\Rightarrow$ Agnostic PAC Learning $\Rightarrow$ Realizable PAC Learning (the second implication is by definition). Note that n_{UC} for $\mathcal{H}$ will scale linearly with $\mathrm{VC}(\mathcal{H})$ and this will give the rate of the fundamental VC theorem, which we now state.

Theorem 1: Fundamental Theorem of Statistical Learning

Let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$. Then the following are equivalent:

1. $\mathcal{H}$ has the uniform convergence property.
2. Any ERM rule is a successful agnostic PAC learner for $\mathcal{H}$.

3. $\mathcal{H}$ is agnostic PAC learnable.
4. $\mathcal{H}$ is PAC learnable (in the realizable setting).
5. Any ERM rule is a successful PAC learner for $\mathcal{H}$.
6. $\mathrm{VC}(\mathcal{H}) < \infty$.

Note that (1) is the quantitative version of realizable PAC learning.

2.1 A Proof of the Fundamental Theorem of PAC Learning

The proof is standard and appears in any introductory learning theory course or book (Shalev-Shwartz and Ben-David, 2014). I will just sketch the main ideas.

Observe that the implications $2 \to 3 \to 4$ and $2 \to 5$ are immediate. Hence, the interesting parts are $1 \to 2$ and $6 \to 1$. The implications $4 \to 6$ and $5 \to 6$ follow from the No-Free-Lunch Theorem (shortly, if $\mathcal{H}$ shatters some set of size $2m$, then we cannot learn $\mathcal{H}$ using m examples.).

Theorem 2: No-Free-Lunch Theorem

Let $\mathcal{H}$ be a hypothesis class of functions from $\mathcal{X}$ to $\{0,1\}$. Let m be a training set size. Assume that there exists a set $C \subset \mathcal{X}$ of size $2m$ that is shattered by $\mathcal{H}$. Then, for any learning algorithm A, there exist a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ and a classifier $h \in \mathcal{H}$ such that $R(h) = 0$ but with probability of at least $1/7$ over the choice of S we have that $R(A(S)) \geq 1/8$.

For the proof we refer to Shalev-Shwartz and Ben-David (2014). This result immediately implies that if the class has infinite VC dimension, then it is not PAC learnable. We now proceed to the other implications.

Idea for ($1 \to 2$) This essentially follows from the definitions of ERM and UC. Consider a training set S with the property that for any $h \in \mathcal{H}$, $|\widehat{R}_S(h) - R(h)|$ is small (say at most $\epsilon/2$). Such a set will be called $\epsilon/2$-good. Then it is not difficult to see that for any output h_{ERM} of the algorithm $\mathrm{ERM}_{\mathcal{H}}(S)$ with input S, the following will hold. First,

$$R(h_{\mathrm{ERM}}) \leq \widehat{R}_S(h_{\mathrm{ERM}}) + \epsilon/2\,,$$

by definition of the set S. Next, because of ERM, for any $h \in \mathcal{H}$,

$$\widehat{R}_S(h_{\mathrm{ERM}}) + \epsilon/2 \leq \widehat{R}_S(h) + \epsilon/2\,.$$

Again by the definition of S,

$$\widehat{R}_S(h) + \epsilon/2 \leq R(h) + \epsilon\,.$$

Chaining these inequalities, we get that $1 \to 2$ because (i) $R(h_{\mathrm{ERM}}) \leq \inf_{h \in \mathcal{H}} R(h) + \epsilon$ and (ii) n_{UC} measures the (minimal) sample complexity of obtaining the uniform convergence property, namely, how many examples we need to ensure that with probability of at least $1 - \delta$, the sample S would be ϵ-good.

Idea for $(6 \to 1)$ VC dimension controls the effective size of a class $\mathcal{H}$. We define the projection of $\mathcal{H}$ to a set $\{x_1, ..., x_m\}$ as $\mathcal{H}_{\{x_1,...,x_m\}} = \{(h(x_1), ..., h(x_m)) : h \in \mathcal{H}\}$. Typically, if $\mathrm{VC}(\mathcal{H}) = d$, then when projecting it to a finite set $C \subset \mathcal{X}$, its effective size $|\mathcal{H}_C|$ is $O(|C|^d)$ (and not $2^{|C|}$). This claim is often referred to as Sauer- Shelah-Perles lemma. Then one can show that uniform convergence holds whenever the hypothesis class has a "small effective size", i.e., when $|\mathcal{H}_C|$ grows polynomially with $|C|$.

To see this, we define the growth function as

$$\tau_{\mathcal{H}}(m) = \max_{C \subset \mathcal{X} : |C| = m} |\mathcal{H}_C|\,.$$

The SSP lemma reads as follows.

Lemma 1: Sauer- Shelah-Perles Lemma

Let $\mathcal{H}$ be a concept class with $\mathrm{VC}(\mathcal{H}) \leq d < \infty$. For any sample size $m \in \mathbb{N}$, it holds that

$$\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{d} \binom{m}{i}.$$

In particular, if $m > d+1$, then $\tau_{\mathcal{H}}(m) \leq (em/d)^d$ and if $m < d$, then $\tau_{\mathcal{H}}(m) = 2^m$.

Then for any $\mathcal{D}$, if one manages to show that

$$\mathop{\mathbf{E}}_{S\sim\mathcal{D}^m}\left[\sup_{h\in\mathcal{H}} |R(h) - \widehat{R}_S(h)|\right] \leq \frac{4+\sqrt{\log(\tau_{\mathcal{H}}(2m))}}{\sqrt{2m}},$$

then we get, by Markov's inequality, that with high probability, $\mathcal{H}$ has the uniform convergence property given that $m \approx \mathrm{VC}(\mathcal{H})$. Showing the above inequality is the non-trivial (and most beautiful) part of the proof and requires the *symmetrization* technique, which we now discuss.

We can write $R(h) = \mathbf{E}_{S'\sim\mathcal{D}^m}[\widehat{R}_{S'}(h)]$, where $S' = z'_1, ..., z'_m$ is a 'ghost' sample. Then we can upper bound the LHS by

$$\mathop{\mathbf{E}}_{S,S'\sim\mathcal{D}^m} \sup_{h\in\mathcal{H}} \left|\frac{1}{m}\sum_{i=1}^{m} \ell(h,z_i) - \ell(h,z'_i)\right|$$

where $z_1, ..., z_m$ and $z'_1, ..., z'_m$ are i.i.d. draws from $\mathcal{D}$. This follows from the fact that suprermum of expectation is smaller than expectation of supremum. Here $\ell(h,z)$ is the 0-1 loss of the classifier h on the example z. Since these two samples are i.i.d., nothing will change if we interchange the random sample z_i with z'_i. This allows us to write that for any fixed $\sigma \in \{-1,1\}^m$,

$$\mathop{\mathbf{E}}_{S,S'\sim\mathcal{D}^m} \sup_{h\in\mathcal{H}} \frac{1}{m}\left|\sum_{i=1}^{m} \sigma_i(\ell(h,z_i) - \ell(h,z'_i))\right|.$$

The above also holds if we sample each component of σ uniformly at random from the uniform distribution, i.e., we can also write

$$\mathop{\mathbf{E}}_{S,S'\sim\mathcal{D}^m} \mathop{\mathbf{E}}_{\sigma\sim U(\{-1,1\}^m)} \sup_{h\in\mathcal{H}} \frac{1}{m}\left|\sum_{i=1}^{m} \sigma_i(\ell(h,z_i) - \ell(h,z'_i))\right|.$$

Fixing two datasets S, S', let C be the instances of $\mathcal{X}$ appearing in them; we can consider the supremum over the projection set $\mathcal{H}_C$ (this reduces $\mathcal{H}$ to a finite class over the elements $x_1, x'_1, ..., x_m, x'_m$). Then, we have that

$$\mathop{\mathbf{E}}_{\sigma\sim U(\{-1,1\}^m)} \max_{h\in\mathcal{H}_C} \frac{1}{m}\left|\sum_{i=1}^{m} \sigma_i(\ell(h,z_i) - \ell(h,z'_i))\right|.$$

For any fixed h, consider the random variable $\theta_h = \frac{1}{m}\sum_{i=1}^{m} \sigma_i(\ell(h,z_i) - \ell(h,z'_i))$, where the randomness arises from σ. Note that θ_h is zero-mean and is a sum of independent random variables on $[-1,1]$. Then $\mathbf{Pr}[|\theta_h| > t] \leq \exp\left(-2mt^2\right)$ and so,

$$\mathbf{Pr}\left[\max_{h\in\mathcal{H}_C} |\theta_h| > t\right] \leq |\mathcal{H}_C| \cdot \exp\left(-2mt^2\right).$$

The result follows since we can control the tails of the above non-negative random variable. In fact, its expected value is bounded by

$$\mathbf{E}\left[\max_{h\in\mathcal{H}_C} |\theta_h|\right] \leq \frac{4+\sqrt{\log|\mathcal{H}_C|}}{\sqrt{2m}}.$$

3 Reducing Agnostic to Realizable PAC Learning without UC?

As we saw, uniform convergence implies agnostic PAC learning, which trivially implies learnability in the realizable setting. Moreover, finite VC dimension implies uniform convergence. Hence, this establishes a perhaps surprising equivalence between realizable and agnostic PAC learning.

One of the goals of this course is to argue about learnability and generalization by avoiding uniform convergence arguments. As an example, in this section, we will ask whether the equivalence between realizable and agnostic PAC learning is fundamentally based on uniform convergence or if it can be shown in a more direct manner.

Vapnik and Chervonenkis (1971); Blumer et al. (1989)'s result is certainly a breakthrough in its own right, but its proof technique is too indirect to reveal any deeper connections between realizable and agnostic learning beyond the PAC setting. This led Hopkins et al. (2022) to raise the following natural question:

Is the equivalence of realizable and agnostic learning a fundamental property of learnability, or simply a happy coincidence derived from the uniform convergence?

Can we prove the equivalence between realizable and agnostic PAC learning result *without uniform convergence?* The answer is interestingly yes. This will be the content of this section, where we give a direct reduction from agnostic to realizable learning. The reduction is very simple and is presented right after:

- Input: Realizable PAC learner A, Unlabeled Sample Oracle O_U, Labeled Sample Oracle O_L.
- Algorithm:
  1. Draw an unlabeled sample $S_U \sim O_U$.
  2. Run A on all possible labelings of S_U to get
  $$C(S_U) = \{A(S_U, h(S_U)) : h \in H_{S_U}\}.$$
  3. Draw a labeled sample $S_L \sim O_L$.
  4. Output the element of $C(S_U)$ with the minimum training loss on the labeled sample S_L.

Note that as expected this algorithm is computationally inefficient since it requires the construction of the set $C(S_U)$ but also a search that set of hypotheses.

At its most basic, the correctness of the reduction is based on two observations.

- The set $C(S_U)$ almost certainly contains a near-optimal hypothesis. This will follow from the fact that A is a realizable PAC learner for $\mathcal{H}$.
- Second, we have to argue that Step 4 will actually find that near-optimal classifier. But since we know that it exists and since $C(S_U)$ has bounded size, we can employ standard uniform convergence for finite classes. This implies that if S_L is of order $\log |C(S_U)|$, ERM will find a hypothesis that is close to hypothesis that achieves $\min_{h \in \mathcal{H}} \mathbf{Pr}_{\mathcal{D}}[h(x) \neq y]$.

The key observation in this process is really that $C(S_U)$ 'acts like a cover' of $\mathcal{H}$ in the following weak sense we call non-uniform covering. In contrast to more classical notions, a non-uniform cover is a distribution over subsets of hypotheses that covers any fixed hypothesis in the class with high probability, but may fail to cover all hypotheses simultaneously.

Definition 5: Non-Uniform Cover

Let $\mathcal{H}$ be a hypothesis class over $\mathcal{X}$, $\mathcal{D}_X$ a marginal distribution over $\mathcal{X}$, and C a random variable over the power set $P(\mathcal{H})$. We call C a non-uniform (ϵ, δ)-cover of $\mathcal{H}$ with respect to $\mathcal{D}_X$ if

$$\forall h \in \mathcal{H} : \ \mathop{\mathbf{Pr}}_{C}[\exists h' \in C : \mathop{\mathbf{Pr}}_{x \sim \mathcal{D}_X}[h(x) \neq h'(x)] \leq \epsilon] \geq 1 - \delta \,.$$

Observe that if the quantifier $\forall h \in \mathcal{H}$ was inside $\mathbf{Pr}_C[\cdot]$, then the cover would be uniform since C would cover simultaneously all $h \in \mathcal{H}$. Here, in contrast, for every fixed hypothesis h in the class, C is very likely to contain a hypothesis close to h.

It is left to observe that $C(S_U)$ in Algorithm 1 is actually a non-uniform cover, but this is essentially immediate from the definition of realizable learning. Realizable learning promises that for every fixed hypothesis $h \in \mathcal{H}$, when A receives samples labeled by h it outputs a hypothesis close to h with high probability. $C(S_U)$ is generated by running A across all $h \in \mathcal{H}$, so this guarantee exactly translates to the above.

The above are summarized in two lemmas. First, we will show that the set $C(S_U)$ of outputs corresponding to running the realizable learner A across all possible labelings of the unlabeled sample S_U is a good approximation of the class $\mathcal{H}$.

Lemma 2

For any distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$, there exists a hypothesis $h' \in C(S_U)$ with high probability such that

$$R(h') \leq \mathrm{OPT}_{\mathcal{D}} + \epsilon/2 \,.$$

If this lemma holds, the standard Chernoff concentration and a union bound over the elements in $C(S_U)$ implies that ERM will find a near-optimal classifier. It remains to argue about the above lemma. For this step, we will use the fact that $C(S_U)$ is a non-uniform cover: for any fixed $h \in \mathcal{H}$, $C(S_U)$ contains a hypothesis close to h with high probability.

Lemma 3

For any distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ and any hypothesis $h \in \mathcal{H}$, with high probability, there exists $h' \in C(S_U)$ such that $\mathbf{Pr}_{\mathcal{D}_X}[h(x) \neq h'(x)] \leq \epsilon/2$.

This lemma follows from the guarantees of a realizable PAC learner: algorithm A promises that for any $h \in \mathcal{H}$ and distribution $\mathcal{D}_X$, an $1 - \delta/2$ fraction of labeled samples $(S, h(S)) \sim \mathcal{D}^{n(\epsilon/2, \delta/2)}$ satisfy

$$\mathop{\mathbf{Pr}}_{\mathcal{D}_X \times h}[A(S, h(S))] = \mathbf{Pr}[h(x) \neq h'(x)] \leq \epsilon/2 \,.$$

Here h' is the output of A on $(S, h(S))$. Since $C(S_U)$ contains the output of A on any $h \in \mathcal{H}$, the result follows.

Essentially we have to apply the above Lemma for the hypothesis that realizes $\mathrm{OPT}_{\mathcal{D}}$.

4 Course Plan: Going Beyond VC Theory

In this course, we will try to revisit settings where VC theory and the uniform convergence (UC) tool are not sufficient to explain generalization and learnability. At the same time, the notion of stability as a property of the algorithm will be a key tool in going beyond the predictions of the classical theory. Let us see a particular motivation, which can provide some insight.

Insight 1: Generalization as a Property of Algorithms

Observe that in the UC definition, the supremum is taken over all $h \in \mathcal{H}$. Why this is bad? To see an example, think of perhaps the most standard learning theory problem: running stochastic gradient descent on the class of feed-forwards neural networks (NNs) with L layers and some nonlinear activation such as ReLU.

First, the VC dimension of these models grows at least linearly with the number of parameters in the network. Hence to achieve small excess risk or uniform convergence of sample averages to probabilities for discrete losses, the sample size must be large compared to the number of parameters in these networks. This behavior clearly does not explain *over-parameterization.*

However, there is a catch in the definition of uniform convergence: it is completely algorithm independent! While the space of possible NNs is very rich and complicated (having a VC dimension that scales with the number of parameters), maybe the standard training algorithm (say SGD) only visits a few of these networks. Hence, we do not want to have a bound that holds uniformly over the whole space of possible functions since we are interested in algorithms that may not explore it.

The plan for this course is the following:

Lectures 2, 3, 4. We aim to understand generalization as a property of the learning algorithm. In Lecture 2, we will introduce the notion of stability (Bousquet and Elisseeff, 2002) to do that and see how it implies generalization bounds. In Lecture 3, we will further examine the stability properties of SGD based on the work of Hardt et al. (2016) but also we will discuss about failures of uniform convergence, indicated by the work of Zhang et al. (2021). In Lecture 4, we will talk about failures of uniform convergence based on the work of Nagarajan and Kolter (2019) but also discuss domain adaptation (where the training and test distributions do not coincide) (Ben-David et al., 2006).

Lectures 5, 6, 7. Stability is quite rich of a notion. It can capture various fundamental notions of learning such as private learning. This will be the content of Lectures 5, 6 and 7. In terms of techniques, standard uniform convergence does not suffice to explain private learning. In Lecture 5, we will revisit one of the seminal papers studying private learning (Kasiviswanathan et al., 2011). In Lectures 5 and 6, we will deal with a more modern (and challenging) result: private learning is equivalent to online learning for binary classification (Alon et al., 2019; Bun et al., 2020).

Lecture 8. In this lecture, we will see two other different forms of stability: one-way perfect generalization and replicability (Impagliazzo et al., 2022; Bun et al., 2023; Kalavasis et al., 2023). We will see how these two notions are closely related to differential privacy. This indicates that many well-known stability notions share (perhaps surprisingly) many conceptual and technical similarities.

Lecture 9. Memorization is a big mystery in the ML community. In this lecture, we will try to understand how memorization is related to stability and generalization. We will present two important works (Feldman, 2020; Brown et al., 2021).

Lecture 10 In this lecture, we will see an important setting where VC theory fails. The topic of this lecture is a fundamental work of Bousquet et al. (2021). The goal is to answer the following important question: how quickly can a given class of concepts be learned from examples? The standard measure the performance of a supervised machine learning algorithm is its learning curve, i.e., the decay of the error rate as a function of the number of training examples. We will see how stability can be used to give surprising results.

Lecture 11. In Lecture 11, we will see how stability and online learning are crucial to problems beyond prediction. We will focus on the classical problem of language identification and the recent reformulation of language generation (Gold, 1967; Kleinberg and Mullainathan, 2024).

Lecture 12. In this Lecture, we will talk about diffusion models, sample generation and distribution learning Chen et al. (2022); Chewi et al. (2025).

Lecture 13. In this final Lecture, there will be presentations for recent interesting papers, related to the course's topics.

Lecture 2: Generalization Bounds via Algorithmic Stability

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

In Lecture 1, we revisited the most prominent method to estimate the accuracy of a learning algorithm which is based on the theory of *uniform convergence* of empirical quantities to their mean (see e.g., Shalev-Shwartz and Ben-David (2014)). This theory provides ways to estimate the risk (or generalization error) of a learning system based on an empirical measurement of its accuracy and a measure of its complexity, such as the Vapnik-Chervonenkis (VC) dimension.

On the negative side, this theory cannot explain the success of various algorithms such as the k-NN algorithm: VC theory focuses on the size of the search space (the VC dimension of all concepts induced by k-NN is infinite) while other approaches focus on *how* the algorithm searches that space.

> **Insight 2: VC Theory does not Suffice for Generalization**
>
> Unlike VC-based approaches where the only property of the algorithm that matters is the size of the space to be searched, we are going to focus on *how the machine learning algorithm searches the space*. As we did in the end of Lecture 1, you should think of SGD and neural networks.

In Lecture 1, we used Uniform Convergence to show that the empirical loss of any hypothesis in the class is close to its population loss. Recall that uniform convergence makes no reference to the algorithm.

The general goal of Lecture 2 is to provide a collection of results of the flavor "Stability Implies Generalization"; here, stability is related to some *algorithmic property* and generalization means that the empirical loss will be close to the population one). We will discuss about the following four results-tools:

1. Uniform Stability Implies Generalization
2. Sample Compression Implies Generalization
3. Bounded Mutual Information Implies Generalization
4. Bounded Conditional Mutual Information Implies Generalization

The important difference compared to Lecture 1 is that now we will study generalization via *properties of the learning algorithm* and not of the *function class, i.e., the search space.*

> **Insight 3: Showing Generalization Bounds**
>
> If we fix some (binary) classifier we can show, using standard concentration bounds, that its performance on a sample is close to its performance on the underlying population. However, when we train an ML algorithm A using a dataset S to output a classifier h we cannot just use the fact that it has small loss on S to claim that its loss on the population is small because h depends on S. The above tools will allow us to get such results for algorithms A that satisfy some property among (1)-(4).

5 Local Rules and Generalization

The first stability-based results were obtained by Devroye, Rogers and Wagner in the seventies (see Rogers and Wagner (1978); Devroye and Wagner (1979a)). Rogers and Wagner first showed that the variance of the

leave-one-out error can be upper bounded by what Kearns and Ron (1997) later called hypothesis stability. This quantity measures how much the function learned by the algorithm will change when one point in the training set is removed. The main distinctive feature of their approach is that, unlike VC-theory based approaches where the only property of the algorithm that matters is the size of the space to be searched, it focuses on how the algorithm searches the space (e.g., by making use of that the learning algorithm is 'local'). A standard example is nearest-neighbor-based algorithms: Let K be the set of classifiers implemented by the 1-NN algorithm (for details see Devroye et al. (2013)). Then $\mathrm{VC}(K) = \infty$. However, guarantees about the generalization error of k-NN are known, e.g., see Rogers and Wagner (1978) and Devroye and Wagner (1982) for a survey.

For instance, consider the multiclass classification setting, where (X, Y) is drawn from some distribution and $Y \in \{1, ..., K\}$. For a new point X', the estimate $\widehat{Y}$ of its label based on the k-NN algorithm is based on the most frequent label among the k nearest points of X'. More generally, if (X^j, Y^j) represents the j-th closest point to X', one can think of k-local rules as mappings

$$\widehat{Y} = g((X^1, Y^1), ..., (X^k, Y^k))$$

for some function g. For simplicity, we ignore ties in the above discussion. The loss is $R = \mathbf{Pr}[Y \neq \widehat{Y}|S_n]$, where S_n is the training set of size n. A natural estimate of R is the delete estimate $\widehat{R}_n = \frac{1}{n}\sum_{i=1}^{n} 1\{y_i \neq \widehat{y}_i\}$, where $\widehat{y}_i$ is the estimate of y_i from X_i and S_n with (X_i, y_i) deleted (here $k \leq n-1$).

Theorem 3: Rogers and Wagner (1978)

For all distributions (X, Y) and any k-local rule,

$$\mathbf{E}(R - \widehat{R}_n)^2 \leq \frac{2k + 1/4}{n} + \frac{2k\sqrt{2k + 1/4}}{n^{3/2}} + \frac{k^2}{n^2}\,.$$

6 What is Algorithmic Stability in Machine Learning?

Inspired by the above semilar results, Bousquet and Elisseeff (2002) introduced a general stability framework for ML algorithms. In this section, we define notions of stability for learning algorithms from Bousquet and Elisseeff (2002) and show how to use these notions to derive generalization error bounds based on the empirical error and the leave-one-out error.

Definition 6: Learning Algorithm

A learning algorithm A is a (potentially randomized) mapping from $(\mathcal{X} \times \mathcal{Y})^n$ to $\mathcal{Y}^{\mathcal{X}}$ which maps a training set S of size n to a function $A(S)$ from $\mathcal{X}$ to $\mathcal{Y}$.

In general, learning algorithms are randomized, so they map datasets to a distribution over functions ($A(S)$ is a distribution over $\mathcal{Y}^{\mathcal{X}}$) but for simplicity we will assume that A is deterministic and that all functions are measurable and all sets are countable. For deterministic machine learning algorithms, the only source of randomness comes from the training data.

We will denote datasets either with S or Z. Recall the following notation from Lecture 1:

Definition 7: Empirical and Population Loss

Given algorithm A and training set S, define the population loss on a fresh sample z as

$$R(A, S) = \mathop{\mathbf{E}}_{z} \ell(A(S), z)\,.$$

Also, let the empirical loss on $S = \{z_1 = (x_1, y_1), ..., z_n = (x_n, y_n)\}$ be

$$R_{emp}(A, S) = \frac{1}{n} \sum_{i=1}^{n} \ell(A(S), z_i) .$$

Note that the ERM algorithm on some class $\mathcal{H}$ is a mapping from datasets S to the function

$$\text{argmin}_{h \in \mathcal{H}} \sum_{z \in S} \ell(h, z) .$$

We may also use the notation $\ell(A(S), \cdot)$ to denote a vector of loss values of the predictor $A(S)$ over all possible examples.

Understanding the generalization properties of an algorithm A reduces to the following problem:

Problem 1: High-Probability Generalization Bounds for specific algorithm A

For any ϵ, upper bound the tail probability

$$\underset{S}{\mathbf{Pr}} \left[|R_{emp}(A, S) - R(A, S)| > \epsilon \right] .$$

We remark that given that an algorithm A generalizes in the above sense, does not mean that it is a good learning algorithm. It simply guarantees that with high probability, the performance of the algorithm in the training set will be reflected to the population loss, i.e., A outputs a model that generalizes to the underlying distribution, rather than overfitting its training data.

Let us first recall how this compares to uniform convergence and why then ERM is a natural approach.

Insight 4: Uniform Convergence

Contrast the above with

$$\underset{S}{\mathbf{Pr}} \left[\sup_{f \in \mathcal{H}} \left| \frac{1}{n} \sum_{x \in S} \ell(f, x) - \underset{z}{\mathbf{E}} \ell(f, z) \right| > \epsilon \right] .$$

In Lecture 1, we saw that when the above holds, then ERM is the way to go. In contrast, in this Lecture, we do not want to have a bound that holds uniformly over the whole space of possible functions $\mathcal{H}$ since we are interested in algorithms that *may not explore it*.

Given the above insight, we are looking to deal with algorithms that will not explore everything in a very complicated function space. An intuitive idea for neural network training is that SGD is somehow conservative in its updates exploring a small subset of potential neural networks. The notion of being 'conservative' will be captured by *algorithmic stability*.

There are many ways to define and quantify the stability of a learning algorithm. The natural way of making such a definition is to start from the goal: we want to get bounds on the generalization error of specific learning algorithm and we want these bounds to be tight when the algorithm satisfies the stability criterion.

As one may expect, the more restrictive a stability criterion is, the tighter the corresponding bound will be.

In the learning model we consider, the randomness comes from the sampling of the training set. We will thus consider stability with respect to changes in the training set. To model stability, we will consider only restricted changes such as the removal or the replacement of one single example in the training set.

Definition 8: Uniform Stability

An algorithm A has **uniform stability** β with respect to ℓ if

$$\forall S \in (\mathcal{X} \times \mathcal{Y})^n,\ \forall i \in [n],\ \|\ell(A(S), \cdot) - \ell(A(S_{-i}), \cdot)\|_\infty \leq \beta\,,$$

where $S_{-i} = \{z_1, ..., z_{i-1}, z_{i+1}, ..., z_n\}$ corresponds to the training set S with the i-th example removed.

Note that the above is extremely strong: it is worst-case over all possible datasets that the algorithm could encounter and worst-case over examples from the domain. Moreover, stability is tightly related to a loss function ℓ.

Definition 9: Hypothesis Stability

An algorithm A has **hypothesis stability** β with respect to ℓ if

$$\forall i \in [n],\ \mathop{\mathbf{E}}_{S,z} |\ell(A(S), z) - \ell(A(S_{-i}), z)| \leq \beta\,,$$

where $S_{-i} = \{z_1, ..., z_{i-1}, z_{i+1}, ..., z_n\}$ corresponds to the training set S with the i-th example removed.

Definition 10

An algorithm A has **pointwise hypothesis stability** β with respect to ℓ if

$$\forall i \in [n],\ \mathop{\mathbf{E}}_{S} |\ell(A(S), z_i) - \ell(A(S_{-i}, z_i)| \leq \beta\,,$$

where $S_{-i} = \{z_1, ..., z_{i-1}, z_{i+1}, ..., z_n\}$ corresponds to the training set S with the i-th example removed.

We will only deal with uniformly stable algorithms but similar generalization bounds can be obtained under weaker assumptions.

Insight 5: The practical perspective

Practical methods have been developed to deal with instability of learning algorithms. In particular, Breiman introduced the *bagging* technique which is presented as a method to combine single classifiers in such a way that the variance of the overall combination is decreased. Recently, it was shown that bagging is an optimal PAC learner (Larsen, 2023).

7 Generalization of Uniformly Stable Algorithms

A stable algorithm has the property that removing one element in its learning set does not change much of its outcome. As a consequence, the difference between empirical and generalization error, if thought as a random variable, should have a small variance. If its expectation is small, stable algorithms should then be good candidates for their empirical error to be close to their generalization error. This assertion is formulated in the following theorem:

Theorem 4: Uniform Stability $\Rightarrow$ Generalization

Let A be an algorithm with uniform stability β and assume that $0 \leq \sup_S \sup_z \ell(A(S), z) \leq M$. For any sample size $n \geq 1$ and confidence $\delta \in (0, 1)$, the following generalization bound holds with

probability $1-\delta$ over the training set S:

$$R(A,S) \leq R_{emp}(A,S) + 2\beta + (4n\beta + M)\sqrt{\log(1/\delta)/n}\,.$$

This theorem gives tight bounds when the stability scales as $1/n$ ($1/n$ is the rate one should expect for the stability parameter $\beta = \{\beta_n\}$). We shortly mention that for classification tasks, the stability parameter is binary, i.e., $\beta = 0$ or $\beta = 1$. To deal with it, we usually consider algorithms with real-valued outputs. Then, the label predicted by such an algorithm is the sign of its real-valued output.

7.1 Proving that Uniform Stability Implies Generalization

We are now ready to prove the following result. It will be assumed that the algorithm A is symmetric with respect to S, i.e., it does not depend on the order of the elements in the training set.

Theorem 5: Uniform Stability Implies Generalization

Let A be a symmetric algorithm with uniform stability β and assume that $0 \leq \sup_S \sup_z \ell(A(S),z) \leq M$. For any sample size $n \geq 1$ and confidence $\delta \in (0,1)$, the following generalization bound holds with probability $1-\delta$ over the training set S:

$$R(A,S) \leq R_{emp}(A,S) + 2\beta + (4n\beta + M)\sqrt{\log(1/\delta)/n}\,.$$

Proof. Our goal is to apply McDiarmid's Inequality.

Fix $z_1,...,z_n,z_i' \in Z$. Let $S = \{z_1,...,z_n\}$ and define S^i to be obtained by S by removing z_i and adding z_i'.

Let $f : Z^n \to \mathbb{R}$ be any measurable function for which there exist constants c_i such that

$$\sup_{S\in Z^n, z_i'\in Z} |f(S) - f(S^i)| \leq c_i\,.$$

Then, it holds

$$\mathop{\mathbf{Pr}}_S[|f(S) - \mathop{\mathbf{E}}_S f| \geq \epsilon] \leq \exp\left(-2\epsilon^2 / \sum_i c_i^2\right).$$

Exercise 1

Prove McDiarmid's Inequality (for the proof, see Van Handel (2014).).

As an application of McDiarmid's inequality, let us find c_i for our functions of interest:

$$|R - R_{-i}| \leq \mathop{\mathbf{E}}_z |\ell(A(S),z) - \ell(A(S_{-i},z)| \leq \beta\,,$$

and

$$|R_{emp} - R_{emp,-i}| \leq \frac{\ell(A(S),z_i)}{n} + \frac{1}{n-1}\sum_{j\neq i} |\ell(A(S),z_j) - \ell(A(S_{-i},z_j)| \leq \beta + M/n\,.$$

This means that

$$|R - R^i| \leq |R - R_{-i}| + |R_{-i} - R^i| \leq 2\beta\,.$$

and

$$|R_{emp} - R^i_{emp}| \leq \frac{1}{n}\sum_{j\neq i} |\ell(A(S),z_j) - \ell(A(S_{-i},z_j)| + \frac{1}{n}\sum_{j\neq i} |\ell(A(S_{-i}),z_j) - \ell(A(S^i,z_j)| + \frac{1}{n}|\ell(A(S),z_i) - \ell(A(S^i),z_i')|.$$

The above is at most $2\beta + M/n$. Hence, let us set $f = R - R_{emp}$, which satisfies McDiarmid's inequality with $c_i = 4\beta + M/n$. Since by uniform stability and symmetry

$$\mathop{\mathbf{E}}_{S} R(A,S) - R_{emp}(A,S) \leq 2\beta\,,$$

we get the desired tail bound. The above inequality is shown as follows:

$$\mathop{\mathbf{E}}_{S} R(A,S) - R_{emp}(A,S) = \mathop{\mathbf{E}}_{S,z_i'} \ell(A(S),z_i') - \mathop{\mathbf{E}}_{S} R_{emp}(A,S)$$

and

$$\mathop{\mathbf{E}}_{S} R_{emp}(A,S) = \mathop{\mathbf{E}}_{S,z_i'} \frac{1}{n}\sum_i \ell(A(S),z_i)\,.$$

By the i.i.d. assumption and symmetry, let us rename any z_i as z_i' (the random variable $\ell(A(\{z_1,...,z_i,...,z_n\}),z_i)$ will have the same law as $\ell(A\{z_1,...,z_i',...,z_n\},z_i')$ and get for all $i \in [n]$:

$$\mathop{\mathbf{E}}_{S} R_{emp}(A,S) = \mathop{\mathbf{E}}_{S,z_i'} \frac{1}{n}\sum_i \ell(A(S),z_i) = \mathop{\mathbf{E}}_{S,z_i'} \ell(A(S^i),z_i')\,.$$

Now using uniform stability, we get the result. □

7.2 Applications of Stability

Local Algorithms A k-Local Rule is a classification algorithm that determines the label of an instance x based on the k closest instances in the training set. The simplest example of such a rule is the k-Nearest Neighbors (k-NN) algorithm which computes the label by a majority vote among the labels of the k nearest instances in the training set.

With respect to the $\{0,1\}$-loss function, the k-NN classifier has hypothesis stability

$$\beta \leq \frac{4}{n}\sqrt{k/2\pi}\,.$$

This was proven in Devroye and Wagner (1979a).

Regularization Uniform stability may appear as a strict condition. However, many existing learning methods exhibit a uniform stability which is controlled by the regularization parameter and can thus be very small (for details see Bousquet and Elisseeff (2002)).

7.3 Extensions of Algorithmic Stability

The main object of study in Lecture 2 is uniform stability, a property of the learning algorithm that captures how sensitive the (loss of the) algorithm is when changing a single element of its training set.

> **Definition 11: Uniform Stability (Revisited)**
>
> Algorithm A is said to have uniform stability β with respect to ℓ if for any pair of datasets $S, S' \in \mathcal{Z}^n$ that differ in a single element and every $z \in \mathcal{Z}$, $|\ell(A(S),z) - \ell(A(S'),z)| \leq \beta$.

Denote the generalization gap of A trained on S as

$$\Delta_S(A) = R(A,S) - R_{emp}(A,S)$$

Recall that R is the population loss while R_{emp} is the loss on the training set. For a uniformly stable algorithm, we have the following properties for any distribution $\mathcal{D}$ over Z and $\delta > 0$:

1. $|\mathbf{E}_{S\sim\mathcal{D}^n}[\Delta_S(A)]| \leq \beta$;
2. $\mathbf{E}_{S\sim\mathcal{D}^n}[(\Delta_S(A))^2] \leq \frac{1}{2n} + 6\beta$;
3. $\mathbf{Pr}_{S\sim\mathcal{D}^n}[\Delta_S(A) \geq (4\beta + 1/n)\sqrt{n\log(1/\delta)/2} + 2\beta] \leq \delta$.

Bound (3) does not lead to meaningful generalization bounds in many common settings where $\beta > 1/\sqrt{n}$. Feldman and Vondrak (2018) improve on (2) and (3) (using tools from differential privacy) by showing that

1. $\mathbf{E}_{S\sim\mathcal{D}^n}[(\Delta_S(A))^2] \leq \frac{2}{n} + 16\beta^2$;
2. $\mathbf{Pr}_{S\sim\mathcal{D}^n}[\Delta_S(A) \geq (2\beta + 1/n)\sqrt{\log(8/\delta)}] \leq \delta$.

Finally, Feldman and Vondrak (2019) show that

$$\mathop{\mathbf{Pr}}_{S\sim\mathcal{D}^n}\left[\Delta_S(A) \geq c(\beta\log n\log(1/\delta) + \frac{\sqrt{\log(1/\delta)}}{\sqrt{n}})\right] \leq \delta.$$

We also note that differential privacy and VC dimension can be incorporated into the above stability framework (see Dagan and Feldman (2019)).

8 General Theories for Generalization

8.1 Generalization via Sample Compression

There are other ways to show that a specific algorithm A generalizes. Algorithms whose output essentially only depends on a few of the input data points (e.g., SVM or the median), as formalized by compression schemes (Littlestone and Warmuth, 1986), can be shown to generalize.

Definition 12: Sample Compression

A sample compression scheme takes a long list of samples and compresses it to a short sub-list of samples in a way that allows to invert the compression. Formally, a sample compression scheme for C with kernel size k and side information I, where I is a finite set, consists of two maps κ, ρ for which the following hold:

1. The compression map
$$\kappa : L_{\mathcal{H}}(\infty) \to L_{\mathcal{H}}(k) \times I$$
taking the training set (X, y) (labeled by a function in $\mathcal{H}$) and compressing it to $((Z, z), i)$ where $Z \subset X$ and $z = y|_Z$.
2. The reconstruction map
$$\rho : L_{\mathcal{H}}(k) \times I \to \{0,1\}^{\mathcal{X}}$$
which has the property that for all $(X, y) \in L_{\mathcal{H}}(\infty)$ it holds $\rho(\kappa(X, y))|_X = y$.

The size of the compression scheme is $k + \log|I|$.

The side information I can be thought of as list decoding: the map ρ has a short list of possible reconstructions of a given dataset (X, y), and the information $i \in I$ indicates which element in the list is the correct one.

Littlestone and Warmuth showed that every compression scheme yields a natural learning procedure: Given a labeled sample $(S, h(S))$, the learner compresses it to $\kappa(S, h(S))$ and outputs the hypothesis $h = \rho(\kappa(S, h(S))$. They proved that this is indeed a PAC learner.

Theorem 6: Sample Compression Implies Generalization (Littlestone and Warmuth, 1986)

Let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, and let κ, ρ be a sample compression scheme for $\mathcal{H}$ of size k. Let $n \geq (k \log(1/\epsilon) + \log(1/\delta))/\epsilon$. Then, the learning map $(S, h(S)) \to \rho(\kappa(S, h(S)))$ is PAC learning $\mathcal{H}$ with n samples, generalization error ϵ and success probability $1 - \delta$.

The proof idea is as follows: given a training set $X = \{x_1, ..., x_n\}$, fix a subset T of size k and information $i \in I$: this choice induces a hypothesis $h_{T,i}(x) = \rho((x_S, c|_S), i)$. The random function $h_{T,i}$ is independent of the remaining dataset X_{-S}. Since the size of the scheme is k, the number of pairs (T, i) is of order n^k and the probability that $h_{T,i}$ that has ϵ-generalization error agrees with X_{-S} is of order $(1-\epsilon)^{n-k}$. Hence, by union bound, the probability that the scheme outputs a hypothesis with ϵ-loss (since the scheme should output a mapping such that $h|_X = c|_X$) is $n^k(1-\epsilon)^{n-k} \leq \delta$ and the sample complexity follows.

The other direction is more interesting. Moran and Yehudayoff (2016) show the following surprising result: VC classes have sample compression schemes of finite size (independent of the size of the training set).

Theorem 7: PAC Learning Implies Sample Compression (Moran and Yehudayoff, 2016)

If $\mathcal{H}$ has VC dimension d, then $\mathcal{H}$ has a sample compression scheme of size 2^d.

Exercise 2

Read the proof of Theorem 5 from Moran and Yehudayoff (2016).

It remains open to improve this bound (see also Chase et al. (2024)). Before moving on, we would like to mention that compression and generalization have close connections in many areas of research, such as memory and cognition, e.g., recall the chess experiment that we discussed in the class Frey and Adesman (1976). The study aimed to investigate how expertise in chess affects the ability to recall chessboard configurations. The methodology of the experiment was as follows:

- *Participants:* Chess players of varying skill levels, categorized as novices or experts.
- *Stimuli:* Two types of chessboard configurations were used:
  1. *Meaningful positions:* Derived from actual games (plausible configurations based on chess rules).
  2. *Random positions:* Created by placing pieces randomly on the board, breaking typical game patterns.
- *Procedure:* Participants were briefly shown a chess position for a limited exposure time. Afterward, they were asked to recreate the position from memory on a blank chessboard.

Experts significantly outperformed novices in recalling meaningful positions. For random positions, experts and novices showed similar levels of recall performance, indicating that the expertise advantage relies on the meaningfulness of the stimuli. Experts' superior performance was attributed to their ability to *compress* information, grouping related pieces into familiar patterns. In random positions, the absence of familiar patterns reduced their advantage.

8.2 Generalization via Mutual Information

The mutual information (MI) of two random variables is a measure of the *mutual dependence* between the two variables. It quantifies the amount of information (in units such as bits or nats) obtained about one random variable by observing the other random variable.

Definition 13: Mutual Information

For two random variables X, Y with joint probability distribution $P_{X,Y}$, let us denote by P_X and P_Y their marginals. The mutual information is defined as

$$I(X;Y) = \mathrm{KL}(P_{X,Y} \,\|\, P_X \otimes P_Y)\,.$$

Note that $I(X;Y)$ is equal to zero precisely when the joint distribution coincides with the product of the marginals. It is closely related to the entropy: $I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$ and is symmetric: $I(X;Y) = I(Y;X)$.

We will use the following notation:

1. $\mathcal{Z}$ will contain labeled examples $z = (x, y)$.
2. $\mathcal{H}$ is the space of functions.
3. $Z \sim \mathcal{D}^n$ is a dataset consisting of n i.i.d. labeled examples drawn from $\mathcal{D}$.

A recent line of work has studied generalization using mutual information and related quantities (Russo and Zou, 2016; Raginsky et al., 2016, etc.). Specifically, for a (possibly randomized) algorithm $A : \mathcal{Z}^n \to \mathcal{H}$ and a dataset $Z \sim \mathcal{D}^n$, we consider the quantity $I(A(Z); Z)$. This measures how much information the output of A contains about its input.

The following is shown in Russo and Zou (2016); Xu and Raginsky (2017).

Theorem 8: Bounded Mutual Information Implies Generalization (Xu and Raginsky, 2017)

If $\ell : \mathcal{H} \times \mathcal{Z} \to [0,1]$, $A : \mathcal{Z}^n \to \mathcal{H}$ and $Z \sim \mathcal{D}^n$, then

$$|\mathbb{E}\ell(A(Z), Z) - \ell(A(Z), \mathcal{D})| \leq \sqrt{\frac{2}{n} \cdot I(A(Z); Z)}\,.$$

Note that this bound controls the expected generalization error. There exist algorithms that can make the RHS trivially zero (e.g., use the constant function independently of the dataset). This algorithm however has very bad generalization error. The above inequality does not guarantee anything about the quality of the algorithm, it only allows to argue that the empirical and the population losses will be close.

The key idea behind this result is the following Lemma.

Lemma 4

Consider random variables (X, Y) with joint distribution $P_{X,Y}$. Consider an independent copy $\bar{X}$ of X and an independent copy $\bar{Y}$ of Y such that $P_{\bar{X},\bar{Y}} = P_X \otimes P_Y$. If $f(\bar{X}, \bar{Y})$ is σ-subgaussian with respect to $P_{\bar{X},\bar{Y}}$, then

$$|\,\mathbf{E}[f(X,Y)] - \mathbf{E}[f(\bar{X},\bar{Y})]| \leq \sqrt{2\sigma^2 I(X;Y)}\,.$$

Exercise 3

Prove Lemma 1.

To use this result, we set $X = S$ (a drawn dataset) and Y to be a classifier drawn from the algorithm $P_{h|S}$. Note that $P_{X,Y} = \mathcal{D}^n \times P_{h|S}$. Now we create the independent copies: $\bar{X}$ will be a new dataset and $\bar{Y}$ will be an independent copy of Y (which is independent of $\bar{X}$ and is drawn from the marginal). Hence the expected

generalization error can be written as

$$\mathbf{E}[f(\bar{X}, \bar{Y})] - \mathbf{E}[f(X, Y)]$$

where $f(s, h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h, z_i)$.

Bounds on mutual information can be obtained from differential privacy or from bounds on the entropy of the output of A. Specifically, if A is ε-differentially private, then $I(A(Z); Z) \leq \frac{1}{2}\varepsilon^2 n$ (McGregor et al., 2010; Bun and Steinke, 2016).

Unfortunately, mutual information can easily be infinite even in settings where generalization is easy to prove. Bassily et al. (2018); Nachum et al. (2018) showed that *any* proper and consistent learner for threshold functions A must have $I(A(Z); Z) \geq \Omega\left(\frac{\log\log|\mathcal{Z}|}{n^2}\right)$ when $Z \sim \mathcal{D}^n$ for some worst-case distribution $\mathcal{D}$. In contrast, the VC dimension of threshold functions is 1, which implies strong uniform convergence bounds even for infinite domains Z.

8.3 Generalization via CMI

The issue with the mutual information approach is that even a single data point has infinite information content if the distribution is *continuous*. This makes it difficult to bound the mutual information – an algorithm revealing a single data point is not an issue for generalization, but it is for mutual information.

Steinke and Zakynthinou (2020) introduce the conditional mutual information (CMI) framework for reasoning about the generalization properties of machine learning algorithms. CMI is a quantitative property of an algorithm A. (Note that it does *not* depend on the loss function of interest; compare this to uniform stability.)

The definition of CMI will be inspired by the symmetrization technique, which we used in Lecture 1 in order to show that finite VC dimension implies uniform convergence. Intuitively, CMI measures how well we can "recognize" the input (i.e., training data) given the output (i.e., trained model) of the algorithm. Recognizing the input is formalized by considering a set consisting of $2n$ data points – namely the n input data points mixed with n "ghost" data points (as we did in the symmetrization trick) – and measuring how well it is possible to distinguish the true inputs from their ghosts.

The supersample is randomly partitioned into the input and the ghost samples. We then measure how much information the output reveals about this partition using mutual information, where we take the supersample to be known (i.e., we condition on the supersample and the unknown information is how it is partitioned). We now state the formal definition of CMI:

Definition 14: Conditional Mutual Information

Let $A : \mathcal{Z}^n \to \mathcal{H}$ be a randomized algorithm. Let $\mathcal{D}$ be a probability distribution on $\mathcal{Z}$ and let $Z' \in \mathcal{Z}^{n\times 2}$ consist of $2n$ samples drawn independently from $\mathcal{D}$. Let $S \in \{0,1\}^n$ be uniformly random and independent from Z' and the randomness of A.
Define $\mathcal{Z}'_S \in \mathcal{Z}^n$ by $(\mathcal{Z}'_S)_i = Z'_{i,S_i+1}$ for all $i \in [n]$ – that is, Z'_S is the subset of Z' indexed by S.
The *conditional mutual information (CMI) of A with respect to $\mathcal{D}$* is

$$\mathsf{CMI}_{\mathcal{D}}(A) := I(A(Z'_S); S | Z').$$

The *(distribution-free) conditional mutual information (CMI) of A* is

$$\mathsf{CMI}(A) := \sup_{z' \in \mathcal{Z}^{n\times 2}} I(A(z'_S); S).$$

The above quantity captures the following: Given the supersample Z', how much information does the output of A trained in Z'_S reveal about S?

The following result shows that CMI controls the gap between the empirical and the generalization error.

Theorem 9: Bounded CMI Implies Generalization (Steinke and Zakynthinou, 2020)

Let $A : \mathcal{Z}^n \to \mathcal{H}$ and $\ell : \mathcal{H} \times \mathcal{Z} \to [0,1]$. Let $\mathcal{D}$ be a distribution on $\mathcal{Z}$ and define $\ell(w, \mathcal{D}) = \mathbf{E}_{Z \sim \mathcal{D}} \ell(w, Z)$ and $\ell(w, z) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, z_i)$ for all $w \in \mathcal{H}$ and $z \in \mathcal{Z}^n$. Then

$$\left| \mathop{\mathbf{E}}_{Z \sim \mathcal{D}^n, A} \ell(A(Z), Z) - \ell(A(Z), \mathcal{D}) \right| \leq \sqrt{\frac{2}{n} \cdot \mathsf{CMI}_{\mathcal{D}}(A)}, \tag{2}$$

A definition is useful if it can be used to capture various existing approaches; this is exactly the case for CMI, as we will review below.

Compression Schemes If an algorithm $A : \mathcal{Z}^n \to \mathcal{H}$ has a compression scheme of size k, then $\mathsf{CMI}(A) \leq O(k \cdot \log n)$. Intuitively, this is in agreement with the fact that an algorithm revealing k of the input points and nothing about the rest would have a CMI of k bits.

The argument goes intuitively as follows: Let Z'_S be the random training set chosen by S. Since A is based on sample compression, $I(A(Z'_S); S) = I(\rho(\kappa(Z'_S)); S)$ (let us ignore the information set for simplicity). By the data-processing inequality, this information is at most $I(\rho(\kappa(Z'_S)); S) \leq I((Z'_S)_{K(Z'_S)}; S)$, where $K(Z'_S)$ is the set of indices chosen by the compression algorithm κ on input Z'_S. This means that the information revealed is of order k bits. Since Z' is fixed and S is random, this information is at most $k \log(2n)$, since the possible subsets are at most $\binom{2n}{k}$.

Uniform Convergence & VC Dimension Bounded VC dimension is a necessary and sufficient condition for uniform convergence and is hence a sufficient condition for generalization (recall Lecture 1). Note that CMI is a property which depends on the algorithm, whereas the VC dimension is a property of the output space; this appears to cause an incompatibility between the two methods. However, one can show the following: for any hypothesis class of bounded VC dimension, there always *exists* some ERM algorithm $A : \mathcal{Z}^n \to \mathcal{H}$ with bounded CMI:

Theorem 10

Let $\mathcal{Z} = \mathcal{X} \times \{0,1\}$ and let $\mathcal{H} = \{h : \mathcal{X} \to \{0,1\}\}$ be a hypothesis class with VC dimension d. Then, there exists an empirical risk minimizer $A : \mathcal{Z}^n \to \mathcal{H}$ for the 0-1 loss such that $\mathsf{CMI}(A) \leq d \log n + 2$.

Note that it is not true that *every* empirical risk minimizer for a class of bounded VC dimension has bounded CMI.

Distributional Stability & Differential Privacy Finally, we show that distributional stability implies CMI bounds. Differential privacy is the most well-known form of distributional stability and its generalization properties are well-established (Dwork et al., 2015; Bassily et al., 2016; Jung et al., 2019).

Theorem 11

Let $A : \mathcal{Z}^n \to \mathcal{H}$ be a randomized algorithm. The following conditions imply that $\mathsf{CMI}(A) \leq \varepsilon n$.

(i) A is $\sqrt{2\varepsilon}$-differentially private (Dwork et al., 2006).

(ii) A is ε-TV stable (Bassily et al., 2016).[a]

[a] A randomized algorithm is ϵ-TV stable if for any pair of samples that differ on one element, $\mathrm{TV}(A(S), A(S')) \leq \epsilon$.

We will formally define and study the above notions of stability later in the course.

Loss Stability/Uniform Stability A long line of work Rogers and Wagner (1978); Devroye and Wagner (1979b); Bousquet and Elisseeff (2002) has proven generalization bounds by showing that various algorithms have the property that their loss changes very little if a single input datum is replaced or removed (recall the first part of Lecture 2). Uniform stability (one of the strongest and most well-studied variants of loss stability) has the advantage that it readily yields high probability generalization bounds. However, one limitation of the loss stability approach is that the loss function is an integral part of the definition, whereas CMI do not depend on the loss function. It is of course natural to ask whether loss stability and CMI can be unified in some way. Steinke and Zakynthinou (2020) propose a variant of CMI (Evaluated CMI or eCMI) that takes the loss function into account and allows us to translate between the notions.

Here we define

$$\mathrm{eCMI}_{\mathcal{D}}(\ell(A)) = I(\ell(A(Z'_S), Z'); S|Z')\,.$$

By the data-processing inequality, $\mathrm{eCMI}_{\mathcal{D}}(\ell(A)) \leq \mathrm{CMI}_{\mathcal{D}}(A)$. It turns out given a uniformly stable algorithm A under loss ℓ, one can bound the evaluated CMI of an algorithm $\widetilde{A}$, which corresponds to A with some 'Gaussian perturbations' to its loss.

Lecture 3: Stability of SGD and Randomization Tests

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

In Lecture 3, we will first continue our discussion on algorithmic stability. We will study the work of Hardt et al. (2016) on the stability of Stochastic Gradient Descent, probably the most applied ML algorithm. Next, we will depart from the framework of stability and discuss the experimental work of Zhang et al. (2021) on questioning the theories for generalization in modern deep learning.

9 Stochastic Gradient Descent

The most widely used optimization method in machine learning practice is stochastic gradient method (SGM). Stochastic gradient methods aim to minimize the empirical risk of a model by repeatedly computing the gradient of a loss function on a single training example, or a batch of few examples, and updating the model parameters accordingly. SGM is scalable, robust, and performs well across many different domains ranging from smooth and strongly convex problems to complex non-convex objectives.

- Let $L(w) = \frac{1}{n}\sum_i L_i(w) = \frac{1}{n}\sum_i \ell(w; x_i)$.
- Standard Gradient Update: $w \leftarrow w - \eta \nabla_w L(w)$.

In contrast, SGD performs updates as follows:

- Set $w_0 = 0$.
- Randomly shuffle samples in the training set.
- for $i = 1, ..., n : w \leftarrow w - \eta \nabla_w L_i(w)$.

More generally, SGD requires an unbiased estimate of the true gradient.

Insight 6: An advantage of SGD compared to GD

SGD is in many cases crucial for *computational efficiency* compared to vanilla gradient descent on the empirical loss. As an illustration, consider the truncated statistics setting in high dimensions (Daskalakis et al., 2018). Here, there is an unknown normal distribution $\mathcal{N}(\mu^*)$ (with identity covariance matrix for simplicity) in d dimensions and a truncation set $S \subseteq \mathbb{R}^d$, which we can access only with a membership oracle $1_S : \mathbb{R}^d \to \{0, 1\}$. We assume that the set has no particular structure except that is satisfies that $\int_S \mathcal{N}(\mu^\star; y)dy = \alpha > 1\%$.

The goal is to estimate μ^* in poly(d) time using i.i.d. samples from the truncated distribution $\mathcal{N}_S(\mu^\star)$, where $\mathcal{N}(\mu^\star; x) = \mathcal{N}(\mu^\star; x)1_S(x) / \int_S \mathcal{N}(\mu^\star; y)dy$.

A natural strategy to solve this problem is to run gradient descent on the log-likelihood objective (for any exponential family, the truncated negative log-likelihood is convex). Writing down the (negative) log-likelihood for a single sample x at our current guess μ, we can observe that

$$-\log \mathcal{N}(\mu; x) = \frac{1}{2}\|x - \mu\|^2 + \log\left(\int_S \exp\left(-\frac{1}{2}\|z - \mu\|^2\right)dz\right) .$$

Note that running gradient descent on this objective is intractable in high dimensions, since it in-

volves integrating over this high dimensional set S (and we only have membership access to the set). However, if we compute the gradient, we can see that

$$\nabla_\mu(-\log \mathcal{N}(\mu; x)) = -x + \mathop{\mathbf{E}}_{z\sim\mathcal{N}_S(\mu)}[z]\,.$$

This means that even if we cannot compute this gradient efficiently, we can compute efficiently an unbiased estimate of it (since we can sample from $\mathcal{N}_S(\mu)$, where μ is our current guess) given that the normal centered at μ places a lot of mass to S. This allows us to run SGD efficiently.

10 Stability of Stochastic Gradient Descent

10.1 Some Preliminaries from Lecture 2

We will need some notation from Lecture 2. Given algorithm A and training set S, define the population loss on a fresh sample z as

$$R(A, S) = \mathop{\mathbf{E}}_{z}\ell(A(S), z)\,.$$

Also, let the empirical loss on $S = \{z_1 = (x_1, y_1), ..., z_n = (x_n, y_n)\}$ be

$$R_{emp}(A, S) = \frac{1}{n}\sum_{i=1}^{n}\ell(A(S), z_i)\,.$$

Problem 2: High-Probability Generalization Bounds for specific algorithm A

For any ϵ, upper bound the tail probability

$$\mathop{\mathbf{Pr}}_{S}\left[|R_{emp}(A, S) - R(A, S)| > \epsilon\right]\,.$$

We remind the reader that even if an algorithm A generalizes in the above sense, it does not mean that it is a good learning algorithm. If one additionally manages to show that A achieves small training error, we can guarantee that it also gets small generalization error.

In this Lecture, we will show that parametric models trained by a stochastic gradient method (SGM) with few iterations have vanishing generalization error. We prove our results by arguing that SGD is algorithmically stable in the sense of Lecture 2.

Definition 15: Uniform Stability (Bousquet and Elisseeff, 2002)

A randomized algorithm A has **uniform stability** ϵ_{stab} with respect to ℓ if for all datasets $S, S' \in Z^n$ that differ in at most one example, we have

$$\sup_z \mathop{\mathbf{E}}_{A}|\ell(A(S), z) - \ell(A(S'), z)| \le \epsilon_{stab}\,,$$

where the expectation is is taken only over the internal randomness of A.

In words, an algorithm is stable if the training error it achieves varies only slightly if we change any single training data point. The precise notion of stability we use is known as uniform stability. It states that a randomized algorithm A is uniformly stable if for all data sets differing in only one element, the learned models produce nearly the same predictions.

Observe that the two executions are *coupled* since the randomness is shared across them. This will be crucial for the upcoming analysis. (Think of it as sharing the random seed across the two executions.)

In what follows the algorithm $A : (\mathcal{X} \times \mathcal{Y})^n \to W$ takes as input a training set and runs T steps of SGD on a parametric model (e.g., weights of a neural network) to output parameters w_T.

10.2 Setup for SGD Analysis

We start with the work of Hardt et al. (2016). Given n samples $S = \{z_1, ..., z_n\}$, consider a decomposable function

$$f(w) = \frac{1}{n} \sum_i f(w; z_i) \,,$$

where $f(w; z_i)$ is the loss of w on the example z_i. The SGD update is

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t; z_{i_t}) \,.$$

The important thing here is the index i_t : The SGD algorithm is obtained by performing the SGD update T times where the indices $i_1, ..., i_T$ are *randomly and independently chosen*. There are two popular schemes for choosing the examples' indices. One is to pick it uniformly at random in $[n]$ at each step. The other is to choose a random permutation over $[n]$ and cycle through the examples repeatedly in the order determined by the permutation. The results hold for both variants.

10.3 The Main Result: SGD is Stable

In order to prove that the stochastic gradient method is stable, we will analyze the output of the algorithm on two data sets that differ in precisely one location.

Step 1. If the loss function is L-Lipschitz with respect to w for every example z, we have

$$\mathbf{E} \, |f(w; z) - f(w'; z)| \leq L \, \mathbf{E} \, \|w - w'\|$$

for all w and w'. This means that $\|\nabla_w f(w; z)\| \leq L$ for any z. Hence, it suffices to analyze how w_t and w'_t diverge in the domain as a function of time t.

Step 2. Because of the recursive structure of SGD, our goal is to control $\delta_t = \|w_t - w'_t\|$ in expectation as a function of δ_{t-1}.

Step 3. Based on the random index choice and the fact that S, S' differ in one element, there are two cases to consider.

- In the first case, SGD selects the index of an example at step t which is identical in S and S'. Unfortunately, it could still be the case that δ_t grows, since w_t and w'_t differ and so the gradients at these two points may still differ.
- The second case to consider is when SGD selects the one example to update in which S and S' differ. Note that this happens only with probability $1/n$ if examples are selected randomly. In that case,

  $$\delta_t = \|w_{t-1} - \alpha_{t-1} \nabla_w f(w_{t-1}; z) - w'_{t-1} + \alpha_{t-1} \nabla_w f(w'_{t-1}; z')\| \leq \delta_{t-1} + 2\alpha_{t-1} L \,,$$

  since f is L-Lipschitz.

It remains to combine the two cases to complete the proof.

Step 4. Let us begin with the simple case where f is convex.

Definition 16

A function $f : X \to \mathbb{R}$ is convex if for all $x, y \in X$, it holds

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) \,.$$

A function $f : X \to \mathbb{R}$ is β-smooth if for all $x, y \in X$, it holds

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\| \,.$$

Theorem 12: Convex Case (Hardt et al., 2016)

Assume that the loss function $f(\cdot; z)$ is β-smooth, convex and L-Lipschitz for any z. Suppose that we run SGD (with either the random index resampling scheme or the permutation one) with step sizes $\alpha_t \leq 2/\beta$ for T iterations with the same initialization. Then, SGD satisfies uniform stability with $\epsilon_{stab} \leq \frac{2L^2}{n} \sum \alpha_t$.

Proof. It suffices to bound $\mathbf{E}[\delta_T]$. Observe that at step t, with probability $1 - 1/n$, the example selected by SGD is the same in both S and S'. This means that

$$\mathbf{E}[\delta_{t+1}] \leq (1 - 1/n) \cdot S_t^{(1)} + 1/n \cdot S_t^{(2)}$$

where $S_t^{(1)}$ corresponds to the case where the method selects a common point of S, S' :

$$S_t^{(1)} = \mathbf{E} \, \|w_t - \alpha_t \nabla_w f(w_t; z) - w_t' + \alpha_t \nabla_w f(w_t'; z)\|$$

and $S_t^{(2)}$ corresponds to picking the index of the differing point:

$$S_t^{(2)} = \mathbf{E} \, \|w_t - \alpha_t \nabla_w f(w_t; z) - w_t' + \alpha_t \nabla_w f(w_t'; z')\| \,.$$

Analysis for $S_t^{(1)}$. We will now use convexity and smoothness. An update rule G is η-expansive if for all $w_1, w_2 \in \Omega$, $\|G(w_1) - G(w_2)\| \leq \eta \|w_1 - w_2\|$. It is not difficult to see that the SGD update rule is 1-expansive if f is β-smooth and $\alpha \leq 2/\beta$ (see Lemma 3.7.2 in Hardt et al. (2016)). Note that if we only use smoothness we get that G is $(1 + \alpha\beta)$-expansive. But, convexity and smoothness imply that the gradients are co-coercive, namely

$$\langle \nabla g(w_1) - \nabla g(w_2), w_1 - w_2 \rangle \geq \frac{1}{\beta} \|\nabla g(w_1) - \nabla g(w_2)\|^2 \,.$$

Exercise 4

Prove that ∇f is co-coersive if f is convex and smooth.
(It suffices to show that $f(y) - f(x) - \nabla f(x)^\top (y - x) \geq \frac{1}{2\beta} \|\nabla f(x) - \nabla f(y)\|^2$.)

This now gives that

$$\|G(w_1) - G(w_2)\|^2 = \|w_1 - w_2\|^2 + \alpha^2 \|\nabla g(w_1) - \nabla g(w_2)\|^2 - 2\alpha \langle \nabla g(w_1) - \nabla g(w_2), w_1 - w_2 \rangle \leq \|w_1 - w_2\|^2 \,.$$

This means that $S_t^{(1)} \leq \mathbf{E} \, \delta_t$.

Analysis for $S_t^{(2)}$. Based on the discussion in Step 3, we will get that $S_t^{(2)} \leq \mathbf{E} \, \delta_t + 2\alpha_t L$.

Combining the two and using recursion, we get that $\mathbf{E} \, \delta_T \leq 2L \sum_t \frac{\alpha_t}{n}$. This means that SGD is uniformly stable with parameter $2L^2 \sum_t \frac{\alpha_t}{n}$ (here we used the fact that we started from the same initial condition). □

Non-Convex Regime The above extends to non-convex objectives. The main idea is to observe that SGD typically makes several steps before it even encounters the one example on which two data sets in the stability analysis differ.

Lemma 5: (Hardt et al., 2016)

Assume that the loss function $f(\cdot; z)$ is nonnegative and L-Lipschitz for all z. Let S and S' be two samples of size n differing in only a single example. Denote by w_T and w'_T the output of T steps of SGD on S and S'. respectively. Then, for every $z \in Z$ and every $t_0 \in \{0, 1, ..., n\}$, under both the random update rule and the random permutation rule, we have

$$\mathbf{E}\left|f(w_T; z) - f(w'_T; z)\right| \leq \frac{t_0}{n} \sup_{w,z} f(w, z) + L\,\mathbf{E}[\delta_T | \delta_{t_0} = 0]\,.$$

This can be used to show that

Theorem 13: Stability of SGD (Hardt et al., 2016)

Assume that $f(\cdot; z) \in [0, 1]$ is an L-Lipschitz and β-smooth loss function for every z. Suppose that we run SGD for T steps with monotonically non-increasing step sizes $\alpha_t \leq c/t$. Then, SGD satisfies uniform stability with

$$\epsilon_{stab} = \frac{T^{1-\frac{1}{\beta c+1}}}{n}\,.$$

This shows that the method generalizes provided that (i) the steps are sufficiently small α_t is small) and (ii) the number of iterations is not too large (T is small). More specifically,the number of steps of stochastic gradient can grow as n^C for a small $C > 1$. This can be used to provide some explanation on why neural networks can be trained for multiple epochs of stochastic gradient descent (multiple passes of the training set) and still exhibit good generalization.

Insight 7

The above bounds are algorithm specific. They specifically apply to SGD. Since the number of iterations we allow can be larger than the sample size, an arbitrary algorithm could easily achieve small training error by *memorizing all training data* with no generalization ability. In contrast, if the stochastic gradient method manages to fit the training data in a reasonable number of iterations, it is guaranteed to generalize.

We remind the reader that the above results do not focus on algorithms that yield low training error, but provide insights on algorithms that yield low generalization error. If, in addition, one can achieve low training error quickly on a non-convex problem with stochastic gradient, the above results guarantee that the resulting model generalizes well.

Exercise 5

Complete the proofs of the above results.

11 Rethinking Generalization in Deep Learning

There are a variety of theories proposed to explain generalization: Uniform convergence (recall Lecture 1), and algorithmic stability (Lectures 2,3) are but a few of the important conceptual tools to reason about generalization. The work of Zhang et al. (2021) aims to interrogate these different theories of generalization via simple experiments.[3]

11.1 The Randomization Test

The first experiment is very simple and is based on a long statistical literature (Edgington and Onghena, 2007). They create a copy of the training data where they replace each label independently by a random label chosen from the set of valid labels.

A dog picture labeled "dog" in the true dataset might thus become a dog picture labeled "cat". The crucial point is that *the randomization breaks any relationship between the instance, for example, the image, and the label.* Then they run the learning algorithm both on the natural data and on the randomized data with identical settings and model choice. By design, no generalization is possible on the randomized data (it is information-theoretically impossible to get better than $1/2$ even with 2 labels).

For any purported measure of generalization, it is now possible to compare how it fares on the natural data versus the randomized data. If it turns out to be the same in both cases, it could not possibly be a good measure of generalization for it cannot even distinguish learning from natural data (where generalization is possible) from learning on randomized data (where no generalization is possible). The main observation is:

Deep neural networks can easily fit random labels.

More precisely, when trained on a completely random labeling of the true data, neural networks achieve zero training error. The test error, of course, is no better than random chance as there is no correlation between the training labels and the test labels.

- The effective capacity[4] of neural networks is sufficient for memorizing the entire data set. This is interesting since one may suspect that the reason that neural networks could fit the training data is that they make use of abstractions between the images and the labels. However, since the training loss on the randomized dataset is also zero, this is not the case.
- Plot (b) deals with the training overhead: On the one side, it is important to see that even the optimization part on random labels remains 'easy'. In fact, training time increases only by a small constant factor compared with training on the true labels (see Plot (b) for the training overhead for different models). Hence what drives generalization cannot be identical to what makes optimization of deep neural networks easy in practice. On the other side, fitting random labels takes 3x time compared to the true ones, which is an important training overhead.

Connection to Hardt et al. (2016) and Uniform Stability Uniform stability of a learning algorithm is *independent of the labeling of the training data*. Hence, the concept is not strong enough to distinguish between the models trained on the true labels (small generalization error) and models trained on random labels (high generalization error). This also highlights why the analysis of Hardt et al. (2016) for non-convex optimization was rather pessimistic, allowing only a very few passes over the data (recall Theorem 2). The randomization test essentially shows that training neural networks is not uniformly stable for many passes over the data. However, it seems that there exists something reassuring about the pessimistic analysis of

[3]This paper also appears in this YouTube video by 3Blue1Brown: https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=3&ab_channel=3Blue1Brown

[4]By effective capacity, the authors of Zhang et al. (2021) informally refer to the size of the subset of models that is effectively achievable by the learning algorithm.

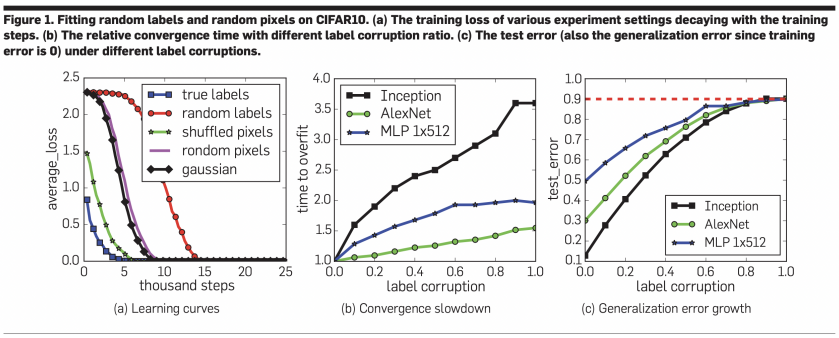


Figure 1: Taken from Zhang et al. (2021).

Theorem 2: when the number of steps are sufficiently small, then SGD is actually stable and the training error is close to the population error; however, in the randomized dataset, the training error is pretty high after a few iterations (it requires several steps to go to 0) and this is aligned to the prediction of stability theory.

Implication to Rademacher Complexity Recall that the empirical Rademacher complexity of a function class $\mathcal{F}$ on an unlabeled dataset $\{x_1, ..., x_n\}$ is defined as

$$\widehat{R}_n(\mathcal{F}) = \mathop{\mathbf{E}}_{\sigma} \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i \in [n]} \sigma_i f(x_i) \,.$$

The Rademacher complexity measures the ability of a function class to fit random binary label assignments, which closely resemble the randomization test (the points $x_1, ..., x_n$ are drawn from the marginal distribution on features, which is not changing when applying the randomization test). Since the neural networks fit random labels, we can see that $\widehat{R}_n(\mathcal{F}) \approx 1$. Often one bounds the Rademacher complexity of a function class $\mathcal{L} = \{\ell(f(x), y) : f \in \mathcal{F}\}$. For L-Lipschitz loss functions ℓ and real-valued class $\mathcal{F}$, it holds

$$\widehat{R}_n(\mathcal{L}) \leq L\widehat{R}_n(\mathcal{F}).$$

Since the neural networks fit random labels, Zhang et al. (2021) expect that $\widehat{R}_n(\mathcal{L})$ approximately achieves the maximum for the corresponding loss class.

Note that a trivial bound on the Rademacher complexity that does not lead to useful generalization bounds in realistic settings, since roughly speaking $\mathrm{err}_{\mathcal{D}}(h) - \mathrm{err}_S(h) \leq 2R_m(\mathcal{H}) + \sqrt{\log(1/\delta)/m)}$, where $R_m(\mathcal{H}) = \mathbf{E}_S[\widehat{R}_m(\mathcal{H})]$. Hence, the claim of Zhang et al. (2021) is that the effective capacity of neural networks trained by SGD cannot be small enough to explain the success of the model on generalizing on the natural data.[5]

11.2 The Role of Regularization

The second part of Zhang et al. (2021) studies regularization. Apart from the effective capacity of the model family, conventional wisdom attributes small generalization error to regularization techniques used during training.

Regularizers are the standard tool in theory and practice to mitigate overfitting in the regime when there are more parameters than data points. The basic idea is that although the original hypothesis is too large to generalize well, regularizers help confine learning to a subset of the hypothesis space with manageable complexity. By adding an explicit regularizer, say by penalizing the norm of the optimal solution, the effective Rademacher complexity of the possible solutions is dramatically reduced.

[5]Thought: However the effective size of the class of models explored by SGD is likely to depend on the training distribution, which changes between the two experiments.

- Data augmentation: augment the training set via domain-specific transformations. For image data, commonly used transformations include random cropping, random perturbation of brightness, saturation, hue and contrast.
- Weight decay: equivalent to a ℓ_2 regularizer on the weights; also equivalent to a hard constrain of the weights to an Euclidean ball, with the radius decided by the amount of weight decay.

Table 1. The training and test accuracy (in %) of various models on the CIFAR10 dataset.

Model	# params	Random crop	Weight decay	Train accuracy	Test accuracy
Inception	1,649,402	Yes	Yes	100.0	89.05
		Yes	No	100.0	89.31
		No	Yes	100.0	86.03
		No	No	100.0	85.75
(fitting random labels)		No	No	100.0	9.78
Inception w/o BatchNorm	1,649,402	No	Yes	100.0	83.00
		No	No	100.0	82.00
(fitting random labels)		No	No	100.0	10.12
Alexnet	1,387,786	Yes	Yes	99.90	81.22
		Yes	No	99.82	79.66
		No	Yes	100.0	77.36
		No	No	100.0	76.07
(fitting random labels)		No	No	99.82	9.86
MLP 3 × 512	1,735,178	No	Yes	100.0	53.35
		No	No	100.0	52.39
(fitting random labels)		No	No	100.0	10.48
MLP 1 × 512	1,209,866	No	Yes	99.80	50.39
		No	No	100.0	50.51
(fitting random labels)		No	No	99.34	10.61

Figure 2: Performance with and without data augmentation and weight decay are compared. Taken from Zhang et al. (2021).

Figure 2 shows that even with all of the regularizers turned off, all of the models still generalize well. Observe that even with the regularizers turned on the models can fit the random labels on ImageNet (see Figure 3).

Table 2: The top-1 and top-5 accuracy (in percentage) of the Inception v3 model on the ImageNet dataset. We compare the training and test accuracy with various regularization turned on and off, for both true labels and random labels. The original reported top-5 accuracy of the Alexnet on ILSVRC 2012 is also listed for reference. The numbers in parentheses are the best test accuracy during training, as a reference for potential performance gain of early stopping.

data aug	dropout	weight decay	top-1 train	top-5 train	top-1 test	top-5 test
ImageNet 1000 classes with the original labels						
yes	yes	yes	92.18	99.21	77.84	93.92
yes	no	no	92.33	99.17	72.95	90.43
no	no	yes	90.60	100.0	67.18 (72.57)	86.44 (91.31)
no	no	no	99.53	100.0	59.80 (63.16)	80.38 (84.49)
Alexnet (Krizhevsky et al., 2012)			-	-	-	83.6
ImageNet 1000 classes with random labels						
no	yes	yes	91.18	97.95	0.09	0.49
no	no	yes	87.81	96.15	0.12	0.50
no	no	no	95.20	99.14	0.11	0.56

Table 2 shows the performance on Imagenet with true labels and random labels, respectively.

Figure 3: Taken from Zhang et al. (2021).

Lecture 4: Failures of Uniform Convergence and Domain Adaptation

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

In Lecture 4, we will discuss the work of Nagarajan and Kolter (2019) on why uniform convergence fails to explain generalization in deep learning. In the second part of Lecture 4, we will talk about domain adaptation and extrapolation.

12 Uniform Convergence Fails to Explain Deep Learning

The presentation of this Section follows the work of Nagarajan and Kolter (2019) but is also inspired by https://locuslab.github.io/2019-07-09-uniform-convergence/.

Explaining why overparameterized deep networks generalize well has become an important open question in deep learning. To theoretically study the generalization behavior of a model, one must upper bound the gap between the population error (expected error under the underlying distribution) and the training error (under the assumption that the observations are i.i.d from the same populations). As we saw in Lectures 1 and 2, generalization bounds have the following form:

$$\text{Population Error} \leq \text{Training Error} + O\left(\frac{\text{Complexity Measure}}{\sqrt{m}}\right),$$

where m is the sample size. The way the complexity measure term is quantified in deep networks has evolved over time. We will now provide a quick overview.

Classics Standard generalization bounds (based e.g., on the VC dimension) consider a uniform convergence bound on the class of functions representable by a deep network to control the complexity of the concept class.

This approach is already known to fail in deep learning as we saw in the previous Lectures (e.g., (Zhang et al., 2021)). Recall that standard VC bounds fail because the obtained complexity measure for neural networks scales as width times depth (i.e., proportionally to the size of the network) and so the generalization bound is:

$$\text{Population Error} \leq \text{Training Error} + O\left(\frac{\text{width} \times \text{depth}}{\sqrt{m}}\right).$$

which is vacuous in the overparameterized setting (where the number of parameters is much larger than the sample size). As a remark, the above bound does not say much about the Rademacher complexity of the models (for nice data distributions, it could be much smaller than the above pessimistic bound).

Modern Approach. The issues based on the classical approach were reconsider in the work of Neyshabur et al. (2015), which sparked several new ideas and work in the deep learning theory community.

The key intuition is the one we already discussed in Lecture 1: *understand the inductive bias of SGD in deep learning*. The main question raised was to identify how does the underlying data distribution and the training algorithm as a pair restrict the "complexity" of the search (i.e., representation) space of the deep networks to a "simple" enough class of functions? Given this simple class, we could obtain generalization bounds based on its complexity, which will be significantly smaller and less pessimistic.

For instance, the generalization error will look like this:

$$\text{Population Error} \leq \text{Training Error} + O\left(\frac{\text{Complexity of Capacity Controlled NN}}{\sqrt{m}}\right).$$

Roughly speaking, these capacity controlled neural networks will correspond for instance to the *norms of the network weights* controlled by SGD:

$$\text{PopulationError} \leq \text{TrainingError} + O\left(\frac{\text{Weight Norms Controlled by SGD for Given Distribution}}{\sqrt{m}}\right).$$

This inspired several lines of work using various tools such as

1. Rademacher Complexity (Neyshabur et al., 2015; Golowich et al., 2018)
2. Covering Numbers (Bartlett et al., 2017)
3. PAC-Bayes (Neyshabur et al., 2017; Dziugaite and Roy, 2017)
4. Compression (Arora et al., 2018)

Recall that the empirical Rademacher complexity is defined as

$$\widehat{R}_m(\mathcal{H}, \{x_1, ..., x_m\}) = \mathop{\mathbf{E}}_{\sigma \sim \mathcal{U}(\{\pm 1\}^m)} \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i h(x_i).$$

The standard modern approach (Neyshabur et al., 2015; Golowich et al., 2018) is to give bounds on the Rademacher complexity (sometimes independent of $x_1, ..., x_m$ as long as they are assumed to have norm at most B), with respect to classes of neural networks with various norm constraints. Using standard arguments, an upper bound on the Rademacher complexity can be converted to bounds on the generalization error, assuming access to a sample of m i.i.d. training examples.

For instance, let us consider standard depth-d neural networks, of the form

$$x \mapsto W_d \sigma_{d-1}(W_{d-1}\sigma_{d-2}(...W_2\sigma_1(W_1 x))).$$

Theorem 14: Neyshabur et al. (2015)

If the parameter matrices $W_1, ..., W_d$ for each one of the d layers have Frobenious norm bounds $M_F(d), ..., M_F(d)$, $\|x\| \leq B$ and under suitable assumptions on the activation functions, the generalization error scales (with high probability) as

$$\frac{B \cdot 2^d \cdot \prod_i M_F(i)}{\sqrt{m}}.$$

Theorem 15: Golowich et al. (2018)

If the parameter matrices $W_1, ..., W_d$ for each one of the d layers have Frobenious norm bounds $M_F(d), ..., M_F(d)$, $\|x\| \leq B$ and under suitable assumptions on the activation functions, the generalization error scales (with high probability) as

$$\frac{B \cdot \sqrt{d} \cdot \prod_i M_F(i)}{\sqrt{m}}.$$

However, behind all these results, the main tool needed is still uniform convergence. A lot of ongoing efforts in this space has been focused on developing more novel or refined variations of such uniform convergence-based generalization bounds to better capture generalization in deep learning. However, the goal of providing a complete explanation of generalization through these bounds appears to be elusive.

Observation The first observation of Nagarajan and Kolter (2019) is that the norm-based bounds are usually vacuous because as the training size increases, the weight norms also increase.

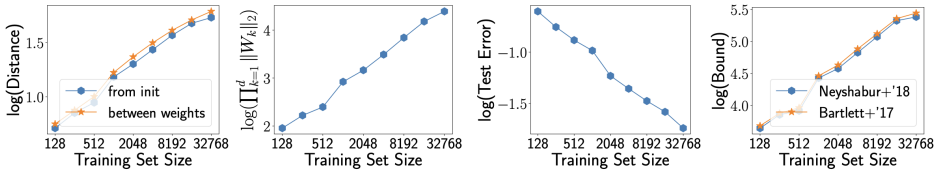


Figure 1: **Experiments in Section 2:** In the **first** figure, we plot (i) ℓ_2 the distance of the network from the initialization and (ii) the ℓ_2 distance between the weights learned on two random draws of training data starting from the same initialization. In the **second** figure we plot the product of spectral norms of the weights matrices. In the **third** figure, we plot the test error. In the **fourth** figure, we plot the bounds from [32, 3]. Note that we have presented log-log plots and the exponent of m can be recovered from the slope of these plots.

Figure 4: Weight Norms grow with the sample size. **Taken from Nagarajan and Kolter (2019).**

Uniform Convergence For a given $\delta \in (0,1)$, the generalization error of the algorithm is essentially a bound on the difference between the error of the hypothesis h_S learned on a training set S and the expected error over D, that holds with high probability of at least $1-\delta$ over the draws of S. More formally:

Definition 17: Generalization Error

The generalization error of A with respect to L is the smallest $\epsilon_{gen}(n,\delta)$ such that

$$\Pr_{S\sim\mathcal{D}^n}[L_{\mathcal{D}}(A(S)) - L_S(A(S)) \leq \epsilon_{gen}(n,\delta)] \geq 1-\delta$$

As we saw in Lecture 1, the standard way to control the generalization error is uniform convergence:

Definition 18: Uniform Convergence

The uniform convergence bound with respect to loss L is the smallest value $\epsilon_{unif}(n,\delta)$ such that

$$\Pr_{S\sim\mathcal{D}^n}[\sup_{h\in\mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) \leq \epsilon_{unif}(n,\delta)] \geq 1-\delta$$

Equivalently, we can say that $\epsilon_{unif}(n,\delta)$ is the smallest value for which there exists a set of sample sets $S_\delta \subseteq (\mathcal{X}\times\{0,1\})^n$ with $\mathbf{Pr}_S(S\in S_\delta) \geq 1-\delta$ and $\sup_{S\in S_\delta} \sup_{h\in\mathcal{H}} |L_{\mathcal{D}}(h) - L_S(h)| \leq \epsilon_{unif}(n,\delta)$.

Given the discussions in Lectures 2 and 3, one might suspect that pruning the search space and keeping only the concepts explored by the training algorithm, would imply generalization bounds (via uniform convergence in the pruned space). The main goal of Nagarajan and Kolter (2019) is to show that this intuition is also false.

Tightest algorithm-dependent uniform convergence. The bound given by ϵ_{unif} can be tightened by ignoring many extraneous hypotheses in $\mathcal{H}$ never picked/explored by the algorithm A for a given distribution $\mathcal{D}$. This is typically done by focusing on a norm-bounded class of hypotheses that the algorithm A implicitly restricts itself to (recall the bounds we saw before).

The main idea of Nagarajan and Kolter (2019) is to take this intuition to the extreme and apply uniform convergence on the *smallest possible class* of concepts, namely, only those hypotheses that are picked by A under D, excluding everything else. The reason is that pruning the hypothesis class any further would not imply a bound on the generalization error, and hence applying uniform convergence on this extremely

pruned hypothesis class would yield the tightest possible uniform convergence bound.

Definition 19: Algorithm-Dependent Uniform Convergence

The tightest algorithm-dependent uniform convergence bound with respect to loss L is the smallest value $\epsilon_{unif,alg}(n,\delta)$ for which there exists a set of sample sets S_δ such that $\mathbf{Pr}_{S\sim\mathcal{D}^n}[S \in S_\delta] \geq 1-\delta$ and and if we define the space of hypotheses explored by A as $\mathcal{H}_\delta = \cup_{S\in S_\delta} A(S) \subseteq \mathcal{H}$ then $\sup_{S\in S_\delta} \sup_{h\in\mathcal{H}_\delta} |L_\mathcal{D}(h) - L_S(h)| \leq \epsilon_{unif,alg}(n,\delta)$.

Through examples of overparameterized models trained by GD (or SGD), Nagarajan and Kolter (2019) argue how even the above tightest algorithm-dependent uniform convergence can fail to explain generalization. i.e., in these settings, even though ϵ_{gen} is smaller than a negligible value ϵ, they show that $\epsilon_{unif,alg}$ is large (close to $1-\epsilon$).

Sketch of the Idea. Consider a scenario where the algorithm generalizes well i.e., for every training set S, $h_S = A(S)$ has zero error on S and has small test error. This means that the generalization error is small.

While this means that h_S has small error on random draws of a test set, it may still be possible that for every such h_S, there exists a corresponding "bad" dataset S' – that is not random, but rather dependent on S – on which h_S has a large empirical error (say 1).

Unfortunately, uniform convergence runs into trouble while dealing with such bad datasets. Specifically, as we can see from the above definition, uniform convergence demands that $L_\mathcal{D}(h_S) - L_S(h_S)$ be small on all datasets in S_δ , which excludes a δ fraction of the datasets.

While it may be tempting to think that we can somehow exclude the bad dataset as part of the δ-fraction, there is a significant catch here: we *can not* carve out a δ fraction specific to each hypothesis; we *can ignore only a single chunk* of δ-mass common to all hypotheses in $\mathcal{H}_\delta$.

This restriction turns out to be a tremendous bottleneck: despite ignoring this δ fraction, for most $h \in \mathcal{H}_\delta$, the corresponding bad set S' would still be left in S_δ.

Then, for all such h_S in $\mathcal{H}_\delta$, $L_\mathcal{D}(h_S)$ will be small but in the definition of uniform convergence we can put $S' \in S_\delta$: this will make $L_{S'}(h_S)$ large when we take the supremum over the sets S_δ: this means that $\epsilon_{unif,alg}$ is indeed vacuous.

The above argument is formalized as follows.

Theorem 16: Nagarajan and Kolter (2019)

Consider an algorithm that has zero training error and a test error of ϵ (on almost all draws of the training set). Assume the algorithm is such that, for nearly all draws of S, one can construct another dataset S' of size n (which can be depend on S) such that:

- h_S misclassifies S' completely.
- The law of S' satisfies $S' \sim \mathcal{D}^n$ (note that S' is a random variable that is a function of S, and here we want the distribution of S', with S marginalized out, to be equal to $\mathcal{D}^n$).

If the algorithm satisfies these requirements, then: $\epsilon_{unif,alg} \geq 1-\epsilon$.

Exercise 6

Understand the proofs in the paper.

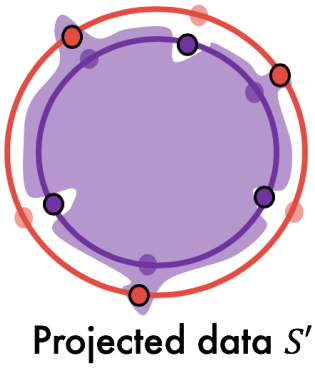


Figure 5: We design the bad dataset S' merely by projecting every point in S to its opposite hypersphere and flipping its label. That is, every point on the inner hypersphere is projected onto the outer hypersphere and vice versa, and then the labels are fixed accordingly. **Taken from**: https://locuslab.github.io/2019-07-09-uniform-convergence/

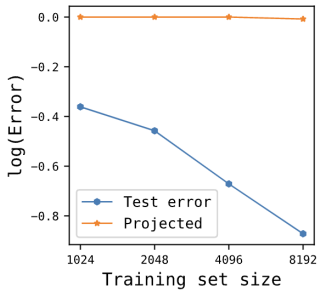


Figure 6: While the test loss goes to 0 with the number of samples, the loss on S' is not improving indicating that UC fails. **Taken from** https://locuslab.github.io/2019-07-09-uniform-convergence/.

Insight 8

The decision boundary learned by SGD on overparameterized deep networks can have certain complexities that hurt uniform convergence without hurting generalization.
The conjecture of Nagarajan and Kolter (2019) was the following:

1. the decision boundary is macroscopically simple, leading to good generalization (the trained classifier will not encounter the "hard" designed dataset).
2. The complex skews in the decision boundary are only microscopic (i.e., they cover only low probability regions in the input space). Hence, they will not affect generalization error. However, uniform convergence, which is by its nature extremely worst-case (even in the algorithm-dependent version) cannot ignore these skews since it takes a supremum over pairs in $(S_\delta, \mathcal{H}_\delta)$.

13 Domain Adaptation

Having extensively discussed about generalization, we have ignored an important aspect of it: Perhaps the most concerning issue regarding generalization in Machine Learning is the assumption that the training distribution is exactly the same as the testing distribution.

The ultimate of Machine Learning is to shed light on understanding how algorithms can be trained on data from the source domain and then adeptly applied to the target domain. The most interesting examples correspond to cases where the target distribution of data, denoted as Q, differs from that of the source domain, characterized by distribution P.

13.1 Domain Adaptation in Regression

In the second part of Lecture 4, we will study domain adaptation through the lens of regression. Here, our goal is to identify a *regressor* f that aims to minimize the mean squared error $\mathbf{E}_{x\sim P}[(f(x)-f^\star(x))^2]$, using samples in the form (x,y), where x is drawn from distribution P and $\mathbf{E}[y|x]=f^\star(x)$. The twist in transfer learning emerges when the objective shifts towards minimizing the expected error $\mathbf{E}_{x\sim Q}[(f(x)-f^\star(x))^2]$, despite the fact that our available samples are drawn from P. This raises the pivotal question: is it possible to establish a relationship between the expected errors $\mathbf{E}_{x\sim P}[(f(x)-f^\star(x))^2]$ and $\mathbf{E}_{x\sim Q}[(f(x)-f^\star(x))^2]$? In essence, this inquiry delves into the capability of the regressor f to *extrapolate* effectively to samples that are out-of-distribution, originating from the target distribution Q, based on its training with the source distribution P.[6]

A natural approach to compare the mean squared error with respect to P with the mean squared error with respect to Q is to apply a change of measure:

$$\begin{aligned}\mathop{\mathbf{E}}_{x\sim Q}[(f(x)-f^\star(x))^2] &= \mathop{\mathbf{E}}_{x\sim P}\left[\frac{Q(x)}{P(x)}(f(x)-f^\star(x))^2\right]\\ &\le \left\|\frac{dQ}{dP}\right\|_\infty \mathop{\mathbf{E}}_{x\sim P}[(f(x)-f^\star(x))^2]\,, \end{aligned} \tag{3}$$

where $\|dQ/dP\|_\infty = \sup_{x\in\mathrm{supp}(Q)} Q(x)/P(x)$.

Due to the many instances where the ratio dQ/dP is unbounded (Kalavasis et al., 2024c), a pertinent question arises:

Is transfer learning possible when $\|dQ/dP\|_\infty \to \infty$*?*

We consider the expressive and well-studied class of *low-degree polynomials*. We will show general transfer inequalities, for functions in this class, under mild assumptions on the distribution pair P and Q, going well beyond the boundedness of the ratio dQ/dP. A distribution with density Q is log-concave if $-\log Q$ is convex.

Theorem 17: Kalavasis et al. (2024c)

For any pair of degree-d polynomials $f, f^\star : \mathbb{R}^n \to \mathbb{R}$, any log-concave distribution Q and any continuous distribution P, it holds that

$$\mathop{\mathbf{E}}_{Q}[(f(x)-f^\star(x))^2] \le C_d \cdot \left\|\frac{dP}{dQ}\right\|_\infty^{2d} \cdot \mathop{\mathbf{E}}_{P}[(f(x)-f^\star(x))^2]\,, \tag{4}$$

where $C_d > 0$ is a constant depending only on the degree d. Furthermore, even when Q is not log-concave, it holds that

$$\mathop{\mathbf{E}}_{Q}[(f(x)-f^\star(x))^2] \le C_d \cdot \inf_{\mu\in\mathcal{L}} \left\|\frac{dQ}{d\mu}\right\|_\infty \left\|\frac{dP}{d\mu}\right\|_\infty^{2d} \cdot \mathop{\mathbf{E}}_{P}[(f(x)-f^\star(x))^2]\,, \tag{5}$$

where the infimum is over the set $\mathcal{L}$ of log-concave distributions over $\mathbb{R}^n$ and P, Q are arbitrary[a]

[a] Observe that it is necessary for the inequality not being void to take the infimum over measures $\mu \in \mathcal{L}$ whose support contains $\mathrm{supp(P)} \cup \mathrm{supp(Q)}$.

[6] When talking about the Euclidean domain, we consider a probability space $(\Omega, \mathcal{F}, (P,Q))$ where $\Omega = \mathbb{R}^n$ and for clarity we will assume that both distributions P and Q are *continuous* distributions.

13.2 The Proof: Anti-Concentration Implies Extrapolation

For $\alpha \geq 1$, we define the divergence $D_\alpha(P \parallel Q) \triangleq \left(\mathbf{E}_{x\sim Q}\left[\left(\frac{P(x)}{Q(x)}\right)^\alpha\right]\right)^{1/\alpha}$; we denote both distributions and associated density functions by P, Q overloading the notation. For $\alpha = \infty : D_\alpha(P \parallel Q) = \|dP/dQ\|_\infty$.

Let us fix some $\mu \in \mathcal{L}$ whose support contains both that of Q and that of P. We will overload the notation and denote by P (resp. Q, μ) the density of the associated distribution. The first step of the proof is to upper bound the expectation of f under Q by that under μ. This is done by a simple change of measure and an application of Hölder's inequality with parameters α, β:

$$\mathop{\mathbf{E}}_{Q}|f| = \mathop{\mathbf{E}}_{x\sim\mu}\left[\frac{Q(x)}{\mu(x)}|f(x)|\right] \leq D_\alpha(Q \parallel \mu)\left(\mathop{\mathbf{E}}_{\mu}|f|^\beta\right)^{1/\beta}. \tag{6}$$

As a next step we need to relate expectations under the log-concave measure μ and under the general measure P, which is the non-trivial part of the argument. We will first lower bound the value of $\mathbf{E}_P|f|^\beta$. For some $\gamma > 0$ to be decided, we have that

$$\mathop{\mathbf{E}}_{P}|f|^\beta \geq \mathop{\mathbf{E}}_{x\sim P}\left[|f|^\beta \,\middle|\, |f(x)|^\beta \geq \gamma\right] \mathop{\mathbf{Pr}}_{x\sim P}\left[|f(x)|^\beta \geq \gamma\right] \geq \gamma \mathop{\mathbf{Pr}}_{x\sim P}\left[|f(x)|^\beta \geq \gamma\right].$$

Hence we get that

$$\mathop{\mathbf{E}}_{P}|f|^\beta \geq \gamma\left(1 - \mathop{\mathbf{Pr}}_{x\sim P}\left[|f(x)|^\beta \leq \gamma\right]\right). \tag{7}$$

In order to further lower bound the right-hand side of the above inequality, it suffices to obtain an upper bound on the probability mass of the event $\{x \in \mathbb{R}^n : |f(x)|^\beta \leq \gamma\}$ under P. We have that

$$\mathop{\mathbf{Pr}}_{x\sim P}\left[|f(x)| \leq \gamma^{1/\beta}\right] = \mathop{\mathbf{E}}_{x\sim\mu}\left[\frac{P(x)}{\mu(x)}\,1\{|f(x)| \leq \gamma^{1/\beta}\}\right].$$

We can upper bound this quantity using Hölder's inequality for $\alpha, \beta \in [1, \infty]$ with $1/\alpha + 1/\beta = 1$:

$$\mathop{\mathbf{Pr}}_{x\sim P}\left[|f(x)|^\beta \leq \gamma\right] \leq D_\alpha(P \parallel \mu)\left(\mathop{\mathbf{Pr}}_{x\sim\mu}\left[|f(x)| \leq \gamma^{1/\beta}\right]\right)^{1/\beta} \tag{8}$$

We now use the following result.

Theorem 18: Carbery and Wright (2001)

There exists an absolute constant C such that the following holds: let $f : \mathbb{R}^n \to \mathbb{R}$ be a non-zero polynomial of degree at most d, μ be a log-concave probability distribution over $\mathbb{R}^n$ and $q, \gamma \in (0, \infty)$. Then it holds that

$$\mathop{\mathbf{Pr}}_{x\sim\mu}\left[|f(x)| \leq \gamma\right] \leq \frac{Cq\gamma^{1/d}}{\left(\mathbf{E}_\mu|f|^{q/d}\right)^{1/q}}.$$

In particular, for $q = d$, it holds that $\mathbf{Pr}_{x\sim\mu}\left[|f(x)| \leq \gamma\right] \leq \frac{Cd\gamma^{1/d}}{\left(\mathbf{E}_\mu|f|\right)^{1/d}}$.

Applying the Carbery-Wright inequality in (8), by picking $q = \beta d$, we get that

$$\mathop{\mathbf{Pr}}_{x\sim P}\left[|f(x)| \leq \gamma^{1/\beta}\right] \leq D_\alpha(P \parallel \mu)\left(\frac{C\beta d\gamma^{1/(\beta d)}}{\left(\mathbf{E}_\mu|f|^\beta\right)^{1/\beta d}}\right)^{1/\beta} \tag{9}$$

We are now ready to pick γ. We choose γ so that $D_\alpha(P \parallel \mu)\left(\frac{C\beta d\gamma^{1/(\beta d)}}{\left(\mathbf{E}_\mu |f|^\beta\right)^{1/(\beta d)}}\right)^{1/\beta} = 1/2$, which implies that $\gamma = \frac{\mathbf{E}_\mu |f|^\beta}{\left(C\beta d 2^\beta D_\alpha(P\|\mu)^\beta\right)^{\beta d}}$. Using this choice we can return to (7) and get that

$$\mathop{\mathbf{E}}_{P}|f|^\beta \geq \frac{\mathbf{E}_\mu |f|^\beta}{2\left(Cd2^\beta D_\alpha(P \parallel \mu)^\beta\right)^{\beta d}}$$

We can now complete the proof using (6), which gives that

$$\mathop{\mathbf{E}}_{Q}|f| \leq D_\alpha(Q \parallel \mu)\left(\mathop{\mathbf{E}}_{\mu}|f|^\beta\right)^{1/\beta} \leq 2C^d d^d 2^{d\beta} D_\alpha(Q \parallel \mu) D_\alpha(P \parallel \mu)^{\beta d}\left(\mathop{\mathbf{E}}_{P}|f|^\beta\right)^{1/\beta}.$$

By minimizing over $\mu \in \mathcal{L}$, the proof is complete. The statement of Theorem 4 follows by setting $\alpha = \infty$.

13.3 Extrapolation in Classification

This is a research direction which is still wide open. Note that the above result about regression can be directly used for classification (why?).

Regarding binary classification, even identification is highly non-obvious, which we will shortly discuss.

Definition 20: Extrapolation

Fix distribution family $\mathcal{P}$ and survival set S. We will say that a concept $h^\star \in \mathcal{H} \subseteq \{0,1\}^{\mathbb{R}^d}$ can be extrapolated with respect to $(\mathcal{P}, S)$ if for any $Q \in \mathcal{P}$ and for any $h \in \mathcal{H}$, it holds that

$$h^\star \neq h \Rightarrow Q_{h^\star,S} \neq Q_{h,S}\,.$$

Assume that

1. It holds that $\partial h^\star \cap S \neq \emptyset$ for each connected component of $h^\star$.
2. It holds that $Q(S) > 0$.

We note that without capturing every connected decision boundary by the survival set, it is information theoretically impossible to extrapolate. We denote by $\mathrm{PTF}_{k,d}$ the class of polynomial-threshold functions of degree k in dimension d.

Theorem 19: Extrapolation of 2D PTFs

Any $h^\star$ in $\mathrm{PTF}_{k,2}$ can be extrapolated from any marginal distribution Q and survival set S that satisfy Conditions 1 and 2.

Exercise 7

Prove this result.

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.

14 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 21: 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.

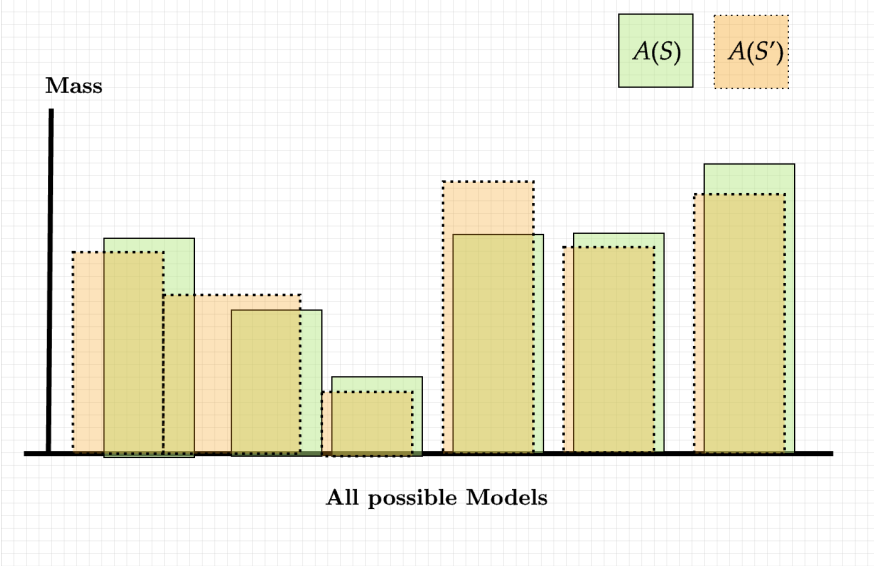


Figure 7: Illustration of how the p.m.f. changes when training the algorithm with S' instead of S.

Definition 22: 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).

15 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 23: 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 $\mathcal{D}$.

16 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 20

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 $\text{err}_S(h) = -q(S,h)/n$. For a fixed hypothesis, this random variable concentrates:

$$\mathop{\mathbf{Pr}}_S[|\text{err}(h) - \text{err}_S(h)| > t] \leq 2\exp\Big(-2t^2 n\Big)$$

Hence the probability that there exists a hypothesis in the finite class $\mathcal{H}_d$ that violates the inequality $|\text{err}(h) - \text{err}_S(h)| \leq t$ 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

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

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

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

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

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

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 $\text{OPT} + t$ 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 9**
>
> Pure DP is characterized by the probabilistic representation dimension (Beimel et al., 2013a), which roughly speaking allows one to reduce the problem to an analysis similar to the above: discretize the class and run the exponential mechanism.

17 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.

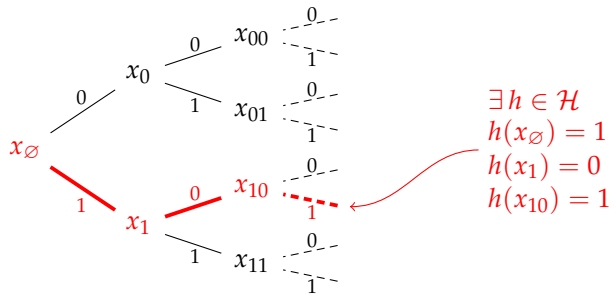


Figure 8: A Littlestone tree of depth 3. Every branch is consistent with a concept $h \in \mathcal{H}$. This is illustrated here for one of the branches. **Taken from Bousquet et al. (2021).**

Definition 24: 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 25: 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 21: 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 26: 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 6

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).

Lecture 6: Private PAC Learning Implies Online Learning

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

Acknowledgement Thanks to Argyris Mouzakis for improving the first version of this draft.

18 Which concept classes can we learn privately?

In Lecture 5, we focused on pure differential privacy. Here we relax the notion of privacy.

Definition 27: 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.

We are interested in understanding which concept classes are PAC learnable with (ϵ, δ)-DP constraints.[7] We will show a surprising result, shown by Alon, Livni, Malliaris and Moran in Alon et al. (2019):

(ϵ, δ)-Private PAC learning implies finite Littlestone dimension.

19 Learning Thresholds with Privacy Constraints

Learning thresholds is one of the most basic problems in machine learning. This problem consists of an unknown threshold function $c : \mathbb{R} \to \{-, +\}$, an unknown distribution $\mathcal{D}$ over $\mathbb{R}$, and the goal is to output a hypothesis $h : \mathbb{R} \to \{-, +\}$ that is close to c (in terms of classification loss), given access to a labeled examples $(x_1, c(x_1)), ..., (x_m, c(x_m))$, where the x_i's are drawn independently from $\mathcal{D}$.

Standard PAC learning of thresholds without privacy constraints is known to be easy and can be done using a constant number of examples (since the VC dimension is 1). In contrast, whether thresholds can be learned privately turned out to be more challenging:

- Based on Lecture 5, the result of Kasiviswanathan et al. (2011) implies a pure differentially private algorithm that learns thresholds over a finite $X \subseteq \mathbb{R}$ of size n with $O(\log n)$ examples. Feldman and Xiao (2014) showed a matching lower bound for any pure differentially private algorithm. Beimel et al. (2013b) showed that by relaxing the privacy constraint to approximate differential privacy, one can significantly improve the upper bound to $2^{O(\log^* n)}$ samples. Bun et al. (2015) gave a lower bound of $\Omega(\log^* n)$ that applies for any proper learning algorithm.

[7] Using standard boosting arguments, we can focus on the regime $\epsilon = 0.1$ and $\delta = o(1/m)$ for sample size m.

Exercise 8

Read the algorithm of Beimel et al. (2013b) for singletons and thresholds based on quasi-concave promise problems.

20 Some Deep Connections: Thresholds and Littlestone Classes

It turns out that there is an close connection between thresholds and the Littlestone dimension, which goes back to Shelah (1972):

A class $\mathcal{H}$ has a finite Littlestone dimension if and only if it does not embed thresholds as a subclass.

More formally, Shelah's result (which is part of the so-called Unstable Formula Theorem) (see (Malliaris and Moran, 2022) for a more modern reference) provides a simple and elegant connection between Littlestone dimension and th class of thresholds:

Theorem 22: Littlestone Dimension and Thresholds (Shelah, 1972)

Let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a hypothesis class. The following hold true:

- If $\mathrm{Ldim}(\mathcal{H}) \geq d$, then $\mathcal{H}$ contains $\lfloor \log d \rfloor$ thresholds.
- If $\mathcal{H}$ contains d thresholds then $\mathrm{Ldim}(\mathcal{H}) \geq \lfloor \log d \rfloor$.

The second part of the statement is relatively easy. If $\mathcal{H}$ contains 2^t thresholds then there are $h_i \in \mathcal{H}$ for $0 \leq i < 2^t$ and there are x_j for $0 \leq j < 2^t - 1$ such that $h_i(x_j) = 0$ for $j < i$ and $h_i(x_j) = 1$ for $j \geq i$. Now we can define a labeled binary tree of height t corresponding to the binary search process that will witness the Littlestone dimension bound. For the first part, we refer to Alon et al. (2019) for a combinatorial proof.

In this Lecture, we will discuss two theorems. We focus on the standard DP regime where $\epsilon = 0.1$ and $\delta = o(1/m)$, where m is the number of samples.[8]

Theorem 23: Thresholds are not privately learnable (Alon et al., 2019)

Let $\mathcal{X} \subseteq \mathbb{R}$ with $|\mathcal{X}| = n$ and let $\mathcal{A}$ be an $(1/16, 1/16)$-accurate learning algorithm for the class of thresholds over $\mathcal{X}$ with sample complexity m which satisfies (ϵ, δ)-differential privacy with $\epsilon = 0.1$ and $\delta = O(\frac{1}{m^2 \log m})$. Then

$$m \geq \Omega(\log^* n)\,.$$

In particular, the class of thresholds over an infinite $\mathcal{X}$ cannot be learned privately.

The above is a strong negative result showing that thresholds over infinite domains are not privately PAC learnable (even in constant accuracy).

Definition 28: Iterated Logarithm

We define $\log^{(0)} x = \log x$ and $\log^{(m)} x = 1 + \log^{(m-1)} \log x$. The function $\log^* x$ equals the number of times the iterated logarithm must be applied before the result is less than or equal to 1. We define

$$\log^* x = (1 + \log^* \log x)\mathbf{1}\{x > 1\}\,.$$

[8] The studied privacy regime is the standard one appearing in practice. A natural interpretation of the negligible $\delta > 0$ parameter is that there is a very small probability of a very bad event (in which even all the input data is leaked) but otherwise the algorithm satisfies pure differential privacy.

The function $\log^* x$ equals the number of times the iterated logarithm must be applied before the result is less than or equal to 1.

Combining the above result with the above deep connection between thresholds and Littlestone dimension, we get the following:

Theorem 24: Private Learning implies finite Littlestone dimension (Alon et al., 2019)

Let $\mathcal{H}$ be a hypothesis class with Littlestone dimension $d \in \mathbb{N} \cup \{\infty\}$. Let $\mathcal{A}$ be an $(1/16, 1/16)$-accurate learning algorithm for $\mathcal{H}$ with sample complexity m which satisfies (ϵ, δ)-differential privacy with $\epsilon = 0.1$ and $\delta = O(\frac{1}{m^2 \log m})$. Then

$$m \geq \Omega(\log^* d)\,.$$

In particular, any class that is privately learnable has a finite Littlestone dimension.

To see this, $\mathcal{H}$ contains $c = \lfloor \log d \rfloor$ thresholds, i.e., there exist $x_1, ..., x_c$ and $h_1, ..., h_c \in \mathcal{H}$ such that $h_i(x_j) = 1$ for all $j \geq i$. But then, Theorem 2 implies a lower bound of $m \geq \Omega(\log^* c) = \Omega(\log^* d)$.

21 Ramsey Theory and Sketch of the Reduction

It suffices to prove Theorem 2. We begin by considering an arbitrary differentially private algorithm A that learns the class of thresholds over an ordered domain $\mathcal{X}$ of size n. Our goal is to show a lower bound of order $\Omega(\log^* n)$ on the sample complexity of A.

Before the work of Alon et al. (2019), we already knew of lower bounds for proper algorithms. The central challenge in the proof is the potential *improper* nature of the learner A, i.e., A may output arbitrary hypotheses (which is in contrast with proving impossibility results for proper algorithms where the structure of the learned class can be exploited).

Some rough ideas about the proof are as follows:

1. Construct a collection of hard datasets (which are algorithm-dependent)
2. These datasets will exploit some structure of the algorithm, making use of the fact that it learns thresholds.

The core property that the hard datasets will have is the following: they will come from a critical subset of the domain on which the learner behaves *in a highly regular way*. How do we find some a regular structure?

Insight 10: Ramsey Theory

Ramsey theory is a branch of the mathematical field of combinatorics that focuses on the appearance of **order in a substructure given a structure of a known size**. We use Ramsey theory to deal with the fact that the algorithm is improper by finding some structure in its behavior.

Ramsey theory handles the above challenge by showing that for any algorithm (in fact, for any mapping that takes input samples to output hypotheses) there is a large subset of the domain that is *homogeneous with respect to the algorithm.* This notion of homogeneity places useful restrictions on the algorithm when restricting it to the homogeneous set.

We will need the following definition.

Definition 29: Increasing and Balanced Dataset

A sample $S = (x_1, y_1), ..., (x_m, y_m)$ of an even size is realizable (by thresholds), balanced, and increasing if and only if $x_1 < x_2 < ... < x_m$ and the first half of y_i's is -1 and the second one $+1$.

Reduction to homogeneous set. As discussed above, the first step in the proof is about identifying a large homogeneous subset of the input domain $\mathcal{X}$ on which we can control the output of A:

Definition 30: Homogeneous Set

A subset $\mathcal{X}' \subseteq \mathcal{X}$ is called m-homogeneous with respect to A if there is a list of numbers $p_0, p_1, ..., p_m$ such that for every increasing balanced sample $S \in (\mathcal{X}' \times \{-1, 1\})^m$ of points from $\mathcal{X}'$ and every $x' \in \mathcal{X}'$ with $\text{ord}_S(x') = i$:

$$|A_S(x') - p_i| = O(1/m)\,.$$

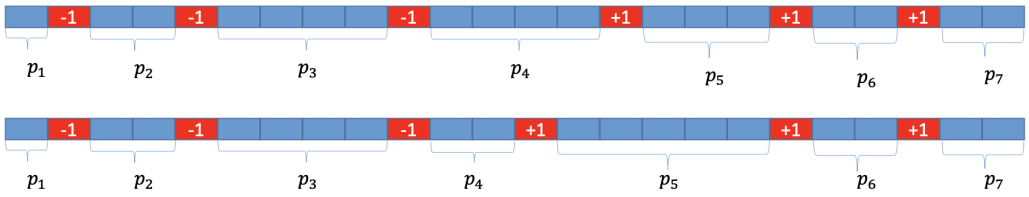


Figure 1: Depiction of two possible outputs of an algorithm over an homogeneous set, given two input samples from the set (marked in red). The number p_i denote, for a given point x, the probability that $h(x) = 1$, where $h \sim A(S)$ is the hypothesis h outputted by the algorithm on input sample S. These probabilities depends (up to a small additive error) only on the interval that x belongs to. In the figure above we changed in the input the fourth example – this only affects the interval and not the values of the p_i's (again, up to a small additive error).

Figure 9: Taken from Alon et al. (2019).

The intuition is the following: the probability for any x' in the homogeneous subset $\mathcal{X}' \setminus S$ to receive the label 1 by the algorithm A trained on S depends only on the relative ordering with respect to training examples S.

Existence of Large Homogeneous Sets. Do homogeneous sets exist? This is where Ramsey theory comes in.

Lemma 7: Every algorithm has a large homogeneous set

Let A be a (possibly randomized) algorithm that is defined over input samples of size m over a domain $\mathcal{X} \subseteq \mathbb{R}$ of size $|\mathcal{X}| = n$. Then, there is a set $\mathcal{X}' \subseteq \mathcal{X}$ that is m-homogeneous with respect to A of size

$$|\mathcal{X}'| \geq \frac{\log^{(m)} n}{2^{m \log m}}\,.$$

The intuition is the following: Consider the complete m-uniform hypergraph with n vertices. If n is appropriately large and we color the hyperedges with k colors, there must exist a large monochromatic hyperclique. Roughly speaking, in our case:

1. vertices are domain points.
2. m-uniform hyperedges are all datasets that are increasing and balanced.
3. colors are lists of probabilities.
4. monochromatic hyperclique is a homogeneous set.

Now Ramsey's Theorem implies that for any algorithm A there is a large subset $\mathcal{X}' \subseteq \mathcal{X}$ homogeneous with respect to A.

More typically, define a coloring on the $(m+1)$-subsets of $\mathcal{X}$ as follows. Let $D = \{x_1 < x_2 < ... < x_{m+1}\}$ be such a subset. For each $i \leq m+1$, define $D^{-i} = D \setminus \{x_i\}$ and let S^{-i} denote the balanced increasing sample on D^{-i}. Set p_i to be the element of $\{t/10m^2\}_t$ that is closest to $A_{S^{-i}}(x_i)$ ($10m^2+1$ choices). The coloring assigned is then the list $(p_1, p_2, ..., p_{m+1})$.

Theorem 25: Quantitative Version of Ramsey Theorem

Let $s > t \geq 2$ and q be integers and let

$$N \geq \text{twr}_t(3sq \log q)\,.$$

Then for every coloring of the subsets of size t of a universe of size N using q colors there is a homogeneous subset of size s (a subset of the universe is homogeneous if all of its t-subsets have the same color.).

Recall that $\text{twr}_1(x) = x$ and $\text{twr}_i(x) = 2^{\text{twr}_{i-1}(x)}$.

In the above, we have $q = (10m^2+1)^{m+1}$ colors, $t = m+1$ and $N = n$. Solving for s in the above, we can show that there is a set $\mathcal{X}' \subseteq X$ such that $|\mathcal{X}'| \geq \frac{\log^{(m)} n}{2^{m \log m}}$ such that all $m+1$-subsets have the same color.

Implications of Homogeneous Sets. We can now use the structure of these sets for the next result.

Theorem 26: Large homogeneous sets imply lower bounds for private learning

Let A be an $O(0.1, \delta)$-differentially private algorithm with sample complexity m and $\delta \leq \frac{1}{1000m^2 \log m}$.
Let $\mathcal{X} = \{1, ..., k\}$ be m-homogeneous with respect to A.
Then if A *empirically* learns thresholds over $\mathcal{X}$ with $(1/16, 1/16)$-accuracy, then

$$m \geq \frac{\sqrt{\log k}}{\log \log k}\,.$$

Using Lemma 5.9 in Bun et al. (2015), we can transfer the above sample complexity lower bound from empirical learners to population ones (with the same sample complexity lower bound).

Combining the above with Lemma 1, we get that $m \geq \Omega(\log^* n)$. Let us see why:

1. Lemma 1 implies that $k \geq \log^{(m)} n / 2^{O(m \log m)}$
2. Theorem 5 implies that $k = 2^{O(m^2 \log^2 m)}$.

Combining the two bounds, we get that

$$\log^{(m)} n \leq 2^{c \cdot m^2 \log m}\,.$$

Apply the iterated logarithm t times where $t = \log^*(2^{O(m^2 \log^2 m)}) = \log^*(m) + O(1)$ to the above inequality and get that

$$\log^{(m+t)} n = \log^{(m+\log^* m + O(1))} n \leq 1 .$$

This means that $\log^* n \leq m + \log^* m + O(1)$. This implies that $m \geq \log^* n$ and gives the desired lower bound on the sample complexity.

To show Theorem 5, we need to obtain a large class of hard datasets. Roughly speaking, if A is an accurate and private empirical learner for thresholds over a homogeneous set, its list of probabilities $p_1, ..., p_{m+1}$ should have a gap, i.e., there is some i such that $|p_i - p_{i-1}| \geq 1/m$.

We will use this property to construct a hard family of distributions. Let's give some intuition: Let this index be i^* and construct hard datasets as follows: Pick an increasing realizable sample $S \in (\mathcal{X}' \times \{0,1\})^m$ of size m so that the interval $J \subseteq \mathcal{X}'$ between x_{i^*-1} and x_{i^*+1} is of size $k - m$. For every $x \in J$, let S_x be the neighboring sample of S that is obtained by replacing x_i with x.

Lemma 8: Hard Family (Informal)

Let m, k as in Theorem 5. Let $A, \mathcal{X}'$ be as above. Let $n = m - k$. There exists a family $\{P_i : i \leq n\}$ over $\{-1,1\}^n$ such that (i) every P_i, P_j are $(0.1, \delta)$-indistinguishable and (ii) there exists r such that for all i : $\mathbf{Pr}_{v \sim P_i}[v(j) = 1] = (r - 1/m)1\{j < i\} + (r + 1/m)1\{j \geq i\}$.

Then one can show that $k - m = |\text{size of family}| \leq 2^{m^2 \log^2 m}$, which implies that $k = 2^{O(m^2 \log^2 m)}$.

Conclusion The main result of this Lecture is a lower bound on the sample complexity of private learning in terms of the Littlestone dimension.

In the next Lecture, we will see whether the opposite is true: can every class with finite Littlestone dimension be learned privately?

Lecture 7: Online Learning Implies Private PAC Learning

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

In the previous lecture, we proved that private PAC learning requires a finite Littlestone dimension. In Lecture 7, we will sketch the ideas for the breakthrough result of Bun, Livni and Moran (Bun et al., 2020), showing that finite Littlestone dimension is also sufficient for learning with differential privacy.

This establishes a qualitative equivalence between online learning and PAC learning with approximate differential privacy.

22 Differential Privacy, Online Learning, and Stability

At a first glance it may seem that online learning (which is characterized by the finiteness of the Littlestone dimension) and differentially private PAC learning have little to do with one another: e.g., in online learning there is no underlying distribution while in PAC learning, each learning task corresponds to some probability distribution over labeled examples.

However, a long line of works has revealed a tight connection between the two. For instance, in Lecture 6, we showed that online learnability is necessary for private PAC learning.

This connection was culminated in the work of Bun et al. (2020), where Bun, Livni, and Moran proved that every binary concept class with finite Littlestone dimension can be learned by an approximate differentially private algorithm. This result, together with the main result of the previous Lecture, demonstrate an equivalence between online and private classification.

At a high-level, this connection seems to boil down to the notion of *stability*, which plays a key role in both topics. On one hand, the definition of differential privacy is itself a form of stability; it requires that the distribution over outputs of the learning algorithm is "stable" with respect to small changes of the training set. On the other hand, stability also arises as a central motif in online learning paradigms, where an online learner eventually "stabilizes" to some hypothesis given feedback from the environment and stops making mistakes.

In their monograph, Dwork and Roth (Dwork et al., 2014) identified stability as a common factor of learning and differential privacy: "*Differential privacy is enabled by stability and ensures stability...we observe a tantalizing moral equivalence between learnability, differential privacy, and stability.*"

Essentially, the works of Alon et al. (2019) and Bun et al. (2020) formalize the above intuitive statement.

23 Privacy Reminder

Our main object of interest is approximate differential privacy which we recall below.

Definition 31: Approximate Differential Privacy

A randomized algorithm A is (ϵ, δ)-differentially private if for all neighboring datasets 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.

24 Online Learning Implies DP PAC Learning

The main result of Bun et al. (2015) is that Littlestone classes can be PAC learning with approximate differential privacy.

Theorem 27: Bun et al. (2020)

Let $\mathcal{H}$ be a class of Littlestone dimension d, let $\epsilon, \delta \in (0,1)$ be privacy parameters and let $\alpha, \beta \in (0, 1/2)$ be accuracy parameters. For

$$n = \frac{2^{\widetilde{O}(2^d)} + \log 1/\beta\delta}{\alpha\epsilon},$$

there exists an (ϵ, δ)-DP algorithm A such that for any realizable distribution $\mathcal{D}$, given as input $S \sim \mathcal{D}^n$ samples, the output hypothesis $f \sim A(S)$ satisfies $R_{\mathcal{D}}(f) \leq \alpha$ with probability $1 - \beta$, where the randomness is over S and A and $R_{\mathcal{D}}$ is the population error under $\mathcal{D}$.

Combining this result with the main Theorem of the previous lecture, one gets the following surprising equivalence.

Theorem 28: Alon et al. (2022)

For a class $\mathcal{H} \subseteq \{-1, 1\}^{\mathcal{X}}$, $\mathcal{H}$ is online learnable if and only if $\mathcal{H}$ is (ϵ, δ)-DP PAC learnable.

25 The Proof

The starting point of the proof is the following key definition.

Definition 32: Global Stability (Bun et al., 2020)

For $\eta > 0$ and $n \in \mathbb{N}$, a learning algorithm A is (n, η)-globally stable with respect to a distribution $\mathcal{D}$ over examples if there is a hypothesis h whose frequency as an output is at least η, i.e.,

$$\Pr_{S \sim \mathcal{D}^n}[h = A(S)] \geq \eta.$$

The word *global* highlights a difference with other forms of algorithmic stability. Indeed, previous forms of stability such as differential privacy (Dwork et al., 2014) and uniform stability (Bousquet and Elisseeff, 2002) are local in the sense that they require output robustness subject to local changes in the input. However, the property required by global stability captures stability with respect to resampling the entire input.

For instance, every learner using n examples which produces a hypothesis in a finite hypothesis class $\mathcal{H}$ is $(n, 1/|\mathcal{H}|)$-globally stable.

Having this definition at hand, it is a natural first step to ask: which concept classes can be learned by globally stable algorithms?

Proof Outline. The proof consists of two parts. First, we will see that every concept class with a finite Littlestone dimension can be learned by a globally stable algorithm. Next, we have to transform the globally stable learner to a DP one (using mostly standard tools from differential privacy).

25.1 Littlestone Classes can be Learned by Globally Stable Learners

Let $\mathcal{H}$ be a concept class with Littlestone dimension d. Global stability is quite strong of a notion. Our goal is to give an (n, η)-globally-stable learning algorithm for $\mathcal{H}$ with (very small) stability parameter $\eta = 2^{-2^{O(d)}}$ and (very large) sample complexity $n = 2^{2^{O(d)}}$.

Let $\mathcal{D}$ be the unknown realizable learning distribution. So, $\mathcal{D}$ is a distribution over examples $(x, c(x))$ where $c \in \mathcal{H}$ is an unknown target concept.

Since $d < \infty$, we know that $\mathcal{H}$ is online learnable in the realizable setting with at most d mistakes. Let A be the algorithm that online learns $\mathcal{H}$ with mistake bound d. Consider an execution of that online algorithm to the training set $(x_1, c(x_1)), ..., (x_n, c(x_n))$ drawn i.i.d. from $\mathcal{D}^n$.

This execution induces a random variable M, which corresponds to the number of mistakes that A makes. By definition of the algorithm (and online learning), we know that

$$0 \leq M \leq d.$$

Hence, we have that there exists $i \in \{0, 1, ..., d\}$ such that

$$\mathbf{Pr}[M = i] \geq \frac{1}{d+1}.$$

The first observation is that this index can be identified from samples: with high probability, an i with $\mathbf{Pr}[M = i] \geq 1/(2d)$ can be found by independently running A on $O(d)$ traces from $\mathcal{D}^n$.

This index i has $d + 1$ possible values. Let us first assume that $i = d$, i.e.,

$$\mathbf{Pr}[M = d] \geq \frac{1}{2d}.$$

This is a good scenario for our learning algorithm because, after making d mistakes, we have *identified* the true hypothesis c (since otherwise the adversary could force us make more mistakes). Hence, if $i = d$, then $A(S) = c$ with probability at least $1/(2d)$. In this case, the samples were very informative and we could identify the target hypothesis. What if $i < d$?

Let us assume that $i = d - 1$. To give some intuition, we plan to follow an inductive argument "over a Littlestone tree of depth d" and this is why we get double-exponential dependencies on the parameters.

The issue with $i = d - 1$ is that A can output many different hypotheses before making the d-th mistake. Hence, we need to find a way to make the algorithm stable: we should artificially force the learner make one more mistake.

Let us draw two samples $S_1, S_2 \sim \mathcal{D}^n$ independently and let $f_i = A(S_i)$. Consider the event that the number of mistakes made by A on each S_i is exactly $d - 1$. We know that this event occurs with probability $1/(2d)^2$. Let us condition on that event. There are two cases: either $\mathbf{Pr}[f_1 = f_2] < 1/4$ or $\geq 1/4$.

If the collision occurs with probability at least $1/4$, then this means that there exists an h such that

$$\mathbf{Pr}[A(S) = h] \geq \frac{1}{4} \cdot \frac{1}{(2d)^2}$$

and we are done. The difficult case is when the collision is rare. For that case, we have to run a "random contest" between S_1, S_2:

- Pick x such that $f_1(x) \neq f_2(x)$ and draw $y \sim \{-1, 1\}$ uniformly at random.
- If $f_1(x) \neq y$, then the output of the contest is $A(S_1 \circ (x, y))$ (since then A makes d mistakes in the online game)
- If $f_2(x) \neq y$, the output of the contest is $A(S_2 \circ (x, y))$.

Adding the ghost example (x, y) forces A to make d mistakes on $S_i \circ (x, y)$. Now if $y = c(x)$, then by the mistake bound of A, it will hold that $A(S_i \circ (x, y)) = c$. Crucially, since y was picked at random, it will hold that $\mathbf{Pr}_y[c(x) = y] = 1/2$. Hence,

$$\mathop{\mathbf{Pr}}_{S_1, S_2, y}[A(S_i \circ (x, y)) = c] \geq \frac{1}{(2d)^2} \cdot 3/4 \cdot 1/2 = \Omega(1/d^2)\,.$$

Now, we can use induction for the other cases. But now, we need to guess more labels, to enforce mistakes on the algorithm. As we guess more labels the success rate reduces, nevertheless we never need to make more then 2^d such guesses.

Summary. We can get global stability using the above inductive argument with stability parameter doubly exponentially small in the depth d.

Theorem 29: Bun et al. (2020)

Let $\mathcal{H}$ be a concept class with Littlestone dimension d. Let $n = \text{poly}(2^{2^d}/\alpha)$. There exists an algorithm A that uses n samples $S \sim \mathcal{D}^n$ from the realizable distribution $\mathcal{D}$ such that

$$\mathbf{Pr}[A(S) = f] \geq 1/2^{2^d}$$

for some hypothesis f and

$$R_{\mathcal{D}}(f) \leq \alpha.$$

We finally add the actual algorithm for completeness. For the details, we refer to Bun et al. (2020).

The Distributions $\tilde{\mathcal{D}}_k$ (a Monte-Carlo variant of $\mathcal{D}_k$)

1. Let n be the auxiliary sample size and N be an upper bound on the number of examples drawn from $\mathcal{D}$.
2. $\tilde{\mathcal{D}}_0$: output the empty sample $\emptyset$ with probability 1.
3. For $k > 0$, define $\tilde{\mathcal{D}}_k$ recursively by the following process:
   - (*) **Throughout the process, if more than N examples from $\mathcal{D}$ are drawn (including examples drawn in the recursive calls), then output "Fail".**
   - (i) Draw $S_0, S_1 \sim \tilde{\mathcal{D}}_{k-1}$ and $T_0, T_1 \sim \mathcal{D}^n$ independently.
   - (ii) Let $f_0 = \mathsf{SOA}(S_0 \circ T_0)$, $f_1 = \mathsf{SOA}(S_1 \circ T_1)$.
   - (iii) If $f_0 = f_1$ then go back to step (i).
   - (iv) Else, pick $x \in \{x : f_0(x) \neq f_1(x)\}$ and sample $y \sim \{\pm 1\}$ uniformly.
   - (v) If $f_0(x) \neq y$ then output $S_0 \circ T_0 \circ ((x, y))$ and else output $S_1 \circ T_1 \circ ((x, y))$.

Figure 10: Taken from Bun et al. (2020). Tournament Process: For $k = 1$, we run A in two different datasets T_0, T_1 until ot outputs two different hypotheses f_0, f_1. Then we force the algorithm to make a mistake and output the extended dataset.

25.2 From Global Stability to Differential Privacy

The above argument gives a way to design a globally stable algorithm for Littlestone classes. We now have to convert it into a differentially private one. This step is not quite difficult.

If A is a (η, n)-globally stable learner with respect to a distribution $\mathcal{D}$, we obtain a DP learner using roughly n/η samples from $\mathcal{D}$ as follows.

Algorithm G

1. Consider the distribution $\tilde{\mathcal{D}}_k$, where the auxiliary sample size is set to $n = \lceil \frac{2^{d+2}}{\alpha} \rceil$ and the sample complexity upper bound is set to $N = 2^{2^{d+2}+1} 4^{d+1} \cdot n$.
2. Draw $k \in \{0, 1, \ldots, d\}$ uniformly at random.
3. Output $h = \mathsf{SOA}(S \circ T)$, where $T \sim \mathcal{D}^n$ and $S \sim \tilde{\mathcal{D}}_k$.

Figure 11: Taken from Bun et al. (2020). Globally Stable Algorithm: Given this algorithm G, which randomly draws the index k of the number of mistakes, one can show that there exists a hypothesis f such that $\mathbf{Pr}[G(S) = f] > \frac{1}{d+1} 2^{-2^{d+2}}$ and $\text{loss}_{\mathcal{D}}(f) < \alpha$.

- First, run A on $k \approx 1/\eta$ independent batches, non-privately producing a list of k hypotheses.
- We next use the 'Stable Histograms' algorithm to this list. This algorithm privately publishes a short list of hypotheses that appear with frequency $\Omega(\eta)$. It privately finds the heavy elements of the list.
- Global stability of the learner A guarantees that with high probability, this list contains some hypothesis h with small population loss.
- Think of this small list as a finite class. We can then apply a generic differentially-private learner (based on the exponential mechanism) on a fresh set of examples to identify such an accurate hypothesis from this short list.

Theorem 30: Stable Histograms Algorithm

Let $\mathcal{X}$ be any data domain. For

$$n \geq O\left(\frac{\log(1/(\eta\beta\delta))}{\eta\epsilon}\right)$$

there exists an $(\epsilon, \delta)-$differentially private algorithm which, with probability at least $1 - \beta$, on input $S = (x_1, ..., x_n)$ outputs a list $L \subseteq \mathcal{X}$ and a sequence of estimates $a \in [0, 1]^{|L|}$ such that

- For every x with $\text{freq}_S(x) \geq \eta$, it holds that $x \in L$.
- For every $x \in L$, it holds that $|a_x - \text{freq}_S(x)| \leq \eta$.

Roughly speaking the idea given the short list of hypotheses is to use the exponential mechanism, which takes in the following input: (i) a dataset $S \in Z^n$, (ii) a set of objects H and (iii) a score function $s : Z^n \times H \to \mathbb{R}$. The only private information is the dataset S. We define the sensitivity of the score function with respect to the dataset only:

$$\Delta_s = \max_{h \in H} \max_{S, S'} |s(S, h) - s(S', h)| .$$

The exponential mechanism outputs an object $h \in H$ with probability proportional to $\exp\left(\frac{\epsilon s(S,h)}{2\Delta_s}\right)$. This mechanism is ϵ-DP and the score of its output is close to $\max_{h \in H} s(S, h)$ with high probability

26 Differential Privacy meets Graph Theory

In this section, we are going to demonstrate a link (by Alon et al. (2024)) between private learnability (both pure and approximate) and *cliques* in certain graphs, which we call *contradiction graphs*.

Definition 33: Clique

A clique in a graph G is a set δ of vertices such that every pair of distinct vertices in δ is connected by an edge.

A fractional clique is a standard LP relaxation of a clique.

Definition 34: Fractional Clique

A function $\delta : V \to [0,1]$ is called a fractional clique if for every independent set $I \subseteq V$, it holds that $\sum_{v \in I} \delta(v) \leq 1$. The size of a fractional clique δ is the sum $\sum_{v \in V} \delta(v)$.

- The clique number of G, denoted $\omega(G)$, is the largest size of a clique in G.
- The fractional clique number of G, denoted $\omega^\star(G)$, is the largest size of a fractional clique in G.

Definition 35: Contradiction Graph

Let $\mathcal{H} \subseteq \{-1,1\}^{\mathcal{X}}$ be a concept class and let $m \in \mathbb{N}$. The contradiction graph of order m of $\mathcal{H}$ is an undirected graph G_m whose vertices are datasets of size m that are consistent with $\mathcal{H}$. Two datasets are connected by an edge whenever they contradict each other.

The vertices of the contradiction graph $G_m(\mathcal{H})$ are $\mathcal{H}$-realizable sequences of length m and $\{S, S'\}$ is an edge of the contradiction graph if there is an example $x \in \mathcal{X}$ such that $(x,0) \in S$ and $(x,1) \in S'$. Given the above, we can get graph-theoretic dimensions of $\mathcal{H}$ that characterize private learning.

Theorem 31: Alon et al. (2024)

The following hold:

- For every $\mathcal{H}$, either $\omega_m = 2^m$ for all m or $\omega_m \leq \text{poly}(m)$.
- For every $\mathcal{H}$, either $\omega^\star_m = 2^m$ for all m or $\omega^\star_m \leq \text{poly}(m)$.
- $\mathcal{H}$ is pure DP learnable if and only if $\omega^\star_m \leq \text{poly}(m)$. ("small" fractional cliques.)
- $\mathcal{H}$ is approximate DP learnable if and only if $\omega_m \leq \text{poly}(m)$. ("small" cliques.)

Observe that the clique and fractional clique dimensions satisfy a (polynomial vs. exponential) dichotomy similar to the Sauer-Shelah-Perles (SSP) dichotomy of the VC dimension.

It is an open problem to use such graph-theoretic notions to characterize other learning settings such as PAC learning.

Lecture 8: Replicable and DP PAC Learning

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

In Lecture 8, we discuss about connections between replicability, total variation (TV) indistinguishability, and differential privacy in the context of PAC learning.

27 Pseudo-determinism and Replicability

In Lecture 7, we introduced the notion of *global stability* (Bun et al., 2020) that was crucial in order to show that online learnability (i.e., finite Littlestone dimension) implies DP PAC learnability of a concept class. Another way to view global stability is in the context of *pseudo-deterministic algorithms* (Gat and Goldwasser, 2011; Chen et al., 2023; Grossman and Liu, 2019; Goldwasser et al., 2017).

A pseudo-deterministic algorithm is a randomized algorithm which yields some fixed output with high probability (i.e., it looks like a deterministic algorithm to a computationally bounded observer). Thinking of a realizable distribution $\mathcal{D}$ as an instance on which PAC learning algorithm has oracle access, a globally-stable learner is one which is "weakly" pseudo-deterministic: it produces some fixed output with probability bounded away from zero (recall that the global stability parameter that we could achieve was quite small – doubly exponentially small in the Littlestone dimension).

The definition of pseudo-deterministic algorithms inspired the work of Impagliazzo et al. (2022) to introduce the following definition to capture replicability for ML algorithms.

Definition 36: Replicability (Impagliazzo et al., 2022)

Fix some source of randomness $\mathcal{R}$. An algorithm A is ρ-replicable if for any $S, S' \sim \mathcal{D}^n$, it holds that

$$\mathop{\mathbf{Pr}}_{S,S' \sim \mathcal{D}^n, r \sim \mathcal{R}}[A(S,r) \neq A(S',r)] \leq \rho\,.$$

This definition says that with probability at least $1-\rho$, the outputs of the algorithm trained on two i.i.d. datasets S and S' with shared randomness (i.e., Python seed) will be *exactly* the same. The shared random string can be seen as a way to achieve a *coupling* between two executions of the algorithm A. An interesting aspect of this definition is that replicability is verifiable; replicability under Definition 27 can be tested using polynomially many samples, random seeds r and queries to A.

Impagliazzo et al. (2022) proved an interesting positive result: any SQ algorithm can be made replicable.

Recall that a statistical query oracle for a distribution $\mathcal{D}$ over Z with accuracy τ is an oracle that gets as input a statistical query $q : Z \to [-1,1]$ and returns a value $v \in [\mathbf{E}_{\mathcal{D}}\, q - \tau, \mathbf{E}_{\mathcal{D}}\, q + \tau]$.

Exercise 9

Prove that there is an algorithm that simulates the statistical query oracles using samples from $\mathcal{D}$. Then prove that this algorithm can be made replicable with a blow-up in the sample complexity.

28 A Weaker Requirement: Total Variation Indistinguishability

At first glance, replicability seems like a rather strong requirement. We will now see a weaker definition, studied in Kalavasis et al. (2023). Total variation distance between two distributions P and Q over the probability space (Ω, Σ_Ω) can be expressed as

$$\begin{aligned} \mathrm{TV}(P,Q) &= \sup_{A \in \Sigma_\Omega} P(A) - Q(A) \\ &= \inf_{(X,Y)\sim\Pi(P,Q)} \mathbf{Pr}[X \neq Y]\,, \end{aligned} \tag{10}$$

where the infimum is over all couplings between P and Q so that the associated marginals are P and Q respectively. A *coupling* between the distributions P and Q is a set of variables (X, Y) on some common probability space with the given marginals, i.e., $X \sim P$ and $Y \sim Q$. We think of a coupling as a construction of random variables X, Y with the prescribed laws.

This definition leads to the following relaxation of replicability.

Definition 37: Total Variation Indistinguishability

A learning rule A is n-sample ρ-TV indistinguishable if for any distribution over inputs $\mathcal{D}$ and two independent sets $S, S' \sim \mathcal{D}^n$ it holds that

$$\mathop{\mathbf{E}}_{S,S'\sim\mathcal{D}^n}[\mathrm{TV}(A(S), A(S'))] \leq \rho\,.$$

This information-theoretic definition of TV indistinguishability seems to put weaker restrictions on learning rules than the notion of replicability in two ways:

- it allows for *arbitrary* couplings between the two executions of the algorithm (recall the coupling definition of TV distance), and,
- it allows for *different* couplings between every pair of datasets S, S' (the optimal coupling in the definition of TV distance will depend on S, S').

In short, our definition allows for *arbitrary data-dependent* couplings, instead of just sharing the randomness across two executions. We note that this definition is not new and it turns out that it corresponds to one-way perfect generalization (Cummings et al., 2016).

29 Privacy Reminder

The notion of algorithmic replicability that we have discussed so far has close connections with the classical definition of approximate differential privacy. For $a, b, \epsilon, \delta \in [0, 1]$, let $a \approx_{\epsilon,\delta} b$ denote the statement $a \leq e^\epsilon b + \delta$ and $b \leq e^\epsilon a + \delta$. We say that two probability distributions P, Q are (ϵ, δ)-indistinguishable if $P(E) \approx_{\epsilon,\delta} Q(E)$ for any measurable event E.

Definition 38: Approximate Differential Privacy

A learning rule A is an n-sample (ϵ, δ)-differentially private if for any pair of samples $S, S' \in (\mathcal{X} \times \{0,1\})^n$ that disagree on a single example, the induced posterior distributions $A(S)$ and $A(S')$ are (ϵ, δ)-indistinguishable.

As we saw in previous lectures, in the context of PAC learning, any hypothesis class $\mathcal{H}$ can be PAC-learned by an approximate differentially-private algorithm if and only if it has a finite Littlestone dimension $\mathrm{Ldim}(\mathcal{H})$, i.e., there is a qualitative equivalence between online learnability and private PAC learnability.

The results of the next sections appear in Kalavasis et al. (2023). Similar results independently appear in Bun et al. (2023).

30 Replicability $\iff$ TV Indistinguishability

We will prove the following result.

Theorem 32: Replicability $\Rightarrow$ TV Indistinguishability

If a learning rule A is n-sample ρ-replicable, then it is also n-sample ρ-TV indistinguishable.

Proof. Fix some distribution $\mathcal{D}$ over inputs. Let A be n-sample ρ-replicable with respect to $\mathcal{D}$. For the random variables $A(S), A(S')$ where $S, S' \sim \mathcal{D}^n$ are two independent samples and using Eq.(10), we have

$$\mathop{\mathbf{E}}_{S,S' \sim \mathcal{D}^n}[\mathrm{TV}(A(S), A(S'))] = \mathop{\mathbf{E}}_{S,S' \sim \mathcal{D}^n}\left[\inf_{(h,h') \sim \Pi(A(S),A(S'))} \mathbf{Pr}[h \neq h']\right] . \tag{11}$$

Let $\mathcal{R}$ be the source of randomness that A uses. The expected optimal coupling of Eq.(11) is at most $\mathbf{E}_{S,S' \sim \mathcal{D}^n}\left[\mathbf{Pr}_{r \sim \mathcal{R}}[A(S,r) \neq A(S',r)]\right]$. This inequality follows from the fact that using shared randomness between the two executions of A is a particular way to couple the two random variables. To complete the proof, it suffices to notice that this upper bound is equal to

$$\mathop{\mathbf{Pr}}_{S,S' \sim \mathcal{D}^n, r \sim \mathcal{R}}[A(S,r) \neq A(S',r)] \leq \rho .$$

The last inequality follows since A is ρ-replicable. □

We now deal with the opposite direction, i.e., we show that TV indistinguishability implies replicability. In order to be formal, we need to discuss some measure theoretic properties first. Let us recall the definition of absolute continuity for two measures.

Definition 39: Absolute Continuity

Consider two measures P, Q on a σ-algebra $\mathcal{B}$ of subsets of Ω. We say that P is absolutely continuous with respect to Q if for any $E \in \mathcal{B}$ such that $Q(E) = 0$, it holds that $P(E) = 0$.

Since the learning rules induce posterior distributions over hypotheses, this definition extends naturally to such rules.

Definition 40

Given learning rule A, distribution over inputs $\mathcal{D}$ and reference probability measure $\mathcal{P}$, we say that A is absolutely continuous with respect to $\mathcal{P}$ on inputs from $\mathcal{D}$ if, for almost every sample S drawn from $\mathcal{D}$, the posterior distribution $A(S)$ is absolutely continuous with respect to $\mathcal{P}$.

In the previous definition, we fixed the data-generating distribution $\mathcal{D}$. We next consider its distribution-free version.

Definition 41

Given learning rule A and reference probability measure $\mathcal{P}$, we say that A is absolutely continuous with respect to $\mathcal{P}$ if, for any distribution over inputs $\mathcal{D}$, A is absolutely continuous with respect to $\mathcal{P}$ on inputs from $\mathcal{D}$.

If $\mathcal{X}$ is finite, then one can take $\mathcal{P}$ to be the uniform probability measure over $\{0,1\}^{\mathcal{X}}$ and any learning rule is absolutely continuous with respect to $\mathcal{P}$. We now show how we can find such a prior $\mathcal{P}$ in the case where $\mathcal{X}$ is countable.

Lemma 9: Reference Probability Measure for Countable Domains

Let $\mathcal{X}$ be a countable domain and A be a learning rule. Then, there is a reference probability measure $\mathcal{P}$ such that A is absolutely continuous with respect to $\mathcal{P}$.

Proof. Since $\mathcal{X}$ is countable, for a fixed n, we can consider an enumeration of all the n-tuples $\{S_i\}_{i\in\mathbb{N}}$. Then, we can take $\mathcal{P}$ to be a countable mixture of these probability measures, i.e., $\mathcal{P} = \sum_{i=1}^{\infty} \frac{1}{2^i} A(S_i)$. Notice that since, each $A(S_i)$ is a measure and $1/2^i > 0$ for $i \in \mathbb{N}$, and, $\sum_{i=1}^{\infty} 1/2^i = 1$, we have that $\mathcal{P}$ is indeed a probability measure. We now argue that each $A(S_i)$ is absolutely continuous with respect to $\mathcal{P}$. Assume towards contradiction that this is not the case and let $E \in \mathcal{B}$ be a set such that $\mathcal{P}(E) = 0$ but $A(S_j)(E) \neq 0$, for some $j \in \mathbb{N}$. Notice that $A(S_j)$ appears with coefficient $1/2^j > 0$ in the mixture that we consider, hence if $A(S_j)(E) > 0 \implies 1/2^j A(S_j)(E) > 0$. Moreover $A(S_i)(E) \geq 0, \forall i \in \mathbb{N}$, which means that $\mathcal{P}(E) > 0$, so we get a contradiction. □

We next define when two learning rules A, A' are equivalent.

Definition 42: Equivalent Learning Rules

Two learning rules A, A' are equivalent if for every sample S it holds that $A(S) = A'(S)$, i.e., for the same input they induce the same distribution over hypotheses.

In the next result, we show that for every TV indistinguishable algorithm A, that is absolutely continuous with respect to some reference probability measure $\mathcal{P}$, there exists an equivalent learning rule which is replicable.

Theorem 33: TV Indistinguishability ⇒ Replicability

Let $\mathcal{P}$ be a reference probability measure over $\{0,1\}^{\mathcal{X}}$, and let A be a learning rule that is n-sample ρ-TV indistinguishable and absolutely continuous with respect to $\mathcal{P}$. Then, there exists an equivalent learning rule A' that is n-sample $\frac{2\rho}{1+\rho}$-replicable.

This implies the following:

Theorem 34

Let $\mathcal{X}$ be a countable domain and let A be a learning rule that is n-sample ρ-TV indistinguishable. Then, there exists an equivalent learning rule A' that is n-sample $\frac{2\rho}{1+\rho}$-replicable.

Proof Sketch. Let us consider a learning rule A satisfying the conditions of Theorem 30. Fix a distribution $\mathcal{D}$ over inputs. The crux of the proof is that given two random variables X, Y whose TV distance is bounded by ρ, we can couple them using only a carefully designed source of shared randomness $\mathcal{R}$ so that the probability that the realizations of these random variables differ is at most $2\rho/(1+\rho)$. We can instantiate this observation with $X = A(S)$ and $Y = A(S')$. Crucially, in the countable $\mathcal{X}$ setting, we can pick the shared randomness $\mathcal{R}$ in a way that only depends on the learning rule A, but not on S or S'. Let us now describe how this coupling works. Essentially, it can be thought of as a generalization of the von Neumann

rejection-based sampling which does not necessarily require that the distribution has bounded density. Following Angel and Spinka (2019), we pick $\mathcal{R}$ to be a Poisson point process which generates points of the form (h, y, t) with intensity[9] $\mathcal{P} \times \text{Leb} \times \text{Leb}$, where $\mathcal{P}$ is a reference probability measure with respect to which A is absolutely continuous and Leb is the Lebesgue measure over $\mathbb{R}_+$. Intuitively, $h \sim \mathcal{P}$ lies in the hypotheses' space, y is a non-negative real value and t corresponds to a time value. The coupling mechanism performs *rejection sampling* for each distribution we would like to couple (here $A(S)$ and $A(S')$): it checks (in the ordering indicated by the time parameter) for each point (h, y, t) whether $f(h) > y$ (i.e., if y falls below the density curve f at h) and accepts the first point that satisfies this condition. In the formal proof, there will be two density functions; f (resp. f') for the density function of $A(S)$ (resp. $A(S')$). We also refer to 12. One can show that $\mathcal{R}$ gives rise to a coupling between $A(S)$ and $A(S')$ under the condition that both measures are absolutely continuous with respect to the reference probability measure $\mathcal{P}$.

This coupling technique appears in Angel and Spinka (2019), which shows:

Theorem 35: Pairwise Optimal Coupling

Let $\mathcal{S}$ be any collection of random variables that are absolutely continuous with respect to a common probability measure[a] μ. Then, there exists a coupling of the variables in $\mathcal{S}$ such that, for any $X, Y \in \mathcal{S}$,

$$\mathbf{Pr}[X \neq Y] \leq \frac{2\text{TV}(X, Y)}{1 + \text{TV}(X, Y)}.$$

Moreover, this coupling requires sample access to a Poisson point process with intensity $\mu \times \text{Leb} \times \text{Leb}$, where Leb is the Lebesgue measure over $\mathbb{R}_+$, and full access to the densities of all the random variables in $\mathcal{S}$ with respect to μ.

[a] This result extends to the setting where μ is a σ-finite measure, but it is not needed for the purposes of our work.

Insight 11: Intuition of the above Result

Crucially, the points generated by the Poisson point process $\mathcal{R}$ are generated once. However, we may accept a different point for X and a different point for Y. The key property is that the probability that we pick a different point is roughly speaking the total variation distance between the two distributions.

Insight 12: Correlated Sampling

The above rejection sampling process is one way to couple the random variables. Another way to couple the two random variables is via **correlated sampling.** We refer to Bavarian et al. (2016) and Bun et al. (2023) for details.

We can then apply it and get

$$\mathop{\mathbf{Pr}}_{r \sim \mathcal{R}}[A(S, r) \neq A(S', r)] \leq \frac{2\text{TV}(A(S), A(S'))}{1 + \text{TV}(A(S), A(S'))}.$$

Taking the expectation with respect to the draws of S, S', we show (after some algebraic manipulations) that $\mathbf{Pr}_{S,S' \sim \mathcal{D}^n r \sim \mathcal{R}}[A(S, r) \neq A(S', r)] \leq 2\rho/(1 + \rho)$. We conclude this section with the following remarks.

[9] Roughly speaking, a point process is a (general) Poisson point process with intensity λ if (i) the number of points in a bounded Borel set E is a Poisson random variable with mean $\lambda(E)$ and (ii) the numbers of points in n disjoint Borel sets forms n independent random variables.

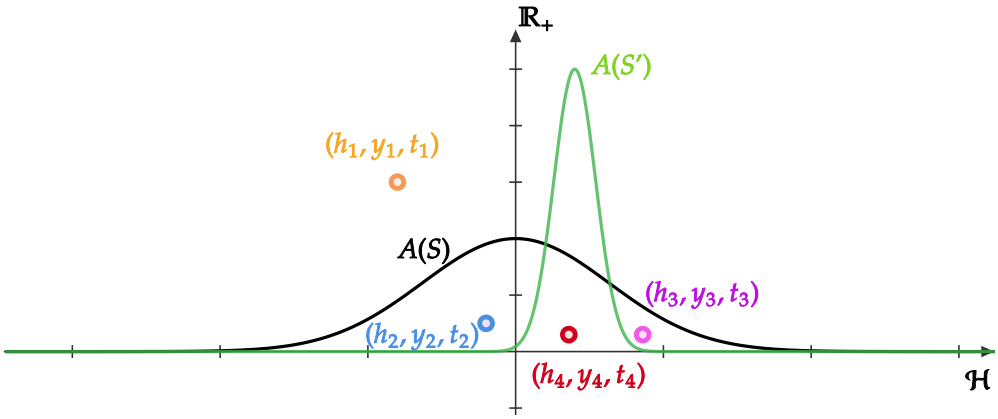


Figure 12: Taken from Kalavasis et al. (2023). Our goal is to couple $A(S)$ with $A(S')$, where these two distributions are absolutely continuous with respect to the reference probability measure $\mathcal{P}$. A sequence of points of the form (h, y, t) is generated by the Poisson point process with intensity $\mathcal{P} \times \text{Leb} \times \text{Leb}$ where $h \sim \mathcal{P}, (y, t) \in \mathbb{R}_+^2$ and Leb is the Lebesgue measure over $\mathbb{R}_+$ (note that we do not have upper bounds for the densities). Intuitively, h lies in the hypotheses' space, y is a non-negative real value and t corresponds to a time value. Let f be the Radon-Nikodym derivate of $A(S)$ with respect to $\mathcal{P}$. We assign the first (the one with minimum t) value h to $A(S)$ that satisfies the property that $f(h) > y$, i.e., y falls below the density curve of $A(S)$. We assign a hypothesis to $A(S')$ in a similar manner. This procedure defines a data-independent way to couple the two random variables and naturally extends to multiple ones. In the example, we set $A(S) = h_2$ and $A(S') = h_4$ given that $t_1 < t_2 < t_3 < t_4$.

31 TV Indistinguishability $\iff$ DP

In this section we study the connections with differential privacy. We start with the following result.

Theorem 36: (ϵ, δ)-DP $\Rightarrow$ TV Indistinguishability

Let $\mathcal{X}$ be a (possibly infinite) domain and $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$. Let $\gamma \in (0, 1/2), \alpha, \beta, \rho \in (0,1)^3$. Assume that $\mathcal{H}$ is learnable by an n-sample $(1/2 - \gamma, 1/2 - \gamma)$-accurate $(0.1, 1/(n^2 \log(n)))$-differentially private learner. Then, it is also learnable by an (α, β)-accurate ρ-TV indistinguishable learning rule.

Proof Sketch The proof goes through the notion of global stability. The existence of an (ϵ, δ)-DP learner implies that the hypothesis class $\mathcal{H}$ has finite Littlestone dimension (Alon et al., 2019). Thus, we know that there exists a ρ-globally stable learner for $\mathcal{H}$ (Bun et al., 2020). The next step is to use the replicable heavy-hitters algorithm (Impagliazzo et al., 2022) with frequency parameter $O(\rho)$ and replicability parameter $O(\rho')$, where $\rho' \in (0,1)$ is the desired TV indistinguishability parameter of the learning rule.

Global stability implies that the list of heavy-hitters will be non-empty and it will contain at least one hypothesis with small error rate, with high probability. Finally, since the list of heavy-hitters is finite and has bounded size, we feed the output into the replicable agnostic learner. Thus, we have designed a replicable learner for $\mathcal{H}$, and Theorem 30 shows that this learner is also TV indistinguishable.

We proceed to the opposite direction where we provide an algorithm that takes as input a TV indistinguishable learning rule for $\mathcal{H}$ and outputs a learner for $\mathcal{H}$ which is (ϵ, δ)-DP. In this direction countability of $\mathcal{X}$ is crucial.

Theorem 37: TV Indistinguishability $\Rightarrow (\epsilon, \delta)$-DP

Let $\mathcal{X}$ be a countable domain. Assume that $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ is learnable by an (α, β)-accurate ρ-TV indistinguishable learner A, for some $\rho \in (0,1), \alpha \in (0, 1/2), \beta \in \left(0, \frac{1-\rho}{1+\rho}\right)$. Then, for any $(\alpha', \beta', \varepsilon, \delta) \in (0,1)^4$, it is also learnable by an $(\alpha + \alpha', \beta')$-accurate (ε, δ)-differentially private learner A'.

On a high level, the proof resembles the approaches of Bun et al. (2020); Ghazi et al. (2021); Bun et al. (2023) to show that (pseudo-)global stability implies differential privacy. We consider k' different batches of k datasets of size n. For each such batch, our goal is to couple the k different executions of the algorithm on an input of size n, so that most of these outputs are, with high probability, the same.

One first approach would be to use a random variable as a "pivot" element in each batch: we first draw $A(S_1^1)$ according to its distribution and the remaining $\{A(S_i^1)\}_{i \in [k] \setminus \{1\}}$ from their optimal coupling with $A(S_1^1)$, given its realized value. Even though this coupling has the property that, in expectation, most of the outputs will be the same, it is not robust at all. If the adversary changes a point of S_1^1, then the values of all the outputs will change! This is not privacy preserving. For this reason, we use the coupling that we used before. We use the fact that $\mathcal{X}$ is countable to design a reference probability measure $\mathcal{P}$ that is independent of the data. This is the key step that leads to privacy-preservation. Then, we can argue that if we follow this approach for multiple batches, there will be a classifier whose frequency and performance are non-trivial. The next step is to feed all these hypotheses into the Stable Histograms algorithm, which will output a list of frequent hypotheses that includes the non-trivial one we mentioned above. Finally, we feed these hypotheses into the Generic Private Learner and we get the desired result.

Hence the algorithm looks as follows:

1. Since $\mathcal{X}$ is countable, we define $\mathcal{P} = \sum A(S_i)/2^i$ for some enumeration of the n-element sets.
2. Now we perform the von-Neumann coupling $\Pi_{\mathcal{R}}$ in each batch to get classifiers
$$(h_1^j, ..., h_k^j) \leftarrow \Pi_{\mathcal{R}}(A(S_1^j), ..., A(S_k^j)), j \in [k'].$$
3. Next, we apply to each batch the stable histograms algorithm (e.g., check Lecture 7) which outputs the most frequent elements of a list in a private manner. We get that
$$L^j = \text{StableHist}(h_1^j, ..., h_k^j)\,.$$
4. Remove rare elements from the lists and get $\widetilde{L}^j$.
5. Apply a standard DP PAC learner for the finite concept class $\cup_{j \in [k']} \widetilde{L}^j$.

Lecture 9: Memorization in Machine Learning

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

Lecture 9 focuses on two important theoretical works by Feldman (2020) and Brown et al. (2021) on memorization in Machine Learning, and its connections to privacy and stability. Moreover, we discuss memorization in generative models and, in particular, diffusion models (Shah et al., 2025).

32 Memorization, Learning, and Privacy

Deep learning models are over-parameterized, meaning they have far more tunable parameters than available data points. As a result, they can easily "overfit" by memorizing training labels rather than using the training points to generalize to unseen data. As we have already seen in prior lectures, state-of-the-art image recognition models, which achieve near-perfect training accuracy, are known to fit even randomly assigned labels (Zhang et al., 2021). The main question we would like to understand is the following:

Is memorization necessary for learning (e.g., image classification) and generative modeling (e.g., diffusion models)?

Do near-optimal classifiers have to memorize training examples? Apart from classification, questions about generation and memorization have deep connections to various areas of research such as linguistics (Slobin, 2013).

Beyond its theoretical implications, memorization raises serious *privacy risks*, particularly when training data includes sensitive personal information.

- Regarding classification, there are various works studying memorization or overfitting (Zhang et al., 2019, 2021; Belkin et al., 2018). Many works focus on *label memorization* e.g., Feldman (2020); Arpit et al. (2017); Ma et al. (2018) or on memorization of the entire training set (Brown et al., 2021). Zhang et al. (2021) empirically demonstrate that DNNs are capable of fitting random data, which implicitly necessitates some high degree of memorization. Feldman (2020) rigorously shows that, for some problems, label memorization is necessary for achieving near-optimal accuracy on test data.
- Carlini et al. (2023); Daras et al. (2023, 2024); Somepalli et al. (2022, 2023); Ross et al. (2024) show that diffusion models memorize the training data and often replicate them at generation time.
- Carlini et al. (2021) demonstrate that modern models for next-word prediction memorize large chunks of text from the training data verbatim, including personally identifiable and sensitive information such as phone numbers.

33 Regularization and Memorization

The standard theoretical framework for preventing overfitting relies on *regularization*, which ensures a balance between model complexity and empirical error. The idea is that fitting outliers requires increasing model complexity (intuitively increasing the VC dimension or model capacity), and by tuning this trade-off, algorithms can capture meaningful patterns without overfitting. However, this perspective contradicts empirical observations in modern deep learning.

- Models trained on image and text classification datasets often achieve near-perfect (95–100%) accuracy on training data.

- This happens even when test accuracy is significantly lower (typically 50–80%).
- Such high training accuracy (with low test error) implies memorization of mislabeled data and outliers, which are unavoidable in large datasets.
- Moreover, even with strong regularization, models achieve over 90% training accuracy on ImageNet even when labels are randomly assigned (Zhang et al., 2021). This suggests that current regularization techniques are not strong enough to prevent memorization.

34 Label Memorization: A Short Tale about a Long Tail

Feldman (2020) proposes a conceptually simple explanation and supporting theory for

why memorization of seemingly useless labels may be necessary to achieve close-to-optimal generalization error.

It is based on the viewpoint that the primary hurdle to learning an accurate model is not the noise inherent in the labels but rather an insufficient amount of data to predict accurately on rare and atypical instances.

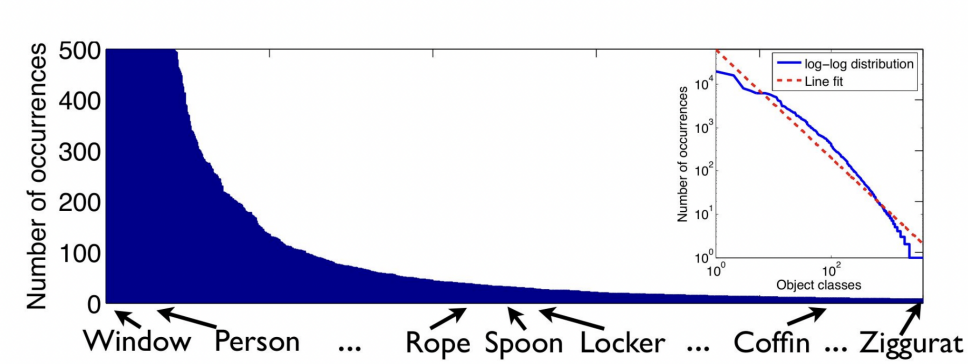
Insight 13: TLDR

Feldman (2020) theoretically shows strong trade-offs between memorization and generalization by showing that memorization is *necessary* for (optimal) *classification*.
The need for memorization in (Feldman, 2020) is associated with the frequencies of different subpopulations (e.g., cats, dogs, etc.) that appear in the dataset. The key observation is that the distribution of the frequencies is usually *heavy-tailed* (Zhu et al., 2014), i.e., roughly speaking in a dataset of size n, there will be many classes with frequency around $1/n$. This means that the training algorithm will only observe a single representative from those subpopulations and cannot distinguish between the following two cases:
Case 1. If the unique example comes from an extremely rare subpopulation (with frequency $\ll 1/n$), then memorizing it has no significant benefits, and,
Case 2. If the unique example comes from a subpopulation with $1/n$ frequency, then memorizing it will probably improve the accuracy on the entire subpopulation and decrease the generalization error by $\Omega(1/n)$. Hence, the optimal classifier should memorize these unique examples to avoid paying Case 2 in the error.
Rare or atypical instances are also the cause of language model hallucinations. The existence of such rare instances was the key tool for Kalai and Vempala to prove that calibrated language models must hallucinate (Kalai and Vempala, 2024).

34.1 What is a "rare" example?

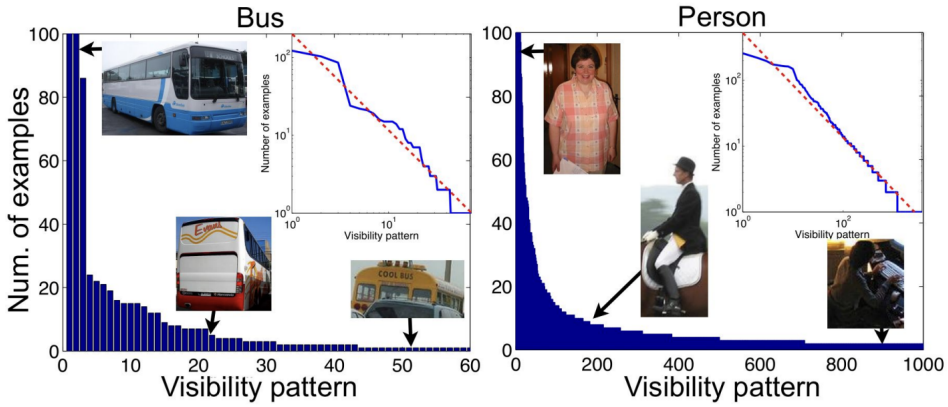
In practice, rare instances are referred as the "long tail" of the data distribution. Modern datasets used for visual object recognition and text labeling follow classical long-tailed distributions such as Zipf distribution (or more general power law distributions); i.e., the frequencies of the classes of the datasets follow a heavy-tailed distribution (see Figure 13).

To be more specific, let us consider a multiclass prediction problem. To formalize the notion of having a "long (or heavy) tail", we will model the data distribution of each class *as a mixture of distinct subpopulations*. For example, images of birds include numerous different species photographed from different perspectives and under different conditions (such as close-ups, in foliage and in the sky). Naturally, the subpopulations may have different frequencies (which correspond to mixture coefficients). We model long-tailed data distributions as distributions in which the frequencies of subpopulations are long-tailed (see Figure 14).



(a) The number of examples by object class in SUN dataset

Figure 13: Taken from Feldman (2020). The distribution of the frequencies of the classes in a dataset is heavy-tailed.



(b) Distributions of the visibility patterns for bus and person

Figure 14: Taken from Feldman (2020). For a fixed (coarse) label (e.g., bus or person), the distribution of the frequencies of the "sub-populations" (fine labels) in a dataset is heavy-tailed.

34.2 What do we predict on "rare" examples?

We make two reasonable hypotheses:

1. Before seeing the dataset, the learning algorithm does not know the frequencies of subpopulations.
2. The algorithm is not able to predict accurately on a subpopulation until at least one example from the subpopulations is observed.

Feldman's reasoning then goes as follows:

1. A dataset of n samples from a long-tailed mixture distribution will have some subpopulations from which just a single example was observed (and some subpopulations from which none at all).
2. To predict more accurately on a subpopulation from which only a single example was observed (and to fit the example) the learning algorithm needs to memorize the label of the example.

Is this behavior necessary for close-to-optimal generalization error. The crucial observation of Feldman (2020) is that this depends on the frequency of the subpopulation.

- If the unique example from a subpopulation (or singleton) comes from an extremely rare (or "outlier") subpopulation (say with mixing coefficient 2^{-n}) then memorizing it has no significant benefits.
- At the same time, if the singleton comes from an "atypical" subpopulation with frequency on the order of $1/n$, then memorizing such an example is likely to improve the accuracy on the entire subpopulation and thereby reduce the generalization error by $\Omega(1/n)$.

The key point of Feldman (2020) is that based on observing a single sample from a subpopulation, *it is impossible to distinguish samples from "atypical" subpopulations from those in the "outlier" ones.*

Hence, an algorithm can only avoid the risk of missing "atypical" subpopulations by memorizing the labels of singletons from the "outlier" subpopulations (since it cannot distinguish between them). The next step is to measure the mass of the $1/n$ coefficients: in a long-tailed distribution of frequencies, the total weight of frequencies on the order of $1/n$ is significant enough that ignoring these subpopulations will hurt the generalization error substantially.

Combining the above, any algorithm that achieves near-optimal generalization against such heavy-tailed instances, has to memorize the labels of outliers.

34.3 The Key Result

Consider a discrete population $X = [N]$ and label space $Y = [m]$.

Consider a set of functions $\mathcal{F}$ mapping X to Y. Fix a known prior $\pi = (\pi_1, ..., \pi_N)$. We consider a random distribution over X as follows: for any $x \in X$, set p_x randomly and independently from π. Define the pdf on X as $D(x) = p_x / \sum_x p_x$. We also let $\overline{\pi}$ the resulting marginal distribution over the frequency of any single element in x.

An instance of our learning problem is generated by picking a random $D \sim \mathcal{D}_\pi$ and a random true labeling $f \sim \mathcal{F}$.

We let

$$\overline{\text{err}}(\pi, \mathcal{F}, \mathcal{A}) = \mathop{\mathbf{E}}_{D \sim \mathcal{D}_\pi, f \sim \mathcal{F}} \text{err}_{D,f}(\mathcal{A}) .$$

We can now define

$$\tau_1 = \frac{\mathbf{E}_a[a^2(1-a)^{n-1}]}{\mathbf{E}_a[a(1-a)^{n-1}]}$$

where a is drawn from the actual marginal distribution over frequencies that results from our process (see Feldman (2020)). This essentially captures the expected frequency of a sample conditioned on observing it in the dataset exactly once. We can extend this definition to τ_ℓ for $\ell \in [n]$.

The main result is as follows.

Theorem 38

For every learning algorithm $\mathcal{A}$ and any dataset $Z \in (X \times Y)^n$,

$$\overline{\text{err}}(\pi, \mathcal{F}, \mathcal{A}|Z) \geq \text{opt}(\pi, \mathcal{F}|Z) + \sum_{\ell \in [n]} \tau_\ell \cdot \text{errn}_Z(\mathcal{A}, \ell) .$$

Proof. The key observation is that D and f are picked independently during the generation of the learning instance. We write $G(\cdot|Z)$ as the distribution that generates the pair (D, f) conditioned on Z; note that this

is a product measure. We can write

$$\overline{\mathrm{err}}(\pi, \mathcal{F}, \mathcal{A}|Z) = \mathop{\mathbf{E}}_{(D,f)\sim G(\cdot|Z)} \mathop{\mathbf{E}}_{h\sim\mathcal{A}(Z)} \sum_{x\in X} 1\{h(x) \neq f(x)\} D(x)\,.$$

We now decompose X as the elements x that do not appear in the training set Z (i.e., $x \in X_{Z\#0}$) and the elements x that appear exactly ℓ times ($x \in X_{Z\#\ell}$ for any $\ell \in [n]$). Let $X_Z = \cup_\ell X_{Z\#\ell}$. We can hence decompose $\sum_{x\in X} = \sum_{x\in X_{Z\#0}} + \sum_{x\in X_Z}$. For every $x \in X_{Z\#0}$, we get

$$\mathop{\mathbf{E}}_{(D,f)\sim G(\cdot|Z)} \mathop{\mathbf{E}}_{h\sim\mathcal{A}(Z)} 1\{h(x) \neq f(x)\} D(x) = \mathop{\mathbf{Pr}}_{f\sim\mathcal{F}, h\sim\mathcal{A}(Z)}[h(x) \neq f(x)] \mathop{\mathbf{E}}_{D\sim\mathcal{D}(\cdot|Z)}[D(x)]\,.$$

Similarly, for $x \in X_{Z\#\ell}$,

$$\mathop{\mathbf{E}}_{(D,f)\sim G(\cdot|Z)} \mathop{\mathbf{E}}_{h\sim\mathcal{A}(Z)} 1\{h(x) \neq f(x)\} D(x) = \mathop{\mathbf{Pr}}_{f\sim\mathcal{F}, h\sim\mathcal{A}(Z)}[h(x) \neq f(x)] \mathop{\mathbf{E}}_{D\sim\mathcal{D}(\cdot|Z)}[D(x)]\,,$$

and we observe that for that x (which appears exactly ℓ times):

$$\mathop{\mathbf{E}}_{D\sim\mathcal{D}(\cdot|Z)}[D(x)] = \tau_\ell\,.$$

Hence, we get that

$$\overline{\mathrm{err}}(\pi, \mathcal{F}, \mathcal{A}|Z) = \sum_{\ell\in[n]} \tau_\ell \cdot \mathrm{errn}_Z(\mathcal{A}, \ell) + \sum_{x\in X_{Z\#0}} \mathop{\mathbf{Pr}}_{f,h}[h(x) \neq f(x)] \mathop{\mathbf{E}}_{D}[D(x)]\,.$$

This implies the claimed inequality since the right-hand side is minimizes when for all $\ell \in [n]$, $\mathrm{errn}_Z(\mathcal{A}, \ell) = 0$ and for all unseen x, it holds that $\mathbf{Pr}_{h,f}[h(x) \neq f(x)] = \min_{y\in Y} \mathbf{Pr}_f[f(x) \neq y]$. Note that this minimum is minimized by the algorithm $\mathcal{A}^*$ that memorizes the examples in Z and predicts the minimizing label y for all $x \in X_{Z\#0}$. Hence, $\overline{\mathrm{err}}(\pi, \mathcal{F}, \mathcal{A}^*|Z) = \mathrm{opt}(\pi, \mathcal{F})$.

□

We can remove the conditioning on Z by taking an expectation over the Z-marginal.

Note that the above can be generalized to continuous populations. Here, we model the unlabeled data distribution as a mixture of a large number of fixed distributions $M_1, ..., M_N$ and we sample the mixing coefficients using the same random process as before to get a mixture $M(x) = \sum_i D_i M_i(x)$. To put some structure, we assume that the entire subpopulation X_i is labeled by the same label and now we look at the multiplicity of sub-domains and not points themselves and count mistakes just once per sub-domain.

34.4 Heavy-Tailed Distributions

We are going to formally explain what it means for the frequencies of the original dataset to be heavy-tailed (Zhu et al., 2014; Feldman, 2020). This heavy-tailed structure will then allow us to control the generalization error. We will be interested in subpopulations that have only one representative in the training set Z (these are the examples that will cost roughly τ_1 in the error). We will refer to them as *single* subpopulations.

For this to happen given that $|Z| = n$, it should be roughly speaking the case where some frequencies D_i are of order $1/n$. The quantity that controls how many of the frequencies D_i will be of order $1/n$ is the mass that the distribution $\overline{\pi}(a) = \mathbf{Pr}_D[D_i = a]$ assigns to the interval $[1/(2n), 1/n]$.

Typically, we will call a list of frequencies π *heavy-tailed* if

$$\mathrm{weight}\left(\overline{\pi}, \left[\frac{1}{2n}, 1/n\right]\right) = \Omega(1)\,. \tag{12}$$

In words, there should be a constant number of subpopulations with frequencies of order $O(1/n)$. This definition is important because it can then lower bound the value τ_1 and hence it can lower bound the generalization loss of not fitting single subpopulations.

Lemma 10

Consider a dataset of size n and assume that π is heavy-tailed, as in (12). Then $\tau_1 = \Omega(1/n)$.

On the contrary, when π is not heavy-tailed, τ_1 will be small and hence generalization is not hurt by not memorizing.

34.5 Memorization and Stability

Using the above key result, we have that

$$\overline{\mathrm{err}}(\pi, \mathcal{F}, \mathcal{A}) \geq \mathrm{opt}(\pi, \mathcal{F}) + \tau_1 \cdot \mathop{\mathbf{E}}_{Z} \mathrm{errn}_Z(\mathcal{A}, 1)\,.$$

This inequality relates the sub-optimality gap of an arbitrary algorithm A to its error in elements that appear exactly once in the dataset.

Let us now introduce the notion of memorization. Let us, in particular, see the definition of *label memorization* according to Feldman (2020):

$$\mathrm{mem}(A, S, i) = \mathop{\mathbf{Pr}}_{h \sim A(S)}[h(x_i) = y_i] - \mathop{\mathbf{Pr}}_{h \sim A(S_{-i})}[h(x_i) = y_i]$$

or

$$\mathrm{mem}(A, S, i) = \mathop{\mathbf{Pr}}_{h \sim A(S_{-i})}[h(x_i) \neq y_i] - \mathop{\mathbf{Pr}}_{h \sim A(S)}[h(x_i) \neq y_i]\,.$$

We think of a label as memorized when this value is larger than some fixed positive constant.

Observe that this definition is closely related to the classical leave-one-out notion of stability but focuses on the change in the label and not in the incurred loss.

The expectation over the choice of dataset of the average memorization value is equal to the expectation of the generalization gap:

$$\frac{1}{n} \mathop{\mathbf{E}}_{S \sim P^n} \sum_{i \in [n]} \mathrm{mem}(A, S, i) = \mathop{\mathbf{E}}_{S' \sim P^{n-1}}[\mathrm{err}_P(A(S'))] - \mathop{\mathbf{E}}_{S \sim P^n}[\mathrm{err}_S(A(S))]$$

where

$$\mathrm{err}_S(A(S)) = \frac{1}{n} \sum_i \mathop{\mathbf{Pr}}_{h \sim A(S)}[h(x_i) \neq y_i]\,, \quad \mathrm{err}_P(A(S')) = \mathop{\mathbf{Pr}}_{(x,y) \sim P} \mathop{\mathbf{Pr}}_{h \sim A(S'))}[h(x) \neq y]\,.$$

Thus a large generalization gap implies that a significant fraction of labels is memorized. An immediate corollary of this definition is that an algorithm with a limited ability to memorize labels will not fit the singleton data points well whenever the algorithm cannot predict their labels based on the rest of the dataset.

Lemma 11

For any dataset $S \in (X \times Y)^n$, any algorithm A and $i \in [n]$,

$$\mathop{\mathbf{Pr}}_{h \sim A(S)}[h(x_i) \neq y_i] = \mathop{\mathbf{Pr}}_{h \sim A(S_{-i})}[h(x_i) \neq y_i] - \mathrm{mem}(A, S, i)\,.$$

This means that

$$\mathrm{errn}_S(A, 1) = \sum_{i \in [n], x_i \in X_{S\#1}} \mathop{\mathbf{Pr}}_{h \sim A(S_{-i})}[h(x_i) \neq y_i] - \mathrm{mem}(A, S, i)\,.$$

Hence, if the first term in the RHS is large (the algorithm cannot predict the label based on the rest of the training set) then memorization must be large in order to make the LHS small.

35 Memorization of Training Data

We next shortly discuss the results of the work of Brown et al. (2021). We will study instances in which most of the information about entire high-dimensional training examples (and not only the labels) must be encoded by near-optimal learning algorithms.

We define a problem instance P as a distribution over labeled examples, i.e., P is a distribution over where $\mathcal{S} = X \times Y$ is a space of examples X paired with labels in Y. A dataset $S \in \mathcal{S}^n$ is generated by sampling i.i.d. from such a distribution. We assume that d is the dimension of the data; hence S can be described by $\Theta(nd)$ bits (this will be important for the information-theoretic result that follows). The instance P is itself drawn from a meta-distribution q, dubbed the learning task. The learning task q is assumed to be known to the learner, but the specific problem instance is a priori unknown.

Given $S \sim P^n$, the algorithm A produces a model $M = A(S)$. We are interested in the following loss:

$$\mathrm{err}_{q,n}(A) = \mathop{\mathbf{Pr}}_{P\sim q} \mathop{\mathbf{Pr}}_{S\sim P^n} \mathop{\mathbf{Pr}}_{(z,y)\sim P} \mathop{\mathbf{Pr}}_{\text{coins}A,M} [M(z) \neq y]$$

Theorem 39

Assume that the examples lie in $\{0,1\}^d$. For all n and d, there exist natural tasks q for which any algorithm A that satisfies, for small constant ϵ,

$$\mathrm{err}_{q,n}(A) \leq \mathrm{err}_{q,n}(A_{OPT}) + \epsilon$$

also satisfies

$$I(S; M|P) = \Omega(nd)\,.$$

This essentially means that any algorithm with near-optimal accuracy on q must memorize $\Omega(nd)$ bits about the sample.

Insight 14: Interpretation of this result

Recall that the conditional mutual information is defined via two conditional entropy terms:

$$I(S; M|P) = H(S|P) - H(S|M, P)\,.$$

Consider an observer who knows the full data distribution P (but not X).

1. The term $I(S; M|P)$ captures how the observer's uncertainty about the set S is reduced after observing the model M.
2. The term $H(S|P)$ captures the uncertainty about what is unique to the data set, such as noise or irrelevant features.
3. So $I(S; M|P) = \Omega(nd)$ means that (i) the learning algorithm must encode a constant fraction of the information it receives (since after seeing M, there is a very large drop in the uncertainty), but also (ii) a constant fraction of what the model encodes is irrelevant to the task (since the drop is against $H(S|P)$ which are the unique properties of the dataset unrelated to the general task P).

Insight 15: Why do we need a meta-distribution?

The meta-distribution q captures the learner's initial uncertainty about the problem, and is essential to the result: if the exact distribution P were known to the learner A, it could simply ignore X and

write down an optimal classifier for P as its model M. In that case, we would have $I(X; M|P) = 0$. That said, since conditional information is an average over realizations of P, our result also means that for every learner, $I(M; X)$ is large for some particular p in the support of q.

36 Memorization and Diffusion Models

For a more empirical perspective on memorization and diffusion models, we refer to Somepalli et al. (2022, 2023); Daras et al. (2023); Shah et al. (2025).

In this section, we will shortly discuss about the work of Shah et al. (2025). We first mention how one measures empirically quality and memorization.

Quality We compute the Fréchet Inception Distance (Heusel et al., 2017) (FID) between 50,000 generated samples and 50,000 dataset samples as a measure of quality.

Memorization We measure memorization by computing the similarity score (i.e., inner product) of each generated sample to its nearest neighbor in the embedding space of DINOv2 (Oquab et al., 2023).

Diffusion The first step in diffusion modeling is to design a corruption process. We define a sequence of increasing corruption levels indexed by $t \in [0, 1]$, with:

$$X_t = \sqrt{1 - \sigma_t^2} X_0 + \sigma_t Z, \quad Z \sim \mathcal{N}(0, I_d), \tag{13}$$

where the map $\sigma_t := \sigma(t)$ is the noise schedule and X_0 is drawn from the clean distribution p_0. Our ultimate goal is to sample from the unknown distribution p_0. The key idea behind diffusion modeling is to learn the score functions, defined as $\nabla \log p_t(\cdot)$, for different noise levels t, where $X_t \sim p_t$.

one can train directly for the score function using the noise prediction loss (Ho et al., 2020; Vincent, 2011):

$$J(\theta) = \mathop{\mathbf{E}}_{x_0, x_t, t} \left[\left\| s_\theta(x_t, t) - \frac{\sqrt{1 - \sigma_t^2} x_0 - x_t}{\sigma_t^2} \right\|^2 \right]. \tag{14}$$

Given access to the score function for different times t, one can sample from the distribution of p_0 by running the process (Song et al., 2020):

$$\mathrm{d}x = \left(-x - \frac{(\mathrm{d}\sigma_t / \mathrm{d}t)\sigma_t}{1 + \sigma_t^2} \nabla \log p_t(x_t) \right) \mathrm{d}t. \tag{15}$$

The underlying distribution of x_0 is continuous, but in practice we only optimize this objective over a finite distribution of training points. Prior work has shown that when the expectation is taken over an empirical distribution $\widehat{p}_0$, the optimal score can be written in closed form (Scarvelis et al., 2023; Biroli et al., 2024; De Bortoli, 2022; Kamb and Ganguli, 2024; Benton et al., 2024). Specifically, the optimal score for the empirical distribution is a gradient filed where each point x_0 in the finite sample S (i.e., the empirical distribution $\widehat{p}_0$) is pulling the noisy iterate x_t towards itself, where the weight of the pull depends on the distance of each training point to the noisy point. The optimal score will lead to a diffusion model that only *replicates* the training points during sampling (Scarvelis et al., 2023; Kamb and Ganguli, 2024). Hence, any potential creativity that is observed experimentally in diffusion models comes from the failure to perfectly optimize the training objective.

Ambient Score Matching One way to mitigate memorization is to never see the training data. Consider the case where we are given samples from a noisy distribution p_{t_n} (where t_n stands for t-nature) and we desire to learn the score at time t for $t > t_n$. The Ambient Score Matching loss (Daras et al., 2024), defined as:

$$J_{\text{ambient}}(\theta) = \mathbb{E}_{x_{t_n}} \mathbb{E}_{(x_t,t)|x_{t_n}} \left[\left\| \frac{\sigma_t^2 - \sigma_{t_n}^2}{\sigma_t^2 \sqrt{1-\sigma_{t_n}^2}} h_\theta(x_t, t) + \frac{\sigma_{t_n}^2}{\sigma_t^2} \sqrt{\frac{1-\sigma_t^2}{1-\sigma_{t_n}^2}} x_t - x_{t_n} \right\|^2 \right], \tag{16}$$

can learn the conditional expectation $\mathbf{E}[x_0|x_t]$ without ever looking at clean data from p_0.

> **Exercise 10**
>
> Prove that Ambient Score matching learns the correct conditional expectation.

The intuition behind this objective is that to denoise the noisy sample x_t, we need to find the direction of the noise and then rescale it appropriately. The former can be found by denoising to an intermediate level t_n and the rescaling ensures that we denoise all the way to the level of clean images. Once the conditional expectation $\mathbf{E}[x_0|x_t]$ is recovered, we get the score by using Tweedie's Formula. We remark that this objective can only be used for $t > t_n$.

We are now ready to present our framework for training diffusion models with limited data that will allow creativity without sacrificing quality. Our key observation is that the diversity of the generated images is controlled in the high-noise part of the diffusion trajectory (Dieleman, 2024; Li and Chen, 2024). Hence, if we can avoid memorization in this regime, it is highly unlikely that we will replicate training examples at inference time, even if we memorize at the low-noise part. Our training algorithm can "copy" details from the training samples and still produce diverse outputs.

> **Insight 16**
>
> The key property of Feldman (2020) seems to break when noise is added to the images. That is because different subpopulations start to merge and the heavy-tails of the weights' distribution disappear. Interestingly, diffusion models learn the (score of the) distribution at different levels of noise. This indicates that, in principle, it is feasible to avoid memorization in the high-noise regime (without sacrificing too much quality).

Algorithm The algorithm of Shah et al. (2025) works by splitting the diffusion training time into two parts, $t \leq t_n$ and $t > t_n$, where t_n (t-nature) is a free parameter to be controlled. For the regime, $t \leq t_n$, we train with the regular diffusion training objective, and (assuming perfect optimization) we know the exact score. To train for $t > t_n$, we first create the set S_{t_n} which has *one* noisy version of each image in the training set. Then, we train using the set S_{t_n} and the Ambient Score Matching loss.

> **Insight 17**
>
> It is useful to build some intuition about why this algorithm avoids memorization and at the same time produces high-quality outputs. Regarding memorization: 1) the learned score function for times $t \geq t_n$ does not point directly towards the training points since Ambient Diffusion aims to predict the noisy points (recall that the optimal DDPM solution points towards scalings of the training points) and 2) the noisy versions x_{t_n} are harder to memorize than x_0, since noise is not compressible. At the same time, if the dataset size were to grow to infinity, both our algorithm and the standard diffusion objective would find the true solution: the score of the underlying continuous distribution. In fact,

Algorithm 1 Algorithm for training diffusion models using limited data.

Require: untrained network h_θ, set of samples S, noise level t_n, noise scheduling $\sigma(t)$, batch size B, diffusion time T

$S_{t_\text{n}} \leftarrow \{\sqrt{1-\sigma_{t_\text{n}}^2}x_0^{(i)} + \sigma_{t_\text{n}}\varepsilon^{(i)} | x_0^{(i)} \in S, \varepsilon^{(i)} \sim \mathcal{N}(0, I_d)\}$ ▷ Noise the training set at level t_n.
while not converged **do**
  Form a batch $\mathcal{B}$ of size B uniformly sampled from $S \cup S_{t_\text{n}}$
  loss $\leftarrow 0$ ▷ Initialize loss.
  for each sample $x \in \mathcal{B}$ **do**
    $\varepsilon \sim \mathcal{N}(0, I)$ ▷ Sample noise.
    if $x \in S_{t_\text{n}}$ **then**
      $x_{t_\text{n}} \leftarrow x$ ▷ We are dealing with a noisy sample.
      $t \sim \mathcal{U}(t_\text{n}, T)$ ▷ Sample diffusion time for noisy sample.
      $x_t \leftarrow \sqrt{\frac{1-\sigma_t^2}{1-\sigma_{t_\text{n}}^2}}x_{t_\text{n}} + \sqrt{\frac{\sigma_t^2-\sigma_{t_\text{n}}^2}{1-\sigma_{t_\text{n}}^2}}\varepsilon$ ▷ Add additional noise.
      loss $\leftarrow$ loss $+ \left\| \frac{\sigma_t^2-\sigma_{t_\text{n}}^2}{\sigma_t^2\sqrt{1-\sigma_{t_\text{n}}^2}} h_\theta(x_t, t) + \frac{\sigma_{t_\text{n}}^2}{\sigma_t^2}\sqrt{\frac{1-\sigma_t^2}{1-\sigma_{t_\text{n}}^2}}x_t - x_{t_\text{n}} \right\|^2$ ▷ Ambient Score Matching.
    else
      $x_0 \leftarrow x$ ▷ We are dealing with a clean sample.
      $t \sim \mathcal{U}(0, t_\text{n})$ ▷ Sample diffusion time for clean sample.
      $x_t \leftarrow \sqrt{1-\sigma_t^2}x_0 + \sigma_t\varepsilon$ ▷ Add noise.
      loss $\leftarrow$ loss $+ \|h_\theta(x_t, t) - x_0\|^2$ ▷ Regular Denoising Score Matching.
    end if
  end for
  loss $\leftarrow \frac{\text{loss}}{B}$ ▷ Compute average loss.
  $\theta \leftarrow \theta - \eta\nabla_\theta$loss ▷ Update network parameters via backpropagation.
end while

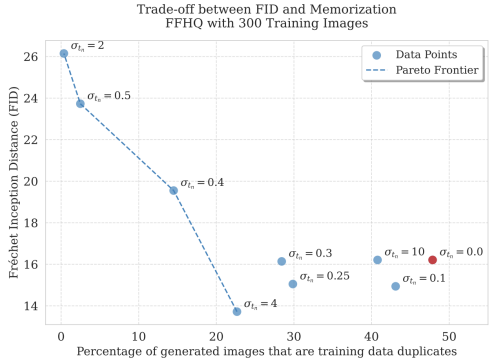


Figure 1: (FID, Memorization) pairs for different values of σ_{t_n} used in our proposed Algorithm 1 (presented in Section 3) for training diffusion models from limited data. The standard DDPM objective corresponds to $\sigma_{t_\text{n}} = 0$ and it is not in the Pareto frontier. Setting σ_{t_n} too low or too high reverts back to the DDPM behavior. Values for $\sigma_{t_\text{n}} \in [0.4, 4]$ strike different balances between memorization and quality of generated images. The models in this Figure are trained on only 300 images from FFHQ.

Figure 15: Taken from Shah et al. (2025). Various instantiations of the algorithm of Shah et al. (2025).

the algorithm learns the same score function for times $t \leq t_\text{n}$ as DDPM. This contributes to generating samples with high-quality details, copied from the training set.

Table 1: FID and Memorization results comparing DDPM and Algorithm 1. Memorization is measured as DINOv2 similarity between generated samples and their nearest training neighbors. We achieve the same or better FID with significantly lower memorization.

		# Train Images					
		300		1k		3k	
		DDPM	Ours	DDPM	Ours	DDPM	Ours
CIFAR-10	FID	25.1	**23.91**	10.46	**10.36**	14.73	**14.26**
	S>0.9	78.96	**44.84**	75.86	**69.08**	53.40	**52.24**
	S>0.925	67.2	**20.22**	57.98	**47.26**	11.92	**11.36**
	S>0.95	56.56	**9.64**	43.44	**26.34**	0.08	**0.06**
FFHQ	FID	16.21	**15.05**	12.26	**11.3**	**6.42**	6.46
	S>0.85	63.38	**49.68**	55.36	**32.08**	21.58	**20.08**
	S>0.875	55.48	**40.01**	43.82	**17.48**	4.98	**4.53**
	S>0.9	47.86	**29.86**	33.92	**7.52**	0.46	**0.42**
ImageNet	FID	——		50.2	**47.19**	40.66	**39.87**
	S>0.9	——		54.72	**26.68**	32.86	**28.40**
	S>0.925	——		41.66	**15.56**	12.32	**9.44**
	S>0.95	——		25.86	**5.54**	6.08	**4.02**

Figure 16: Taken from Shah et al. (2025). Comparison between the above algorithm and standard DDPM.

Lecture 10: A Theory of Learning Curves

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

Lecture 10 presents a setting that classical VC theory fails to explain. We will be interested in *learning curves*. This problem was studied by Bousquet et al. (2021). At a technical level, stability will be crucial in order to obtain one of the key results of the paper.

37 A Theory of Learning Curves

How quickly does the error of a learning algorithm drop for a given class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ of concepts as the number of samples increases?

The standard way to measure the performance of a learning algorithm is by plotting its *learning curve*, i.e., by plotting the decay of the error rate as a function of the number n of training examples. This is also standard in *scaling laws*, when one compares performance by scaling some resource such as training dataset size or model size.

A natural first question is to wonder whether the classical theory of learning, the PAC model of Vapnik-Chervonenkis and Valiant, could be used to explain learning curves. The answer is *no.*

The reason is that the PAC model adopts a *minimax* perspective. Let us denote by $\mathrm{RE}(H)$ the family of distributions P for which the concept class H is realizable, i.e., $\inf_{h \in H} \mathbf{E}_P[\mathrm{er}(h)] = 0$. The fundamental result of PAC learning theory states that

$$\inf_{\hat{h}_n} \sup_{P \in \mathrm{RE}(H)} \mathbf{E}[\mathrm{er}(\hat{h}_n)] \asymp \min\left(\frac{\mathrm{vc}(H)}{n}, 1\right),$$

where $\mathrm{vc}(H)$ is the VC dimension of H.

In other words, PAC learning theory is concerned with the best *worst-case* error over all realizable distributions, that can be achieved by means of a learning algorithm $\hat{h}_n$.

PAC model can only bound the upper envelope of the learning curves over all possible data distributions. Observe that the worst-case distribution can change with the sample size n! However, this is not reflected in real world, where the data source is typically fixed in any given scenario. The learner may choose the number of training examples to use but, as the sample increases, the data source remains the same.

VC theory immediately implies a fundamental *dichotomy* of uniform rates: every concept class H has a uniform rate that is either *linear* $\frac{c}{n}$ or *bounded away from zero*, depending on the finiteness of the combinatorial parameter $\mathrm{vc}(H)$. The term *uniform* comes from the fact that it holds uniformly over all distributions R.

38 Trichotomy of Possible Universal Rates

Bousquet et al. (2021) propose a model for learning curves, called the *universal rates* models. The term *universal* is important and replaces the term uniform: a given property of the learner (such as consistency or rate) should hold for *every* realizable distribution P, but *not* uniformly over all distributions. For example, a class H is universally learnable at rate R if the following holds:

$$\exists\, \hat{h}_n \quad \text{s.t.} \quad \forall\, P \in \mathrm{RE}(H), \quad \exists\, C, c > 0 \quad \text{s.t.} \quad \mathbf{E}[\mathrm{er}(\hat{h}_n)] \leq CR(cn) \text{ for all } n.$$

The crucial difference between universal rates and the uniform PAC setting is that *the constants C, c are allowed to depend on P*: thus universal learning is able to capture *distribution-dependent* learning curves for a given learning problem.

Question. Given a class H, what is the fastest rate at which H can be universally learned?

We will need the following definitions.

Definition 43: Learning Rates

Let H be a concept class, and let $R : \mathbb{N} \to [0, 1]$ with $R(n) \to 0$ be a rate function.

- H is learnable at rate R if there is a learning algorithm $\hat{h}_n$ such that for every realizable distribution P, there exist $C, c > 0$ for which $\mathbf{E}[\text{er}(\hat{h}_n)] \leq CR(cn)$ for all n.
- H is not learnable at rate faster than R if for every learning algorithm $\hat{h}_n$, there exists a realizable distribution P and $C, c > 0$ for which $\mathbf{E}[\text{er}(\hat{h}_n)] \geq CR(cn)$ for infinitely many n.
- H is learnable with optimal rate R if H is learnable at rate R and H is not learnable faster than R.
- H requires arbitrarily slow rates if, for every $R(n) \to 0$, H is not learnable faster than R.

The main result of Bousquet et al. (2021) is a remarkable **trichotomy**: there are only three possible rates of universal learning. More precisely, we show that the learning curves of any given concept class decay either at an exponential, linear, or arbitrarily slow rates. Moreover, each of these cases is completely characterized by appropriate combinatorial parameters, and we exhibit optimal learning algorithms that achieve the best possible rate in each case.

Theorem 40: Trichotomy of Universal Learning Rates

For every concept class H with $|H| \geq 3$, exactly one of the following holds.

- H is learnable with optimal rate e^{-n}.
- H is learnable with optimal rate $\frac{1}{n}$.
- H requires arbitrarily slow rates.

The condition $|H| \geq 3$ avoids some trivially learnable classes.

More to that, one can provide combinatorial measures to characterize when each rate occurs.

Theorem 41: Combinatorial Characterization of Universal Rates

For every concept class H with $|H| \geq 3$, the following hold:

- If H does not have an infinite Littlestone tree, then H is learnable with optimal rate e^{-n}.
- If H has an infinite Littlestone tree but does not have an infinite VCL tree, then H is learnable with optimal rate $\frac{1}{n}$.
- If H has an infinite VCL tree, then H requires arbitrarily slow rates.

Such results have been extended to multiple labels (Kalavasis et al., 2022; Hanneke et al., 2023), real-valued

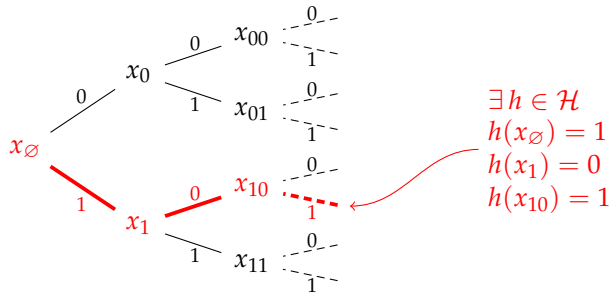


Figure 17: Figure taken from Bousquet et al. (2021). A Littlestone tree of depth 3. Every branch is consistent with a concept $h \in \mathcal{H}$. This is illustrated here for one of the branches. From Bousquet et al. (2021).

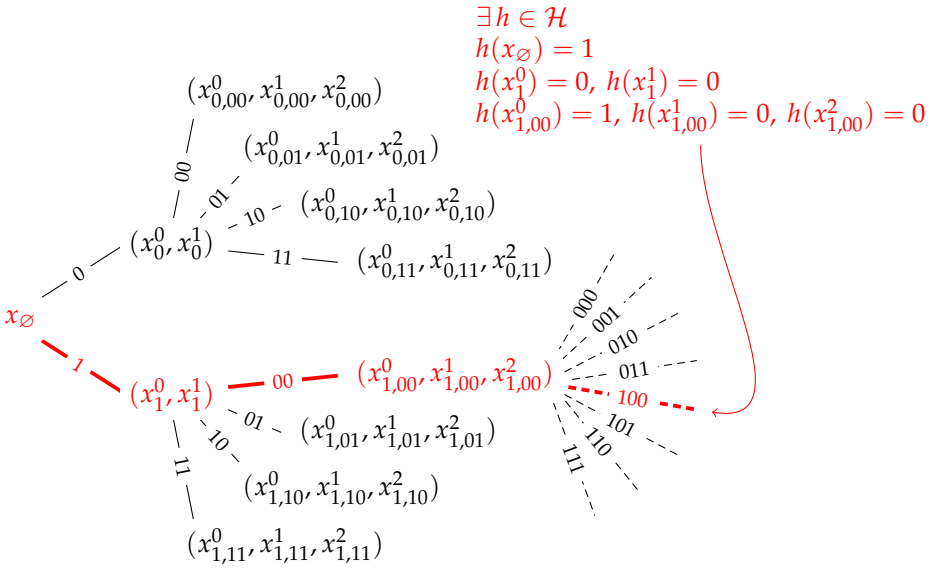


Figure 18: Figure taken from Bousquet et al. (2021). A VCL tree of depth 3. Every branch is consistent with a concept $h \in \mathcal{H}$. This is illustrated here for one of the branches.

regression (Attias et al., 2024), and other settings (see e.g., (Hanneke and Xu, 2024; Kalavasis et al., 2024a; Hanneke and Xu, 2025)).

39 The Exponential Rates Proof - Stability is the Key

The proof of the exponential rates result is pretty surprising and the route one has to cross the get it is highly non-trivial and novel. Stability will be a key to that result.

39.1 The Online Learning Game

Let $\mathcal{X}$ be a domain, and let the concept class $\mathcal{H}$ be a collection of functions $h : \mathcal{X} \to \{0,1\}$. We consider a two-player game between the *learner* P_L and an *adversary* P_A. The game is played in rounds. In each round $t \geq 1$:

- P_A chooses a point $x_t \in \mathcal{X}$.
- P_L predicts a label $\hat{y}_t \in \{0,1\}$ for that point.
- P_A reveals the true label $y_t = h(x_t)$ for some function $h \in \mathcal{H}$ that is consistent with the previous label assignments $h(x_1) = y_1, \ldots, h(x_{t-1}) = y_{t-1}$.

The learner makes a mistake in round t if $\hat{y}_t \neq y_t$. The goal of the learner is to make as few mistakes as possible and the goal of the adversary is to cause as many mistakes as possible.

The adversary need not choose a target concept $h \in H$ in advance, but must ensure that the sequence $\{(x_t, y_t)\}_{t=1}^{\infty}$ is *realizable* by H in the sense that for all $T \in \mathbb{N}$ there exists $h \in H$ such that $h(x_t) = y_t$ for all $t \leq T$.

We will say that $\mathcal{H}$ is online learnable if there is a strategy

$$\hat{y}_t = \hat{y}_t(x_1, y_1, \ldots, x_{t-1}, y_{t-1}, x_t),$$

that makes only finitely many mistakes, regardless of what realizable sequence $\{(x_t, y_t)\}_{t=1}^{\infty}$ is presented by the adversary.

Based on what we have already seen in the previous lectures, this game is quite similar to how we derived the Littlestone dimension. However there is a crucial catch. Here we ask only that the strategy makes a finite number of mistakes on any input, without placing a uniform bound on the number of mistakes.

Theorem 42

For any concept class $\mathcal{H}$, we have the following dichotomy.

- If $\mathcal{H}$ does not have an infinite Littlestone tree, then there is a strategy for the learner that makes only finitely many mistakes against any adversary.
- If $\mathcal{H}$ has an infinite Littlestone tree, then there is a strategy for the adversary that forces any learner to make a mistake in every round.

In particular, $\mathcal{H}$ is online learnable if and only if it has no infinite Littlestone tree.

Question: Is an infinite Littlestone tree and an infinite Littlestone dimension the same? (Think).

39.2 A Gale-Stewart Game

The first surprising step of the proof is the following connection. We view the online learning game from a new perspective that fits to classical game theory. However, classical game theory usually deals with finite games. Here we are playing an *infinite two-player game.*

From a game-theoretic perspective, we can ask: How should an adversary force mistakes to the learning player?

For $x_1, \ldots, x_t \in \mathcal{X}$ and $y_1, \ldots, y_t \in \{0,1\}$, consider the class

$$\mathcal{H}_{x_1,y_1,\ldots,x_t,y_t} := \{h \in \mathcal{H} : h(x_1) = y_1, \ldots, h(x_t) = y_t\}$$

of hypotheses that are consistent with the sequence of moves $x_1, y_1, \ldots, x_t, y_t$. An adversary P_A who aims to maximize the number of mistakes the learner makes will pick a sequence of x_t, y_t with $y_t \neq \hat{y}_t$ for as many rounds in a row as possible.

Using the above notation, the adversary tries to keep $\mathcal{H}_{x_1,1-\hat{y}_1,\ldots,x_t,1-\hat{y}_t} \neq \varnothing$ as long as possible. When this set would become empty (for every possible x_t), however, the only consistent choice of label is $y_t = \hat{y}_t$, so the learner makes no mistakes from that point onwards.

We can define the following game $\mathfrak{G}$. There are two players P_A and P_L. In each round τ:

- Player P_A chooses a point $\xi_\tau \in \mathcal{X}$ and shows it to Player P_L.
- Then, Player P_L chooses a point $\eta_\tau \in \{0,1\}$.

Player P_L wins the game in round τ if $\mathcal{H}_{\xi_1,\eta_1,\ldots,\xi_\tau,\eta_\tau} = \varnothing$. Player P_A wins the game if the game continues indefinitely. In other words, the set of winning sequences for P_L is

$$\mathsf{W} = \{(\boldsymbol{\xi}, \boldsymbol{\eta}) \in (\mathcal{X} \times \{0,1\})^\infty : \mathcal{H}_{\xi_1,\eta_1,\ldots,\xi_\tau,\eta_\tau} = \varnothing \text{ for some } 0 \leq \tau < \infty\}$$

This set of sequences W is *finitely decidable* in the sense that the membership of $(\boldsymbol{\xi}, \boldsymbol{\eta})$ in W is witnessed by a *finite subsequence*. Games that satisfy this property are called *Gale-Stewart (GS) games*. These games come with a fundamental property:

exactly one of P_A or P_L has a winning strategy in this game.

Let us now go back to the online game of the previous section. The game $\mathfrak{G}$ is fundamentally related to Littlestone trees:

$$P_A \text{ has a winning strategy in the Gale-Stewart game } \mathfrak{G} \iff \mathcal{H} \text{ has an infinite Littlestone tree.}$$

This is interesting but how is it helpful to our goal? The main challenge in the proof is constructing a learning algorithm that achieves exponential rate for every realizable P. However, up to now, we only deal with the online setting where there is no distribution P.

Let us assume in the remainder that $\mathcal{H}$ has no infinite Littlestone tree. Winning strategies of Gale-Stewart games for P_L essentially yield the existence of a sequence of universally measurable functions $\hat{Y}_t : (\mathcal{X} \times \{0,1\})^{t-1} \times \mathcal{X} \to \{0,1\}$ that solve the online learning problem. We will use these winning strategies as classifiers for the statistical setting.

Define the data-dependent classifier

$$\hat{y}_{t-1}(x) := \hat{Y}_t(X_1, Y_1, \ldots, X_{t-1}, Y_{t-1}, x).$$

The first observation is that this online algorithm is also applicable in the statistical setting. In particular, one can show that

$$\mathbf{P}\{\mathrm{er}(\hat{y}_t) > 0\} \to 0 \text{ as } t \to \infty.$$

This certainly shows that $\mathbf{E}[\mathrm{er}(\hat{y}_t)] \to 0$ as $t \to \infty$. Thus the online learning algorithm yields a consistent algorithm in the statistical setting. This is how we used stability (i.e., online learnability) to get a consistent learning algorithm in the probabilistic setting.

This, however, does not yield any bound on the learning rate. It could be the case that the error of $\hat{y}_t$ goes to 0 quite slowly.

The idea now is to *boost* the rate of this algorithm. We will build a new algorithm on the basis of $\hat{y}_t$ that guarantees an exponential learning rate.

As a first observation, suppose we knew a number t^* so that $\mathbf{P}\{\mathrm{er}(\hat{y}_{t^*}) > 0\} < \frac{1}{4}$. This t^* exists and is finite (why?).

Then we could output a classifier $\hat{h}_n$ with exponential rate; the idea is just classical boosting argument.

- Break up the data $X_1, Y_1, \ldots, X_n, Y_n$ into $\lfloor n/t^* \rfloor$ batches, each of length t^*.
- Train a classifier $\hat{y}_{t^*}$ separately for each batch.
- Choose $\hat{h}_n$ to be the majority vote among these classifiers.

Now, by the definition of t^* and Hoeffding's inequality, the probability that more than $1/3$ of the classifiers has positive error is exponentially small! It follows that the majority vote $\hat{h}_n$ errs only with exponentially small probability, meaning that its learning curve drops exponentially fast. Note that this t^* depends on P and so the constants in the rate depend on P.

This last comment is exactly the problem with this idea: t^* depends on the unknown distribution P, so we cannot assume it is known to the learner. However, there is a smart fix to that: first, we will construct an estimate $\hat{t}_n$ for t^* from the data; and then we apply the above majority algorithm with batch size $\hat{t}_n$ instead t^*.

The estimation of t^* can be done using samples as follows:

Lemma 4.4. *There exist universally measurable $\hat{t}_n = \hat{t}_n(X_1, Y_1, \ldots, X_n, Y_n)$, whose definition does not depend on P, so that the following holds. Given t^* such that*

$$\mathbf{P}\{\mathrm{er}(\hat{y}_{t^*}) > 0\} \le \tfrac{1}{8},$$

there exist $C, c > 0$ independent of n (but depending on P, t^*) so that

$$\mathbf{P}\{\hat{t}_n \in \mathcal{T}_{\mathrm{good}}\} \ge 1 - Ce^{-cn},$$

where

$$\mathcal{T}_{\mathrm{good}} := \{1 \le t \le t^* : \mathbf{P}\{\mathrm{er}(\hat{y}_t) > 0\} \le \tfrac{3}{8}\}.$$

Proof For each $1 \le t \le \lfloor \frac{n}{2} \rfloor$ and $1 \le i \le \lfloor \frac{n}{2t} \rfloor$, let

$$\hat{y}_t^i(x) := \hat{Y}_{t+1}(X_{(i-1)t+1}, Y_{(i-1)t+1}, \ldots, X_{it}, Y_{it}, x)$$

be the learning algorithm from Section 3.1 that is trained on batch i of the data. For each t, the classifiers $(\hat{y}_t^i)_{i \le \lfloor n/2t \rfloor}$ are trained on subsamples of the data that are independent of each other and of the second half $(X_s, Y_s)_{s > n/2}$ of the data. Thus $(\hat{y}_t^i)_{i \le \lfloor n/2t \rfloor}$ may be viewed as independent draws from the distribution of $\hat{y}_t$. We now estimate $\mathbf{P}\{\mathrm{er}(\hat{y}_t) > 0\}$ by the fraction of $\hat{y}_t^i$ that make an error on the second half of the data:

$$\hat{e}_t := \frac{1}{\lfloor n/2t \rfloor} \sum_{i=1}^{\lfloor n/2t \rfloor} \mathbf{1}_{\{\hat{y}_t^i(X_s) \ne Y_s \text{ for some } n/2 < s \le n\}}.$$

Figure 19: Taken from Bousquet et al. (2021). Estimation of the critical time where the algorithms stops making mistakes with some constant probability $> 1/2$.

40 The Landscape of Universal Rates

In this section we provide further examples. The main aim of this section is to illustrate important distinctions with the uniform setting and other basic concepts in learning theory, which are illustrated schematically in Figure 20.

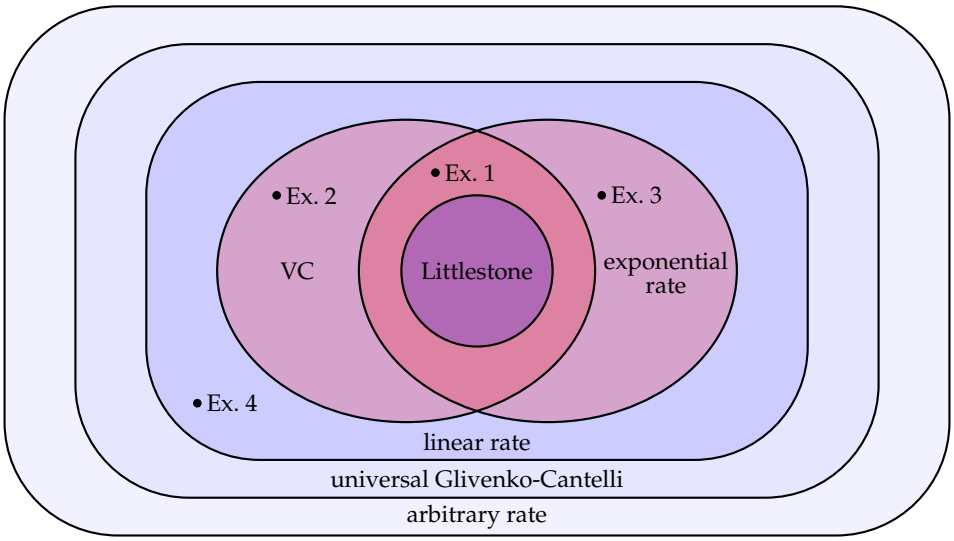


Figure 20: Figure taken from Bousquet et al. (2021). A Venn diagram depicting the trichotomy and its relation with uniform and universal learnability. While the focus here is on statistical learning, note that this diagram also captures the distinction between uniform and universal online learning.

We begin by giving four examples that illustrate that the classical PAC learning model (which is characterized by finite VC dimension) is not comparable to the universal learning model.

Example 3: VC with exponential rate

Consider the class $\mathcal{H} \subseteq \{0,1\}^{\mathbb{N}}$ of all threshold functions $h_t(x) = \mathbf{1}_{x \geq t}$ where $t \in \mathbb{N}$.

1. It is a VC class with VC dimension 1.
2. It is learnable at an exponential rate since it does not have an infinite Littlestone tree.
3. It 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.

Example 4: VC with linear rate

Consider the class $\mathcal{H} \subseteq \{0,1\}^{\mathbb{R}}$ of all threshold functions $h_t(x) = \mathbf{1}_{x \geq t}$, where $t \in \mathbb{R}$.

1. It is a VC class with VC dimension 1.
2. It is not learnable at an exponential rate (it has an infinite Littlestone tree).
3. The optimal rate is linear.

Example 5: Exponential rate but not VC

Let $\mathcal{X} = \bigcup_k \mathcal{X}_k$ be the disjoint union of finite sets $|\mathcal{X}_k| = k$. For each k, let $\mathcal{H}_k = \{\mathbf{1}_S : S \subseteq \mathcal{X}_k\}$, and consider the concept class $\mathcal{H} = \bigcup_k \mathcal{H}_k$.

1. This class has an unbounded VC dimension.
2. It is universally learnable at an exponential rate. To establish the latter, it suffices to prove that $\mathcal{H}$ does not have an infinite Littlestone tree. Indeed, once we fix any root label $x \in \mathcal{X}_k$ of a Littlestone tree, only $h \in \mathcal{H}_k$ can satisfy $h(x) = 1$, and so the hypotheses consistent with the subtree corresponding to $h(x) = 1$ form a finite class. This subtree can therefore have only finitely many leaves, contradicting the existence of an infinite Littlestone tree.

Example 6: Linear rate but not VC

Assume that $\mathcal{X}$ is the disjoint union of $\mathbb{R}$ and finite sets $\mathcal{X}_k$ with $|\mathcal{X}_k| = k$, and $\mathcal{H}$ is the union of the class of all threshold functions on $\mathbb{R}$ and the classes $\mathcal{H}_k = \{\mathbf{1}_S : S \subseteq \mathcal{X}_k\}$.

1. This class has an unbounded VC dimension.
2. It is universally learnable at a linear rate. To establish the latter, it suffices to note that $\mathcal{H}$ has an infinite Littlestone tree, but $\mathcal{H}$ cannot have an infinite VCL tree. Indeed, once we fix any root label $x \in \mathcal{X}$, the class $\{h \in \mathcal{H} : h(x) = 1\}$ has finite VC dimension, and thus the corresponding subtree of the VCL tree must be finite.

Lecture 11: Language Identification and Generation

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

In Lecture 11, we present the Gold-Angluin online model for language identification in the limit (Gold, 1967; Angluin, 1980). Then we will discuss the online model for language generation in the limit by Kleinberg and Mullainathan (2024).

41 Countable Sets and Enumerations

Countable sets will be very important for this Lecture. So let us recall some basic examples.

1. Finite sets, naturals, integers, rationals are countable
2. Given a finite alphabet (like the letters of the English alphabet), the set of all finite-length words that can be formed using that alphabet is countable.
3. The set of all finite subsets of natural numbers: Each finite subset can be represented by a binary sequence, making this set countable.
4. (Assuming the axiom of countable choice) The union of countably many countable sets is countable.

Let us now recall some standard uncountable sets.

1. Cantor's Theorem implies that the powerset of $\mathbb{N}$ is uncountable.
2. The set of real numbers is uncountable.

Given a countable set S, an *enumeration* of S is a listing of all of its elements with the property that each $x \in S$ appears in a fixed finite position. For instance, the standard ordering of $\mathbb{N}$ is an enumeration since any x appears in the x-th position but the list $1, 3, 5, 7, ..., 2, 4, 6, ...$ is not an enumeration.

42 Language Identification in the Limit

The problem of language identification in the limit from positive examples was introduced by Gold (1967) and further studied by Angluin (1979, 1980). The setting is specified by a countable collection of languages $\mathcal{L} = \{L_1, L_2, \dots\}$. For a fixed collection $\mathcal{L}$, an adversary and an identifier play the following game: The adversary chooses a language K from $\mathcal{L}$ without revealing it to the identifier, and it begins *enumerating* the strings of K (potentially with repetitions) $x_1, x_2, \dots$ over a sequence of time steps $t = 1, 2, 3, \dots$. The adversary can repeat strings in its enumeration, but the crucial point is that for every string $x \in K$, there must be at least one finite time step t at which it appears.

At each time t, the identification algorithm I, given the previous examples $x_1, x_2, \dots, x_t$, outputs an index i_t that corresponds to its guess for the true language K.

Definition 44: Language Identification in the Limit (Gold, 1967)

Fix some language K from the language collection $\mathcal{L}$. The identification algorithm I identifies K in the limit if there is some $t^* \in \mathbb{N}$ such that for all steps $t > t^*$, the identifier's guess i_t satisfies $i_t = i_{t-1}$

and $L_{i_t} = K$. The language collection $\mathcal{L}$ is identifiable in the limit if there is an identifier that identifies in the limit any $K \in \mathcal{L}$, for any enumeration of K.

I wish to construct a precise model for the intuitive notion "able to speak a language" in order to be able to investigate theoretically how it can be achieved artificially. Since we cannot explicitly write down the rules of English which we require one to know before we say he can "speak English," an artificial intelligence which is designed to speak English will have to learn its rules from implicit information. That is, its information will consist of examples of the use of English and/or of an informant who can state whether a given usage satisfies certain rules of English, but cannot state these rules explicitly.

For the purpose of artificial intelligence, a model of the rules of usage of natural languages must be general enough to include the rules which do occur in existing natural languages. This is a lower bound on the generality of an acceptable linguistic theory. On the other hand, the considerations of the last paragraph impose an upper bound on generality: For any language which can be defined within the model there must be a training program, consisting of implicit information, such that it it possible to determine which of the definable languages is being presented.

Therefore this research program consists of the study of two subjects: Linguistic structure and the learnability of these structures. This report describes the first step of this program. A very naive model of language is assumed, namely, a language is taken to be a distinguished set of strings. Such a language is too simple to do anything with (for instance, to give information or to pose problems), but it has enough structure to allow its learnability to be investigated as follows: Models of information presentation are defined, and for each I ask "For which classes of languages does a learning algorithm exist?"

Figure 21: Taken from Gold (1967). Gold's motivation for the model.

Gold (1967) showed that collections of finite cardinality languages, i.e., each language in the collection $\mathcal{L}$ is finite, can be identified in the limit from positive examples. This is true since in the limit, one will see all the elements of the target (finite) language, at which point it can be identified. The identification algorithm is the following: at time t, guess L to consist solely of the elements that have occurred in the sequence. Since L is finite, there will be a finite time after which all elements of L will have been revealed, so after that the algorithm will have identified the target.

A super-finite collection of languages denotes any collection which contains all languages of finite cardinality and at least one of infinite cardinality. Gold showed that super-finite collections of languages cannot be identified in the limit from positive examples. Further, he showed that negative examples help: any super-finite collection can be identified in the limit using positive and negative examples[10] (the idea is simple: keep guessing the infinite language until seeing a negative example; then it reduces to the finite case).

Theorem 43: (Gold, 1967)

Let $\mathcal{L} = \{L_\infty, L_1, L_2, \dots\}$ be the language collection with $L_1 \subset L_2 \subset \cdots \subset L_\infty = \cup_{i \geq 1} L_i$ and for each i, $|L_i| < \infty$. Then, there is no algorithm that identifies $\mathcal{L}$ in the limit from positive examples. Moreover, this collection can be identified in the limit when the algorithm has access to both positive and negative examples.

[10] This means that the adversary presents an enumeration of the whole domain $\mathcal{X}$, with a label indicating whether the example is in the target language.

For instance, let $L_i = [i]$ and $L_\infty = \mathbb{N}$.

Exercise 11

Show that the above language collection is not identifiable in the limit.

The above result already shows a separation in terms of identification between observing only positive examples and observing positive *and* negative examples in Gold's model. Moreover, it raises the question of whether there exist non-trivial collections of languages identifiable in the limit from positive examples. In that direction, Angluin (1979) studied pattern languages (whose definition is not important for our work) and showed that for that collection identification in the limit is possible only with positive examples.

Before proceeding with the characterization result, there are two important aspects that we have not paid extensive attention:

1. How does the algorithms access the collection $\mathcal{L}$? Usually, the only type of access the algorithm has is *membership oracle access*, i.e., given element x and index i, the oracle returns $1\{x \in L_i\}$. For instance, the algorithm for the finite cardinality languages works with just that type of access. However, it could e.g., be the case that the algorithm uses some other type of oracle, e.g., a *subset oracle*, where given indices i, j, the oracle returns $1\{L_i \subseteq L_j\}$.
2. Can the algorithm be collection-dependent? For instance, if the collection has some particular property, can the algorithm make use of it? As we will see, e.g., the algorithm of Kleinberg and Mullainathan (2024) is agnostic to the collection; it only uses membership oracles.

Let us consider another example. Let $\mathcal{L}$ be some *finite* collection of languages. Is this identifiable in the limit? Let $\mathcal{L} = \{L_1, L_2, ..., K, ..., L_T\}$ be the finite collection and let K be the unknown true language. Which languages can we eventually exclude easily?

For any $i \in [T]$ with $K \setminus L_i \neq \emptyset$, it means that eventually the algorithm will observe a point in K that is not in L_i (can check it with membership oracle); hence, in finite time, this language will become inconsistent.

However, we are not done: there is some possibility that $K \subset L_i$. In that case, any algorithm that starts predicting L_i will never exclude it since any element of K also belongs to L_i. If the algorithm could output from the minimal consistent, then we would be fine. For instance, if the algorithm is collection-dependent, then the pairwise subset relations can be encoded in the algorithm.

The next question is whether one can get a *characterization* of the language collections that can be identified from positive examples. Angluin (1980) resolved this problem.

Definition 45: Angluin's Condition (Angluin, 1980)

Fix a language collection $\mathcal{L} = \{L_1, L_2, \dots\}$. Suppose there is a membership oracle which, given a string x and index i, answers $1\{x \in L_i\}$. The collection $\mathcal{L}$ is said to satisfy Angluin's condition if there is a *tell-tale* oracle that given an index i enumerates a set of *finite* strings T_i such that

$$T_i \subseteq L_i \text{ and for all } j \geq 1, \text{ if } T_i \subseteq L_j \text{ then } L_j \text{ is not a proper subset of } L_i.$$

The difficulty in trying to identify a language from positive examples is the problem of *over-generalization*. If while seeing positive examples the algorithm specifies a language that is a proper superset of the true answer K, then by only seeing positive examples it will never see a counterexample to that language. This would be avoided with positive and negative examples. Angluin's condition essentially ensures this over-generalization problem can be avoided by from just positive examples (without the help of negative examples).

Before proceeding to Angluin's result, we stress one important point: inspecting Angluin's definition, we can see that it requires access to a procedure that *finds* this set of strings T_i. This oracle is called a *tell-tale*

oracle and is quite crucial for Angluin's algorithm to work.

Definition 42 led to the following characterization.

Theorem 44: (Angluin, 1980)

A countable language collection $\mathcal{L}$ is identifiable in the limit if and only if it satisfies Angluin's criterion.

Let us see why Angluin's condition is sufficient for identification in the limit. Assume that $\mathcal{L}$ satisfies Angluin's criterion.

For any $t \geq 1$:

1. Get x_t and let S_t be the training set observed until time t.
2. Find the least positive integer $g \leq t$ (if any) such that
   - $S_t \subseteq L_g$
   - $T_g^{(t)} \subseteq S_t$, where $T_g^{(t)}$ is the set of strings produced in the first t steps of the enumeration of T_g.
3. If no such index g exists, output a random index and go to the next iteration.
4. Otherwise output g.

To see why this works, let k be an index and L_k the true language. Let $s_1, s_2, ...$ be an enumeration of L_k. Let m be the index of the first appearance of L_k in the enumeration of $\mathcal{L}$. For each $0 < i < m$, either

1. $L_m \setminus L_i \neq \emptyset$. This means that there is an element in the enumeration that does not belong to L_i. Hence, there is a finite time that L_i will be rejected. Say n_i that time, which is associated with L_i.
2. If the above is not true, then $L_m \subset L_i$. In general, this case is hard when only positive examples are available, because L_i cannot be rejected. But, here is where Angluin's condition comes in. Let us pick n_i sufficiently large such that $T_i^{(n_i)} = T_i$ for all $t \geq n_i$.

Let n_m be sufficiently large such that $T_m \subseteq S_t$ for all $t \geq n_m$. Let $N = \max\{n_1, n_2, ..., n_m, m\}$. Now for all $t \geq N$, we have $m \leq t, S_t \subseteq L_m$, and $T_m \subseteq S_t$.

Now if for $i < m$ and L_i does not belong to Case 1, then $L_m \subset L_i$, then T_i cannot be a subset of L_m, and so $T_i^{(t)}$ is not a subset of L_m. Hence, the tell-tale for L_i serves a counterexample for L_i. This means that the index converges to m and stabilizes at that point.

Finally, let us consider the case of language identification with both positive and negative examples, i.e., when the adversary provides an enumeration of the whole domain $\mathcal{X}$ and every example has a label indicating whether it is in the true language K. We mention that focusing on algorithms equipped with membership oracle, the following result appears in Gold (1967).

Theorem 45: (Gold, 1967)

Any countable language collection is identifiable in the limit from positive and negative examples.

To see how the algorithm works, let $\mathcal{L} = \{L_1, L_2, \ldots\}$ and denote by L_z the smallest indexed language in $\mathcal{L}$ for which $L_z = K$. The algorithm observes an enumeration of the form $(x_t, y_t) \in \times\{0,1\}$ for $t \geq 1$. Recall this means that $1\{x_t \in K\} = y_t$. The algorithm works as follows: in every timestep $t \in \mathbb{N}$, it predicts the lowest index of a consistent language, i.e., the smallest $j \in \mathbb{N}$ for which $1\{x_\tau \in L_j\} = y_\tau$ for all $\tau \leq t$. Consider two cases: if $z = 1$, then the algorithm will never predict any language $L_{z'}, z' \geq 2$, so it will be correct from the first step. If $z > 1$, then for all $L_{z'}, z' < z$, that come before L_z in the enumeration of $\mathcal{L}$, there is a finite time $t_{z'}$ when the example $(x_{t_{z'}}, y_{t_{z'}})$ contradicts the language $L_{z'}$.

43 Language Generation in the Limit

We now move to language generation in the limit from positive examples, introduced by Kleinberg and Mullainathan (2024). The setup is exactly the same as in the Gold-Angluin model (the adversary provides an enumeration of K), but now the goal of the learner is to *generate unseen examples* from K instead of identifying the index of K. Their formal definition is the following.

Definition 46: Language Generation in the Limit (Kleinberg and Mullainathan, 2024)

Fix some language K from the collection $\mathcal{L} = \{L_1, L_2, \dots\}$ and a generating algorithm $\mathcal{G}$. At each step t, let $S_t \subseteq K$ be the set of all strings that the algorithm $\mathcal{G}$ has seen so far. $\mathcal{G}$ must output a string $x_t \notin S_t$ (its guess for an unseen string in K). The algorithm $\mathcal{G}$ consistently generates from K in the limit if, for all enumerations of K, there is some $t^* \in \mathbb{N}$ such that for all steps $t \geq t^*$, the algorithm's guess a_t belongs to $K \setminus S_t$. The collection $\mathcal{L}$ allows for consistent generation in the limit if there is an algorithm $\mathcal{G}$ that, for any choice of the target language $K \in \mathcal{L}$, it consistently generates from K in the limit.

Definition 43 straightforwardly generalizes to randomized algorithms; consider the same setup as before except that now the output string a_t may be randomized. The definition of generation is also the same except that instead of requiring $a_t \in K \setminus S_t$ one requires that the support A_t of the distribution from which a_t is sampled is non-empty and satisfies $A_t \subseteq K \setminus S_t$.

Insight 18

One way to think of the elements of some language L is as valid *sentences* and not just words. In this way, the above model aims to abstract the whole training process of LLMs. In the real world, first a huge collection of texts (say Wikipedia articles) are collected; these correspond to the elements of the true language. Then in practice, the way that pre-training is applied is via next-token prediction: each article is separated into tokens, and the model is trained to predict the next token given the previous one. In the generation phase, where the weights are frozen, the model also generates in a next-token based manner: given an input prompt, it samples the next token and then recursively generates the whole sentence (which is another element from the language in our formulation). In this way, the above model abstracts all these steps and simply gets training samples (which are sentences from a language) and aims to learn how to generate new sentences from that language.

Observe that language generation requires that the algorithm's outputs are *consistent* with K (in the limit), but allows the algorithm to not generate certain strings from K. For instance, if K is the set of all strings, then the algorithm that always outputs even length strings (not in S_t), generates from K in the limit but also misses infinitely many strings in K (namely, all strings of odd length). Consistency is clearly a desirable notion: without consistency, algorithms may keep outputting strings outside the target language K which, when K is the set of all meaningful and true strings, inevitably leads to hallucinations.

A trivially consistent generator is one that outputs data already seen in the training set. As we already mentioned, we count such outputs as mistakes. This form of predicting unseen positive examples makes the task of generation interesting. At first sight, it seems that there is an easy strategy that achieves generation in the limit: given an enumeration of all hypotheses $L_1, L_2, \dots$, we sequentially generate from L_i $(i = 1, 2, \dots)$ until it becomes inconsistent with the sample S_n; then we move to L_{i+1}. This strategy seems natural for generation because we know that there is some index k such that the true language $K = L_k$. This idea has a fundamental issue, already reported by Kleinberg and Mullainathan (2024): if there exists an index i such that $i < k$ and $L_k \subsetneq L_i$, then the generator will get stuck at L_i and never update.

43.1 Pick the First Consistent

In thinking about approaches to generation in the limit, there is a natural strategy that at first seems to solve the problem directly, but in fact does not work. Its failure is useful to discuss, since it motivates the more involved solution that follows.

The strategy is to move through the list of languages $\mathcal{L} = \{L_1, L_2, L_3, \ldots\}$ in order, treating each language L_i as a hypothesis for K until the sample S_t proves otherwise. That is, we start with L_1, and we generate strings from $L_1 - S_t$ until we encounter (if ever) a step t in which S_t is not a subset of L_1. At this point we know that L_1 cannot be the true language K, and so we continue the process with L_2.

The nice idea that underpins this strategy is that the true language K is equal to L_z for some index z. So if our process were to reach L_z at some step t_*, it would never move on from L_z, and so we would be generating from $K - S_t$ for all $t \geq t_*$.

Unfortunately, there is a deep issue with this approach: there may be a language $L_i \in \mathcal{L}$ with the property that L_i comes before L_z and L_i properly contains L_z (that is, $i < z$, and $L_z \subsetneq L_i$). In this case, our procedure would stop at the first such L_i forever: since it is only ever shown samples in S_t that come from the language L_z, and since $L_z \subseteq L_i$, it would never encounter a string in S_t that didn't belong to L_i, and so it we would never move on to L_{i+1}. And when this procedure generated from $L_i - S_t$, there is no guarantee that it would choose strings from L_z.

Note that if this approach worked, we could also identify in the limit. This is exactly where Angluin's criterion was used.

The important observation is that if the algorithm is maintaining hypotheses for the true language K over time, it can provably *never know whether its current hypothesis is correct*; instead, it must be always moving further down the collection of languages, potentially considering languages that are not K, but in such a way that it is eventually always generating from $K - S_t$.

43.2 The Algorithm

A non-trivial solution to this problem was given by Kleinberg and Mullainathan (2024). They show that all countable sets of languages in countable domains allow for generation in the limit from positive examples; this is in stark contrast with identification in the limit from positive examples.

Theorem 46: Theorem 1 in Kleinberg and Mullainathan (2024)

There is an algorithm with the property that for any countable collection of languages $\mathcal{L} = \{L_1, L_2, \ldots\}$, any target language $K \in \mathcal{L}$, and any enumeration of one of these languages K, the algorithm generates from K in the limit with positive examples.

We say that a language L_i is consistent with the sample S_t at time t if S_t is contained in L_i.

Definition 47

A language L_n is critical at step t if L_n is consistent with S_t, and for every language $L_i \in \{L_1, ..., L_n\}$ that is consistent with S_t, we have $L_n \subseteq L_i$.

Note that at any time t, there is at least one critical language: the language with the smallest index that is consistent with S_t.

1. Let $\mathcal{L} = \{L_1, L_2, ...\}$ be the enumeration of the language class. For any step $t \geq 1$, we proceed as follows:
2. Let $\{L_1, ..., L_t\}$ be the first t languages.

3. Given the examples S_t until time t, identify all the critical languages in $\{L_1, ..., L_t\}$ at time t.
4. Among those, let L_{n_t} be the critical language with the largest index ($n_t \leq t$).
5. Output some element from $L_{n_t} \setminus S_t$.

We now provide some intuition on how this algorithm works. Let $L_1, L_2, \ldots$ be an enumeration of the collection of languages and K be the true language. Let z be an index such that $L_z = K$. Now assume that we have two languages L_i and L_j with $L_i \subseteq L_j$ which are both consistent with S_t. Then, it is clear that the generating algorithm should prefer to generate from L_i rather than L_j: any $w \in L_i \setminus S_t$ satisfies $w \in L_j \setminus S_t$. This property inspired Kleinberg and Mullainathan (2024) to define the notion of a *critical language*, which we introduced before. Let $C_n = \{L_1, L_2, \ldots, L_n\}$.

Recall that a language L_n is critical at step t if L_n is consistent with S_t and for every $L_i \in C_n$ that is consistent with S_t, it must be $L_n \subseteq L_i$. There are some key properties upon which the generating algorithm is built:

- At any time, there is at least one language consistent with S_t, the true one $L_z = K$. Also, there is at least one critical language at any step t: for any t, the consistent language L_i with the lowest index i must be critical at step t, as it is the only consistent language in C_i.
- There exist times t for which L_z (which is K) is not critical. But eventually, L_z will become critical at some step and then remain critical forever after that. Also, any critical language coming after L_z must be a subset of L_z, thus it is safe to generate from it.
- Hence the algorithm, roughly speaking, keeps track of a list of critical languages and generates from the last one in the list; this is because, after some finite index, all the critical languages are subsets of L_z and, hence, it is safe to generate from any of them.

We remark that the above algorithm requires, apart from membership oracle access, access to an oracle that checks where $L_i \subseteq L_j$?. This is called a subset oracle. A natural question is whether a generation in the limit algorithm can be solely based on membership queries. The answer is affirmative. One can relax the notion of criticality by considering *projections* of languages:

Definition 48

Let t and m be positive integers. A language L_n is (t, m)-critical if L_n is consistent with S_t, and for every language $L_i \in \{L_1, ..., L_n\}$ such that L_i is consistent with S_t, we have $L_n[m] \subseteq L_i[m]$.

Here $L[m]$ are the first m elements in the enumeration of L. Note that in the above, as m increases, (t, m)-criticality tends to become the actual t-critical definition, since larger and larger projections of the languages are considered.

The idea is again to generate a new string from the highest-indexed critical language. The difficulty now is that we have to find a new string. This is also doable, see Kleinberg and Mullainathan (2024) for details.

Related Works. Soon after the work of Kleinberg and Mullainathan (2024), there has been exciting progress on the topic of language generation (see e.g., (Charikar and Pabbaraju, 2024; Kalavasis et al., 2024b; Li et al., 2024; Peale et al., 2025; Kleinberg and Wei, 2025)). We refer to `https://languagegeneration.github.io/` for recent updates.

References

Alon, N., Bun, M., Livni, R., Malliaris, M., and Moran, S. (2022). Private and online learnability are equivalent. *J. ACM*, 69(4).

Alon, N., Livni, R., Malliaris, M., and Moran, S. (2019). Private pac learning implies finite littlestone dimension. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 852–860.

Alon, N., Moran, S., Schefler, H., and Yehudayoff, A. (2024). A unified characterization of private learnability via graph theory. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 94–129. PMLR.

Angel, O. and Spinka, Y. (2019). Pairwise optimal coupling of multiple random variables. *arXiv preprint arXiv:1903.00632*.

Angluin, D. (1979). Finding patterns common to a set of strings (extended abstract). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, page 130–141, New York, NY, USA. Association for Computing Machinery.

Angluin, D. (1980). Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. (2018). Stronger generalization bounds for deep nets via a compression approach. In *International conference on machine learning*, pages 254–263. PMLR.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. (2017). A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR.

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.

Attias, I., Hanneke, S., Kalavasis, A., Karbasi, A., and Velegkas, G. (2024). Universal rates for regression: Separations between cut-off and absolute loss. In Agrawal, S. and Roth, A., editors, *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pages 359–405. PMLR.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. (2017). Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30.

Bassily, R., Moran, S., Nachum, I., Shafer, J., and Yehudayoff, A. (2018). Learners that use little information. In Janoos, F., Mohri, M., and Sridharan, K., editors, *Proceedings of Algorithmic Learning Theory*, volume 83 of *Proceedings of Machine Learning Research*, pages 25–55. PMLR.

Bassily, R., Nissim, K., Smith, A., Steinke, T., Stemmer, U., and Ullman, J. (2016). Algorithmic stability for adaptive data analysis. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 1046–1059, New York, NY, USA. ACM.

Bavarian, M., Ghazi, B., Haramaty, E., Kamath, P., Rivest, R. L., and Sudan, M. (2016). Optimality of correlated sampling strategies. *arXiv preprint arXiv:1612.01041*.

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

Beimel, A., Nissim, K., and Stemmer, U. (2013b). Private learning and sanitization: Pure vs. approximate differential privacy. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 363–378. Springer.

Belkin, M., Hsu, D. J., and Mitra, P. (2018). Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. *Advances in neural information processing systems*, 31.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2006). Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19.

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

Benton, J., Bortoli, V., Doucet, A., and Deligiannidis, G. (2024). Nearly d-linear convergence bounds for diffusion models via stochastic localization.

Biroli, G., Bonnaire, T., De Bortoli, V., and Mézard, M. (2024). Dynamical regimes of diffusion models. *Nature Communications*, 15(1):9957.

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.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965.

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.

Bousquet, O., Hanneke, S., Moran, S., and Zhivotovskiy, N. (2020). Proper learning, helly number, and an optimal svm bound. In *Conference on Learning Theory*, pages 582–609. PMLR.

Brown, G., Bun, M., Feldman, V., Smith, A., and Talwar, K. (2021). When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing*, pages 123–132.

Bun, M., Gaboardi, M., Hopkins, M., Impagliazzo, R., Lei, R., Pitassi, T., Sivakumar, S., and Sorrell, J. (2023). Stability is stable: Connections between replicability, privacy, and adaptive generalization. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 520–527, New York, NY, USA. Association for Computing Machinery.

Bun, M., Livni, R., and Moran, S. (2020). An equivalence between private classification and online prediction. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 389–402. IEEE.

Bun, M., Nissim, K., Stemmer, U., and Vadhan, S. (2015). Differentially private release and learning of threshold functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 634–649. IEEE.

Bun, M. and Steinke, T. (2016). Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer.

Carbery, A. and Wright, J. (2001). Distributional and lq norm inequalities for polynomials over convex bodies in rn. *Mathematical research letters*, 8(3):233–248.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., and Wallace, E. (2023). Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., et al. (2021). Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650.

Charikar, M. and Pabbaraju, C. (2024). Exploring facets of language generation in the limit. *arXiv preprint arXiv:2411.15364*.

Chase, Z., Chornomaz, B., Hanneke, S., Moran, S., and Yehudayoff, A. (2024). Dual vc dimension obstructs sample compression by embeddings. *arXiv preprint arXiv:2405.17120*.

Chen, L., Lu, Z., Oliveira, I. C., Ren, H., and Santhanam, R. (2023). Polynomial-time pseudodeterministic construction of primes. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1261–1270. IEEE.

Chen, S., Chewi, S., Li, J., Li, Y., Salim, A., and Zhang, A. R. (2022). Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. *arXiv preprint arXiv:2209.11215*.

Chewi, S., Kalavasis, A., Mehrotra, A., and Montasser, O. (2025). Ddpm score matching and distribution learning. *arXiv preprint arXiv:2504.05161*.

Cummings, R., Ligett, K., Nissim, K., Roth, A., and Wu, Z. S. (2016). Adaptive learning with robust generalization guarantees. In *Conference on Learning Theory*, pages 772–814. PMLR.

Dagan, Y. and Feldman, V. (2019). Pac learning with stable and private predictions. *arXiv preprint arXiv:1911.10541*.

Daras, G., Dimakis, A. G., and Daskalakis, C. (2024). Consistent diffusion meets tweedie: Training exact ambient diffusion models with noisy data. *arXiv preprint arXiv:2404.10177*.

Daras, G., Shah, K., Dagan, Y., Gollakota, A., Dimakis, A., and Klivans, A. (2023). Ambient diffusion: Learning clean distributions from corrupted data. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Daskalakis, C., Gouleakis, T., Tzamos, C., and Zampetakis, M. (2018). Efficient statistics, in high dimensions, from truncated samples. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 639–649. IEEE.

De Bortoli, V. (2022). Convergence of denoising diffusion models under the manifold hypothesis. *arXiv preprint arXiv:2208.05314*.

Devroye, L., Györfi, L., and Lugosi, G. (2013). *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media.

Devroye, L. and Wagner, T. (1979a). Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25(2):202–207.

Devroye, L. and Wagner, T. (1979b). Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604.

Devroye, L. and Wagner, T. J. (1982). 8 nearest neighbor methods in discrimination. *Handbook of Statistics*, 2:193–197.

Dieleman, S. (2024). Diffusion is spectral autoregression.

Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., and Roth, A. L. (2015). Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126. ACM.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg. Springer-Verlag.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Dziugaite, G. K. and Roy, D. M. (2017). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*.

Edgington, E. and Onghena, P. (2007). *Randomization Tests*. CRC Press.

Feldman, V. (2020). Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959.

Feldman, V. and Vondrak, J. (2018). Generalization bounds for uniformly stable algorithms. *Advances in Neural Information Processing Systems*, 31.

Feldman, V. and Vondrak, J. (2019). High probability generalization bounds for uniformly stable algorithms with nearly optimal rate. In *Conference on Learning Theory*, pages 1270–1279. PMLR.

Feldman, V. and Xiao, D. (2014). Sample complexity bounds on differentially private learning via communication complexity. In *Conference on Learning Theory*, pages 1000–1019. PMLR.

Frey, P. W. and Adesman, P. (1976). Recall memory for visually presented chess positions. *Memory & Cognition*, 4:541–547.

Gat, E. and Goldwasser, S. (2011). Probabilistic search algorithms with unique answers and their cryptographic applications. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 18, pages 1–3.

Ghazi, B., Kumar, R., and Manurangsi, P. (2021). User-level private learning via correlated sampling. *arXiv preprint arXiv:2110.11208*.

Gold, E. M. (1967). Language identification in the limit. *Information and control*, 10(5):447–474.

Goldwasser, S., Grossman, O., and Holden, D. (2017). Pseudo-deterministic proofs. *arXiv preprint arXiv:1706.04641*.

Golowich, N., Rakhlin, A., and Shamir, O. (2018). Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pages 297–299. PMLR.

Grossman, O. and Liu, Y. P. (2019). Reproducibility and pseudo-determinism in log-space. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 606–620. SIAM.

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.

Hanneke, S. (2016). The optimal sample complexity of pac learning. *Journal of Machine Learning Research*, 17(38):1–15.

Hanneke, S., Moran, S., and Zhang, Q. (2023). Universal rates for multiclass learning. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 5615–5681. PMLR.

Hanneke, S. and Xu, M. (2024). Universal rates of empirical risk minimization. *Advances in Neural Information Processing Systems*, 37:116291–116331.

Hanneke, S. and Xu, M. (2025). Universal rates of erm for agnostic learning. *arXiv preprint arXiv:2506.14110*.

Hardt, M., Recht, B., and Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Hopkins, M., Kane, D. M., Lovett, S., and Mahajan, G. (2022). Realizable learning is all you need. In *Conference on Learning Theory*, pages 3015–3069. PMLR.

Impagliazzo, R., Lei, R., Pitassi, T., and Sorrell, J. (2022). Reproducibility in learning. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 818–831.

Jung, C., Ligett, K., Neel, S., Roth, A., Sharifi-Malvajerdi, S., and Shenfeld, M. (2019). A new analysis of differential privacy's generalization guarantees. *arXiv preprint arXiv:1909.03577*.

Kalai, A. T. and Vempala, S. S. (2024). Calibrated language models must hallucinate. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, page 160–171, New York, NY, USA. Association for Computing Machinery.

Kalavasis, A., Karbasi, A., Moran, S., and Velegkas, G. (2023). Statistical indistinguishability of learning algorithms. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15586–15622. PMLR.

Kalavasis, A., Mehrotra, A., and Velegkas, G. (2024a). Characterizations of language generation with breadth. *arXiv preprint arXiv:2412.18530*.

Kalavasis, A., Mehrotra, A., and Velegkas, G. (2024b). On the limits of language generation: Trade-offs between hallucination and mode collapse. *arXiv preprint arXiv:2411.09642*.

Kalavasis, A., Velegkas, G., and Karbasi, A. (2022). Multiclass learnability beyond the pac framework: Universal rates and partial concept classes. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20809–20822. Curran Associates, Inc.

Kalavasis, A., Zadik, I., and Zampetakis, M. (2024c). Transfer learning beyond bounded density ratios. *arXiv preprint arXiv:2403.11963*.

Kamb, M. and Ganguli, S. (2024). An analytic theory of creativity in convolutional diffusion models. *arXiv preprint arXiv:2412.20292*.

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.

Kearns, M. and Ron, D. (1997). Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. In *Proceedings of the tenth annual conference on Computational learning theory*, pages 152–162.

Kleinberg, J. and Mullainathan, S. (2024). Language generation in the limit. In *Advances in Neural Information Processing Systems*, volume 37.

Kleinberg, J. and Wei, F. (2025). Density measures for language generation. *arXiv preprint arXiv:2504.14370*.

Larsen, K. G. (2023). Bagging is an optimal pac learner. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 450–468. PMLR.

Li, J., Raman, V., and Tewari, A. (2024). Generation through the lens of learning theory. *arXiv preprint arXiv:2410.13714*.

Li, M. and Chen, S. (2024). Critical windows: non-asymptotic theory for feature emergence in diffusion models. *arXiv preprint arXiv:2403.01633*.

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

Littlestone, N. and Warmuth, M. K. (1986). Relating data compression and learnability. Technical report.

Ma, S., Bassily, R., and Belkin, M. (2018). The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. In *International Conference on Machine Learning*, pages 3325–3334. PMLR.

Malliaris, M. and Moran, S. (2022). The unstable formula theorem revisited via algorithms. *arXiv preprint arXiv:2212.05050*.

McGregor, A., Mironov, I., Pitassi, T., Reingold, O., Talwar, K., and Vadhan, S. (2010). The limits of two-party differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 81–90. IEEE.

Moran, S. and Yehudayoff, A. (2016). Sample compression schemes for vc classes. *Journal of the ACM (JACM)*, 63(3):1–10.

Nachum, I., Shafer, J., and Yehudayoff, A. (2018). A direct sum result for the information complexity of learning. *arXiv preprint arXiv:1804.05474*.

Nagarajan, V. and Kolter, J. Z. (2019). Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. (2017). A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*.

Neyshabur, B., Tomioka, R., and Srebro, N. (2015). Norm-based capacity control in neural networks. In *Conference on learning theory*, pages 1376–1401. PMLR.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. (2023). Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*.

Peale, C., Raman, V., and Reingold, O. (2025). Representative language generation. *arXiv preprint arXiv:2505.21819*.

Raginsky, M., Rakhlin, A., Tsao, M., Wu, Y., and Xu, A. (2016). Information-theoretic analysis of stability and bias of learning algorithms. *2016 IEEE Information Theory Workshop (ITW)*, pages 26–30.

Rogers, W. H. and Wagner, T. J. (1978). A finite sample distribution-free performance bound for local discrimination rules. *The Annals of Statistics*, pages 506–514.

Ross, B. L., Kamkari, H., Wu, T., Hosseinzadeh, R., Liu, Z., Stein, G., Cresswell, J. C., and Loaiza-Ganem, G. (2024). A geometric framework for understanding memorization in generative models. *arXiv preprint arXiv:2411.00113*.

Russo, D. and Zou, J. (2016). Controlling bias in adaptive data analysis using information theory. In Gretton, A. and Robert, C. C., editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1232–1240, Cadiz, Spain. PMLR.

Scarvelis, C., Borde, H. S. d. O., and Solomon, J. (2023). Closed-form diffusion models. *arXiv preprint arXiv:2310.12395*.

Shah, K., Kalavasis, A., Klivans, A. R., and Daras, G. (2025). Does generation require memorization? creative diffusion models using ambient diffusion. *arXiv preprint arXiv:2502.21278*.

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

Shelah, S. (1972). A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247–261.

Slobin, D. I. (2013). Crosslinguistic evidence for the language-making capacity. In *The crosslinguistic study of language acquisition*, pages 1157–1256. Psychology Press.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. (2022). Diffusion art or digital forgery? investigating data replication in diffusion models. *arXiv preprint arXiv:2212.03860*.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. (2023). Understanding and mitigating copying in diffusion models. *arXiv preprint arXiv:2305.20086*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

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.

Steinke, T. and Zakynthinou, L. (2020). Reasoning about generalization via conditional mutual information. In *Conference on Learning Theory*, pages 3437–3452. PMLR.

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

Van Handel, R. (2014). Probability in high dimension. *Lecture Notes (Princeton University)*, 2(3):2–3.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vapnik, V. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Xu, A. and Raginsky, M. (2017). Information-theoretic analysis of generalization capability of learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2524–2533.

Zhang, C., Bengio, S., Hardt, M., Mozer, M. C., and Singer, Y. (2019). Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Zhu, X., Anguelov, D., and Ramanan, D. (2014). Capturing long-tail distributions of object subcategories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922.