Large (robust) models from computational constraints

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large models with thousands of parameters have been hugely successful. In this work, we ask: can the need for large models be due to the *computational* limitation of the learner? Additionally, we ask, is this situation exacerbated for *robust* learning? We show that this indeed could be the case. We show learning tasks for which computationally bounded learners need *significantly more* model parameters than those of information-theoretic learners. Furthermore, we show that even more model parameters could be necessary for the case of robust learning. In particular, for computationally bounded learners, we boost the recent result of Bubeck and Sellke [NeurIPS'2021], which shows that robust models might need more parameters, to the computational regime and show that bounded learners could provably need an even larger number of parameters. Next, we ask: can we hope to remedy the situation for robust computationally bounded learning by restricting adversaries to also be computationally bounded? Here again, we show that this might be possible. Specifically, building on Garg, Jha, Mahloujifar, and Mahmoody [ALT'2020], we demonstrate a learning task that can be learned robustly and efficiently against a computationally bounded attacker with a small model. On the other hand, to be robust against an information-theoretic attacker requires the learner to output a much larger model.

1 Introduction

The fundamental theory of learning theory characterizes the set of hypothesis functions that are learnable (e.g. to have finite VC dimension) and further shows that the simple "empirical risk minimization" can always do the job. In particular, in the above (realizable) setting, it is enough to output any $f \in \mathcal{H}$ that is consistent with the training set. This means that if $\mathcal{H}$ is a "simple" class, then at least *information theoretically*, there is no need to have a more complex model than what the hypothesis class $\mathcal{H}$ requires. More specifically, if all $h \in \mathcal{H}$ can be represented with ℓ number of bits (also referred to as parameters), then the learner, (again information-theoretically) does not need more parameters in the output model f than what h needs.

What requires large models? In recent years, deep neural nets with millions or even billions of parameters[1] [Wortsman et al., 2022, Dai et al., 2021, Yu et al., 2022] have come out as one of the most powerful models for very basic tasks such as image classification. A magic of DNNs is that they generalize without falling into the classical theories mentioned above, and hence they are the subject of an active line of work aiming to understand how DNNs generalize Arora et al. [2019], Allen-Zhu et al. [2019b,a], Novak et al. [2018], Neyshabur et al. [2014] and the various benefits of overparametrized regimes Xu et al. [2018], Chang et al. [2020]. In fact, the number of parameters in such models is so large that it is enough to memorize (and fit) to a large number of *random labels*

[1] See https://paperswithcode.com/sota/image-classification-on-imagenet for the size of the most successful models for image classification of Imagenet.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

[Zhang et al., 2021]. This leads us to our first main question, in which we study one potential cause for having large models:

> *Are there any learning tasks for which computationally bounded learners need to output significantly more model parameters than needed information-theoretically?*

One should be cautious in how to formulate the question above. That is because, many simple (information-theoretically learnable) tasks are believed to be computationally hard to learn (e.g., learning parity with noise [Pietrzak, 2012]). In that case, one can interpret this as saying that the efficient learner requires *infinite* number of parameters, as a way of saying that the learning is not possible at all! However, as explained above, we are interested in understanding the reason behind having large number of parameters when learning *is possible*.

Could robustness also be a cause? A highly sought-after property of machine learning models is their *robustness* to the so-called adversarial examples [Goodfellow et al., 2014]. Here we would like to find a model f such that $f(x) = f(x')$ holds with high probability when $x \leftarrow D$ is an honestly sampled instance and $x' \approx x$ is a *close* instance that is perhaps minimally (yet adversarially) perturbed.[2]. A recent work of Bubeck and Sellke [2021] showed that having large model parameters *could* be due to the robustness of the model. Here, we are asking whether such phenomenon can have a computational variant that perhaps leads to needing *even more* parameters when the robust learner is running in polynomial time.

> *Are there any learning tasks for which computationally bounded **robust** learning comes at the cost of having even more model parameters than non-robust learning?*

Can we remedy the situation? We ask whether natural restrictions on the adversary can help reduce model sizes. In particular, we consider the restriction to the class of polynomially bounded adversaries. In fact, when it comes to robust learning, the computationally bounded aspect could be imposed both on the learner as well as the *adversary* [Mahloujifar and Mahmoody, 2019, Bubeck et al., 2019, Garg et al., 2020b, Degwekar et al., 2019].

> *Are there any learning tasks for which computationally bounded **robust** learning can be done with fewer model parameters for **polynomial-time** adversaries in comparison with information-theoretic adversaries?*

1.1 Our results

Here we describe our results. In summary, we prove that under the computational assumption that one-way functions exist, the answer to both of our main questions above is positive. Indeed, we show that the computational efficiency of the learner could be the cause of having large models. In particular, we prove the following theorem, in which a learning task is parameterized by λ, a hypothesis class $\mathcal{H}$ and a class of input distributions $\mathcal{D}$ (see Section 2.1 for formal definitions).

Theorem 1 (Main results, informal)**.** *If one-way functions exist, and for parameter λ and arbitrary polynomials $n < \alpha < \beta$ (e.g., $n = \lambda^{0.1}, \alpha = \lambda^5, \beta = \lambda^{10}$) the following holds.*
Part 1: *There is a learning task parameterized by λ and a robustness bound (to limit how much an adversary can perturb the inputs) such that:*

- *The input/instance size is $\Theta(n)$.*
- *There is a robust learner that uses $\Theta(\lambda)$ parameters.*
- *Any* polynomial-time *learner needs $\Theta(\alpha)$ parameters to learn the task.*
- *Any* polynomial-time *learner needs $\Theta(\beta)$ parameters to robustly learn the task.*

Part 2: *There is a learning task parameterized by λ and a robustness bound (to limit how much an adversary can perturb the inputs) such that:*

[2]The closeness here could be aiming to guarantee that a human cannot tell the difference between the two images x, x'.

- *The input/instance size is* $\Theta(n)$.
- *When the adversary that generates the adversarial examples runs in polynomial time, there is an (efficient) learner that outputs a model with* $O(1)$ *parameters that robustly predicts the output labels with small error.*
- *Against information-theoretic (computationally unbounded) adversaries that do the instance-perturbations, no learner can produce a model with* $< \alpha$ *parameters that later robustly makes the predictions.*

In Sections 3 and 4, we explain the high-level ideas behind the proof of Theorem 1 and formalize these statements. The full proofs can be found in supplemental material.

2 Preliminaries

For a distribution X, $x \leftarrow X$ denotes that x is sampled according to X. We use $U_{\mathcal{S}}$ for the uniform distribution over $\mathcal{S}$, and we use U_n to represent $U_{\{0,1\}^n}$. For a set $\mathcal{S}$, $x \leftarrow \mathcal{S}$ means $s \leftarrow \mathcal{U}_{\mathcal{S}}$. For any two distributions X and Y over a finite universe Ω, the *statistical distance* between X and Y is defined as

$$\mathsf{SD}\left(X, Y\right) := \frac{1}{2} \cdot \sum_{\omega \in \Omega} \left|\Pr\left[X = \omega\right] - \Pr\left[Y = \omega\right]\right|.$$

We sometimes write $X \approx_{\varepsilon} Y$ to denote $\mathsf{SD}\left(X, Y\right) \leqslant \varepsilon$. For a vector $x = (x_1, \ldots, x_n) \in \mathbb{F}^n$ and a subset $\mathcal{S} = \{i_1, \ldots, i_\ell\} \subseteq \{1, 2, \ldots, n\}$, $x_{\mathcal{S}}$ denotes $(x_{i_1}, \ldots, x_{i_\ell})$. For any two vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$, their *Hamming distance* is defined as $\mathsf{HD}(x, y) = |\{i \in \{1, 2, \ldots, n\} : x_i \neq y_i\}|$. For a set $\mathcal{S}$ and an integer $0 \leqslant t \leqslant |\mathcal{S}|$, $\binom{\mathcal{S}}{t}$ represents the set of subsets of $\mathcal{S}$ of size t. $\mathbb{I}$ stands for the indicator function. The base for all logarithms in this paper is 2.

2.1 Definitions Related to Learning and Attacks

In this section, we present the basic notions and definitions of what it means to learn a class of functions. We use $\mathcal{X}$ to denote the inputs and $\mathcal{Y} = \{0, 1\}$ to denote the outputs. By $\mathcal{H}$ we denote a *hypothesis class* of functions from $\mathcal{X}$ to $\mathcal{Y}$. We use D to denote a distribution over $\mathcal{X} \times \mathcal{Y}$. A learning algorithm, takes a set $\mathcal{S} \in (\mathcal{X} \times \mathcal{Y})^*$ and a parameter λ and outputs a function f that is supposed to predict a fresh sample from the same distribution that has generated the set $\mathcal{S}$. The parameter λ is supposed to capture the complexity of the problem, e.g., by allowing the inputs (and the running times) to grow. (E.g., this could be the VC dimension, but not necessarily so.) In particular, we assume that the members of the sets $\mathcal{X}_\lambda, \mathcal{Y}_\lambda$ can be represented with $\mathsf{poly}(\lambda)$ bits. A *proper* learning for a hypothesis class $\mathcal{H}$ outputs $f \in \mathcal{H}$, while an *improper* learner is allowed to output arbitrary functions. By default, we work with improper learning as we do particularly focus on whether the learning algorithms output a function from the hypothesis class. For a set $\mathcal{S} \subseteq \mathcal{X}$ and a function $h \colon \mathcal{X} \to \mathcal{Y}$, by $\mathcal{S}^h$ we denote the *labeled* set $\{(x, h(x)) \mid x \in \mathcal{S}\}$. For a distribution D and an oracle-aided algorithm A, by A^D we denote giving access to a *sampler* from D to A.

Definition 2 (Learning problems and learners)**.** *We use* $\mathcal{F} = \{F_\lambda = (\mathcal{X}_\lambda, \mathcal{Y}_\lambda, \mathcal{D}_\lambda, \mathcal{H}_\lambda)\}_{\lambda \in \mathbb{N}}$ *to denote a family of learning problems where each* $\mathcal{H}_\lambda$ *is a set of hypothesis functions mapping* $\mathcal{X}_\lambda$ *to* $\mathcal{Y}_\lambda$ *and* $\mathcal{D}_\lambda$ *is a set of distributions supported on* $\mathcal{X}_\lambda$.

- *For function* $\varepsilon(\cdot, \cdot)$*, we say* L ε*-learns* $\mathcal{F}$ *if*
$$\forall \lambda \in \mathbb{N}, D_\lambda \in \mathcal{D}_\lambda, h \in \mathcal{H}_\lambda, n \in \mathbb{N};\ \mathop{\mathbf{E}}_{\mathcal{S} \leftarrow D_\lambda^n; f \leftarrow L(\mathcal{S}^h, \lambda)} [\mathrm{Risk}(h, D_\lambda, f)] \leqslant \varepsilon(\lambda, n).$$
where $\mathrm{Risk}(h, D_\lambda, f) = \Pr_{x \leftarrow D_\lambda}[h(x) \neq f(x)]$.
- L *outputs models with at most* $p(\cdot)$ *bits of parameters if*
$$\forall \lambda \in \mathbb{N};\ \log\left(\left|\mathrm{Supp}\big(L(\cdot, \lambda)\big)\right|\right) \leqslant p(\lambda).$$
- L *is an* ε*-robust learner against (all) adversaries of budget* r *w.r.t. distance metric* d *if*
$$\forall \lambda \in \mathbb{N}, D_\lambda \in \mathcal{D}_\lambda, h \in \mathcal{H}_\lambda, n \in \mathbb{N};\ \mathop{\mathbf{E}}_{\mathcal{S} \leftarrow D_\lambda^n; f \leftarrow L(\mathcal{S}^h, \lambda)} [\mathrm{Risk}_{d,r}(h, D_\lambda, f)] \leqslant \varepsilon(\lambda, n).$$

where

$$\mathrm{Risk}_{(d,r)}(h, D_\lambda, f) = \Pr_{x \leftarrow D_\lambda}[\max_{x'} \mathbb{I}\big(f(x') \neq h(x)\big) \wedge \mathbb{I}\big(d(x, x') \leqslant r\big)].$$

- *L runs in polynomial time if there is a polynomial $\mathsf{poly}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and $\mathcal{S} \in (\mathcal{X}_\lambda, \mathcal{Y}_\lambda)^*$, the running time of $L(\mathcal{S}, \lambda)$ is bounded by $\mathsf{poly}(|\mathcal{S}| \cdot \lambda)$.*
- *L is an ε-robust learner against polynomial-time adversaries of budget r w.r.t distance d, if for any family of $\mathsf{poly}(\lambda)$-time (oracle aided) adversaries $\mathcal{A} = \{A_\lambda^{(\cdot)}\}_{\lambda \in \mathbb{N}}$ we have*

$$\forall \lambda \in \mathbb{N}, D_\lambda \in \mathcal{D}_\lambda, h \in \mathcal{H}_\lambda, n \in \mathbb{N}; \mathop{\mathbf{E}}_{\mathcal{S} \leftarrow D_\lambda^n; f \leftarrow L(\mathcal{S}^h, \lambda)}[\mathrm{Risk}_{d,r,A_\lambda}(h, D_\lambda, f)] \leqslant \varepsilon(\lambda, n).$$

where

$$\mathrm{Risk}_{(d,r,A_\lambda)}(h, D_\lambda, f) = \Pr_{x \leftarrow D_\lambda, x' \leftarrow A^{D_\lambda}(x, h(x), f)}[\mathbb{I}\big(h(x') \neq f(x)\big) \wedge \mathbb{I}\big(d(x, x') \leqslant r\big)].$$

2.2 Cryptographic Primitives

Definition 3 (Negligible function)**.** *A function $\varepsilon(\lambda)$ is said to be negligible (simply denoted by $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$) if for all polynomial $\mathsf{poly}(\lambda)$, it holds that $\varepsilon(\lambda) < 1/\mathsf{poly}(\lambda)$ for all sufficiently large λ.*

Definition 4 (Pseudorandom generator)**.** *Suppose $\alpha > \lambda$. A function $f\colon \{0,1\}^\lambda \to \{0,1\}^\alpha$ is called a pseudorandom generator if for all polynomial-time probabilistic algorithm A, it holds that*

$$\left| \Pr_{x \leftarrow \{0,1\}^\lambda}[A(f(x)) = 1] - \Pr_{y \leftarrow \{0,1\}^\alpha}[A(y) = 1] \right| = \mathsf{negl}(\lambda).$$

The ratio α/λ is refer to as the (multiplicative) stretch of the PRG.

Theorem 5 (Håstad et al. [1999])**.** *Pseudorandom generators (with arbitrary polynomial stretch) can be constructed from any one-way functions.*

Definition 6 (Signature scheme)**.** *A signature scheme consists of three algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.

- $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*: on input the security parameter 1^λ, the randomized algorithm* Gen *outputs a (public) verification key* vk *and a (private) signing key* sk.
- $\sigma = \mathsf{Sign}(\mathsf{sk}, m)$*: on input the signing key* sk *and a message m,* Sign *outputs a signature σ.*
- $b = \mathsf{Verify}(\mathsf{vk}, m, \sigma)$*: on input a verification key* vk*, a message m, and a signature σ,* Verify *outputs a bit b, and $b = 1$ indicates tha the signature is accepted.*

We require a (secure) signature scheme to satisfy the following properties.

- ***Correctness.*** *For every message m, it holds that*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1 : \begin{array}{c} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \sigma = \mathsf{Sign}(\mathsf{sk}, m) \end{array}\right] = 1.$$

- ***Weak unforgeability.***[3] *For any PPT algorithm A and message m, it holds that*

$$\Pr\left[\begin{array}{c} m' \neq m \text{ and} \\ \mathsf{Verify}(\mathsf{vk}, m', \sigma') = 1 \end{array} : \begin{array}{c} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \sigma = \mathsf{Sign}(\mathsf{sk}, m) \\ (m', \sigma') \leftarrow A(\mathsf{vk}, m, \sigma) \end{array}\right] = \mathsf{negl}(\lambda).$$

Theorem 7 (Naor and Yung [1989], Rompel [1990])**.** *The signature scheme can be constructed from any one-way function.*

[3] We consider a rather weak notion of unforgability. Here, we require that the adversary cannot forge a signature when he is given only one valid pair of message and signature. This weaker security notion already suffices for our purposes. In the stronger notion, the adversary is allowed to pick m based on the given vk.

2.3 Coding Theory

Let $\mathbb{F}$ be a finite field. A *linear code* $\mathcal{H}$ with *block length* n and *dimension* k is a k-dimensional subspace of the vector space $\mathbb{F}^n$. The *generator matrix* $G \in \mathbb{F}^{k \times n}$ maps every message $m \in \mathbb{F}^k$ to its encoding, namely, the message m is encoded as $m \cdot G$. The *distance* d of the code $\mathcal{H}$ is the minimum distance between any two codewords. That is, $d = \min_{(x,y \in \mathcal{H}) \wedge (x \neq y)} \mathsf{HD}(x, y)$. The *rate* of the codeword is defined as $R = k/n$.

Reed-Solomon code. In this work, we will mainly use the Reed-Solomon (RS) code. For the RS code, every message $m = (m_1, \ldots, m_k) \in \mathbb{F}^k$ is parsed as a degree $k-1$ polynomial

$$f(x) = m_1 + m_2 \cdot x + \cdots + m_k \cdot x^{k-1}$$

and the encoding is simply $(f(1), f(2), \ldots, f(n)) \in \mathbb{F}^n$.

Definition 8 (List-decodable code)**.** *A code $\mathcal{H} \subseteq \mathbb{F}^n$ is said to be (p, L)-list-decodable if for any string $x \in \mathbb{F}^n$, there are $\leqslant L$ messages whose encoding c satisfies $\mathsf{HD}(x, \mathsf{c}) \leqslant p \cdot n$. We say the code $\mathcal{H}$ is efficiently (p, L)-list-decodable if there is an efficient algorithm that finds all such messages.*

Theorem 9 (Guruswami and Sudan [1998])**.** *For any constant $R > 0$, the Reed-Solomon code with constant rate R and block length n is efficiently $(1 - \sqrt{R}, \mathsf{poly}(n))$-list-decodable.*

Due to space limitations, we include the preliminaries on randomness extraction in Appendix A. This includes the definition of (average-case) min-entropy, samplers, etc. Additional materials on the cryptographic primitives and coding theory are also included in Appendix A.

3 Efficient learner vs. information-theoretic learner

Consider the problem of learning an inner product function IP_P defined as $\mathsf{IP}_P(x) = \langle x, P \rangle$, where the inner product is done in $\mathbb{GF}_2$. Our first observation is that to learn IP_P with small error where P is uniformly random, the number of model parameters that the learner employs must be as (almost) as large as the length of P. Intuitively, one can argue it as follows. Let Z denote the parameters in the model that the learner outputs. Suppose that Z is shorter than P. Then P must contains entropy conditioned on Z.[4] By standard result in randomness extraction, one can argue that, for a uniform x, it holds that[5]

$$(Z, x, \langle x, P \rangle) \approx (Z, x, U_{\{0,1\}}).$$

That is, the label $\langle x, P \rangle$ looks information-theoretically uniform to the classifier who holds Z and x. Therefore, the classifier can only output the correct label with (the trivial) probability $\approx 1/2$.

The conclusion is that learning *all* linear functions needs a learner that outputs as many parameters as the function's description is. However, even an *efficient* learner can do the learner just as well as an information-theoretic one (e.g., using the Gaussian elimination). We now show how to modify this task to make a big difference between an efficient learner and an information-theoretic learner in terms of the number of parameters that they output.

Computational perspective. Now, suppose P is *computationally pseudorandom* [Goldwasser and Micali, 1984] rather than being truly random.[6] That is, let $f : \{0,1\}^\lambda \to \{0,1\}^\alpha$ be a pseudorandom generator and P is distributed as $f(U_\lambda)$, where U_λ denotes the uniform distribution over λ bits. Since (1) an efficient learner cannot distinguish $P \leftarrow f(U_\lambda)$ from $P \leftarrow U_\alpha$ and (2) any learner who tries to learn IP_P with $P \leftarrow U_\alpha$ needs α parameters, it can be proved that an efficient learner who tries to learn IP_P with $P \leftarrow f(U_\lambda)$ must also uses α parameters. However, an information-theoretic learner needs only λ parameters to learn IP_P with $P \leftarrow f(U_\lambda)$ as it can essentially find the seed s such that $P = f(s)$ and output the seed s. To conclude, to learn IP_P for $P \leftarrow f(U_\lambda)$, an information-theoretic learner only needs a few (i.e., λ) parameters and an efficient learner needs a lot of (i.e., α) parameters. We emphasize that for a pseudorandom generator, its output length α could have an arbitrarily large polynomial dependence on its input length λ. For instance, it could be $\alpha = \lambda^{10}$.

[4] Formally, this entropy is measured by the average-case min-entropy (Definition 21).

[5] We use $U_{\{0,1\}}$ to denote the a uniformly random bit and U_n a uniformly random binary string of length n.

[6] A pseudorandom string is merely indistinguishable from a random one in eyes' of computationally bounded distinguishers.

Robustness. We now explain the ideas behind Theorem 32 in which we study the role of model robustness in the size of the model parameters. We now suppose the instance is sampled according to the distribution D_Q, which is defined by

$$\mathsf{Enc}(U) + Q.$$

Here, U is the uniform distribution (over the right number of bits), $\mathsf{Enc}(U)$ is an error-correcting encoding of U, and the addition is coordinate-wise field addition. Moreover, the label for this instance is the inner product $\langle U, P\rangle$ for some vector P. Observe that if the learner learns Q, it can always find the correct label on a perturbed instance due to the error-correcting property of $\mathsf{Enc}(U)$. We argue that, in order to robustly learn this task for a uniformly random Q, the number of parameters in the model that the learner outputs must be almost as large as the length of Q. Intuitively, the argument is as follows. Let Z be the model that the learner outputs. Since $|Z| < |Q|$, then Q contains entropy conditioned on Z. In this case, we prove that

$$(Z, \mathsf{Enc}(U) + Q + \rho) \approx (Z, U').$$

Here, ρ stands for the noise that the adversary adds to the instance. In words, the classifier holding the parameter Z cannot distinguish the perturbed instance $\mathsf{Enc}(U) + Q + \rho$ from a uniformly random string U'. Since U is information-theoretically hidden to the classifier, it can only output the correct label $\langle U, P\rangle$ with probability $\approx 1/2$.

Next, to explore the difference between an efficient and information-theoretic learner, we consider the case where Q is pseudorandomly distributed, i.e. $Q \leftarrow f(U_\lambda)$ for some $f : \{0,1\}^\lambda \to \{0,1\}^\beta$. Again, since (1) an efficient learner cannot distinguish $Q \leftarrow U_\beta$ from $Q \leftarrow f(U_\lambda)$ and (2) any learner needs at least $\approx Q$ parameters to learn the task with $Q \leftarrow U_\beta$, an efficient learner must also need at least $\approx Q$ parameters to learn the task. On the other hand, an information-theoretic learner could again find the seed s such that $Q = f(s)$ and output the seed s as the parameter. To conclude, an information-theoretic learner requires few (i.e., λ) parameters to robustly learn the task and an efficient learner needs a lot of (i.e., $\approx \beta$) parameters to robustly learn the task. (Again, β could have an arbitrary polynomial dependence on λ.)

Making instances small. The learning tasks we considered above suffer from one drawback: the size of the instance is very large, or at least is related to the number of parameters of the model. Here we ask: is it possible to have a small instance size while an efficient learner still needs to output a very large model? We answer this question positively. In particular, for any $n = \mathsf{poly}(\lambda)$ (e.g., $n = \lambda^{0.1}$), we construct a lerning problem where the instance size is $\Theta(n)$ and the efficient learner still needs α (resp. β) parameters to learn (resp. robustly learn) the task. Our construction uses the "sampler" by Vadhan [2003]. Informally, a sampler samp (see Theorem 20) needs a small seed u and outputs a subset of $\{1, 2, \ldots, \alpha\}$ of size n. The sampler comes with the guarantee that if P is a source with high entropy, $P|_{\mathsf{samp}(u)}$ also has high enough entropy. To illustrate the usage of the sampler, consider learning this new inner product function IP_P defined as

$$\mathsf{IP}_P(u, x) = \left\langle x, P|_{\mathsf{samp}(u)} \right\rangle.$$

For uniformly random P, one can similarly argue that a model must output at least $|P|$ parameters to learn the task. Let Z denote the parameters in the model that the learner outputs. If $|Z| < |P|$, then P contains high entropy conditioned on Z. By the property of the sampler, it must hold that $P|_{\mathsf{samp}(u)}$ contains high entropy conditioned on Z and u. Consequently,

$$(Z, u, x, \mathsf{IP}_P(x)) \approx (Z, u, x, U_{\{0,1\}}).$$

Namely, the classifier who sees the parameter Z and the instance (u, x) can only predict the label with probability $\approx 1/2$. Observe that the size of the instance (u, x) is roughly n,[7] while P could have length $\alpha \gg n$. The use of the sampler in the robust learning case is similar to the non-robust case above. We refer the readers to Appendix B for more details.

This concludes the first part of Theorem 1. To sum up, we present our formal construction and formal theorem statement below. The formal proof is presented in the supplemental material.

Construction 10 (Parameter-heavy learning task under efficient learning)**.** *Our construction relies on the following building blocks.*

[7] As the seed u is very small.

- *Let* $f_1 \colon \{0,1\}^\lambda \to \{0,1\}^\alpha$ *and* $f_2 \colon \{0,1\}^\lambda \to \{0,1\}^\beta$ *be PRGs (Definition 4).*
- *Let* Enc *be a Reed-Solomon encoding with dimension* k*, block length* n*, and rate* $R = k/n$*. The rate is chosen to be any constant* $< 1/3$ *and* k *is defined by* R *and* n*. This RS code is over the field* $\mathbb{F}_{2^\ell}$ *for some* $\ell = \Theta(\log \lambda)$*.*
- *Let* $\mathsf{samp}_1 \colon \{0,1\}^{r_1} \to \binom{\{1,\ldots,\alpha\}}{k\cdot\ell}$ *and* $\mathsf{samp}_2 \colon \{0,1\}^{r_2} \to \binom{\{1,\ldots,\beta\}}{n\cdot\ell}$ *be samplers. We obtain these samplers by invoking Theorem 20 with sufficiently small* κ_1 *and* κ_2*. For instance, setting* $\kappa_1 = \Theta(1/\log\lambda)$ *and* κ_2 *to be any small constant suffices. Note that* κ_1 *and* κ_2 *defines* r_1 *and* r_2*.*
- *For any binary string* v*, we use* $[v]$ *for an arbitrary error-correcting encoding of* v *(over the field* $\mathbb{F}_{2^\ell}$*) such that* $[v]$ *can correct* $> (1-R)n/2$ *errors. This can always be done by encoding* v *using RS code with a suitable (depending on the dimension of* v*) rate. Looking forward, we shall consider an adversary that may perturb* $\leqslant (1-R)n/2$ *symbols. Therefore, when a string* v *is encoded as* $[v]$ *and the adversary perturb it to be* $\widetilde{[v]}$*, it will always be error-corrected and decoded back to* v*.*

We now construct the following learning task $F_\lambda = (\mathcal{X}_\lambda, \mathcal{Y}_\lambda, \mathcal{D}_\lambda, \mathcal{H}_\lambda)$*.*

- $\mathcal{X}_\lambda$ *implicitly defined by the distribution* $\mathcal{D}_\lambda$*.*
- $\mathcal{Y}_\lambda = \{0,1\}$*.*
- $\mathcal{D}_\lambda$ *consists of distributions* D_s *for* $s \in \{0,1\}^\lambda$*, where* D_s *is the following.*

$$D_s = \Big([u_1],\ [u_2],\ m,\ \mathsf{Enc}(m) + \Big(f_2(s)\big|_{\mathsf{samp}_2(u_2)}\Big)\Big),$$

  where

  - u_1 *and* u_2 *are sampled uniformly at random from* $\{0,1\}^{r_1}$ *and* $\{0,1\}^{r_2}$*, respectively.*
  - m *is sampled uniformly at random from* $\mathbb{F}_{2^\ell}^k$*.*
  - $f_2(s)\big|_{\mathsf{samp}_2(u_2)}$ *is interpet as a vector over* $\mathbb{F}_{2^\ell}$ *and* $\mathsf{Enc}(m) + \Big(f_2(s)\big|_{\mathsf{samp}_2(u_2)}\Big)$ *is coordinate-wise addition over* $\mathbb{F}_{2^\ell}$*.*

- $\mathcal{H}_\lambda$ *consists of all functions* $h_s : \mathcal{X} \to \mathcal{Y} = \{0,1\}$ *for all* $s \in \{0,1\}^\lambda$*, where* h_s *is defined as follow.*

$$h_s(x) = \Big\langle m, \Big(f_1(s)\big|_{\mathsf{samp}_1(u_1)}\Big)\Big\rangle,$$

  where the inner product is over $\mathbb{F}_2$ *and* m *is interpret as a string* $\in \mathbb{F}_2^{k\cdot\ell}$ *in the natural way.*

- ***Adversary.*** *The entire input* $x = ([u_1], [u_2], m, \mathsf{Enc}(m) + \big(f_2(s)|_{\mathsf{samp}_1(u_1)}\big))$ *is interpret as a vector over* $\mathbb{F}_{2^\ell}$ *and we consider an adversary that may perturb* $\leqslant (1-R)n/2$ *symbols. That is, the distance metric is Hamming distance over* $\mathbb{F}_{2^\ell}$ *and the adversary has a budget of* $(1-R)n/2$*.*

Theorem 11. *An information-theoretic learner can (robustly)* ε*-learn the task of Construction 10 with parameter size* 2λ *and sample complexity* $\Theta(\frac{\lambda}{\varepsilon})$*. Moreover, an efficient learner can (robustly)* ε*-learn this task with parameter size* $\alpha+\beta$ *and sample complexity* $\Theta(\frac{\alpha+\beta}{\varepsilon})$*.*

Theorem 12. *Any efficient learner that outputs models with* $\leqslant \alpha/2$ *parameters cannot* ε*-learn* F_λ *of Construction 10 for* $\varepsilon < 1/3$*.*

Theorem 13. *There exists some constant* c *such that the following holds. In the presence of an adversary that may perturb* $(1-R)n/2$ *symbols, any efficient learner for the task of Construction 10 that outputs a model with* $c \cdot \beta/\log\lambda$ *parameters cannot* ε*-robustly learn* F_λ *for* $\varepsilon < 1/3$*.*

4 Efficient adversary vs. information-theoretic adversary

Next, we explore whether the computational efficiency of the adversary could affect the number of parameters required to robustly learn a task. Previous work Garg et al. [2020a] has considered the

difference between an efficient adversary and an information-theoretic one in terms of whether a successful attack can be launched. We first recall their construction. The learning instance is sampled from the distribution

$$[\mathsf{vk}], b, \mathsf{Sign}(\mathsf{sk}, b).$$

Here, $(\mathsf{vk}, \mathsf{sk})$ is the verification key and signing key pair of a signature scheme (Definition 6);[8] $[\mathsf{vk}]$ is an error-correcting encoding of vk, which ensures that $[\mathsf{vk}]$ can always be recovered after perturbation by the adversary; b is a uniform random bit and the label of the instance is simply b.

The signature scheme ensures that an efficient adversary cannot forge a valid signature and, hence, any efficient adversary that perturbs $(b, \mathsf{Sign}(\mathsf{sk}, b))$ will result in an invalid message/signature pair. A classifier could then detect such perturbations and output a special symbol $\perp$ indicating that perturbation is detected. On the other hand, an information-theoretic adversary could launch a successful attack by forging a valid signature $(1-b, \mathsf{Sign}(\mathsf{sk}, 1-b))$.Therefore, it could perturb the instance to be $[\mathsf{vk}], 1-b, \mathsf{Sign}(\mathsf{sk}, 1-b)$ and, hence, flipping the label of the output.

First idea. We want to construct a learning problem such that the learner needs few parameters against an efficient adversary, but a lot of parameters against an information-theoretic adversary. Our first idea is to add another way of recovering b in the above learning problem. Consider the learning problem where the instance is sampled from distribution D_P defined as

$$[\mathsf{vk}], b, \mathsf{Sign}(\mathsf{sk}, b), [b + \langle u, P\rangle], [u].$$

Here, u is uniformly distributed. Observe that if the learner learns P, it can always recover b correctly from $[b + \langle u, P\rangle]$ and $[u]$ by invoking error-correction decoding. However, if the number of parameters in the model that the learner outputs have $< |P|$ parameters, then $[b + \langle u, P\rangle]$ and $[u]$ information-theoretically hides b. Again, this is because P will contain entropy conditioned on the parameters Z and, hence,

$$Z, u, \langle u, P\rangle \approx Z, u, U_{\{0,1\}}.$$

Consequently, an information-theoretic adversary could again launch the attack that replace $b, \mathsf{Sign}(\mathsf{sk}, b)$ with $1-b, \mathsf{Sign}(\mathsf{sk}, 1-b)$ and successfully flipping the label. Therefore, a learner must employ $|P|$ parameters to robustly learn the task against information-theoretic adversaries.

Second idea. In the above learning task, a learner employing $|P|$ parameters can always recover the correct label against information-theoretic adversaries. However, against efficient adversaries, a learner with fewer parameters may not always recover the correct label but will sometimes output the special symbol $\perp$ indicating that tampering is detected. Could we twist it to ensure that a learner with fewer parameters can also always recover the correct label against efficient adversaries? We positively answer this question by using list-decodable code (Definition 8). Intuitively, list-decoding ensures that given an erroneous codeword, the decoding algorithm will find the list of all messages whose encoding is close enough to the erroneous codeword.

Our new learning task has instances drawing from the distribution D_P defined as

$$[\mathsf{vk}], \mathsf{Enc}(b, \mathsf{Sign}(\mathsf{sk}, b)), [b + \langle u, P\rangle], [u].$$

Here, Enc is the encoding algorithm of the list-decoding code. The main idea is that: after the perturbation, the original message/signature pair $b, \mathsf{Sign}(\mathsf{sk}, b)$ will always appear in the list output by the decoding algorithm. Now, against an efficient adversary, $b, \mathsf{Sign}(\mathsf{sk}, b)$ must be the only valid message/signature pair in the list. Otherwise, this adversary breaks the unforgeability of the signature scheme. Against an information-theoretic adversary, however, it could introduce $(1-b, \mathsf{Sign}(\mathsf{sk}, 1-b))$ into the list recovered by the list-decoding algorithm. Consequently, the learner cannot tell if the correct label is b or $1-b$. Consequently, for this learning task, against information-theoretic adversaries, one still needs $|P|$ parameters. Against efficient adversaries, one does not need to output any parameter and can always recover the correct label.

Making instances small. Again, we have this unsatisfying feature that the instance has the same size as the model. We resolve this issue using the sampler in a similar way. We refer the readers to Appendix C for more details.

This finishes the high-level proof of Part 2 of Theorem 1. To sum up, we present our formal construction and theorem statement below. The formal proof is presented in Appendix C.

[8] Every instance samples a fresh $(\mathsf{vk}, \mathsf{sk})$.

Construction 14 (Learning task for bounded/unbounded attackers)**.** *We first introduce the building blocks.*

- $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *be a signature scheme (see Definition 6).*
- *Let* Enc *be a Reed-Solomon encoding with dimension* k*, block length* n*, and rate* $R = k/n$*. We pick the rate* R *to be any constant* $< 1/4$ *and* k *is defined by* R *and* n*. This RS code is over the field* $\mathbb{F}_{2^\ell}$ *for some* $\ell = \Theta(\log \lambda)$*.*
- *Let* $\mathsf{samp}\colon \{0,1\}^r \to \binom{\{1,\ldots,\alpha\}}{n}$ *be samplers. (We obtain these samplers by invoking Theorem 20 with sufficiently small* κ_1 *and* κ_2*. For instance, setting* $\kappa_1 = \Theta(1/\log \lambda)$ *and* κ_2 *to be any small constant suffices.)*
- *For any binary string* v*, we use* $[v]$ *for an arbitrary error-correcting encoding of* v *(over the field* $\mathbb{F}_{2^\ell}$*) such that* $[v]$ *can correct* $> (1-\sqrt{R})n$ *errors. This can always be done by encoding* v *using RS code with a suitable (depending on the dimension of* v*) rate. Looking forward, we shall consider an adversary that may perturb* $\leqslant (1-\sqrt{R})n$ *symbols. Therefore, when a string* v *is encoded as* $[v]$ *and the adversary perturb it to be* $\widetilde{[v]}$*, it will always be error-corrected and decoded back to* v*.*

We now construct the following learning task $F_\lambda = (\mathcal{X}_\lambda, \mathcal{Y}_\lambda, \mathcal{D}_\lambda, \mathcal{H}_\lambda)$*.*

- $\mathcal{X}_\lambda$ *is* $\{0,1\}^N$ *for some* N *that is implicitly defined by* $\mathcal{D}_\lambda$*.*
- $\mathcal{Y}_\lambda$ *is* $\{0,1\}$*.*
- *The distribution* $\mathcal{D}_\lambda$ *consists of all distribution* D_s *for* $s \in \{0,1\}^\alpha$*, where*

$$D_s = \Big([u], [v],\ [\mathsf{vk}],\ \mathsf{Enc}(b, \mathsf{Sign}(\mathsf{sk}, b)),\ \big[b + \langle v, s|_{\mathsf{samp}(u)}\rangle\big]\Big).$$

  Here,

  - u *is uniformly at random from* $\{0,1\}^r$ *and* v *is uniformly at random from* $\{0,1\}^n$*.*
  - $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *are sampled from the signature scheme.*
  - b *is sampled uniformly at random from* $\{0,1\}$*.* $(b, \mathsf{Sign}(\mathsf{sk}, b))$ *is intepret as* $\mathbb{F}_{2^\ell}^k$ *in the natural way.*
- h_λ *consists of one single function* h*. On input* $x = ([u], [v], \mathsf{Enc}(b, \mathsf{Sign}(\mathsf{sk}, b)), [b + \langle v, s|_{\mathsf{samp}}(u)\rangle])$*,* $h(x)$ *simply decode* $\mathsf{Enc}(b, \mathsf{Sign}(\mathsf{sk}, b))$ *and output* b*.*
- ***Adversary.*** *The entire input* $([u], [v], [vk], \mathsf{Enc}(b, \mathsf{Sign}(\mathsf{sk}, b)), [b + \langle v, s|_{\mathsf{samp}(u)}\rangle])$ *is interpret as a vector over* $\mathbb{F}_{2^\ell}$ *and we consider an adversary that may perturb* $\leqslant (1-\sqrt{R})n$ *symbols. That is, the distance metric is Hamming distance over* $\mathbb{F}_{2^\ell}$ *and the adversary has a budget of* $(1-\sqrt{R})n$*.*

Theorem 15. *For the learning task of Construction 14, there is an efficient learner (with 0 sample complexity) that outputs a model with no parameter and* $\mathsf{negl}(\lambda)$*-robustly learns* F_λ *against efficient adversaries of budget* $(1-\sqrt{R})n$*.*

Theorem 16. *Consider the learning task of Construction 14. Then, for computationally unbounded adversaries, any information-theoretic learner with* $\alpha/2$ *parameters cannot* ε*-robustly learn* F_λ *for* $\varepsilon < 1/3$*.*

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32, 2019a.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019b.

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

Sébastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34, 2021.

Sébastien Bubeck, Yin Tat Lee, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints. In *International Conference on Machine Learning*, pages 831–840. PMLR, 2019.

Xiangyu Chang, Yingcong Li, Samet Oymak, and Christos Thrampoulidis. Provable benefits of overparameterization in model compression: From double descent to pruning neural networks. *arXiv preprint arXiv:2012.08749*, 2020.

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977, 2021.

Akshay Degwekar, Preetum Nakkiran, and Vinod Vaikuntanathan. Computational limitations in robust classification and win-win results. In *Conference on Learning Theory*, pages 994–1028. PMLR, 2019.

Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 654–663, Baltimore, MA, USA, May 22–24, 2005. ACM Press. doi: 10.1145/1060590.1060688.

Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008. doi: 10.1137/060651380. URL https://doi.org/10.1137/060651380.

Sanjam Garg, Somesh Jha, Saeed Mahloujifar, and Mohammad Mahmoody. Adversarially robust learning could leverage computational hardness. In Aryeh Kontorovich and Gergely Neu, editors, *Algorithmic Learning Theory, ALT 2020, 8-11 February 2020, San Diego, CA, USA*, volume 117 of *Proceedings of Machine Learning Research*, pages 364–385. PMLR, 2020a.

Sanjam Garg, Somesh Jha, Saeed Mahloujifar, and Mahmoody Mohammad. Adversarially robust learning could leverage computational hardness. In *Algorithmic Learning Theory*, pages 364–385. PMLR, 2020b.

Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. In *39th Annual Symposium on Foundations of Computer Science*, pages 28–39, Palo Alto, CA, USA, November 8–11, 1998. IEEE Computer Society Press. doi: 10.1109/SFCS.1998.743426.

Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

Saeed Mahloujifar and Mohammad Mahmoody. Can adversarially robust learning leveragecomputational hardness? In *Algorithmic Learning Theory*, pages 581–609. PMLR, 2019.

Omar Montasser, Steve Hanneke, and Nathan Srebro. Vc classes are adversarially robustly learnable, but only improperly. In *Conference on Learning Theory*, pages 2512–2530. PMLR, 2019.

Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, WA, USA, May 15–17, 1989. ACM Press. doi: 10.1145/73007.73011.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Krzysztof Pietrzak. Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 99–114. Springer, 2012.

John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, MD, USA, May 14–16, 1990. ACM Press. doi: 10.1145/100216.100269.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Salil P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 61–77, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. doi: 10.1007/978-3-540-45146-4_4.

Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptol.*, 17(1):43–77, 2004.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*, 2022.

Ji Xu, Daniel J Hsu, and Arian Maleki. Benefits of over-parameterization with em. *Advances in Neural Information Processing Systems*, 31, 2018.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]