Towards a theory of model distillation

Enric Boix-Adserà
eboix@mit.edu

March 14, 2024

Abstract

Distillation is the task of replacing a complicated machine learning model with a simpler model that approximates the original [BCNM06, HVD15]. Despite many practical applications, basic questions about the extent to which models can be distilled, and the runtime and amount of data needed to distill, remain largely open.

To study these questions, we initiate a general theory of distillation, defining PAC-distillation in an analogous way to PAC-learning [Val84]. As applications of this theory: (1) we propose new algorithms to extract the knowledge stored in the trained weights of neural networks – we show how to efficiently distill neural networks into succinct, explicit decision tree representations when possible by using the "linear representation hypothesis"; and (2) we prove that distillation can be much cheaper than learning from scratch, and make progress on characterizing its complexity.

Contents

1 Introduction

This paper studies model distillation, which is the task of replacing a complicated machine learning model with a simpler one that approximates it well [BCNM06, HVD15].

Distillation has significant practical applications. First, distillation is an important tool for improving computational efficiency: large models or ensembles of models can often be distilled to smaller models which are deployable at lower computational cost [BCNM06, HVD15, PPA18, FC18, GYMT21]. Second, a growing body of work uses distillation for interpretability: a "black-box" model such as a neural network can be studied by distilling it to the closest model from a more transparent class of models, such as linear functions or decision trees (see, e.g., [ADRDS+20] and references within).

While distillation is widely used in practice, and sometimes succeeds in replacing large, complex models by smaller or simpler models with a minor loss in accuracy, a number of basic questions remain largely open:

1. *To what extent can a given model be distilled, and on what properties of the model does this depend?*

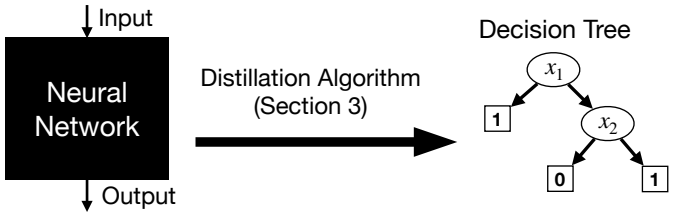
2. *How much data and runtime are needed to distill a given model?*

1.1 Contributions

This paper makes progress on these questions. We summarize our contributions below:

- **Formalization of distillation**: First, we formalize the distillation task via a new definition that we call *PAC-distillation*. Our framework allows us to rigorously approach the above questions, letting us prove statements about the amount of data and runtime needed in order to distill a class of models into another class of models; see Section 2.

- **New tools to distill neural networks**: Using this formalization, we propose novel algorithms to distill neural networks. Our algorithms exploit the structure in trained neural networks' weights to distill the models into more succinct and transparent forms. Most notably, we provide a provably efficient algorithm that distills a network that *implicitly computes a decision tree* into an *explicitly-represented decision tree* which compactly represents the original network; see below for a schematic, and see Section 3 for details.



- **General theory of distillation**: More generally, we initiate a computational and statistical theory of distillation that applies to arbitrary models beyond neural networks. A takeaway of our results is that in many cases distilling a trained model can be a much cheaper task in terms of data and compute than learning a model from scratch. This is an attractive property of distillation, since it means that distillation may be feasible even when few computing and data resources are available; see Sections 4 and 5.

- **Open problems**: Finally, we discuss extensions and new open directions; see Section 6.

Let us now delve into finer detail on each point.

1.1.1 Defining PAC-distillation

We first introduce PAC-distillation, which formalizes the distillation problem from a class of models $\mathcal{F}$ to a class of models $\mathcal{G}$. This paper mostly focuses on the classification setting, where a model is a function $h : \mathcal{X} \to \mathcal{Y}$ from datapoints to labels.

Informal Definition 1.1. *Given a source model $f \in \mathcal{F}$ and i.i.d. samples $x_1, \ldots, x_n$ from an unknown distribution $\mathcal{D}$,* PAC-distillation *is the problem of finding a model $g \in \mathcal{G}$ whose error approximating f over the distribution is small:*

$$\text{error}_{f,\mathcal{D}}(g) = \mathbb{P}_{x\sim\mathcal{D}}[g(x) \neq f(x)]\,.$$

The full definition of PAC-distillation is in Section 2 and variations, such as to the regression setting, are considered in Section 6.

Remark 1.2 (Relation to PAC-learning). In analogy to PAC-learning [Val84], PAC stands for "Probably Approximately Correct", since the distillation process only has to be approximately correct with high probability. In the distillation setting we are given complete access to the source model $f \in \mathcal{F}$ represented in some manner (e.g., if $\mathcal{F}$ were a class of neural networks, f would be given as the weights of the neural network). This contrasts with PAC-learning, where only the labels $f(x_1), \ldots, f(x_n)$ of the data samples would be known to the algorithm.

Since a distillation algorithm gets extra information about the source function f than just the sample labels, it is possible to PAC-distill from class $\mathcal{F}$ to class $\mathcal{G}$ whenever it is possible to PAC-learn concepts from class $\mathcal{F}$ with hypotheses from class $\mathcal{G}$. In this work, we will be primarily concerned with understanding when there is a separation – i.e., when PAC-distillation requires fewer samples and/or runtime than PAC-learning.

With this definition in hand, we now return to the questions raised above. Before describing our general results on fundamental computational and statistical limits for distillation, we focus on distilling neural networks, since these are of contemporary significance.

1.1.2 New algorithmic tools to distill neural networks

Neural networks are a highly expressive function class. As such, one should not expect that arbitrary neural networks can be usefully distilled into simpler models. The answer to whether a neural network can be distilled (question 1 above) therefore must hinge on *some* structure being present in the weights of the trained neural network.

We present a novel distillation algorithm in this paper that makes use of a structure that may be present in a trained neural network: the *linear representation property*. A network satisfies this property if important high-level features of the input can be expressed as a linear function of the trained neural network's internal representation. There is significant empirical evidence [MYZ13, AB16, VC20, EHO+22, THGN23, SPK16, CKL+18, AL23a, LHB+22, NLW23, MT23, LNA21, DDH+21, AL23b, AMS21, BCZ+16, VC20, RTGC22, RGC23, BSJR+23, WGNV23, HLA23, PCV23, BLSR22] and growing theoretical evidence [ALL+16, ABM22, DLS22, BES+22, MHPG+22, ABM23, DKL+23, BBPV23] supporting the hypothesis that trained neural networks satisfy the linear representation property; see Appendix A for a literature review.

Informal Definition 1.3 (Linear representation property)**.** *Let $\phi(x) \in \mathbb{R}^m$ be the neural network's* representation map*, which takes in an input x and computes a vector embedding.*[a]

Let $\mathcal{G}$ be a collection of real-valued functions of the input x. For any $\tau > 0$, the τ-bounded linear representation property with respect to $\mathcal{G}$ states that for each function $g \in \mathcal{G}$, there is $w_g \in \mathbb{R}^m$ such that $\|w_g\| \leq \tau$ and for all inputs x we have

$$w_g^\top \phi(x) = g(x)\,.$$

[a]The representation map $\phi(x) \in \mathbb{R}^m$ is typically the concatenation of the functions computed at the m internal neurons, which is what we use in our experiments. There are other choices, such as the gradient of the network with respect to its parameters, or the activations of the penultimate layer, that also make sense.

In this paper we prove that under the linear representation property a neural network that *implicitly computes a decision tree* can be distilled into an *explicit and succinct* decision tree. The set of functions $\mathcal{G}$ that we assume that the trained network linearly represents is the set of the intermediate computations of the decision tree, consisting of all of AND statements computed by paths starting from the root and going to any internal node or leaf. Our distillation algorithm is efficient (polynomial time).

Informal Theorem 1.4 (The linear representation property can be used to efficiently distill neural networks into decision trees)**.** *Suppose that we are given:*

- *a neural network $f_{\mathsf{NN}}(x)$ that implicitly computes a decision tree of size s and depth r*
- *the network's representation $\phi(x) \in \mathbb{R}^m$, which satisfies the τ-bounded linear representation property with respect to the internal computations of the decision tree*

Then, if the input distribution is uniform over $\{0,1\}^d$, the network can be distilled to an equivalent, explicitly represented tree of size s in $\mathrm{poly}(d, s, \tau, \max_x \|\phi(x)\|, m)$ time, where each network and representation evaluation takes unit time.
If the input distribution $\mathcal{D}$ is arbitrary over $\{0,1\}^d$, then distillation is possible in $\mathrm{poly}(d, 2^r, \tau, \max_x \|\phi(x)\|, m)$ time and samples from $\mathcal{D}$.

Remark 1.5 (Comparison to query learning, and learning from random examples)*.* This constitutes the first polynomial-time algorithm for distillation of neural networks into decision trees. The prior best algorithm for this task uses query access to the neural network model and runs in time $\tilde{O}(d^2) \cdot s^{O(\log\log s)}$ when the input distribution is assumed to be uniform over the binary hypercube [BLQT22]. The caveat in our result is that we additionally require the linear representation property. Nevertheless, we verify experimentally that this property *does* hold in practice. In Section 3.2.2, we implement our algorithm to successfully extract decision trees from deep neural networks that have been trained to learn a decision tree.

Our distillation result should also be contrasted with the current best theoretical guarantee of $d^{O(\log(s))}$ time and samples for learning decision trees from random examples [EH89], and which is not guaranteed to return a s-size tree but can potentially return a tree of $d^{\Omega(\log(s))}$ size. Thus, this is a natural setting where distillation can potentially be much faster than learning from random examples.

Remark 1.6 (Beyond decision trees)*.* Of course, neural networks can and do learn functions that are *not* given by small decision trees, so distillation into small decision trees may not be possible. We believe that under the linear representation property it may be possible to efficiently distill

neural networks into other simple model classes that are more expressive than decision trees but better capture the internal computations of the network. Indeed, we view leveraging of the linear representation property as the more significant conceptual contribution of this theorem, rather than the specific polynomial-time algorithm for decision tree distillation. In ongoing work, we are currently exploring distillation to other classes beyond decision trees, as mentioned in Section 6.

The above result addresses the questions 1 and 2, by giving specific, nontrivial conditions under which models with hidden structure can be efficiently distilled (and more broadly our result suggests that the linear representation property may be useful for distilling networks beyond the setting of decision trees). We now turn to providing theoretical principles on distillation that address questions 1 and 2 in more generality.

1.1.3 General computational and statistical theory of distillation

We initiate a general computational and statistical theory of distillation, analogous to the theory of PAC-learning. First, we observe that distillation can be used to define a web of computational reductions. Distillation creates a partial order between model classes, capturing the *computational feasibility* of converting from one model class to another. We are optimistic that this framework can lead to a computational theory of interpretability, formalizing what it means for a given model class to be uniformly more interpretable than another. This is discussed in Section 4.

Towards deriving fundamental statistical limits, we also explore the sample complexity of distillation. One of our main results is on *perfect distillation*, which is the setting where for every model in $\mathcal{F}$ it is possible to distill arbitrarily well to $\mathcal{G}$ given sufficiently many samples. In this setting we characterize the sample complexity of distillation for finite and countable model classes, and we show that very few samples are needed to distill.

Informal Theorem 1.7 (Whenever it is possible to distill to arbitrary accuracy, very few samples are needed)**.** *Let $\mathcal{F}$ and $\mathcal{G}$ be model classes of countable size. Whenever it is possible to "perfectly distill" from $\mathcal{F}$ into $\mathcal{G}$ (see Definition 5.3), it is possible to ϵ-approximately distill with δ probability of error using only $O(\log(1/\delta)/\epsilon)$ samples. Furthermore, this bound is tight.*

The conceptual takeaway is that in general far fewer samples are needed to distill from class $\mathcal{F}$ to class $\mathcal{G}$ than are needed to learn the model in class $\mathcal{G}$ from scratch. The number of samples needed to distill here is always bounded independently of $\mathcal{F}$ and $\mathcal{G}$, even in cases when $\mathcal{G}$ is not learnable. This supports a machine learning paradigm in which a large institution or company trains models using a huge amount of data, and then smaller entities can distill the trained model using far fewer resources. Of course, Informal Theorem 1.7 does not provide a computationally efficient algorithm, and tradeoffs between the amount of computation and data used to distill are highly interesting to study in future work.

In addition to this result, we show lower bounds on the sample complexity of distillation, prove that the natural analogues of VC-dimension from learning theory do not correctly predict the sample complexity of distillation, and identify a number of new open problems on pinning down the sample complexity of distillation.

1.2 Related work

Distillation applied to efficient inference Model distillation was proposed in [BCNM06] under the name "model compression". The setting in that work was finding a single small model that matched the performance of a large ensemble of models. This was later popularized by [HVD15], which proposed the method of training a small neural network model with a loss function that

drove it to match the logits of the larger ensemble model. Extensive empirical work has shown that this method can often distill large neural network models to smaller ones with minimal loss in accuracy (see e.g., the surveys [GYMT21, XLT+24] and references within). There are alternative methods for distilling large models to smaller ones that also have empirical success, such as methods based on pruning [ZG17, FC18, MLN19, WWL19, BGOFG20], methods based on replacing individual layers with lightweight approximations (e.g., [MPF+23, SAM23]), and methods based on quantizing the weights of the models [GAGN15, GKD+22].

Distillation applied to interpretability Distillation is also an important tool in the model interpretability literature, where it is sometimes known as "model simplification" (see e.g., the survey [ADRDS+20] and references within). This approach to interpretability seeks to extract rules from complicated models such as neural networks by distilling them, if possible, into small logical formulas [MMS91, McM92, Thr93, ADT95], decision trees [CS94, CS95, BS96, VAB07, JSL11, ZH16, BKB17, VLJ+17, FH17], generalized additive models [LDIK08, RSG16, TCHL18], or wavelets [HSL+21]. Those more transparent models can then be understood or interpreted using downstream techniques (such as model purification [LTC+20] in the case of generalized additive models), yielding insight into the original model.

Other approaches to interpretability. It should be noted that there are many other approaches to interpreting black-box models outside of distillation, such as methods based on feature visualization and attribution (see, e.g., [MSK+19, ADRDS+20] and references within). Another angle is to avoid training black-box models and instead train models that are interpretable by design (see, e.g., [RCC+22, CSH+22]).

Mathematical analyses of distillation The work of [USSBD11] studies the related setting of semi-supervised learning, where there is a small amount of labeled data and a large amount of unlabeled data. That paper gives general sufficient conditions under which a two-stage algorithm involving learning and then distilling can reduce the amount of labelled data needed to learn, and it explores these conditions in the context of learning halfspaces and learning with nearest neighbors.

Another pair of highly-relevant works are [ZZH18] and [ZXH22], which study the stability of distilling models, showing that in practice a significant amount of data is needed to ensure that the same model is consistently outputted by distillation across various trials with independent samples. Of these papers, [ZXH22] proposes a general procedure for distillation that treats the process as a multiple-hypothesis testing problem between student hypotheses, and provides upper bounds on the number of samples needed to distill with this procedure, based on certain measures of complexity of the model classes.

In the domain of neural networks, several hypotheses for the practical effectiveness of distillation have been proposed. In [AZL20] it is argued that distillation succeeds because of "multi-view" structure in data that is learned by an ensemble of models, and is captured by the distilled model. In [PCJH22] it is argued that distillation prefers flatter minima in the loss landscape, which can even enable the distilled model to generalize better than the original. The paper [PL19] analyzes the distillation of linear and deep linear models by optimizing the cross-entropy loss on the soft labels of the teacher model. There, it is proved that distillation with a small amount of data can obtain much lower risk than learning from the same amount of data, because of the interaction between the optimizer and the geometry of the data.

	PAC-learning [Val84] concept class $\mathcal{F}$, hypothesis class $\mathcal{G}$	**PAC-distillation** [this work] source class $\mathcal{F}$, target class $\mathcal{G}$
Unknown:	distribution $\mathcal{D}$ concept $f \in \mathcal{F}$	distribution $\mathcal{D}$
Known:	examples $x_1, \ldots, x_n \overset{i.i.d.}{\sim} \mathcal{D}$ labels $f(x_1), \ldots, f(x_n)$	examples $x_1, \ldots, x_n \overset{i.i.d.}{\sim} \mathcal{D}$ model $f \in \mathcal{F}$
Want:	hypothesis $g \in \mathcal{G}$ minimizing $\text{error}_{f,\mathcal{D}}(g)$	model $g \in \mathcal{G}$ minimizing $\text{error}_{f,\mathcal{D}}(g)$

Figure 1: Summary of PAC-learning vs. PAC-distillation as defined in Section 2. The difference is that in PAC-distillation, the source model f is one of the inputs to the distillation algorithm. For example, if the goal is to distill a neural network, then the neural network's weights are inputted to the distillation algorithm.

1.3 Notation

Let $[m] = \{1, \ldots, m\}$ denote the set of m elements. Given a vector $v \in \mathbb{R}^m$, and a subset $S \subseteq [m]$ of indices, let $v_S \in \mathbb{R}^S$ denote the subvector indexed by those indices. Let $\text{poly}(a_1, a_2, \ldots, a_k)$ denote a polynomially-bounded function of $a_1, \ldots, a_k$.

2 PAC-distillation

We formally define *PAC-distillation*, which is the problem of translating a function from source class $\mathcal{F}$ to an approximately equivalent function in target class $\mathcal{G}$. The acronym PAC stands for "Probably Approximately Correct": the distillation must be ϵ-approximately correct with probability at least $1 - \delta$ for some parameters ϵ, δ.

2.1 PAC-learning

Our definition is based on *PAC-learning* [Val84], a foundational definition in computational learning theory [KV94, MRT18]. In the PAC-learning formalism, there is:

- an *input space* $\mathcal{X}$,
  (e.g., the set of all possible images)
- a *label space* $\mathcal{Y}$,
  (e.g., the set of possible classification labels of those images)
- a *distribution* $\mathcal{D}$ over the input space $\mathcal{X}$,
  (e.g., the distribution of natural images on the Internet)
- a *concept class* $\mathcal{F}$, which is a set of functions from $\mathcal{X}$ to $\mathcal{Y}$,
  (e.g., the set of possible ground-truth classifications)
- and a *hypothesis class* $\mathcal{G}$, which is also a set of functions from $\mathcal{X}$ to $\mathcal{Y}$.
  (e.g., the set of neural network classifiers)

The goal of a learning algorithm is to output an approximation $g \in \mathcal{G}$ of a ground-truth function $f \in \mathcal{F}$, with respect to the distribution $\mathcal{D}$. A learning algorithm seeks to minimize the probability that the returned hypothesis makes a mistake on a fresh test data point:

$$\text{error}(g) = \text{error}_{f,\mathcal{D}}(g) = \mathbb{P}_{x \sim \mathcal{D}}[g(x) \neq f(x)]\,. \tag{2.1}$$

A major difficulty is that the learning algorithm does not have direct access to (i) the ground-truth function f, or (ii) the distribution $\mathcal{D}$. The inputs are a set of random examples $S = (x_1, \ldots, x_n) \in \mathcal{X}^n$ drawn i.i.d. from $\mathcal{D}$, and corresponding ground-truth labels $L = (f(x_1), \ldots, f(x_n)) \in \mathcal{Y}^n$. The learner outputs a hypothesis:

$$\mathcal{A}_{\text{learn}}(S, L) \in \mathcal{G}.$$

A concept class $\mathcal{F}$ is efficiently PAC-learnable with hypothesis class $\mathcal{G}$ if there is a learner that outputs a low-error hypothesis using a small number of samples and small amount of time.[1]

Definition 2.1 (PAC-learning; [Val84])**.** *Concept class $\mathcal{F}$ is (ϵ, δ)-learnable by algorithm $\mathcal{A}_{\text{learn}}$ in n samples and t time if for any distribution $\mathcal{D}$ on $\mathcal{X}$, and any concept $f \in \mathcal{F}$, the algorithm runs in $\leq t$ time and*

$$\mathbb{P}_{S \sim \mathcal{D}^n}[\text{error}_{f,\mathcal{D}}(\mathcal{A}_{\text{learn}}(S, L)) \leq \epsilon] \geq 1 - \delta\,.$$

2.2 PAC-distillation

We adapt the definition of PAC-learnability to the distillation setting, where we refer to $\mathcal{F}$ as the *source class*, and to $\mathcal{G}$ as the *target class*. The difference is that the distillation algorithm $\mathcal{A}_{\text{dist}}$ is also given the source function $f \in \mathcal{F}$ as one of its inputs:

$$\mathcal{A}_{\text{dist}}(S, f) \in \mathcal{G}\,.$$

Definition 2.2 (PAC-distillation)**.** *Source class $\mathcal{F}$ is (ϵ, δ)-distillable into target class $\mathcal{G}$ by algorithm $\mathcal{A}_{\text{dist}}$ in n samples and t time if for any distribution $\mathcal{D}$ on $\mathcal{X}$, and any source model $f \in \mathcal{F}$, the algorithm runs in $\leq t$ time and:*

$$\mathbb{P}_{S \sim \mathcal{D}^n}[\text{error}_{f,\mathcal{D}}(\mathcal{A}_{\text{dist}}(S, f)) \leq \epsilon] \geq 1 - \delta\,.$$

PAC-distillation is an easier problem than PAC-learning because given the samples $S = (x_1, \ldots, x_n)$ and the function $f \in \mathcal{F}$, the labels $L = (f(x_1), \ldots, f(x_n))$ can be computed, so a learning algorithm can be applied. Nevertheless, distillation remains challenging from the statistical angle because the distribution $\mathcal{D}$ is unknown, and also from the computational angle because one must convert between the representation of functions in function class $\mathcal{F}$ and those in function class $\mathcal{G}$.

If the source class $\mathcal{F}$ and the target class $\mathcal{G}$ are equal (i.e., $\mathcal{F} = \mathcal{G}$), as is often the case for learning, then the distillation problem is trivial – the algorithm can simply take in $f \in \mathcal{F}$ and output $f \in \mathcal{G}$. The distillation problem is challenging and interesting when $\mathcal{F}$ and $\mathcal{G}$ are distinct function classes.

2.2.1 Agnostic PAC-distillation

We also define an *agnostic* variant of PAC-distillation. The goal is to find $g \in \mathcal{G}$ that competes with the best possible target function in $\mathcal{G}$ instead of having small absolute error.

[1] Typically, time and samples are required to scale polynomially in the input and classifier size for PAC-learnability. The definition here is more convenient to discuss fine-grained runtime and sample complexity bounds as in this paper.

Definition 2.3 (Agnostic PAC-distillation)**.** *Source class $\mathcal{F}$ is (ϵ, δ)-agnostically distillable into target class $\mathcal{G}$ by algorithm $\mathcal{A}_{\text{dist}}$ in n samples and t time if for any distribution $\mathcal{D}$ on $\mathcal{X}$, and any source model $f \in \mathcal{F}$, the algorithm runs in $\leq t$ time and:*

$$\mathbb{P}_{S\sim\mathcal{D}^n}[\text{error}_{f,\mathcal{D}}(\mathcal{A}_{dist}(S, f)) - \min_{g\in\mathcal{G}} \text{error}_{f,\mathcal{D}}(g) \leq \epsilon] \geq 1 - \delta\,.$$

The term *agnostic* is inherited from *agnostic PAC-learning*. See Section 6 for further extensions.

3 Two case studies in distilling neural networks

Since neural networks are the most common class of "black-box functions" in practice, we motivate PAC-distillation with two case studies showing how to distill neural networks that have learned simple logical functions. Our algorithms distill the neural networks into more interpretable forms in time faster than required to learn from scratch:

- In Section 3.1, we warm up with the setting when the neural network has implicitly learned a *junta*, i.e., a function of the input that depends only on a small subset of the features (formal definition below). Here, we use previous results on query learning [BC16] to efficiently distill the model into an *explicit representation* of the junta.
- In Section 3.2, we consider the more general setting when the neural network has implicitly learned a *decision tree*. We provide a new result, showing that under the "linear representation hypothesis" (see, e.g., [MYZ13, EHO+22] and further references in Appendix A) we can distill the neural network into a decision tree in polynomial time.

3.1 Warm-up: provably-efficient distillation of networks into juntas

We warm up with a previously-known result on learning juntas with queries [KM91, Dam98a, Dam98b, DBGV05, BC16]; this will set the stage for our new result in Sections 3.2.

Definition 3.1 (Junta)**.** *A k-junta is a function $f : \{0,1\}^d \to \{0,1\}$ that depends only on a subset $S \subseteq [d]$ of the coordinates, of size $|S| = k$. Formally, $f(\boldsymbol{x}) = h(\boldsymbol{x}_S)$ for some $h : \{0,1\}^k \to \{0,1\}$.*

Learning juntas from random examples is hard Learning k-juntas from noisy data is widely held to be a computationally hard problem: it is conjectured to require at least $\Omega(d^k)$ operations as the dimension d grows [BFJ+94]. This hardness result has been proved rigorously under restricted models of computation such as the Statistical Query (SQ) framework [BFJ+94].

Can we distill networks into juntas efficiently? Is it possible to distill a neural network into a junta in significantly fewer operations than learning the junta from scratch, avoiding the $\Omega(d^k)$ complexity barrier, and instead running in $d^{O(1)}$ time, where the constant in the exponent does not grow with k?

Formally, suppose that we are given a vector of parameters θ that specify a neural network $f_\theta : \{0,1\}^d \to \mathbb{R}$ that implicitly computes a k-junta. We seek to extract an explicit representation of the junta in terms of the subset $S \subseteq [d]$ of the coordinates and a truth table of the function on those coordinates. In other words, we seek to distill from the class

$$\mathcal{F}_{\mathsf{NN},k\text{-juntas}} = \{\text{neural networks } f_\theta \text{ which implicitly compute a } k\text{-junta}\}$$

into the class of k-juntas

$$\mathcal{F}_{k\text{-juntas}} = \{k\text{-juntas, explicitly represented by } S \text{ and } h : \{0,1\}^k \to \{0,1\}\}.$$

This is a nontrivial computational problem, since the neural networks in $\mathcal{F}_{\mathsf{NN},k\text{-juntas}}$ are represented by the vector of weights θ, whereas the functions in $\mathcal{F}_{k\text{-juntas}}$ are represented by the subset of the coordinates on which they depend, and a 2^k-size truth table on those coordinates.

We have been intentionally vague about the neural network's architecture, because it turns out that the architecture does not matter. Using only evaluation access to the network, it is possible to distill the network into a junta very efficiently. This follows from a previously-known result on learning with queries:

Proposition 3.2 (Efficient distillation of networks into juntas; cf. [BC16])**.** *Suppose that each network evaluation takes T time. Then, for any $0 < \delta < 1$, we can $(\epsilon = 0, \delta)$-distill $\mathcal{F}_{\mathsf{NN},k\text{-juntas}}$ into $\mathcal{F}_{k\text{-juntas}}$ in 0 samples and $O(T \cdot k \log d + T \cdot 2^k \log \frac{1}{\delta}) + \mathrm{poly}(2^k, d)$ time.*

Proof. Directly implied by Theorem 13 of [BC16] on the query complexity of learning juntas. Each query can be simulated by one evaluation of the neural network. □

The key points are that: (a) learning the k-junta from noisy data takes at least $\Omega(d^k)$ time under standard hardness assumptions [BFJ+94]; (b) we can distill an explicit representation of the junta from the trained neural network much faster, in $\mathrm{poly}(d, 2^k)$ network evaluations and time, where the exponent on d does not scale in k.

Significance of this result The distillation phase has vanishing computational cost compared to the learning phase. This supports a "*Learn first, distill later*" paradigm of data science, where the bulk of the compute is first dedicated to training a model that fits the data using a flexible and general-purpose learning algorithm such as training a neural network[2], and then the knowledge in this network can be efficiently extracted at much lower cost.

3.2 Provably-efficient distillation of networks into decision trees

Next, we consider the more general setting of distilling neural networks into decision trees, which are popular classifiers that are often touted for their transparency [BFOS84, RCC+22].

We provide a novel algorithm for distilling neural networks into decision trees in polynomial time, which is faster than the current fastest algorithms for learning decision trees from data or queries [EH89, BLQT22], but our algorithm uses the neural network's structure beyond evaluation access to the network.

Recall the definition of a decision tree; see Figure 2 for an example.

Definition 3.3 (Decision tree)**.** *A decision tree T is a binary tree, where each internal node is labeled by a variable in $\{x_1, \ldots, x_d\}$, and each leaf is labeled by the output 0 or 1. In a notational overlaod, the tree represents a function $T : \{0,1\}^d \to \{0,1\}$ where $T(\boldsymbol{x})$ is computed by starting at the root node, and following the path to the leaves determined by $\boldsymbol{x}$.*

[2] In fact, the model training for learning juntas runs in the conjectured optimal run-time [BEG+22, ABM23, EGK+23], and the runtime adapts to hierarchical structure in the data [ABB+21, ABM22, AB22, ABM23].

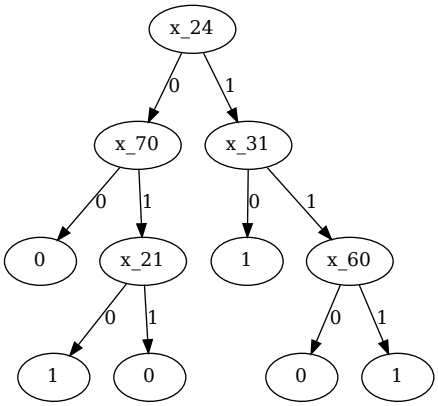


Figure 2: Example of a depth-3 decision tree with 11 nodes and $d = 100$.

Learning decision trees from random examples is hard The problem of finding the smallest decision tree from random examples that fits the data exactly or approximately is NP-hard [Ang, PV88, KST23b, KST23a], which suggests that no polynomial time algorithm may be possible. The fastest approximation algorithm for learning decision trees from data runs in $d^{O(\log(s))}$-time and returns a tree that may be of size $d^{\Omega(\log(s))}$ [EH89], assuming is a ground-truth size-s tree that fits the data perfectly. This algorithm matches corresponding $d^{\Omega(\log(s))}$-complexity lower bounds for learning in the Statistical Query (SQ) model [BFJ+94], indicating it might be optimal.

Learning decision trees from queries is also hard Since learning decision trees in NP-hard, we can ask analogously to the previous section: is it easier to distill a model into a decision tree, than to learn the decision tree from scratch? Formally, is it possible to distill the class

$$\mathcal{F}_{\mathsf{NN},\text{size-}s,\text{depth-}r \text{ trees}} = \{\text{neural networks } f_\theta \text{ which implicitly compute decision trees}\}$$

into the class

$$\mathcal{F}_{\text{size-}s,\text{depth-}r \text{ trees}} = \{\text{decision trees, explicitly represented}\},$$

in polynomial time and number of samples?

Unfortunately, if we only use evaluation access to the network this problem is still NP-hard [KST23a]. And even under the extra assumption that the data distribution $\mathcal{D}$ is uniform on $\{0,1\}^d$, the current best algorithm runs in $\tilde{O}(d^2) \cdot (s/\epsilon)^{O(\log((\log s)/\epsilon))}$ time [BLQT22]. This is better than the naive $d^{O(\log(s))}$ algorithm learning from random samples, but does not yet achieve polynomial time.

We will use additional neural network structure to distill in polynomial time We sidestep the limitations of previous decision tree learning algorithms by using extra structure from the neural network, *beyond evaluation access to the network*. This will ultimately allow us to distill networks into decision trees in polynomial time.[3]

[3]Our algorithm runs $\text{poly}(d, 1/\epsilon, s)$ time when the input distribution is uniform, and $\text{poly}(d, 1/\epsilon, s, 2^r)$ time for arbitrary non-uniform distributions – see Theorem 3.6. For arbitrary non-uniform distributions and $r = O(\log(s))$, this is $\text{poly}(d, 1/\epsilon, s)$ time, whereas previously known algorithms require $d^{\Omega(\log(s))}$-time although they do not require the Linear Representation Hypothesis.

We must first explain the structure of the network that our distillation algorithm uses. We use the vector-valued representation map $\varphi_\theta : \mathcal{X} \to \mathbb{R}^m$ associated with the neural network. The representation $\varphi_\theta(\boldsymbol{x}) \in \mathbb{R}^m$ is the concatenation of all of the internal neurons' activations under input $\boldsymbol{x}$.[4] Surprisingly, this representation $\varphi_\theta(\boldsymbol{x})$ can often represent useful high-level features of the data via linear regression. This finding was called the "linear representation hypothesis" (LRH) by [EHO+22]. Formally:

Definition 3.4 (Linear representation hypothesis)**.** *Let $\mathcal{G}$ be a collection of functions $g : \mathcal{X} \to \mathbb{R}$ for each $g \in \mathcal{G}$. For any $\tau > 0$, the τ-LRH with respect to $\mathcal{G}$ states that for all $g \in \mathcal{G}$ there is a coefficient vector $\boldsymbol{w}_g \in \mathbb{R}^m$ such that $\|\boldsymbol{w}_g\| \leq \tau$ and*

$$\boldsymbol{w}_g \cdot \varphi(\boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x} \in \mathcal{X}\,.$$

The parameter τ is a bound on the norm of the coefficients that controls the complexity of the linear combination of the features in φ_θ. A smaller τ means that the network can more easily represent the functions in $\mathcal{G}$. The set $\mathcal{G}$ varies depending on the setting, but should be thought of as a collection of "high-level features" that could include important intermediate steps in the computation of the neural network.

There is rapidly mounting empirical [MYZ13, THGN23, SPK16, CKL+18, AL23a, LHB+22, NLW23, MT23, LNA21, DDH+21, AL23b, Fel98, AMS21, BCZ+16, VC20, RTGC22, RGC23, BSJR+23, WGNV23, HLA23, PCV23, BLSR22] and theoretical [ALL+16, ABM22, DLS22, BES+22, MHPG+22, ABM23, DKL+23, BBPV23] evidence across several domains that trained neural networks satisfy the LRH, for appropriate sets of features $\mathcal{G}$. To keep the presentation focused on decision tree distillation, we discuss evidence for the LRH in Appendix A.

The LRH for decision trees In decision trees, the intermediate computations of the decision tree correspond to all paths of the tree that start at the root. We postulate that these intermediate computations are linearly encoded by the neural network's learned representation. This is defined formally below.

Definition 3.5. *A* clause *is a set of literals $S \subseteq \{x_1, \ldots, x_d, \neg x_1, \ldots, \neg x_d\}$. The function $\mathrm{AND}_S(\boldsymbol{x})$ is defined to be 1 if all literals in S are true, and 0 otherwise. We say that S is a nondegenerate k-clause if $|S| = k$ and each variable appears at most once in S (i.e., for all i, we have $|S \cap \{x_i, \neg x_i\}| \leq 1$).*

Given a decision tree T, the set of intermediate computations is:

$$\mathcal{G}_T = \{\mathrm{AND}_S(\boldsymbol{x}) \mid \text{clauses } S \text{ formed by a path starting at the root of } T\}$$

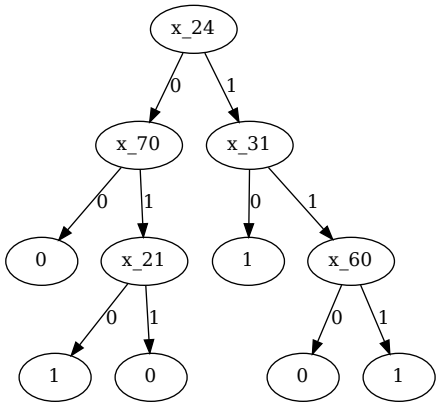
See Figure 3 for an example.

Networks that have learned decision trees and satisfy the LRH belong to the set

$$\mathcal{F}^{\tau\text{-LRH}}_{\mathsf{NN},\text{size-}s,\text{depth-}r \text{ trees}} = \{\text{neural networks } f_\theta \text{ which implicitly compute a decision tree } T \text{ such that the network's representation } \varphi_\theta \text{ satisfies } \tau\text{-LRH for features } \mathcal{G}_T\}$$

To make sure that these definitions are clear, we restate them in slightly different form. Let T be a decision tree, and let $S \subseteq \{x_1, \ldots, x_d, \neg x_1, \ldots, \neg x_d\}$ be a clause formed by taking a path in the

[4] Sometimes φ_θ is instead defined as the penultimate layer of activations, or the gradient of the network with respect to the weights.



(a)

Intermediate computations $\mathcal{G}_T$:

1

$\neg x_{24}$

x_{24}

$\neg x_{24} \wedge \neg x_{70}$

$\neg x_{24} \wedge x_{70}$

$x_{24} \wedge \neg x_{31}$

$x_{24} \wedge x_{31}$

$\neg x_{24} \wedge x_{70} \wedge x_{21}$

$\neg x_{24} \wedge x_{70} \wedge x_{21}$

$x_{24} \wedge x_{31} \wedge \neg x_{60}$

$x_{24} \wedge x_{31} \wedge x_{60}$

(b)

Figure 3: (a) Example of a depth-3 decision tree with 11 nodes and $d = 100$. (b) List of all intermediate computations in this tree; there is one AND function per path starting at the root. Note that these paths do not have to end at the leaves. For convenience, we also include the path of length zero, which is why $\mathrm{AND}_\emptyset \equiv 1$ is also one of the intermediate computations.

tree starting from the root. Then, a network $f_\theta \in \mathcal{F}^{\tau\text{-LRH}}_{\mathsf{NN},\text{size-}s,\text{depth-}r \text{ trees}}$ which has implicitly learned T will have a representation φ_θ such that there is a vector of coefficients $\boldsymbol{w}_S \in \mathbb{R}^m$ of bounded norm $\|\boldsymbol{w}_S\| \leq \tau$ such that

$$\boldsymbol{w}_S \cdot \varphi_\theta(\boldsymbol{x}) = \mathrm{AND}_S(\boldsymbol{x}) \text{ for all } \boldsymbol{x}\,.$$

We are now ready to present our main result in this subsection, which is a polynomial time algorithm for distilling networks into decision trees, under the LRH condition. Each evaluation of the network or the network's representation is assumed to take unit time.

Theorem 3.6 (Under LRH, networks computing decision trees can be efficiently distilled)**.** *For any $\epsilon, \delta \in (0,1)$, there is an algorithm that (ϵ,δ)-distills from $\mathcal{F}^{\tau\text{-LRH}}_{\mathsf{NN},\text{size-}s,\text{depth-}r \text{ trees}}$ to $\mathcal{F}_{\text{size-}s,\text{depth-}r \text{ trees}}$ and runs in $\mathrm{poly}(d, m, 1/\epsilon, s, 2^r, \log(1/\delta), \tau, B)$ time and $\mathrm{poly}(1/\epsilon, s, \log(d/\delta), \log(\tau B))$ samples from $\mathcal{D}$, where $B \geq \max_{\boldsymbol{x}} \|\varphi_\theta(\boldsymbol{x})\|$ is an upper bound on the norm of the network representation.*

Furthermore, if $\mathcal{D}$ is uniform over $\{+1,-1\}$, then there is an algorithm that (ϵ,δ)-distills from $\mathcal{F}^{\tau\text{-LRH}}_{\mathsf{NN},\text{size-}s,\text{depth-}r \text{ trees}}$ to $\mathcal{F}_{\text{size-}s,\text{depth-}r \text{ trees}}$ in $\mathrm{poly}(d, m, 1/\epsilon, s, \log(1/\delta), \tau, B)$ time.

Significance of this result Similarly to Proposition 3.2, this result supports the "*Learn first, distill later*" paradigm. If $\tau, B, m \leq \mathrm{poly}(d)$ are polynomial-size, then our distillation algorithm takes only polynomial $\mathrm{poly}(d, 2^r)$ time. On the other hand, learning a decision tree from random examples is conjectured to require the much larger time $d^{\Omega(r)}$, because r-juntas can be encoded as decision trees of depth r with size 2^r. Thus, once again, the bulk of the compute and data is required for the learning phase, and the distillation phase can be much cheaper.

Furthermore, this result illustrates that using extra neural network structure beyond query access can aid the design of distillation algorithms that run faster than the best known query learners.

For arbitrary distributions, our distillation algorithm takes exponential time in the depth, and thus is truly polynomial-time only when $r = O(\log(s))$ It is an interesting open problem whether it is possible to obtain polynomial dependence on the depth under the LRH condition.

3.2.1 Proof of Theorem 3.6

A building block of our distillation procedure is a "linear probe" subroutine, which checks whether a function $g : \mathcal{X} \to \mathbb{R}$ can be approximated by a low-norm linear function of a representation $\varphi : \mathcal{X} \to \mathbb{R}^m$.

Lemma 3.7 (Linear probe subroutine)**.** *Given a function $g : \mathcal{X} \to [-1, 1]$, a representation map $\varphi : \mathcal{X} \to \mathbb{R}^m$ of norm bounded by $B \geq \max_{\boldsymbol{x}} \|\varphi(\boldsymbol{x})\|$, a coefficient norm-bound $\tau > 0$, error tolerance parameters $\epsilon, \delta > 0$, and a distribution $\mathcal{D}$, there is a subroutine* LinearProbe$(g, \varphi, B, \tau, \epsilon, \delta, \mathcal{D})$ *that runs in* $\mathrm{poly}(1/\epsilon, \log(1/\delta), \tau, B, m)$ *time and draws* $\mathrm{poly}(1/\epsilon, \log(1/\delta), \tau, B)$ *samples from distribution $\mathcal{D}$ and returns* probe $\in \{\mathsf{true}, \mathsf{false}\}$ *such that:*

- *If there is $\boldsymbol{w} \in \mathbb{R}^m$ such that $\|\boldsymbol{w}\| \leq \tau$ and $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \leq \epsilon$, then* probe $=$ true *with probability $\geq 1 - \delta$.*
- *If for all $\boldsymbol{w} \in \mathbb{R}^m$ such that $\|\boldsymbol{w}\| \leq \tau$ we have $\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \geq 2\epsilon$, then* probe $=$ false *with probability $\geq 1 - \delta$.*

Proof of Lemma 3.7. The proof is a standard application of convex optimization and generalization bounds based on Rademacher complexity. See Appendix B.1.1. □

With this subroutine in mind, we present the pseudocode for our distillation procedure in Algorithm 1. This algorithm proceeds in two phases. First, there is a a search phase, where we extract a set of AND functions that can be efficiently represented by the network using the LinearProbe subroutine. Second, there is a stitching phase, where we use dynamic programming to combine these AND functions that we have found to construct the best decision tree.

We hope that this two-phase blueprint will be generally useful for distillation algorithms in other settings: these algorithms could first extract a candidate set of high-level features from the network, and then distill to a model that uses these features as the backbone of its intermediate computations.

Algorithm 1 Distilling a neural network into a decision tree; Theorem 3.6

Inputs: Neural net $f_\theta : \{0,1\}^d \to \{0,1\}$ and its representation $\varphi_\theta : \{0,1\}^d \to \mathbb{R}^m$
Access to random samples from distribution $\mathcal{D}$
Hyperparameters: $R \in \mathbb{N}$, $\epsilon, \delta > 0$

Output: Decision tree $\hat{T}$

Phase 1: Search for ANDs that network can efficiently represent.

$\mathcal{S}_0 \leftarrow \{\emptyset\}$

for $i = 0$ to $i = R-1$ **do**

  $\mathcal{P}_i \leftarrow \{S \in \mathcal{S}_i \text{ s.t. } \textsc{LinearProbe}(\text{AND}_S, \varphi_\theta, B, \tau, 2^{-i-3}, \frac{\delta}{2|\mathcal{S}_i|R}, \text{Unif}\{0,1\}^d) = \mathsf{true}\}$

  $\mathcal{S}_{i+1} \leftarrow \bigcup_{S \in \mathcal{P}_i} \textsc{Successors}(S)$ # defined in (3.1)

end for

$\mathcal{S} \leftarrow \mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_R$

Phase 2: Construct best decision tree.

$\hat{v}_S \leftarrow \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[\text{AND}_S(\boldsymbol{x})(2f_\theta(\boldsymbol{x}) - 1)] \pm \epsilon/s$ **for each** $S \in \mathcal{S}$

Return decision tree $\hat{T}$ maximizing $\text{val}(\hat{T}, \hat{\boldsymbol{v}})$, over all decision trees with $\mathcal{G}_{\hat{T}} \subseteq \{\text{AND}_S \mid S \in \mathcal{S}\}$

Details for Phase 1 In the first phase of the algorithm, we iteratively construct a sequence of sets $\mathcal{S}_0, \ldots, \mathcal{S}_R$ of clauses, which will satisfy two properties with high probability:

(P1) The sets contain intermediate computations of the decision tree: for any $\text{AND}_S \in \mathcal{G}_T$ which is an AND function on $|S| = k$ variables where $k \le R$, we have $S \in \mathcal{S}_k$.

(P2) The size of the sets $\mathcal{S}_i$ is bounded: $|\mathcal{S}_i| \le 2^{3R+2}\tau^2 B^2 d$.

In order to ensure property (P1), we iteratively grow the sets $\mathcal{S}_i$ using the Successors operation, which returns the set of clauses that could possibly be formed by taking one more step down the depth of the decision tree starting at a certain clause. For any clause $S \subseteq \{x_1, \ldots, x_d, \neg x_1, \ldots, \neg x_d\}$, the successors are:

$$\textsc{Successors}(S) = \{S \cup \ell \mid \ell \in \{x_1, \ldots, x_d, \neg x_1, \ldots, \neg x_d\} \text{ and } \ell \notin S \text{ and } \neg\ell \notin S\}. \tag{3.1}$$

For example, the successors of $S = \emptyset$ consist of all 1-clauses $\{x_1\}, \ldots, \{x_d\}$ and $\{\neg x_1\}, \ldots, \{\neg x_d\}$. Similarly, the successors of $S = \{\neg x_1\}$ consist of all 2-clauses where one of the literals is $\neg x_1$ and the other literal is neither x_1 nor x_2, i.e., $\{\neg x_1, x_j\}$ for all $j \neq 1$ and, $\{\neg x_1, \neg x_j\}$ for all $j \neq 1$.

In order to ensure property (P2), that the sets $\mathcal{S}_i$ do not grow too large, we prune the search on each iteration by only adding the successors of clauses S such that AND_S that can be efficiently linearly represented by the network. Our key lemma controls the number of distinct AND functions that can be approximately represented as low-norm linear combinations of the features in φ_θ, and this means that the sizes of the sets $\mathcal{S}_i$ cannot grow too large.

Lemma 3.8 (Linearly representing many ANDs requires high norm)**.** *Let $\mathcal{S}$ be a collection of nondegenerate k-clauses. Let $\mathcal{G} = \{\mathrm{AND}_S \text{ for all } S \in \mathcal{S}\}$. Suppose that $\varphi : \{0,1\}^d \to \mathbb{R}^m$ approximately satisfies the τ-LRH with respect to $\mathcal{G}$ in the sense that for $g \in \mathcal{G}$ there is $\boldsymbol{w}_g \in \mathbb{R}^m$ with $\|\boldsymbol{w}_g\| \le \tau$ such that*

$$\mathbb{E}_{\boldsymbol{x}\sim\mathrm{Unif}\{0,1\}^d}[(\boldsymbol{w}_g \cdot \varphi(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \le 2^{-k-2}\,.$$

Then

$$|\mathcal{S}| \le 2^{3k+4}\tau^2 \mathbb{E}_{\boldsymbol{x}\sim\mathrm{Unif}\{0,1\}^d}[\|\varphi(\boldsymbol{x})\|^2]\,.$$

Proof sketch. The proof of the lemma is deferred to Appendix B.1.2. The key ingredient in the proof is to show that there is a subspace $\Omega \subseteq L^2(\{0,1\}^d)$ of function space such that if we project the set of functions in $\mathcal{G} = \{\mathrm{AND}_S \text{ for all } S \in \mathcal{S}\}$ to Ω, then $P_\Omega \mathcal{G}$ cannot lie in a small-volume ball. We show that if $|\mathcal{S}|$ is very large, then this contradicts the LRH. The subspace Ω that we choose is the subspace of degree-k polynomials, and our arguments are Fourier-analytic and rely on writing each AND_S function as a polynomial. □

Let us apply this lemma to conclude the proof of correctness of Phase 1. For each $i \in \{0, \dots, R-1\}$, let E_i be the event that:

- the set $\mathcal{S}_i$ contains all of the i-clauses in $\mathcal{G}_T$ (i.e., it contains all of the intermediate computations of the decision tree at depth i), and
- all calls to LinearProbe when forming the set $\mathcal{P}_i$ satisfy the guarantees of Lemma 3.7.

For any $0 \le i \le R-1$, conditioned on events $E_0, E_1, \dots, E_i$, it follows that:

- The set $\mathcal{P}_i$ contains all i-clauses of $\mathcal{G}_T$, because $\mathcal{S}_i$ contains all i-clause of $\mathcal{G}_T$, and $\mathcal{S}_i \cap \mathcal{G}_T$ satisfies the τ-LRH. Therefore $\mathcal{S}_{i+1}$ contains all $(i+1)$-clauses of $\mathcal{G}_T$, because it contains all possible successors to the i-clauses of $\mathcal{G}_T$.
- For every clause $S \in \mathcal{P}_i$, there is $\boldsymbol{w} \in \mathbb{R}^m$ such that $\|\boldsymbol{w}\| \le \tau$ and $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \le 2^{-i-2}$. Therefore by Lemma 3.8, we have $|\mathcal{P}_i| \le 2^{3i+4}\tau^2 B^2$. For each clause, there are at most two successors, so $|\mathcal{S}_{i+1}| \le 2d|\mathcal{P}_i| \le 2^{3i+5}\tau^2 B^2 d$.

A union bound on the event that each call to LinearProbe has the guarantees of Lemma 3.7 implies:

$$\mathbb{P}[E_{i+1} \mid E_0, \dots, E_i] \ge 1 - \delta/(2R)\,.$$

The base case of $\mathbb{P}[E_0] \ge 1 - \delta/(2R)$ is also easy to see since we start with $\mathcal{S} = \{\emptyset\}$ and $\emptyset$ is the only 0-clause. Thus, by induction on i and a union bound over the probability of error over all calls of LinearProbe, we have proved that

$$\mathbb{P}[E_0, \dots, E_R] \ge 1 - \delta/2\,.$$

These events imply the correctness of Phase 1, which are the statements that $\mathcal{S}_i$ contains all i-clauses of $\mathcal{G}_T$ and each set $\mathcal{S}_i$ is of bounded size:

$$\text{Properties (P1) and (P2) hold with probability at least } 1 - \delta/2\,.$$

Notice also that under events $E_0, \dots, E_R$ this phase takes time $\mathrm{poly}(2^R, \tau, B, d)$ since the sets are of size $|\mathcal{P}_i| \le |\mathcal{S}_i| \le 2^{3i+2}\tau^2 B^2 d$.

Details for Phase 2 In the second phase, we use a well-known dynamic programming technique from [GLR99, MR02] to find the best decision tree whose intermediate computations are contained in $\mathcal{S}$. By property (P1) of Phase 1, we know that restricting to trees with clauses in $\mathcal{S}$ is without loss of generality, since the ground truth tree's intermediate computations are contained in $\mathcal{S}$.

Thus, in the discussion below, we only consider decision trees whose internal nodes are contained in the set $\mathcal{S}$. For a decision tree $\tilde{T}$, each node can be identified with a clause S. The leaf clauses are denoted by $\text{Leaves}(\tilde{T})$. For any leaf clause S, let $\tilde{T}(S) \in \{0,1\}$ denote the output. If the nodes of $\tilde{T}$ are contained in $\mathcal{S}$, then for any $\boldsymbol{u} \in \mathbb{R}^{\mathcal{S}}$ we can define

$$\text{val}(\tilde{T}, \boldsymbol{u}) = \sum_{S \in \text{Leaves}(\tilde{T})} u_S(2\tilde{T}(S) - 1)\,.$$

If we define $\boldsymbol{v} \in \mathbb{R}^S$ with $v_S = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\text{AND}_S(\boldsymbol{x})(2f_\theta(\boldsymbol{x}) - 1)]$, then

$$\text{val}(\tilde{T}, \boldsymbol{v}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\sum_{S \in \text{Leaves}(\tilde{T})} \text{AND}_S(\boldsymbol{x})(2f_\theta(\boldsymbol{x}) - 1)(2\tilde{T}(S) - 1)] = 2\mathbb{P}_{\boldsymbol{x}\sim\mathcal{D}}[\tilde{T}(\boldsymbol{x}) = f_\theta(\boldsymbol{x})] - 1\,.$$

Therefore, if we have an approximation $\hat{\boldsymbol{v}}$ to $\boldsymbol{v}$ such that $|\hat{v}_S - v_S| \le \epsilon/s$ for all S, we have

$$|\text{val}(\tilde{T}, \hat{\boldsymbol{v}}) - \text{val}(\tilde{T}, \boldsymbol{v})| \le \epsilon\,.$$

Therefore, a size-s, depth-r tree $\hat{T}$ that maximizes $\text{val}(\hat{T}, \hat{\boldsymbol{v}})$ is an ϵ-approximately optimal size-s tree fitting the neural network f_θ.

It remains to show that $\hat{\boldsymbol{v}}$ can be constructed efficiently, and that this optimization can be performed efficiently. The estimation of $\hat{\boldsymbol{v}}$ can be performed in $\text{poly}(1/\epsilon, \log(|S|/\delta))$ samples and time from $\mathcal{D}$ by a Hoeffding bound, and we have $\|\boldsymbol{v} - \hat{\boldsymbol{v}}\|_\infty \le \epsilon$ with probability $\ge 1 - \delta/2$. The maximization of $\text{val}(\hat{T}, \hat{\boldsymbol{v}})$ can be performed in $\text{poly}(|S|, d, s)$ time by a dynamic program that computes, for each clause $S \in \mathcal{S}$ and each tree size $0 \le s' \le s$, the subtree $\hat{T}_{S,s'}$ of size s' rooted at S that maximizes $\text{val}(\hat{T}_{S,s'}, \hat{\boldsymbol{v}})$ [GLR99, MR02].

By guarantee (P2) of Phase 1, we know that $|\mathcal{S}| \le \text{poly}(2^R, \tau B, d)$. Therefore, putting the guarantees of Phase 1 and Phase 2 together proves the theorem for arbitrary distributions if we choose $R = r$ to be the depth of the tree – since then total number of samples is $\text{poly}(1/\epsilon, s, \log(\tau B/\delta))$ and runtime is $\text{poly}(1/\epsilon, 2^r, \tau B, \log(1/\delta))$. For the uniform distribution, we can choose $R = \min(r, O(\log(s/\epsilon)))$, since there is a subtree of depth $r = O(\log(s/\epsilon))$ that ϵ-approximates the original tree under this distribution. □

3.2.2 Empirical evaluation

We have proved that the distillation procedure from neural networks to decision trees is efficient *under the linear representation hypothesis*. It remains to check that this hypothesis is valid for trained networks that have learned decision trees. We provide experimental support for this in this subsection, by implementing Algorithm 1. We show that it can successfully distill models that have been trained on uniformly random trees with $d = 100$ and a certain depth from $\{2, 3, 4, 5\}$. In Figure 4 we visualize examples of these random trees on which we train our network.

For practical implementation purposes, we make one modification to our linear probe subroutine: in the construction of the set $\mathcal{P}_i$ on Line 6 we train our linear probe on 1000 samples with Adam and filter out all but the k AND functions where the probe has the lowest validation loss on a held-out dataset of 10000 samples. Each linear probe is trained for 100 iterations on 1000 samples. In Figure 5 we report the number of linear probes conducted by our algorithm with varying k, compared to the reconstruction accuracy of the final tree in terms of how well it approximates the true

tree computed by the model. This figure shows that our algorithm only needs to probe a very small fraction of the total number of possible AND functions that what would be checked by a brute-force exhaustive search method. See https://github.com/eboix/theory-of-model-distillation for the code.

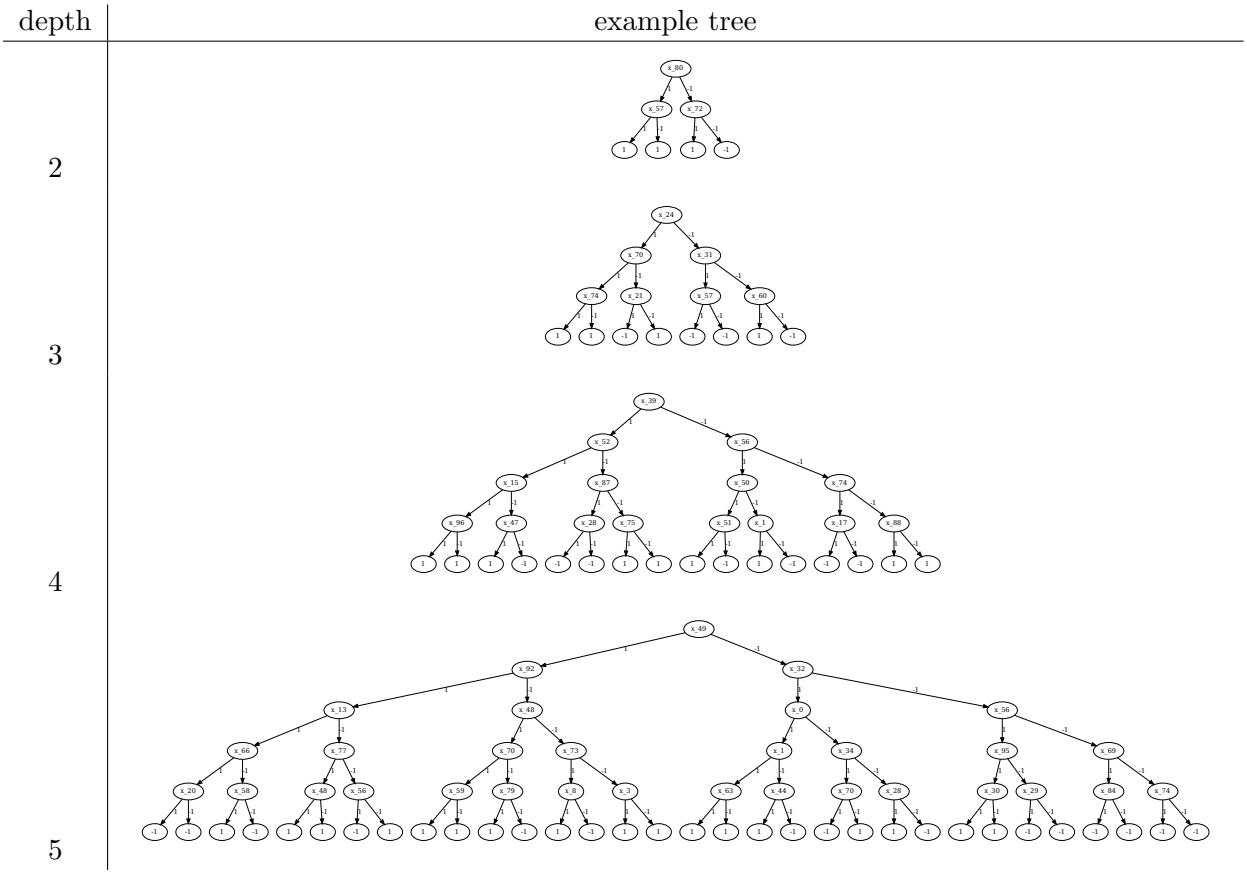


Figure 4: Example random trees of varying depths on which we benchmark our distillation algorithm. We train a 5-layer ResNet network to learn the tree based on 100,000 random samples for the depth-2,3,4 trees and 1,000,000 random samples for the depth-5 trees. Given the trained network, we then recover from the trained network using the distillation procedure. The input space is $\{0,1\}^d$ with input dimension $d = 100$. See Figure 5 for numerical details.

depth	k	fraction of inputs distillation is correct	average number of probes	fraction of possible probes
2	10	$\{1.00, 1.00, 1.00, 1.00, 1.00\}$	2141	0.107045
2	50	$\{1.00, 1.00, 1.00, 1.00, 1.00\}$	8901	0.445028
2	100	$\{1.00, 1.00, 1.00, 1.00, 1.00\}$	15101	0.755012
2	**200**	**{1.00, 1.00, 1.00, 1.00, 1.00}**	**20001**	**1.000000**
3	10	$\{1.00, 1.00, 1.00, 0.87, 1.00\}$	4090	0.003113
3	50	$\{1.00, 1.00, 1.00, 1.00, 1.00\}$	18341	0.013962
3	100	$\{1.00, 1.00, 1.00, 1.00, 1.00\}$	32373	0.024644
3	**200**	**{1.00, 1.00, 1.00, 1.00, 1.00}**	**48894**	**0.037222**
4	10	$\{0.87, 0.81, 0.88, 0.74, 0.75\}$	6023	0.000094
4	50	$\{0.87, 1.00, 0.94, 0.94, 0.87\}$	28035	0.000438
4	100	$\{1.00, 1.00, 1.00, 1.00, 0.94\}$	52943	0.000827
4	**200**	**{1.00, 1.00, 1.00, 1.00, 1.00}**	**93519**	**0.001460**
5	10	$\{0.81, 0.73, 0.69, 0.72, 0.75\}$	7930	0.000003
5	50	$\{0.91, 0.81, 0.84, 0.86, 0.85\}$	37413	0.000015
5	100	$\{0.97, 0.94, 0.91, 0.94, 0.87\}$	71525	0.000029
5	200	$\{1.00, 1.00, 0.94, 1.00, 0.96\}$	131617	0.000053
5	500	$\{1.00, 1.00, 0.97, 1.00, 0.97\}$	289190	0.000117
5	**1000**	**{1.00, 1.00, 0.97, 1.00, 1.00}**	**539362**	**0.000218**

Figure 5: For each tree depth in $\{2, 3, 4, 5\}$ we generate 5 random decision trees, train a depth-5 ResNet on each one with the cross-entropy loss, and distill to a decision tree. We report the results when we vary the hyperparameter k, which controls the size to which we prune the set of probes as we explore the space of functions efficiently represented by the network. In the third column, we report the accuracy of the distilled decision tree, which increases as the hyperparameter k increases and the number of probes that the algorithm can execute increases. The average number of probes with a given depth and k is reported in the fourth column, and is accurate across runs up to $\pm 0.5\%$ accuracy. The final column compares this average number of probes to the total number of possible probes of AND functions that the algorithm could make if it were brute-forcing. For depth r, this total number of possible probes is $\sum_{i=0}^{r} 2^i \binom{d}{i}$. We see in the final column that our algorithm requires only a very small fraction of the brute-force number of probes to succeed in recovering the true tree, supporting the linear representation hypothesis for networks trained on random decision trees. We leave more in-depth experiments beyond synthetic settings to future work.

4 Computational theory: a web of reductions

In this section, we initiate a general computational theory of PAC-distillation. The core idea is that we should view distillation as a reduction between two hypothesis classes, because it satisfies a transitivity property. In order to define these reductions formally, we must define what it means to distill from class $\mathcal{F}$ to class $\mathcal{G}$ in polynomial time. In this section, we assume that each class $\mathcal{F}$ is equipped with a function $\mathsf{size} : \mathcal{F} \to \mathbb{N}$ that gives the representation size of a hypothesis.

Definition 4.1 (Polynomial-time distillation)**.** *Class $\mathcal{F}$ can be distilled into class $\mathcal{G}$ in polynomial time if there is a polynomial $p(\cdot,\cdot,\cdot)$ such that for any $\epsilon, \delta \in (0,1)$ and $m \in \mathbb{N}$, we can (ϵ,δ)-distill from the class $\mathcal{F}_m = \{f \in \mathcal{F} : \mathsf{size}(f) = m\}$ into the class $\mathcal{G}$ in less than $p(1/\epsilon, \delta, m)$ time and samples.*

For technical reasons, we will need a slightly more general definition: distilling a sequence of classes into another sequence of classes in polynomial time. This is more general than the above definition because we can always take $\mathcal{I}$ to be a singleton set.

Definition 4.2 (Polynomial-time distillation between sequences of classes)**.** *Let $\mathcal{I}$ be an index set. Classes $\{\mathcal{F}_i\}_{i\in\mathcal{I}}$ can be distilled into classes $\{\mathcal{G}_i\}_{i\in\mathcal{I}}$ in polynomial time if there is a polynomial $p(\cdot,\cdot,\cdot)$ such that for any $i \in \mathcal{I}$, the class $\mathcal{F}_i$ can be distilled into $\mathcal{G}_i$ in polynomial time bounded by $p(\cdot,\cdot,\cdot)$.*

We use the shorthand $\{\mathcal{F}_i\}_{i\in\mathcal{I}} \lesssim_{p,dst} \{\mathcal{G}_i\}_{i\in\mathcal{I}}$ to denote that the sequence of classes $\{\mathcal{F}_i\}_{i\in\mathcal{I}}$ can be distilled into the sequence of classes $\{\mathcal{G}_i\}_{i\in\mathcal{I}}$ in polynomial time.

Lemma 4.3 (Transitivity of polynomial-time distillation)**.** *If $\{\mathcal{F}_i\}_i \lesssim_{p,dst} \{\mathcal{G}_i\}_i$ and $\{\mathcal{G}_i\}_i \lesssim_{p,dst} \{\mathcal{H}_i\}_i$, then $\{\mathcal{F}_i\}_i \lesssim_{p,dst} \{\mathcal{H}_i\}_i$.*

Proof. Immediate by composing the distillation algorithms for each $i \in \mathcal{I}$, and using a union bound on the probability of error and a triangle bound on the quantity of error. □

Remark 4.4. We remark that for agnostic distillation, the transitivity property in Lemma 4.3 does not hold. However, if $\mathcal{F}$ can be distilled into $\mathcal{G}$ and $\mathcal{G}$ can be agnostically distilled into $\mathcal{H}$, then $\mathcal{F}$ can be agnostically distilled into $\mathcal{H}$; see further discussion in Appendix B.2.

Lemma 4.3 allows us to begin to build a web of reductions between hypothesis classes. This web can be viewed as giving a computational theory for which model classes are uniformly more interpretable than others – if we can efficiently distill from class $\mathcal{F}$ to class $\mathcal{G}$, then we should consider the classifiers in $\mathcal{F}$ to be uniformly at least as interpretable as those in $\mathcal{G}$. Otherwise, given the classifier in $\mathcal{F}$ we could always distill it to a corresponding classifier from $\mathcal{G}$ and use that instead.[5]

We present a very preliminary web of reductions in Figure 4 below, which is based off of the case studies from Section 3. The classes involved are:

- Juntas, as defined in Definition 3.1. This is the sequence of hypothesis classes $\{\mathcal{F}_{k\text{-juntas}}\}_{k\in\mathbb{N}}$, indexed by the size of the junta k. Each k-junta is represented by a subset $S \subseteq [d]$ of the input indices of size $|S| = k$, and a truthtable $h : \{0,1\}^k \to \{0,1\}$ on those indices. We define the representation size of a junta to be $\mathsf{size}((S,h)) = 2^k + d$.

[5] Of course, there are situations in which two model classes may be incomparable in terms of distillation, but one of the model classes may in practice be considered by some users as more interpretable than the other. However, it is less clear in these situations how to develop a mathematical theory of interpretability. Please refer back to the related work in Section 1.2 for more discussion on the literature on interpretability.

- Decision trees as defined in Definition 3.3. We consider the sequence of hypothesis classes $\{\mathcal{F}_{\text{size-}s,\text{depth-}r \text{ trees}}\}_{s\in\mathbb{N},r\in\mathbb{N}}$ indexed by the number of nodes s and depth r of the decision tree. We define the representation complexity of a tree T to be $\mathsf{size}(T) = s + d$, where s is the number of nodes and d is the input dimension.
- Neural networks implicitly computing juntas. This is the sequence of hypothesis classes $\{\mathcal{F}_{\mathsf{NN},k\text{-juntas}}\}_{k\in\mathbb{N}}$, indexed by the size of the junta k. Each neural network f_θ has a representation size given by $\text{size}(f_\theta) = d + 2^k +$ number of network parameters. Here we add the 2^k so that our distillation algorithms are allowed to run in exponential time in k because otherwise it would be impossible to output the truthtable of the k-junta in some cases.
- Neural networks implicitly computing decision trees, under the LRH. This is the sequence of hypothesis classes $\{\cup_{\tau>0}\mathcal{F}^{\tau\text{-LRH}}_{\mathsf{NN},\text{size-}s,\text{depth-}r \text{ trees}}\}_{s\in\mathbb{N},r\in\mathbb{N}}$. Each neural network f_θ with representation φ_θ has representation size given by $\text{size}((f_\theta,\varphi_\theta)) = d + s + \tau \cdot \max_{\boldsymbol{x}} \|\varphi_\theta(\theta)\| +$ number of network parameters. Here we add the s and the dependence on τ and the maximum representation size because our distillation algorithm from Section 3.2 runs in polynomial time in these parameters.

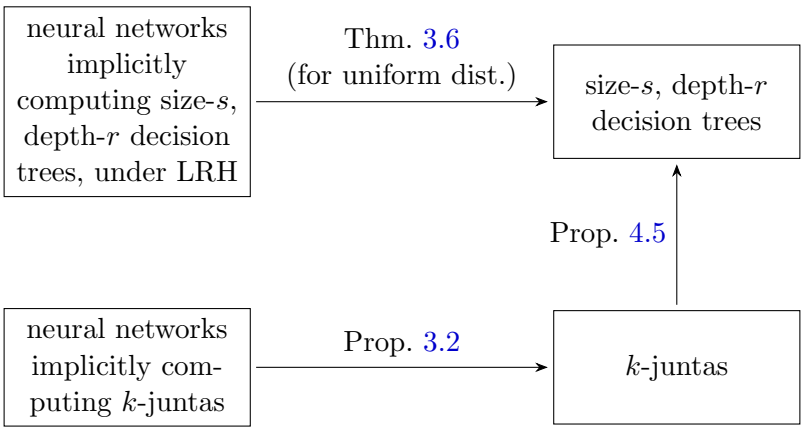


Figure 6: Simple web of reductions based on the distillation algorithms from Section 3. In the distillation from k-juntas to size-s, depth-r trees, we have $s = 2^k$ and $r = k$. By the transitivity property of reductions, this web implies a distillation algorithm from the class of neural networks implicitly computing k-juntas, to the class of size-2^k, depth-k decision trees.

The only missing element in the web is the distillation algorithm from juntas to decision trees, which is folklore:

Proposition 4.5 (Distilling trees into juntas and vice-versa)**.** *$\{\mathcal{F}_{k\text{-juntas}}\}_k$ can be $(\epsilon = 0, \delta = 0)$-distilled into $\{\mathcal{F}_{\text{size-}s,\text{depth-}r \text{ trees}}\}_{s=2^k,r=k}$ in $\text{poly}(2^k, d)$ time and 0 samples.*

Proof. Let (S, h) be a k-junta, where $S = \{i_1, \ldots, i_k\} \subseteq [d]$ is the set of coordinates on which the k-junta depends, and $h : \{0,1\}^k \to \{0,1\}$ is the truth table. The junta is represented by a decision tree of depth k whose jth layer nodes are all labeled by i_j, and whose 2^k leaf nodes are labeled by the entries of the truth table. □

Open direction It is an open direction for future work to expand this web of reductions to include other popular classification algorithms such as (sparse) linear threshold functions, polynomial threshold functions, nearest neighbors, and logical circuits with varying sizes, depths, and gate

types – and also to explore whether these classifiers can be extracted from trained neural networks under structural assumptions such as the LRH. It is also of interest to prove that in some cases distillation is not possible, e.g., to pinpoint which of these hypothesis classes are incomparable.

5 Statistical theory: bounds on sample complexity

We now study the sample complexity of PAC-distillation, without imposing any computational constraints on the algorithms. Our main results in this section are:

- Distilling a class $\mathcal{F}$ into a class $\mathcal{G}$ is no harder than learning $\mathcal{G}$. Furthermore, there are some cases where $\mathcal{G}$ requires a high number of samples to learn, but we can distill from $\mathcal{F}$ to $\mathcal{G}$ with no samples; see Section 5.1.
- The low sample complexity of distillation is a general phenomenon – whenever $\mathcal{F}$ is *perfectly distillable* into $\mathcal{G}$ (defined below), very few samples are needed to do so; see Section 5.2.
- On the other hand, there are cases in which agnostic distillation of $\mathcal{F}$ into $\mathcal{G}$ can have high sample complexity; see Section 5.3.
- Finally, we pose the problem of determining a combinatorial property that controls the sample complexity of distillation. We prove that a natural candidate for this property fails; see Section 5.4.

Throughout this section, we view $f \in \mathcal{F}$ and $g \in \mathcal{G}$ simply as functions, ignoring how they are represented (e.g., neural network, decision tree, or truthtable), which only mattered in previous sections due to the computational constraints om the distillation algorithms.

5.1 Distillation is strictly easier than learning

First we observe that whenever we can (agnostically) learn, we can also (agnostically) distill.

Theorem 5.1 (Distilling is easier than learning)**.** *If concept class $\mathcal{F}$ is (ϵ, δ)-learnable with hypothesis class $\mathcal{G}$ in n samples, then source class $\mathcal{F}$ is (ϵ, δ)-distillable into target class $\mathcal{G}$ in n samples.*
Similarly, if hypothesis class $\mathcal{G}$ is agnostically (ϵ, δ)-learnable in n samples, then any source class $\mathcal{F}$ can be agnostically (ϵ, δ)-distilled into target class $\mathcal{G}$ in n samples.

Proof. Let $\mathcal{A}_{\text{learn}}$ be a learning algorithm that takes in samples $S = (x_1, \ldots, x_n) \sim \mathcal{D}^{\otimes n}$ and their labels $L = (f(x_1), \ldots, f(x_n))$, and outputs a hypothesis $g = \mathcal{A}_{\text{learn}}(S, L) \in \mathcal{G}$ such that $\mathbb{P}[\text{error}_{\mathcal{D}}(g; f) \leq \varepsilon] \geq 1 - \delta$. A distillation algorithm $\mathcal{A}_{\text{dist}}$ can get the same guarantee by outputting $\mathcal{A}_{\text{dist}}(S, f) := \mathcal{A}_{\text{learn}}(S, (f(x_1), \ldots, f(x_n)))$. The proof in the agnostic case is analogous. □

However, the converse is false: in some cases distillation can require strictly fewer samples than learning. In extreme cases, when the concept class is contained in the hypothesis class, distillation is possible with 0 samples, whereas learning might be sample-expensive.

Theorem 5.2 (Distilling can be strictly easier than learning)**.** *Let $\mathcal{X}$ be a finite input space of size $|\mathcal{X}| = m$, let $\mathcal{Y} = \{0, 1\}$, and let $\mathcal{F} = \mathcal{G} = \mathcal{Y}^{\mathcal{X}}$ be all possible functions from $\mathcal{X}$ to $\mathcal{Y}$. Then:*

- *The class $\mathcal{F}$ is $(\epsilon = 0, \delta = 0)$-distillable into $\mathcal{G}$ using 0 samples.*
- *On the other hand, for any $0 < \epsilon \leq 1/16$ and for $\delta = 1/2$, at least $n \geq \Omega(m/\epsilon)$ samples are needed to (ϵ, δ)-learn $\mathcal{F}$ with $\mathcal{G}$.*

Proof. The distillation algorithm $\mathcal{A}_{\text{dist}}(S, f)$ that outputs f is valid since $f \in \mathcal{G}$ because $\mathcal{F} = \mathcal{G}$. It requires no samples and has error $\epsilon = 0$ and soundness $\delta = 0$ for any distribution over the inputs.

The lower bound on the number of samples need to learn $\mathcal{F}$ is well known: it follows from Theorem 3.5 of [KV94] because the "VC dimension" of $\mathcal{F}$ is m. For completeness, we repeat the proof here. Let $\mathcal{X} = \{x_1, \ldots, x_m\}$, and let $\mathcal{D}$ be the distribution over $\mathcal{X}$ that assigns probability mass $1 - 16\epsilon$ to x_1 and $16\epsilon/(m-1)$ to each of $x_2, \ldots, x_m$. Let $\mathcal{I} = \mathcal{X} \setminus S$ be the set of inputs that do not appear in the algorithm's multiset of samples S. If the learning algorithm $\mathcal{A}_{\text{learn}}$ draws $n \leq (m-1)/(64\epsilon)$ samples, then by a Markov bound

$$\mathbb{P}[|\mathcal{I}| \leq \frac{m-1}{2}] \geq 3/4. \tag{5.1}$$

The labels of the inputs in $\mathcal{I}$ are undetermined by the samples: in particular, there is a concept $f \in \mathcal{F}$ agreeing with the samples such that, with respect to this concept, the learning algorithm has error at least $(|\mathcal{I}|/2)(16\epsilon/(m-1))$ in expectation. Therefore

$$\mathbb{P}[\text{error}(\mathcal{A}_{\text{learn}}(S, L)) \geq \frac{2\epsilon|\mathcal{I}|}{m-1}] \geq 3/4. \tag{5.2}$$

By a union bound combining (5.1) and (5.2), we have $\mathbb{P}[\text{error}(\mathcal{A}_{\text{learn}}(S, L)) \geq \epsilon] \geq 1/2$. □

5.2 Whenever perfect distillation is possible, very few samples are needed

In this section, we observe that the low sample complexity of distillation observed in Theorem 5.2 is a general phenomenon. We prove that distillation has a very low sample complexity whenever it is possible to *perfectly distill*, i.e., when it is possible to carry out the distillation to any given level of accuracy. We define *perfect distillation* below.

Definition 5.3. *The class $\mathcal{F}$ is* perfectly distillable *into the class $\mathcal{G}$ if for all $0 < \epsilon < 1$ and $0 < \delta < 1$, there is a (ϵ, δ)-distillation algorithm from $\mathcal{F}$ to $\mathcal{G}$.*

Remark 5.4. We remark that the definition of perfect distillation does not allow the class $\mathcal{G}$ to grow as we vary ϵ, δ. This means that the results in this section do not apply if, e.g., our scenario is distilling neural networks into logical circuits, and we require larger circuits as ϵ and δ decrease. Further research is required to understand the sample complexity in this case; see the open problems in Section 5.4.

First, we consider the case of finite input space, where we show that perfect distillation can always be achieved with 0 samples whenever it is possible.

Theorem 5.5. *Suppose that $\mathcal{F}$ is perfectly distillable into $\mathcal{G}$, and the input space $\mathcal{X}$ is finite. Then $\mathcal{F}$ is $(\epsilon = 0, \delta = 0)$-distillable into $\mathcal{G}$ with 0 samples.*

Proof. Let $\mathcal{D}$ be uniform over $\mathcal{X}$. Then since $\mathcal{F}$ is $(\epsilon = 1/(2|\mathcal{X}|), \delta = 1/2)$-distillable into $\mathcal{G}$, for each $f \in \mathcal{F}$ there is $g \in \mathcal{G}$ such that

$$\mathbb{P}_{x\sim\mathcal{D}}[f(x) \neq g(x)] < 1/(2|\mathcal{X}|)\,,$$

which is possible only if $f(x) = g(x)$ for all x. Therefore $\mathcal{F} \subseteq \mathcal{G}$, and so there is $(0, 0)$-distillation algorithm that simply always outputs $f \in \mathcal{G}$ given $f \in \mathcal{F}$. □

In the case of countable (potentially infinite) input spaces $\mathcal{X}$, we similarly show that perfect distillation can be achieved with very few samples whenever it is possible. The number of samples required is nonzero in this case, but our upper bound scales mildly in ϵ, δ, and does not depend on any extra structure of $\mathcal{F}$ and $\mathcal{G}$.

Theorem 5.6. *Suppose that $\mathcal{F}$ is perfectly distillable into $\mathcal{G}$, and the input space $\mathcal{X}$ is countable. Then for any $0 < \epsilon < 1$ and $0 < \delta < 1$, the class $\mathcal{F}$ is (ϵ, δ)-distillable into $\mathcal{G}$ with $n(\epsilon, \delta) = \lceil \log(1/\delta)/\epsilon \rceil$ samples. Furthermore, there is a pair of classes $\mathcal{F}$ and $\mathcal{G}$ such that this is tight up to a constant.*

Proof. $\implies$. Write $\mathcal{X} = \cup_{m=1}^{\infty} \mathcal{X}_m$ for finite sets $\mathcal{X}_1 \subseteq \mathcal{X}_2 \subseteq \cdots$. A similar argument to the previous theorem shows that for any $f \in \mathcal{F}$ and $m \in \mathcal{N}$, there is $g_m \in \mathcal{G}$ such that $f(x) = g_m(x)$ for all $x \in \mathcal{X}_m$. Consider the distillation algorithm that takes in $\epsilon, \delta \in (0, 1)$ and draws $n(\epsilon, \delta) = \lceil \log(1/\delta)/\epsilon \rceil$ samples $x_1, \ldots, x_n \sim \mathcal{D}$, and lets $m^* = \min\{m : \{x_1, \ldots, x_n\} \subseteq \mathcal{X}_m\}$, and outputs g_{m^*}.

We prove correctness. Define $\tilde{m}_\epsilon = \max\{m : \mathbb{P}_{x\sim\mathcal{D}}[x \notin \mathcal{X}_{m_\epsilon}] > \epsilon\}$. Then the error of the hypothesis returned by the algorithm is:

$$\begin{aligned}
\mathbb{P}_{x_1,\ldots,x_n\sim\mathcal{D}}[\mathrm{error}_{\mathcal{D}}(g_{m^*}; f) > \epsilon] &\leq \mathbb{P}_{x_1,\ldots,x_n\sim\mathcal{D}}[\mathbb{P}_{x\sim\mathcal{D}}[x \notin \mathcal{X}_{m^*}] > \epsilon] \\
&< \mathbb{P}_{x_1,\ldots,x_n\sim\mathcal{D}}[m^* \leq m_\epsilon] \\
&\leq \mathbb{P}_{x_1,\ldots,x_n\sim\mathcal{D}}[x_i \in \mathcal{X}_{m_\epsilon} \text{ for all } i \in [n]] \\
&< (1-\epsilon)^n \\
&< \delta\,.
\end{aligned}$$

$\impliedby$. Now we prove that there is a pair of classes $\mathcal{F}$ and $\mathcal{G}$ such that $\mathcal{F}$ can be perfectly distilled into $\mathcal{G}$ and such that the above sample complexity of distillation is tight up to a constant. Let $\mathcal{X} = \mathbb{N}$ and let $\mathcal{F} = \{\mathrm{zero}\}$ where $\mathrm{zero} : \mathcal{X} \to \{0\}$ is the function that identically outputs zero. Let $\mathcal{G} = \cup_{i=1}^{\infty}\{g_i\}$, where $g_i(x) = 1(x > i)$. Then perfect distillation is possible using the above algorithm with $\mathcal{X}_m = \{1, \ldots, m\}$.

On the other hand, we will show that no algorithm $\mathcal{A}$ can (ϵ, δ)-distill from $\mathcal{F}$ to $\mathcal{G}$ in fewer than $c\log(1/\delta)/\epsilon$ samples, for an absolute constant $c > 0$ and any $0 < \epsilon, \delta < 1/2$. For any distillation algorithm $\mathcal{A}$, let $\mu_{\mathcal{A}}(n, \mathcal{D})$ be the distribution over $\mathbb{N}$ such that $\mathcal{A}$ outputs g_i with probability $[\mu_{\mathcal{A}}(n, \mathcal{D})](i)$ when given n samples from $\mathcal{D}$. For any m, ϵ, let $\mathcal{D}_{m,\epsilon}$ be the distribution that puts probability mass $1 - \epsilon$ on 1 and ϵ on m. Given a number of samples n and parameters ϵ, δ, let

$$m^* = \inf\{m : \mathbb{P}_{i\sim\mu(n,\mathcal{D}_{1,\epsilon})}[i > m] < \delta/2\}\,.$$

Notice that for any $i \in \mathbb{N}$, the error is

$$\mathrm{error}_{\mathcal{D}_{m^*,\epsilon}}(g_i; \mathrm{zero}) = \begin{cases} \epsilon, & i < m^* \\ 0, & i \geq m^*\,. \end{cases} \tag{5.3}$$

Furthermore,

$$\mathrm{TV}(\mathcal{D}_{1,\epsilon}^{\otimes n}, \mathcal{D}_{m^*,\epsilon}^{\otimes n}) \leq 1 - (1-\epsilon)^n \,,$$

so

$$\mathrm{TV}(\mu_{\mathcal{A}}(n, \mathcal{D}_{1,\epsilon}), \mu_{\mathcal{A}}(n, \mathcal{D}_{m^*,\epsilon})) \leq 1 - (1-\epsilon)^n \,. \tag{5.4}$$

Then, if we run algorithm $\mathcal{A}$ on distribution $\mathcal{D}_{m^*,\epsilon}$ with n samples, we have

$$\begin{aligned}
\mathbb{P}_{g_i \sim \mathcal{A}(n, \mathcal{D}_{m^*,\epsilon})}[\mathrm{error}_{\mathcal{D}_{m^*,\epsilon}}(g_i; \mathrm{zero}) \geq \epsilon] &= \mathbb{P}_{i \sim \mu(n, \mathcal{D}_{m^*,\epsilon})}[\mathrm{error}_{\mathcal{D}_{m^*,\epsilon}}(g_i; \mathrm{zero}) \geq \epsilon] \\
&= \mathbb{P}_{i \sim \mu(n, \mathcal{D}_{m^*,\epsilon})}[i \leq m^*] \\
&\geq \mathbb{P}_{i \sim \mu(1, \mathcal{D}_{1,\epsilon})}[i \leq m^*] - \mathrm{TV}(\mu(n, \mathcal{D}_{m^*,\epsilon}), \mu(n, \mathcal{D}_{1,\epsilon})) \\
&> \delta/2 + (1-\epsilon)^n \,.
\end{aligned}$$

So in order to (ϵ, δ)-distill we must have $(1-\epsilon)^n < \delta/2$, so we have to have $n = \Omega(\log(1/\delta)/\epsilon)$. □

5.3 Agnostic distillation may nevertheless require a high number of samples

So far, we have shown that distillation is sometimes far easier than learning. But are there situations where distillation still has a high sample complexity? The answer is yes, and we prove it below.

First, we observe that it might not be possible to distill source class $\mathcal{F}$ with hypothesis class $\mathcal{G}$, even with unboundedly many samples. This is because the functions in $\mathcal{F}$ and $\mathcal{G}$ may be highly mismatched.

Observation 5.7 (Distillation may be impossible, even with unbounded samples)**.** *There are classes $\mathcal{F}, \mathcal{G}$ such that $\mathcal{F}$ cannot be (ϵ, δ)-distilled into $\mathcal{G}$ for any $0 \leq \epsilon < 1$ and $0 \leq \delta < 1$.*

Proof. Let the input space be a singleton set with one element, $\mathcal{X} = \{x\}$, and let $\mathcal{Y} = \{0, 1\}$. Let $\mathcal{F} = \{h_0\}$ and $\mathcal{G} = \{h_1\}$, where $h_i : \mathcal{X} \to \mathcal{Y}$ is the function defined by $h_i(x) = i$. Then the distillation algorithm must output h_1, but $h_1(x) \neq h_0(x)$, so $\mathrm{error}(h_1) = 1$. □

Agnostic distillation bypasses this obstacle, since here the goal is instead to find a hypothesis in $\mathcal{G}$ that approximates the best possible one in $\mathcal{G}$. For finite input spaces, agnostic distillation is always possible with a number of samples that grows as fast as the size of the input space $\mathcal{X}$, because it can be reduced to agnostic learning by Theorem 5.1. It turns out that in some cases this large number of samples is needed in order to agnostically distill:

Theorem 5.8 (Agnostic distillation may require a high number of samples)**.** *Let $\mathcal{X}$ be a finite input space of size $|\mathcal{X}| = 2m$, and let $\mathcal{Y} = \{0, 1\}$. There is a source class $\mathcal{F}$ and a target class $\mathcal{G}$ such that for any $0 < \epsilon \leq 1/100$ and any $0 \leq \delta \leq 1/2$ at least $n \geq m/(20000\epsilon^2)$ samples are needed to agnostically (ϵ, δ)-distill $\mathcal{F}$ into $\mathcal{G}$.*

Proof. Let $\mathcal{F} = \{\mathrm{zero}\}$ be the singleton set consisting only of the trivial function $\mathrm{zero} : \mathcal{X} \to \mathcal{Y}$ that always outputs 0: i.e., $\mathrm{zero}(x) = 0$ for all x.

Next, let $\mathcal{X} = \mathcal{X}_1 \sqcup \mathcal{X}_2$ be a partition of the input space into two subsets of equal sizes with elements $\mathcal{X}_1 = \{x_{1,1}, \ldots, x_{1,m}\}$ and $\mathcal{X}_2 = \{x_{2,1}, \ldots, x_{2,m}\}$. Now, for each vector $\boldsymbol{\theta} \in \{1, 2\}^m$, let $g_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ be the hypothesis with

$$g_{\boldsymbol{\theta}}(x_{i,j}) = \begin{cases} 1, & \text{if } \theta_j = i \\ 0, & \text{otherwise} \end{cases} ,$$

and let $\mathcal{G} = \{g_{\boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \{1,2\}^m\}$. We will show that a large number of samples is needed to agnostically distill $\mathcal{F}$ into $\mathcal{G}$.

Let $0 \le \alpha \le 1$ be a parameter that we will set later. For any $\boldsymbol{\theta} \in \{1,2\}^m$, define the distribution $\mathcal{D}_{\boldsymbol{\theta}}$ over $\mathcal{X}$, which assigns probability mass

$$\mathcal{D}_{\boldsymbol{\theta}}(x) = \begin{cases} \frac{1}{2m} - \frac{\alpha}{2m}, & \text{if } g_{\boldsymbol{\theta}}(x) = 1 \\ \frac{1}{2m} + \frac{\alpha}{2m}, & \text{if } g_{\boldsymbol{\theta}}(x) = 0\,. \end{cases}$$

Notice that under distribution $\mathcal{D}_{\boldsymbol{\theta}}$, the error of hypothesis $g_{\boldsymbol{\theta}'}$ is equal to

$$\mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}}}(g_{\boldsymbol{\theta}'}(x)) = \frac{1}{2} + \frac{\alpha}{2} - \frac{\alpha}{m} \cdot |\{i : \theta_i = \theta'_i\}|\,,$$

which is minimized by taking $\boldsymbol{\theta}' = \boldsymbol{\theta}$.

Let $g_{\boldsymbol{\theta}^{dst}(S)} = \mathcal{A}_{\mathrm{dist}}(S, \mathrm{zero})$ be the hypothesis returned by a distillation algorithm $\mathcal{A}_{\mathrm{dist}}$ when given a tuple of samples $S \in \mathcal{X}^n$ and the concept zero $\in \mathcal{F}$. Let us choose $\boldsymbol{\theta}$ uniformly at random and run $\mathcal{A}_{\mathrm{dist}}$ on samples from the distribution $\mathcal{D}_{\boldsymbol{\theta}}$. Then the expected surplus error of the distilling algorithm is as follows.

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{\theta}\sim\{1,2\}^m}[\mathbb{E}_{S\sim\mathcal{D}_{\boldsymbol{\theta}}^n}[\mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}}}(g_{\boldsymbol{\theta}^{dst}(S)}) - \mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}}}(g_{\boldsymbol{\theta}})]] \\ &\quad = \mathbb{E}_{\boldsymbol{\theta}\sim\{1,2\}^m}[\mathbb{E}_{S\sim\mathcal{D}_{\boldsymbol{\theta}}^n}[\alpha(1 - |\{i : \theta_i^{dst}(S) = \theta_i\}|/m)]] \\ &\quad = \frac{\alpha}{m}\sum_{i=1}^{m} \mathbb{E}_{\boldsymbol{\theta}\sim\{1,2\}^m}[\mathbb{P}_{S\sim\mathcal{D}_{\boldsymbol{\theta}}^n}[\theta_i^{dst}(S) \neq \theta_i]] \end{aligned} \tag{5.5}$$

To bound this, define $\boldsymbol{\theta}^{-i} \in \{1,2\}^m$ to be $\boldsymbol{\theta}$ but with coordinate i flipped. Formally, $\theta_j^{-i} = \theta_j$ if and only if $j \neq i$. Notice that $\boldsymbol{\theta}^{-i}$ has the same distribution as $\boldsymbol{\theta}^i$, so by linearity of expectation:

$$\text{Equation (5.5)} = \frac{\alpha}{2m}\sum_{i=1}^{m} \mathbb{E}^m_{\boldsymbol{\theta}\sim\{1,2\}}[\mathbb{P}_{S\sim\mathcal{D}_{\boldsymbol{\theta}}^n}[\theta_i^{dst}(S) \neq \theta_i] + \mathbb{P}_{S'\sim\mathcal{D}_{\boldsymbol{\theta}^{-i}}^n}[\theta_i^{dst}(S') = \theta_i]] \tag{5.6}$$

For any $\boldsymbol{\theta}$, let $\Gamma_{\boldsymbol{\theta}}$ be a coupling between $\mathcal{D}_{\boldsymbol{\theta}}^n$ and $\mathcal{D}_{\boldsymbol{\theta}^{-i}}^n$. Then, by linearity of expectation

$$\begin{aligned} \text{Equation (5.6)} &= \frac{\alpha}{2m}\sum_{i=1}^{m} \mathbb{E}^m_{\boldsymbol{\theta}\sim\{1,2\}}[\mathbb{E}_{S,S'\sim\Gamma_{\boldsymbol{\theta}}}[1(\theta_i^{dst}(S) \neq \theta_i) + 1(\theta_i^{dst}(S') = \theta_i)]] \\ &\ge \frac{\alpha}{2m}\sum_{i=1}^{m} \mathbb{E}^m_{\boldsymbol{\theta}\sim\{1,2\}}[\mathbb{E}_{S,S'\sim\Gamma_{\boldsymbol{\theta}}}[1(S = S')]]\,. \end{aligned} \tag{5.7}$$

To maximize this lower bound, we should find a coupling for each $\boldsymbol{\theta}$ that maximizes the probability that the samples S and S' are equal. Since the distributions differ only on the probability mass that they put on $x_{i,1}$ and $x_{i,2}$, we can couple them so that $S \sim \mathcal{D}_{\boldsymbol{\theta}}^n$ and $S' \sim \mathcal{D}_{\boldsymbol{\theta}^{-i}}^n$ almost surely agree on all of the samples that are not equal to either $x_{i,1}$ or $x_{i,2}$. Then the coupling on the $x_{i,1}$ and $x_{i,2}$ samples reduces to coupling two Binomial random variables counting the total number of times each appears. Effectively, letting $|S|_x = |\{i : S_i = x\}|$, we have, and letting TV denote total variation distance, we have

$$\begin{aligned} \text{Equation (5.7)} \ge \frac{\alpha}{2m}\sum_{i=1}^{m} \mathbb{E}_{\boldsymbol{\theta}\sim\{1,2\}^m}[\sum_{k=0}^{n} \mathbb{P}_{S\sim\mathcal{D}_{\boldsymbol{\theta}}^n}[|S|_{x_{1,i}} + |S|_{x_{2,i}} = k] \\ \cdot (1 - \mathrm{TV}(\mathrm{Bin}(k, \frac{1-\alpha}{2}), \mathrm{Bin}(k, \frac{1+\alpha}{2})))] \end{aligned}$$

We will use the following calculation bounding the total variation distance between two binomial distributions from [MRT18]:

Claim 5.9 (Lemma 3.21 of [MRT18]). *Let* $\mathrm{Bin}(k,p)$ *denote the Binomial distribution with* k *trials and probability* p*. Then for any* $0 \le \alpha < 1$*, the total variation distance is bounded by:*

$$1 - \mathrm{TV}(\mathrm{Bin}(k, \frac{1-\alpha}{2}), \mathrm{Bin}(k, \frac{1+\alpha}{2})) \ge 2\Phi(k+1,\alpha)\,,$$

where $\Phi(k,\alpha) = \frac{1}{4}(1 - \sqrt{1 - \exp(-k\alpha^2/(1-\alpha^2))})$*, and* $\Phi(\cdot,\alpha)$ *is convex in the first parameter.*

Using this claim, we can continue to bound the expected error of the distilling algorithm. By Jensen's inequality,

$$\begin{aligned}
\text{Equation (5.7)} &\ge \frac{\alpha}{m}\sum_{i=1}^{m} \mathbb{E}_{\boldsymbol{\theta}\sim\{1,2\}^m}[\sum_{k=0}^{n} \mathbb{P}_{S\sim\mathcal{D}_{\boldsymbol{\theta}}^n}[|S|_{x_{1,i}} + |S|_{x_{2,i}} = k] \cdot \Phi(k+1,\alpha)] \\
&\ge \frac{\alpha}{m}\sum_{i=1}^{m} \mathbb{E}_{\boldsymbol{\theta}\sim\{1,2\}^m}[\Phi(\mathbb{E}_{S\sim\mathcal{D}_{\boldsymbol{\theta}}^n}[|S|_{x_{1,i}} + |S|_{x_{2,i}} = k] + 1,\alpha)] \\
&= \alpha \cdot \Phi(n/m+1,\alpha)\,. \qquad (5.8)
\end{aligned}$$

Putting equations (5.5) through (5.8) together, we obtain:

$$\mathbb{E}_{\boldsymbol{\theta}\sim\{1,2\}^m}[\mathbb{E}_{S\sim\mathcal{D}_{\boldsymbol{\theta}}^n}[\mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}}}(g_{\boldsymbol{\theta}^{dst}(S)}) - \mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}}}(g_{\boldsymbol{\theta}})]] \ge \alpha \cdot \Phi(n/m+1,\alpha)\,.$$

Therefore, there is a $\boldsymbol{\theta}^*$ such that when $\mathcal{D}_{\boldsymbol{\theta}^*}$ is the underlying distribution the distillation algorithm has an expected surplus error of at least $\alpha \cdot \Phi(n/m+1,\alpha)$. Since the surplus error is always in the range $[0,\alpha]$, a Markov bound shows:

$$\mathbb{P}_{S\sim\mathcal{D}_{\boldsymbol{\theta}^*}^n}[\mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}^*}}(g_{\boldsymbol{\theta}^{dst}(S)}) - \mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}^*}}(g_{\boldsymbol{\theta}^*}) \ge \alpha\Phi(n/m+1,\alpha)/2] \ge \Phi(n/m+1,\alpha)/2\,.$$

If we choose $\alpha = C\epsilon \le 1/\sqrt{2}$ and $n \le cm/\epsilon^2$, then

$$\begin{aligned}
\Phi(n/m+1,\alpha) &= \frac{1}{4}(1 - \sqrt{1 - \exp(-(c/\epsilon^2+1)(C\epsilon)^2)/(1-(C\epsilon)^2)}) \\
&\ge \frac{1}{4}(1 - \sqrt{1 - \exp(-2(c/\epsilon^2)(C\epsilon)^2 - 1)}) \\
&\ge \frac{1}{4}(1 - \sqrt{1 - \exp(-2cC^2 - 1)})
\end{aligned}$$

Taking $C = 70$ and $c = 1/20000$ thus results in

$$\mathbb{P}_{S\sim\mathcal{D}_{\boldsymbol{\theta}^*}^n}[\mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}^*}}(g_{\boldsymbol{\theta}^{dst}(S)}) - \mathrm{error}_{\mathcal{D}_{\boldsymbol{\theta}^*}}(g_{\boldsymbol{\theta}^*}) \ge \epsilon] \ge 1/70\,.$$

□

Note: this proof shares some elements with the proof that some classes require a large number of samples to agnostically learn (Lemma 3.23 of [MRT18]). This is no coincidence, since lower bounds on the number of samples needed for agnostic distillation also imply lower bounds for agnostic learnability (cf. Theorem 5.1).

5.4 Open problem: combinatorial characterization of sample complexity

We have established that distilling a source class $\mathcal{F}$ into a target class $\mathcal{G}$ is no harder than the corresponding learning problem, but it may still require a large number of samples or may be impossible. In this section we ask: can we identify a combinatorial property of $\mathcal{F}$ and $\mathcal{G}$ that characterizes how many samples are needed to distill? We leave this problem open, and provide partial progress by showing that a natural candidate based on the VC dimension fails.

5.4.1 Sample complexity of learning is determined by the VC dimension

In the case of learnability, it is known that a combinatorial property called the *VC dimension* controls the sample complexity [VC74, BEHW89].

Definition 5.10 (VC dimension [VC71])**.** *Let the label alphabet be $\mathcal{Y} = \{0, 1\}$. A function class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ is said to shatter the set of inputs $\{x_1, \ldots, x_m\} \subseteq \mathcal{X}$ if for each labeling of these inputs $\boldsymbol{\sigma} \in \{0, 1\}^m$ there is a function $h_{\boldsymbol{\sigma}} \in \mathcal{H}$ such that $(h_{\boldsymbol{\sigma}}(x_1), \ldots, h_{\boldsymbol{\sigma}}(x_m)) = \boldsymbol{\sigma}$.*
The VC dimension $\mathrm{VCdim}(\mathcal{H})$ of the function class $\mathcal{H}$ is the size of the greatest shattered subset of inputs (or infinite, if there is no greatest set.)

The VC dimension gives both upper and lower bounds for the number of samples needed to (agnostically) learn [VC74, BEHW89]. We follow the presentation of [AB99].

Proposition 5.11 (VC dimension controls sample complexity of learning; Theorem 5.6 in [AB99])**.** *There are constants $c_1, c_2 > 0$ such that for any $0 < \epsilon < 1/8$ and $0 < \delta < 1/100$, the minimum number of samples $n(\epsilon, \delta)$ needed to (ϵ, δ)-learn concept class $\mathcal{G}$ with hypothesis class $\mathcal{G}$ is:*

$$\frac{c_1}{\epsilon}\left(\mathrm{VCdim}(\mathcal{G}) + \ln(1/\delta)\right) \leq n(\epsilon, \delta) \leq \frac{c_2}{\epsilon}\left(\mathrm{VCdim}(\mathcal{G})\ln(1/\epsilon) + \ln(1/\delta)\right) .$$

If $\mathrm{VCdim}(\mathcal{G}) = \infty$, then $\mathcal{G}$ is not learnable in finitely many samples.

Proposition 5.12 (VC dimension controls sample complexity of agnostic learning; Theorem 5.4 in [AB99])**.** *There are constants $c_1, c_2 > 0$ such that for any $0 < \epsilon < 1/40$ and $0 < \delta < 1/20$ the minimum number of samples $n(\epsilon, \delta)$ needed to agnostically (ϵ, δ)-learn hypothesis class $\mathcal{G}$ is:*

$$\frac{c_1}{\epsilon^2}\left(\mathrm{VCdim}(\mathcal{G}) + \ln(1/\delta)\right) \leq n(\epsilon, \delta) \leq \frac{c_2}{\epsilon^2}\left(\mathrm{VCdim}(\mathcal{G}) + \ln(1/\delta)\right) .$$

If $\mathrm{VCdim}(\mathcal{G}) = \infty$, then $\mathcal{G}$ is not agnostically learnable with finitely many samples.

5.4.2 Is there an analogue for distillation?

Is there a quantity analogous to the VC dimension that determines how many samples are needed to distill? This question can be asked in both the standard and agnostic settings.

Open problem 1. *Find a combinatorial property of the pair $(\mathcal{F}, \mathcal{G})$ that controls the sample complexity of distilling $\mathcal{F}$ into $\mathcal{G}$.*

Open problem 2. *Find a combinatorial property of the pair $(\mathcal{F}, \mathcal{G})$ that controls the sample complexity of agnostically distilling $\mathcal{F}$ into $\mathcal{G}$.*

In the case that the source class can be perfectly distilled into the target class (Definition 5.3), then we have resolved the open problems in Section 5.2. For finite input spaces we have proved that $\mathcal{F}$ is distillable into $\mathcal{G}$ in 0 samples, and for countable input spaces we have proved a $O(\log(1/\delta)/\epsilon)$-sample upper bound, with a matching lower bound. The question is open when perfect distillation is not possible. We make partial progress below.

5.4.3 The VC dimension of the Pareto frontier does not characterize distillation

We give some preliminary results for open problems 1 and 2, but do not fully resolve these problems. In this section, we show that the sample complexity of agnostic distillation is bounded above by a natural combinatorial quantity called the "VC dimension of the Pareto frontier", which can be much smaller than the VC dimension. However, we prove that there is no corresponding lower bound based on the VC dimension of the Pareto frontier.

First, in Lemma 5.13, we reduce distillation without loss of generality to the case where the label alphabet is binary and the source class is a singleton set.[6] The relevant notation for our reduction is as follows. For a function $f \in \mathcal{Y}^{\mathcal{X}}$ and a class of functions $\mathcal{G} \subseteq \mathcal{Y}^{\mathcal{X}}$, define the binary-valued class of functions that represents the error pattern of the hypotheses with respect to $f \in \mathcal{F}$:

$$f \oplus \mathcal{G} = \{f \oplus g \mid g \in \mathcal{G}\} \subseteq \{0,1\}^{\mathcal{X}}, \text{ where } (f \oplus g)(x) = 1(f(x) \neq g(x)).$$

Also, let $\text{zero} : \mathcal{X} \to \{0\}$ be the function that identically outputs 0.

Lemma 5.13 (Reduction to binary labels and singleton source class)**.** *The following are equivalent:*

(a) Class $\mathcal{F}$ is (ϵ, δ)-distillable into class $\mathcal{G}$ in n samples.

(b) For all $f \in \mathcal{F}$, class $\{f\}$ is (ϵ, δ)-distillable into class $\mathcal{G}$ in n samples.

(c) For all $f \in \mathcal{F}$, class $\{\text{zero}\} \subseteq \{0,1\}^{\mathcal{X}}$ is (ϵ, δ)-distillable into class $f \oplus \mathcal{G} \subseteq \{0,1\}^{\mathcal{X}}$ in n samples.

Furthermore, the same is true for agnostic distillation instead of distillation.

Proof. (b) $\implies$ (a): For each $f \in \mathcal{F}$, let $\mathcal{A}_{\text{dist},f}$ be an algorithm distilling $\{f\}$ into $\mathcal{G}$. Then the algorithm $\mathcal{A}_{\text{dist}}$ that outputs $\mathcal{A}_{\text{dist}}(S, f) := \mathcal{A}_{\text{dist},f}(S)$ distills $\mathcal{F}$ into $\mathcal{G}$.

(a) $\implies$ (b): Conversely, given $\mathcal{A}_{\text{dist}}(S, f)$, we can define $\mathcal{A}_{\text{dist},f}(S) := \mathcal{A}_{\text{dist}}(S, f)$ for each f.

(b) $\implies$ (c): Note that $\text{error}_{\mathcal{D}}(g; f) = \text{error}_{\mathcal{D}}(f \oplus g; \text{zero})$. So, given algorithm $\mathcal{A}_{\text{dist},f}$ that distills $\{f\}$ into $\mathcal{G}$, the algorithm $\mathcal{A}'_{\text{dist},f}(S) := f \oplus \mathcal{A}_{\text{dist},f}(S)$ distills $\{\text{zero}\}$ into $f \oplus \mathcal{G}$.

(c) $\implies$ (b): Given $\mathcal{A}'_{\text{dist},f}$, we can let $\mathcal{A}_{\text{dist},f}(S) :=$ "output any $g \in \mathcal{G}$ s.t. $f \oplus g = \mathcal{A}'_{\text{dist},f}(S)$". □

We have reduced the statistical problem of distillation to the case where the source class consists of only the zero function, and the target class represents the error pattern of hypotheses with respect to this function.

Under this lens, one can see that some target functions have strictly more errors than others with respect to the source concept. Thus, an optimal distillation algorithm will always output a target function whose error pattern does not dominate any of the other target functions. Formally, we can reduce to the Pareto frontier of the set of target functions:

[6] The reduction breaks when distillation is extended to different loss functions as discussed in Section 6.

Lemma 5.14 (Reduction to Pareto frontier of target functions)**.** *Let the source class be* $\mathcal{F} = \{\text{zero}\} \subseteq \{0,1\}^{\mathcal{X}}$ *and let the target class be* $\mathcal{G} \subseteq \{0,1\}^{\mathcal{X}}$*. We define the Pareto partial order* $\prec_{Par}$ *between pairs of distinct functions* $g, g' \in \mathcal{G}$*. We say that* $g \prec_{Par} g'$ *if whenever* $g(x) = 0$ *then also* $g'(x) = 0$*. The Pareto frontier of the class* $\mathcal{G}$ *is defined as the set of maximal functions:*

$$\mathrm{PF}(\mathcal{G}) = \{g \in \mathcal{G} \mid \text{ there is no } g' \in \mathcal{G} \text{ s.t. } g \prec_{Par} g'\}\,.$$

Then the following are equivalent:

(a) Class $\mathcal{F} = \{\text{zero}\}$ *is* (ϵ, δ)*-distillable into class* $\mathcal{G}$ *in* n *samples.*

(b) Class $\mathcal{F} = \{\text{zero}\}$ *is* (ϵ, δ)*-distillable into class* $\mathrm{PF}(\mathcal{G})$ *in* n *samples.*

Furthermore, the same is true for agnostic distillation instead of distillation.

Proof. (a) $\implies$ (b): by definition for any $g \in \mathcal{G}$ there is $g' \in \mathrm{PF}(\mathcal{G})$ such that for all distributions $\mathcal{D}$ we have $\mathrm{error}_{\mathcal{D}}(g') \leq \mathrm{error}_{\mathcal{D}}(g)$. Thus an algorithm distilling $\mathcal{F}$ with $\mathcal{G}$ can be converted to outputting a corresponding function in $\mathrm{PF}(\mathcal{G})$ instead without lowering the error.

(b) $\implies$ (a): since $\mathrm{PF}(\mathcal{G}) \subseteq \mathcal{G}$. For agnostic distillation, note the optimum is in $\mathrm{PF}(\mathcal{G})$. □

The above reductions allow us to upper-bound the sample complexity of agnostic distillation in terms of a new combinatorial quantity: the VC dimension of the Pareto frontier.[7]

Theorem 5.15 (The VC dimension of the Pareto frontier upper-bounds the sample complexity of agnostic distillation)**.** *For any source class* $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$ *and any target class* $\mathcal{G} \subseteq \mathcal{Y}^{\mathcal{X}}$*, define*

$$\mathrm{VCdimPF}(\mathcal{F}; \mathcal{G}) = \max_{f \in \mathcal{F}} \mathrm{VCdim}(\mathrm{PF}(f \oplus \mathcal{G}))\,.$$

Then there is a constant $C > 0$ *such that for any* $0 < \epsilon < 1/40$ *and* $0 < \delta < 1/20$*, it is possible to agnostically* (ϵ, δ)*-distill class* $\mathcal{F}$ *into class* $\mathcal{G}$ *in* $n(\epsilon, \delta)$ *samples, where*

$$n(\epsilon, \delta) \leq \frac{C}{\epsilon^2}(\mathrm{VCdimPF}(\mathcal{F}; \mathcal{G}) + \ln(1/\delta))\,.$$

Proof. By Proposition 5.12, for each $f \in \mathcal{G}$ the class $\mathrm{PF}(f \oplus \mathcal{G})$ can be agnostically (ϵ, δ)-learned with $n(\epsilon, \delta)$ samples. By Theorem 5.1, agnostic distillation reduces to agnostic learning so $\{\text{zero}\}$ can be agnostically (ϵ, δ)-distilled into $f \oplus \mathcal{G}$ with $n(\epsilon, \delta)$ samples. This concludes the proof by the reductions in Lemmas 5.13 and 5.14. □

For agnostic distillation, the quantity $\mathrm{VCdimPF}(\mathcal{F}; \mathcal{G})$ matches well with the sample complexity in the examples that we have encountered so far:

- In the example of Theorem 5.2 for when agnostic distillation can be done with 0 samples, we have equal source and target classes $\mathcal{F} = \mathcal{G}$. In this case notice that $\mathrm{VCdimPF}(\mathcal{F}; \mathcal{G}) = \max_{f \in \mathcal{F}} \mathrm{VCdim}(\mathrm{PF}(f \oplus \mathcal{G})) = \max_{f \in \mathcal{F}} \mathrm{VCdim}(\{\text{zero}\}) = 0$.

[7] We remark that Theorem 5.15 does not apply to non-agnostic distillation, which may be impossible in some cases. The example in Observation 5.7 demonstrates that distilling may be impossible even when $\mathrm{VCdimPF}(\mathcal{F}; \mathcal{G}) = 1$.

- In the example of Theorem 5.8 for when agnostic distillation may require a high number of samples, $\mathcal{F} = \{\text{zero}\}$, and $\mathcal{G}$ is large and all its functions are on the Pareto frontier. Thus, $\text{VCdimPF}(\mathcal{F}; \mathcal{G}) = \text{VCdim}(\text{PF}(\mathcal{G})) = \text{VCdim}(\mathcal{G}) = m$, which is large.

In light of these examples, it is natural to conjecture that the VC dimension of the Pareto frontier of functions will fully characterize the sample complexity of agnostic distillation. However, this conjecture turns out to be false, as we show below.

Theorem 5.16 (The VC dimension of the Pareto frontier does not characterize the sample complexity of distillation)**.** *There are classes $\mathcal{F}, \mathcal{G}$ such that $\text{VCdimPF}(\mathcal{F}; \mathcal{G}) = \infty$, but for any $0 < \epsilon < 1$, $0 < \delta < 1$ it is possible to (ϵ, δ)-distill class $\mathcal{F}$ into class $\mathcal{G}$ using 0 samples.*

Proof. Let $\mathcal{F} = \{\text{zero}\}$, let $\mathcal{X} = \{1, 2\} \times \mathbb{N}$, and $\mathcal{Y} = \{0, 1\}$. For any subsets $S, T, U \subseteq \mathbb{N}$, let $g_{S,T,U} : \mathcal{X} \to \mathcal{Y}$ be the hypothesis with

$$g_{S,T,U}((1, i)) = 1(i \in S)\,, \quad g_{S,T,U}((2, i)) = 1(i \in T) \quad g_{S,T,U}((3, i)) = 1(i \in U)\,, \text{ for all } i \in \mathbb{N}\,.$$

Then let $\mathcal{G}$ be the hypothesis class

$$\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2, \qquad \text{where} \qquad \mathcal{G}_1 = \bigcup_{\substack{S \subset \mathbb{N} \\ S \neq \emptyset \\ j \in \mathbb{N}}} \{g_{S,\{j\},\emptyset}\}, \quad \mathcal{G}_2 = \bigcup_{j \in \mathbb{N}} \{g_{\emptyset,\emptyset,\{j\}}\}\,.$$

It can be seen that $g_{S,T,U}$ Pareto-dominates $g_{S',T',U'}$ if and only if $S \subseteq S'$, $T \subseteq T'$, and $U \subseteq U'$ and this containment is strict for one of S, T, U. All of the functions in $\mathcal{G}$ are Pareto-incomparable, so $\text{PF}(\mathcal{G}) = \mathcal{G}$.

Thus $\text{VCdimPF}(\mathcal{F}; \mathcal{G}) = \text{VCdim}(\text{PF}(\mathcal{G})) = \text{VCdim}(\mathcal{G}) \geq \text{VCdim}(\mathcal{G}_1) = \infty$ because for any m the set $\{(1, 1), (1, 2), \ldots, (1, m)\}$ is shattered by $\mathcal{G}_1$.

Finally, given $0 < \epsilon, 0 < \delta$ consider the distillation algorithm $\mathcal{A}_{\text{dist}}$ that outputs $g_{\emptyset,\emptyset,\{j\}}$ by picking j uniformly at random from the interval $\{1, \ldots, \lceil 1/(\epsilon\delta) \rceil\}$. For any distribution $\mathcal{D}$ over the inputs, we have $\mathbb{E}_j[\text{error}_{\mathcal{D}}(g_{\emptyset,\emptyset,\{j\}})] = \mathbb{E}_j[\mathbb{P}_{x \sim \mathcal{D}}[x = (3, j)]] \leq \epsilon\delta$. So, by a Markov bound, $\mathbb{P}_j[\text{error}_{\mathcal{D}}(g_{\emptyset,\emptyset,\{j\}}) \geq \epsilon] \leq \delta$. Thus, this algorithm (ϵ, δ)-distills $\mathcal{F}$ into $\mathcal{G}$ in 0 samples. □

To summarize, we have shown that, for statistical purposes, distillation reduces to considering the Pareto frontier and also to the case $\mathcal{F} = \{\text{zero}\}$ and $\mathcal{Y} = \{0, 1\}$. We have also shown sample complexity upper bounds for agnostic distillation in terms of a new combinatorial quantity VCdimPF. However, we have also shown that VCdimPF fails to characterize the sample complexity of distillation. Thus, we leave problems 1 and 2 on fully characterizing the sample complexity of distillation open for future work.

6 Extensions and future directions

The contributions of this paper are (1) to initiate the study of a general computational and statistical theory of model distillation, and (2) to present case studies on distilling neural networks into more lightweight classifiers.

We hope that this paper encourages further theoretical and empirical exploration into distillation to help realize its promise of improved efficiency and trustworthiness in deep learning. Below, we outline some of the many open directions for future work.

Extending the definition of distillation The definition of PAC-distillation can be easily extended beyond the simple case of classification considered in this work.

- *Beyond classification.* In this paper, we considered classifiers with the zero-one loss (2.1). We could extend our definitions the regression setting, where the output space is the real numbers, and other losses such as the mean-squared error loss are more natural.
- *Promise problems.* In this paper, $\mathcal{D}$ was an adversarially-chosen distribution.[8] One can extend the PAC-distillation framework to cases where we are promised that the input distribution has some constraints and is not worst-case. One example is if we assume that the neural networks in our source class satisfy an approximate version of the LRH, as in Definition A.1 in the appendix.
- *Testing out-of-distribution behavior.* PAC-distillation is defined with matching train and test distributions. Is there a corresponding definition for when there is distribution shift? How does the computational and statistical complexity change in that setting?

Basic statistical and computational theory Fundamental open questions include:

- *Growing the web of reductions.* The web in Figure 4 could be expanded to include other popular classifiers, such as nearest neighbors, linear and polynomial threshold functions, and logical circuits of varying sizes, depths, and gate types – as well as whether these classifiers can be extracted from trained neural networks under structural assumptions such as the LRH.
- *Combinatorial characterization of sample complexity.* An analogue of VC-dimension for (agnostic) distillation remains open (see open problems 1 and 2).
- *Statistical-computational gaps for distillation.* Are there natural cases where it is statistically possible to distill, but it is computationally hard?

New provably-efficient distillation algorithms for neural networks

- *Neural network structure beyond LRH.* What other kinds of structure in trained neural networks can we use to distill efficiently, beyond query access to the network, and the Linear Representation Hypothesis (LRH)? One possibility is to use low-rank weight structure in a trained network [BLA+23], which hints that internally the network may be well-approximated by a composition of multi-index functions [ABM22, DLS22]. Another possibility is to use the attention structure in a transformer, which is often visually inspected for insights into the network's inner workings [Vig19].[9] Yet another possibility is to use internal sparse activation structure [AVPVF23] to distill. It seems that there can be a synergy between research into how neural networks learn, and new distillation algorithms that exploit that research.
- *Obtaining polynomial time in the depth for distilling decision trees.* Our algorithm in Theorem 3.6 for distilling decision trees from neural networks runs in true polynomial time in all parameters when the input distribution is uniform. Is it possible to reduce the current $2^{O(r)}$ dependence on the depth to $\text{poly}(r)$ dependence for arbitrary non-uniform distributions?

[8] Apart from some cases when we considered the uniform distribution.

[9] Nevertheless, recent work [WLLR23] suggests that attention patterns should not be interpreted by themselves without considering how they interact with the feedforward components of the transformer.

- *Distilling into circuits and more expressive classes.* Can we extend the ideas in the distillation algorithm from neural networks into trees so that we instead distill into a class of shallow logical circuits, which are more expressive than decision trees and may better approximate the trained neural network? Beyond small, shallow circuits, it is interesting to distill into more expressive and "human-interpretable" classes, such as, e.g., small Python programs.

Distilling foundation models Scaling these methods to foundation models such as LLMs seems like it will pose a significant engineering challenge, and require new ideas.

- *Into which class of models can we distill an LLM?* Classical hypothesis classes, such as decision trees and very small circuits, are almost certainly insufficient to distill a LLM. One main obstacle is that LLMs are able to memorize a large amount of information which cannot be encoded into a limited-size circuit or decision tree. Therefore, a possible choice is classifiers that consist of a large memory bank, and a circuit that has some limited access to it and also has sparse activation patterns (i.e., one can think of the neurons in the circuits as corresponding to concepts, and only a bounded number of concepts is active in any sentence). Another possibility is a hypothesis class again consisting of a large key-value memory, but now coupled with a small RASP program [WGY21] or other textual program. Yet another possibility is a target class consisting of a large logical circuit where a significant fraction of the nodes correspond to "human-understandable" concepts (e.g., words from a dictionary, or concepts encoded by a more trusted LLM). Another consideration is that it may be possible to distill the higher layers of the LLM, which may be in charge of higher-level reasoning, while maintaining the lower-level layers. One benefit of these distillations might be easier discovery and debugging of reasoning errors in distilled LLMs.

Acknowledgements

I am very grateful to Emmanuel Abbe, Kiril Bangachev, Guy Bresler, Melihcan Erol, Nati Srebro for helpful research discussions. I am also thankful to Guy Bresler for feedback on the exposition. This work was supported by an NSF Graduate Research Fellowship under NSF grant 1745302.

References

[AB99] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*, volume 9. Cambridge University Press, 1999.

[AB16] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[AB22] Emmanuel Abbe and Enric Boix-Adsera. On the non-universality of deep learning: quantifying the cost of symmetry. *Advances in Neural Information Processing Systems*, 35:17188–17201, 2022.

[ABB+21] Emmanuel Abbe, Enric Boix-Adsera, Matthew S Brennan, Guy Bresler, and Dheeraj Nagaraj. The staircase property: How hierarchical structure can guide deep learning. *Advances in Neural Information Processing Systems*, 34:26989–27002, 2021.

[ABM22] Emmanuel Abbe, Enric Boix-Adsera, and Theodor Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of

sparse functions on two-layer neural networks. In *Conference on Learning Theory*, pages 4782–4887. PMLR, 2022.

[ABM23] Emmanuel Abbe, Enric Boix-Adsera, and Theodor Misiakiewicz. Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 2552–2623. PMLR, 2023.

[ADRDS+20] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[ADT95] Robert Andrews, Joachim Diederich, and Alan B Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6):373–389, 1995.

[AL23a] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023.

[AL23b] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023.

[ALL+16] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

[AMS21] Carlos Aspillaga, Marcelo Mendoza, and Alvaro Soto. Inspecting the concept knowledge graph encoded by modern language models. *arXiv preprint arXiv:2105.13471*, 2021.

[Ang] D Angluin. Remarks on the difficulty of finding a minimal disjunctive normal form for boolean functions. *Unpublished manuscript*.

[AVPVF23] Maksym Andriushchenko, Aditya Vardhan Varre, Loucas Pillaud-Vivien, and Nicolas Flammarion. Sgd with large step sizes learns sparse features. In *International Conference on Machine Learning*, pages 903–925. PMLR, 2023.

[AZL20] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

[Bac21] Francis Bach. Learning theory from first principles. *Draft of a book, version of Sept*, 6:2021, 2021.

[BBPV23] Alberto Bietti, Joan Bruna, and Loucas Pillaud-Vivien. On learning gaussian multi-index models with gradient flow. *arXiv preprint arXiv:2310.19793*, 2023.

[BC16] Nader H Bshouty and Areej Costa. Exact learning of juntas from membership queries. In *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings 27*, pages 115–129. Springer, 2016.

[BCNM06] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[BCZ+16] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29, 2016.

[BEG+22] Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[BES+22] Jimmy Ba, Murat A Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. *Advances in Neural Information Processing Systems*, 35:37932–37946, 2022.

[BFJ+94] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 253–262, 1994.

[BFOS84] L Breiman, JH Friedman, R Olshen, and CJ Stone. Classification and regression trees. 1984.

[BGOFG20] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[BKB17] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpretability via model extraction. *arXiv preprint arXiv:1706.09773*, 2017.

[BLA+23] Enric Boix-Adsera, Etai Littwin, Emmanuel Abbe, Samy Bengio, and Joshua Susskind. Transformers learn through gradual rank increase. *arXiv preprint arXiv:2306.07042*, 2023.

[BLQT22] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. *Journal of the ACM*, 69(6):1–19, 2022.

[BLSR22] Enric Boix-Adsera, Hannah Lawrence, George Stepaniants, and Philippe Rigollet. Gulp: a prediction-based metric between representations. *Advances in Neural Information Processing Systems*, 35:7115–7127, 2022.

[BS96] Leo Breiman and Nong Shang. Born again trees. *University of California, Berkeley, Berkeley, CA, Technical Report*, 1(2):4, 1996.

[BSJR+23] Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *arXiv preprint arXiv:2306.03819*, 2023.

[Bub15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[CH90] Kenneth Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[CKL+18] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

[CS94] Mark W Craven and Jude W Shavlik. Using sampling and queries to extract rules from trained neural networks. In *Machine learning proceedings 1994*, pages 37–45. Elsevier, 1994.

[CS95] Mark Craven and Jude Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8, 1995.

[CSH+22] Aditya Chattopadhyay, Stewart Slocum, Benjamin D Haeffele, Rene Vidal, and Donald Geman. Interpretable by design: Learning predictors by composing interpretable queries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7430–7443, 2022.

[Dam98a] Peter Damaschke. Adaptive versus nonadaptive attribute-efficient learning. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 590–596, 1998.

[Dam98b] Peter Damaschke. Computational aspects of parallel attribute-efficient learning. In *International Conference on Algorithmic Learning Theory*, pages 103–111. Springer, 1998.

[DBGV05] Annalisa De Bonis, Leszek Gasieniec, and Ugo Vaccaro. Optimal two-stage algorithms for group testing problems. *SIAM Journal on Computing*, 34(5):1253–1270, 2005.

[DDH+21] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

[DKL+23] Yatin Dandi, Florent Krzakala, Bruno Loureiro, Luca Pesce, and Ludovic Stephan. Learning two-layer neural networks, one (giant) step at a time. *arXiv preprint arXiv:2305.18270*, 2023.

[DLS22] Alexandru Damian, Jason Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Conference on Learning Theory*, pages 5413–5452. PMLR, 2022.

[EGK+23] Benjamin L Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Pareto frontiers in neural feature learning: Data, compute, width, and luck. *arXiv preprint arXiv:2309.03800*, 2023.

[EH89] Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

[EHO+22] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[FC18] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[Fel98] Christiane Fellbaum. *WordNet: An electronic lexical database*. MIT press, 1998.

[FH17] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[GAGN15] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[GKD+22] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[GLR99] David Guijarro, Vıctor Lavın, and Vijay Raghavan. Exact learning when irrelevant variables abound. *Information Processing Letters*, 70(5):233–239, 1999.

[GYMT21] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[HLA23] Evan Hernandez, Belinda Z Li, and Jacob Andreas. Inspecting and editing knowledge representations in language models. *arXiv preprint arXiv:2304.00740*, 2023.

[HSL+21] Wooseok Ha, Chandan Singh, Francois Lanusse, Srigokul Upadhyayula, and Bin Yu. Adaptive wavelet distillation from neural networks through interpretations. *Advances in Neural Information Processing Systems*, 34:20669–20682, 2021.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[JSL11] Ulf Johansson, Cecilia Sönströd, and Tuve Löfström. One tree to explain them all. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1444–1451. IEEE, 2011.

[KM91] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 455–464, 1991.

[KST23a] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Properly learning decision trees with queries is np-hard. *arXiv preprint arXiv:2307.04093*, 2023.

[KST23b] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superpolynomial lower bounds for decision tree learning and testing. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1962–1994. SIAM, 2023.

[KV94] Michael J Kearns and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[LDIK08] Percy Liang, Hal Daumé III, and Dan Klein. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*, pages 592–599, 2008.

[LHB+22] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

[LNA21] Belinda Z Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

[LTC+20] Benjamin Lengerich, Sarah Tan, Chun-Hao Chang, Giles Hooker, and Rich Caruana. Purifying interaction effects with the functional anova: An efficient algorithm for recovering identifiable additive models. In *International Conference on Artificial Intelligence and Statistics*, pages 2402–2412. PMLR, 2020.

[McM92] Clayton McMillan. *Rule induction in a neural network through integrated symbolic and subsymbolic processing*. University of Colorado at Boulder, 1992.

[MHPG+22] Alireza Mousavi-Hosseini, Sejun Park, Manuela Girotti, Ioannis Mitliagkas, and Murat A Erdogdu. Neural networks efficiently learn low-dimensional representations with sgd. *arXiv preprint arXiv:2209.14863*, 2022.

[MLN19] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

[MMS91] Clayton McMillan, Michael C Mozer, and Paul Smolensky. Rule induction through integrated symbolic and subsymbolic processing. *Advances in neural information processing systems*, 4, 1991.

[MPF+23] Stefano Massaroli, Michael Poli, Daniel Y Fu, Hermann Kumbong, Rom N Parnichkun, Aman Timalsina, David W Romero, Quinn McIntyre, Beidi Chen, Atri Rudra, et al. Laughing hyena distillery: Extracting compact recurrences from convolutions. *arXiv preprint arXiv:2310.18780*, 2023.

[MR02] Dinesh Mehta and Vijay Raghavan. Decision tree approximations of boolean functions. *Theoretical Computer Science*, 270(1-2):609–623, 2002.

[MRT18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[MSK+19] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[MT23] Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

[MYZ13] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[NLW23] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

[PCJH22] Minh Pham, Minsu Cho, Ameya Joshi, and Chinmay Hegde. Revisiting self-distillation. *arXiv preprint arXiv:2206.08491*, 2022.

[PCV23] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

[PL19] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International conference on machine learning*, pages 5142–5151. PMLR, 2019.

[PPA18] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[PV88] Leonard Pitt and Leslie G Valiant. Computational limitations on learning from examples. *Journal of the ACM (JACM)*, 35(4):965–984, 1988.

[RCC+22] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys*, 16:1–85, 2022.

[RGC23] Shauli Ravfogel, Yoav Goldberg, and Ryan Cotterell. Log-linear guardedness and its implications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9413–9431, 2023.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[RTGC22] Shauli Ravfogel, Michael Twiton, Yoav Goldberg, and Ryan D Cotterell. Linear adversarial concept erasure. In *International Conference on Machine Learning*, pages 18400–18421. PMLR, 2022.

[SAM23] Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*, 2023.

[SPK16] Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1526–1534, 2016.

[TCHL18] Sarah Tan, Rich Caruana, Giles Hooker, and Yin Lou. Distill-and-compare: Auditing black-box models using transparent model distillation. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 303–310, 2018.

[THGN23] Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *arXiv preprint arXiv:2310.15154*, 2023.

[Thr93] Sebastian Thrun. *Extracting provably correct rules from artificial neural networks*. Citeseer, 1993.

[USSBD11] Ruth Urner, Shai Shalev-Shwartz, and Shai Ben-David. Access to unlabeled data can speed up prediction time. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 641–648, 2011.

[VAB07] Anneleen Van Assche and Hendrik Blockeel. Seeing the forest through the trees: Learning a comprehensible model from an ensemble. In *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18*, pages 418–429. Springer, 2007.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[VC71] VN Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264, 1971.

[VC74] Vladimir Vapnik and A Ya Chervonenkis. The method of ordered risk minimization, i. *Avtomatika i Telemekhanika*, 8:21–30, 1974.

[VC20] Francisco Vargas and Ryan Cotterell. Exploring the linear subspace hypothesis in gender bias mitigation. *arXiv preprint arXiv:2009.09435*, 2020.

[Vig19] Jesse Vig. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*, 2019.

[VLJ+17] Gilles Vandewiele, Kiani Lannoye, Olivier Janssens, Femke Ongenae, Filip De Turck, and Sofie Van Hoecke. A genetic algorithm for interpretable model extraction from decision tree ensembles. In *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2017 Workshops, MLSDA, BDM, DM-BPM Jeju, South Korea, May 23, 2017, Revised Selected Papers 21*, pages 104–115. Springer, 2017.

[WGNV23] Zihao Wang, Lin Gui, Jeffrey Negrea, and Victor Veitch. Concept algebra for score-based conditional model. In *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023.

[WGY21] Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR, 2021.

[WLLR23] Kaiyue Wen, Yuchen Li, Bingbin Liu, and Andrej Risteski. Transformers are uninterpretable with myopic methods: a case study with bounded dyck grammars. *arXiv preprint arXiv:2312.01429*, 2023.

[WWL19] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.

[XLT+24] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.

[ZG17] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[ZH16] Yichen Zhou and Giles Hooker. Interpreting models via single tree approximation. *arXiv preprint arXiv:1610.09036*, 2016.

[ZXH22] Yunzhe Zhou, Peiru Xu, and Giles Hooker. A generic approach for reproducible model distillation. *arXiv preprint arXiv:2211.12631*, 2022.

[ZZH18] Yichen Zhou, Zhengze Zhou, and Giles Hooker. Approximation trees: Statistical stability in model distillation. *arXiv preprint arXiv:1808.07573*, 2018.

A Evidence for the linear representation hypothesis

We discuss evidence for the linear representation hypothesis (LRH), which is that after training a neural network's representation map $\varphi_\theta : \mathcal{X} \to \mathbb{R}^m$ can efficiently represent high-level features and intermediate computations linearly [MYZ13, VC20, EHO+22]. This hypothesis motivated our distillation algorithm in Sections 3.2. The definition of the LRH is restated below:

Definition 3.4 (Linear representation hypothesis)**.** *Let $\mathcal{G}$ be a collection of functions $g : \mathcal{X} \to \mathbb{R}$ for each $g \in \mathcal{G}$. For any $\tau > 0$, the τ-LRH with respect to $\mathcal{G}$ states that for all $g \in \mathcal{G}$ there is a coefficient vector $\boldsymbol{w}_g \in \mathbb{R}^m$ such that $\|\boldsymbol{w}_g\| \leq \tau$ and*

$$\boldsymbol{w}_g \cdot \varphi(\boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x} \in \mathcal{X}\,.$$

We should mention that it is more natural to hypothesize the LRH in its approximate form. This variant was omitted from the main text for simplicity, but the guarantees for the decision tree algorithm could be extended to hold under the approximate LRH as well.

Definition A.1 (Approximate linear representation hypothesis)**.** *Let $\mathcal{G}$ be a collection of "high-level" functions, $g : \mathcal{X} \to \mathbb{R}$ for each $g \in \mathcal{G}$. For any $\tau > 0$ and $\epsilon > 0$, the (τ, ϵ)-LRH under distribution $\mathcal{D}$ is the hypothesis that for all $g \in \mathcal{G}$ there is a coefficient vector $\boldsymbol{w}_g \in \mathbb{R}^m$ such that $\|\boldsymbol{w}_g\| \leq \tau$ and*

$$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(\varphi(\boldsymbol{x}) \cdot \boldsymbol{w}_g(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \leq \epsilon\,.$$

There is quickly growing empirical and theoretical evidence for the LRH.

Empirical evidence for LRH The LRH originated in work on word embeddings [MYZ13], which discovered that linear subspaces of the embedding space encode semantic properties. This resulted in a "vector" algebra between embeddings, with relationships such as $\varphi(\text{"queen"}) \approx \varphi(\text{"king"}) - \varphi(\text{"man"}) + \varphi(\text{"woman"})$ that could be used to solve analogy problems.

Subsequently, this idea of semantic content was used to study neural networks via linear probes [AB16] of their internal representations. Various high-level features are encoded linearly. These include, but are not limited to:

- text sentiment in large language models [THGN23]
- syntactic content of a sentence in a machine-translation model [SPK16, CKL+18]
- the parsing tree in a model trained on a Context-Free-Grammar [AL23a]

- the Othello board's states in a language model trained to play based only off of move sequences [LHB+22, NLW23]
- the truth or falsehood of a statement in a large language model [MT23]
- content about a scene described in text [LNA21]
- learned factual knowledge in pretrained transformers [DDH+21, AL23b]
- semantic WordNet [Fel98] relations between words [AMS21]

The LRH has also been used to control models. One application is erasing concepts from trained models [BCZ+16, VC20, RTGC22, RGC23, BSJR+23], by projecting the internal representation orthogonal to the direction in which the concept is represented. Another application in [WGNV23] shows that the LRH can be used to extract concepts out of representations to control image generation with diffusion model. Additionally, [HLA23] give a method to edit representations of entities based on linear addition of an attribute vector to the entity's representation. Additionally, [PCV23] and suggest a linear transformation of the representation space to disentangle concepts and similarly edit attributes.

Finally, there is evidence that the same collection of linear features might be universally learned by distinct networks trained on the same dataset. Indeed, differently-initialized networks appear to converge to functionally similar representations when trained, in the sense that these representations behave similarly under linear transfer learning [BLSR22].

Theoretical evidence for LRH The LRH has been proved in certain settings.

- *Linear structure in word embeddings*: Under a generative model of text, which encodes topic structure [ALL+16] proves that the PMI method [CH90] constructs word embeddings that satisfy the LRH, in these sense that there are directions of space that encode concepts as observed in [MYZ13].
- *Juntas and multi-index functions*: The LRH has been proved for neural networks that have been trained by gradient-based methods to learn juntas or multi-index functions[10] [ABM22, DLS22, BES+22, MHPG+22, ABM23, DKL+23, BBPV23].
  - Suppose that f_θ is a neural network that has been trained by gradient-based methods to learn a junta $f_*(\boldsymbol{x}) = h(\boldsymbol{x}_S)$. Then the learned representation φ_θ can linearly represent *any* junta depending only on the variables $\boldsymbol{x}_S$, with a coefficient vector of norm $\tau = O_{|S|}(1)$. In contrast, before training, some of these same juntas would have required a much larger coefficient vector of norm $\tau \geq d^{\Omega(|S|)}$. To summarize, learning one junta on the variables S makes the network's representation very efficient at linearly representing any other junta on the same set of variables. See [ABM22] for more details.
  - For multi-index functions, a similar result is known. Multi-index functions are analogous to juntas, except that they depend on a small subspace of the input instead of on a small number of input coordinates. These are functions of the form $f(\boldsymbol{x}) = h(\boldsymbol{P}\boldsymbol{x})$, where $\boldsymbol{P} \in \mathbb{R}^{k \times d}$ is a projection to a k-dimensional subspace. In this case, a network trained on one multi-index function learns a representation that can approximate any other multi-index function on that same subspace with only a $\tau = O_k(1)$-norm vector of coefficients. See [DLS22] for more details.

[10] See Definition 3.1 for juntas, and definition of multi-index functions below.

B Deferred technical proofs

B.1 Lemmas for decision-tree distillation

B.1.1 Proof of linear probe subroutine, Lemma 3.7

Lemma 3.7 (Linear probe subroutine)**.** *Given a function $g : \mathcal{X} \to [-1, 1]$, a representation map $\varphi : \mathcal{X} \to \mathbb{R}^m$ of norm bounded by $B \geq \max_{\boldsymbol{x}} \|\varphi(\boldsymbol{x})\|$, a coefficient norm-bound $\tau > 0$, error tolerance parameters $\epsilon, \delta > 0$, and a distribution $\mathcal{D}$, there is a subroutine* LinearProbe$(g, \varphi, B, \tau, \epsilon, \delta, \mathcal{D})$ *that runs in* $\mathrm{poly}(1/\epsilon, \log(1/\delta), \tau, B, m)$ *time and draws* $\mathrm{poly}(1/\epsilon, \log(1/\delta), \tau, B)$ *samples from distribution $\mathcal{D}$ and returns* probe $\in \{\mathsf{true}, \mathsf{false}\}$ *such that:*

- *If there is $\boldsymbol{w} \in \mathbb{R}^m$ such that $\|\boldsymbol{w}\| \leq \tau$ and $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \leq \epsilon$, then* probe $= \mathsf{true}$ *with probability $\geq 1 - \delta$.*
- *If for all $\boldsymbol{w} \in \mathbb{R}^m$ such that $\|\boldsymbol{w}\| \leq \tau$ we have $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(\boldsymbol{w}\cdot\varphi(\boldsymbol{x})-g(\boldsymbol{x}))^2] \geq 2\epsilon$, then* probe $= \mathsf{false}$ *with probability $\geq 1 - \delta$.*

Proof of Lemma 3.7. Given i.i.d. samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \sim \mathcal{D}$ we can form the empirical loss

$$L_n(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) - g(\boldsymbol{x}_i))^2\,,$$

and construct $\hat{\boldsymbol{w}}$ by optimizing $\min_{\boldsymbol{w}}$ s.t. $_{\|\boldsymbol{w}\|\leq\tau} L_n(\boldsymbol{w})$ up to $\epsilon/4$ additive error in time $\mathrm{poly}(n, \epsilon, \tau, B, m)$ by Frank-Wolfe (e.g., Theorem 3.8 of [Bub15]). Because of the bounds on $g, \boldsymbol{w}, \varphi$, we can write the population loss

$$L(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}))^2]\,,$$

in the form

$$L(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\ell(\boldsymbol{w}, \boldsymbol{x})]\,,$$

where $\ell(\cdot, \cdot)$ is bounded by $(B\tau + 1)^2$ and is $\mathrm{poly}(B, \tau, 1)$-Lipschitz in its first argument. Thus, by standard Rademacher complexity arguments and McDiarmid's inequality (e.g., Proposition 4.5 of [Bac21]), it suffices to take a number of samples $n = \mathrm{poly}(\log(1/\delta), \epsilon, B, \tau)$ in order to guarantee that with probability at least $1 - \delta$ we have $|\min_{\boldsymbol{w}^*\in B(0,\tau)} L(\boldsymbol{w}^*) - L_n(\hat{\boldsymbol{w}})| \leq \epsilon/3$, which allows the probe to satisfy the claimed guarantee. □

B.1.2 Proof of linear representation of AND bound, Lemma 3.8

We prove Lemma 3.8, on the maximum number of ANDs that can have low kernel norm for a given representation. First, we prove a simpler version of this lemma, where we show that it is impossible to represent the AND functions *exactly*, without any error. We will later bootstrap this weaker version into the proof of Lemma 3.8, which shows that is hard to even *approximately* represent a large collection of AND functions.

In order to prove this lemma, we first make a definition of the index set of a clause, and prove an exact version of the lemma.

Definition B.1. *The* index set *of a clause $S \subseteq \{x_1, \ldots, x_d, \neg x_1, \ldots, \neg x_d\}$ is the set $\mathcal{I}(S) = \{i \in [d] : x_i \in S \text{ or } \neg x_i \in S\}$.*

Lemma B.2 (Exact linear representation of many ANDs requires high norm)**.** *Let $\mathcal{S}$ be a set of nondegenerate k-clauses such that all distinct pairs $S, S' \in \mathcal{S}$ have distinct index sets $\mathcal{I}(S) \neq \mathcal{I}(S')$. Let $\mathcal{G} = \{\mathrm{AND}_S \text{ for all } S \in \mathcal{S}\}$. If $\varphi : \{+1,-1\}^d \to \mathbb{R}^m$ satisfies the $(\tau, 0)$-LRH with respect to $\mathcal{G}$. Let $\beta^2 = \mathbb{E}_{\boldsymbol{x}}[\|\varphi(\boldsymbol{x})\|^2]$. Then*

$$\beta^2\tau^2 \geq 2^{-2k-2}|\mathcal{S}|\,.$$

Proof. Without loss of generality, we also use the input alphabet $\{+1,-1\}^d$ instead of $\{0,1\}^d$, since it is more notationally convenient for the Fourier-analytic techniques.

By the linear representation hypothesis, for each $S \in \mathcal{S}$, there $\boldsymbol{w}_S \in B(0,\tau)$ such that

$$\boldsymbol{w}_S \cdot \varphi(\boldsymbol{x}) = \mathrm{AND}_S(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \in \{+1,-1\}^d\,. \tag{B.1}$$

Define the matrix $\boldsymbol{W} \in \mathbb{R}^{\mathcal{S}\times m}$ with rows $\boldsymbol{w}_S$, and the matrix $\boldsymbol{\Phi} \in \mathbb{R}^{m\times 2^d}$ with columns $\varphi(\boldsymbol{x})$ for $\boldsymbol{x} \in \{+1,-1\}^d$. Writing equation (B.1) in matrix-form, we have

$$[\boldsymbol{W}\boldsymbol{\Phi}]_{S,\boldsymbol{x}} = \mathrm{AND}_S(\boldsymbol{x})\,. \tag{B.2}$$

We first prove that these are quite close to linearly independent.

Claim B.3 (Matrix of features has lower-bounded singular values)**.** $\boldsymbol{W}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{W}^\top \gtrsim 2^{d-2k-2}\boldsymbol{I}$

Proof of Claim B.3. Let $\chi_A(\boldsymbol{x}) = \prod_{i\in A} x_i$. Define $\boldsymbol{V} \in \mathbb{R}^{\binom{d}{k}\times 2^d}$ to be the matrix whose rows correspond to the degree-k monomials: $\boldsymbol{V}_{A,\boldsymbol{x}} = \chi_A(\boldsymbol{x})$ for all $A \in \binom{[d]}{k}$. Let $\boldsymbol{P}_k$ be the orthogonal projection matrix to the span of the rows of $\boldsymbol{V}$, and let $\boldsymbol{P}_k^\perp$ be the projection to the orthogonal complement. Since the only degree-k term in the Fourier expansion of AND_S is $\pm\frac{1}{2^{k+1}}\chi_{I(S)}(\boldsymbol{x})$, we have:

$$[\boldsymbol{W}\boldsymbol{\Phi}\boldsymbol{P}_k]_{S,\boldsymbol{x}} = \pm\frac{1}{2^{k+1}}\chi_{\mathcal{I}(S)}(\boldsymbol{x})\,.$$

This implies that

$$\begin{aligned}
\boldsymbol{W}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{W}^\top &= \boldsymbol{W}\boldsymbol{\Phi}(\boldsymbol{P}_k + \boldsymbol{P}_k^\perp)(\boldsymbol{P}_k + \boldsymbol{P}_k^\perp)^\top\boldsymbol{\Phi}^\top\boldsymbol{W}^\top \\
&= \boldsymbol{W}\boldsymbol{\Phi}\boldsymbol{P}_k\boldsymbol{P}_k^\top\boldsymbol{\Phi}^\top\boldsymbol{W}^\top + \boldsymbol{W}\boldsymbol{\Phi}\boldsymbol{P}_k^\perp(\boldsymbol{P}_k^\perp)^\top\boldsymbol{\Phi}^\top\boldsymbol{W}^\top \\
&\gtrsim \boldsymbol{W}\boldsymbol{\Phi}\boldsymbol{P}_k\boldsymbol{P}_k^\top\boldsymbol{\Phi}^\top\boldsymbol{W}^\top \\
&= 2^{d-2k-2}\boldsymbol{I}\,.
\end{aligned}$$

□

Next, we use the linear representation hypothesis (LRH) to prove that the singular values of the feature matrix are upper-bounded, which will yield a contradiction when combined with the previous claim:

Claim B.4 (Singular values upper-bounded by LRH)**.** $\det(\boldsymbol{W}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{W}) \leq (2^d\beta^2\tau^2/|\mathcal{S}|)^{|\mathcal{S}|}$.

Proof of Claim B.4. Notice that $m \geq |\mathcal{S}|$ since otherwise the rank of $\boldsymbol{W}\boldsymbol{\Phi}$ is not large enough for Claim B.3 to hold. Therefore, we can write the matrices in the LQ decomposition as $\boldsymbol{W} = \boldsymbol{L}\boldsymbol{Q}$ where $\boldsymbol{L} \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}$ is a lower-triangular matrix and $\boldsymbol{Q} \in \mathbb{R}^{\mathcal{S}\times m}$ has orthonormal rows. We have

$$\begin{aligned}
\tau^2 \geq \|\boldsymbol{w}_S\|^2 &= \sum_{i\in[m]}[\boldsymbol{L}\boldsymbol{Q}]_{S,i}^2 = \sum_{i\in[m]}\Big(\sum_{S'\in\mathcal{S}} L_{S,S'}\boldsymbol{Q}_{S',i}\Big)^2 \\
&= \sum_{i\in[m]}\sum_{S',S''} L_{S,S'}L_{S,S''}Q_{S',i}Q_{S'',i} = \sum_{S',S''} L_{S,S'}L_{S,S''}\sum_{i\in[m]} Q_{S',i}Q_{S'',i} = \sum_{S'} L_{S,S'}^2 \geq L_{S,S}^2\,.
\end{aligned}$$

So since $\boldsymbol{L}$ is lower-triangular we have

$$|\det(\boldsymbol{L})| = |\prod_{S\in\mathcal{S}} L_{S,S}| \le \tau^{|\mathcal{S}|}.$$

Furthermore, since $\boldsymbol{Q}$ is a semi-orthogonal matrix we have $\|\boldsymbol{Q}\| \le 1$ so each column of $\boldsymbol{Q\Phi}$ has norm at most

$$\|[\boldsymbol{Q\Phi}]_{*,\boldsymbol{x}}\| \le \|\boldsymbol{\Phi}_{*,\boldsymbol{x}}\| = \|\varphi(\boldsymbol{x})\|\,.$$

Therefore

$$\mathrm{tr}(\boldsymbol{Q\Phi\Phi}^\top\boldsymbol{Q}^\top) = \mathrm{tr}(\boldsymbol{\Phi}^\top\boldsymbol{Q}^\top\boldsymbol{Q\Phi}) = \sum_{\boldsymbol{x}} \|[\boldsymbol{Q\Phi}]_{*,\boldsymbol{x}}\|^2 \le \sum_{\boldsymbol{x}} \|\varphi(\boldsymbol{x})\|^2 \le 2^d\beta^2\,.$$

So since $\boldsymbol{Q\Phi\Phi}^\top\boldsymbol{Q}^\top$ is p.s.d. and has dimensions $|\mathcal{S}|\times|\mathcal{S}|$, we have

$$|\det(\boldsymbol{Q\Phi\Phi}^\top\boldsymbol{Q}^\top)| \le (\mathrm{tr}(\boldsymbol{Q\Phi\Phi}^\top\boldsymbol{Q}^\top)/|\mathcal{S}|)^{|\mathcal{S}|} \le (2^d\beta^2/|\mathcal{S}|)^{|\mathcal{S}|}\,.$$

We conclude that

$$\begin{aligned}|\det(\boldsymbol{W\Phi\Phi}^\top\boldsymbol{W})| &= |\det(\boldsymbol{LQ\Phi\Phi}^\top\boldsymbol{Q}^\top\boldsymbol{L}^\top)|\\ &= |\det(\boldsymbol{L})|^2\det(\boldsymbol{Q\Phi\Phi}^\top\boldsymbol{Q}^\top) \le (2^d\beta^2\tau^2/|\mathcal{S}|)^{|\mathcal{S}|}\,.\end{aligned}$$

□

We combine Claims B.3 and B.4:

$$2^{(d-2k-2)|\mathcal{S}|} \le \det(\boldsymbol{W\Phi\Phi}^\top\boldsymbol{W}^\top) \le (2^d\beta^2\tau^2/|\mathcal{S}|)^{|\mathcal{S}|}\,.$$

This implies

$$\beta^2\tau^2 \ge 2^{-2k-2}|\mathcal{S}|\,.$$

and thus we have proved Lemma B.2. □

We can now prove the lemma when the packing is only approximate.

Lemma 3.8 (Linearly representing many ANDs requires high norm)**.** *Let $\mathcal{S}$ be a collection of nondegenerate k-clauses. Let $\mathcal{G} = \{\mathrm{AND}_S \text{ for all } S\in\mathcal{S}\}$. Suppose that $\varphi:\{0,1\}^d\to\mathbb{R}^m$ approximately satisfies the τ-LRH with respect to $\mathcal{G}$ in the sense that for $g\in\mathcal{G}$ there is $\boldsymbol{w}_g\in\mathbb{R}^m$ with $\|\boldsymbol{w}_g\|\le\tau$ such that*

$$\mathbb{E}_{\boldsymbol{x}\sim\mathrm{Unif}\{0,1\}^d}[(\boldsymbol{w}_g\cdot\varphi(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \le 2^{-k-2}\,.$$

Then

$$|\mathcal{S}| \le 2^{3k+4}\tau^2\mathbb{E}_{\boldsymbol{x}\sim\mathrm{Unif}\{0,1\}^d}[\|\varphi(\boldsymbol{x})\|^2]\,.$$

Lemma 3.8. Without loss of generality, we use the input alphabet $\{+1,-1\}^d$ instead of $\{0,1\}^d$, since it is more notationally convenient for the Fourier-analytic techniques.

Assume without loss of generality that for all $S,S'\in\mathcal{S}$, the variables on which S depends are a distinct set from the variables on which S' depends. This can be ensured by shrinking $|\mathcal{S}|$ by a factor of at most 2^k. Thus, it suffices to show the following:

$$\tau^2\mathbb{E}_{\boldsymbol{x}}[\|\varphi(\boldsymbol{x})\|^2] \ge 2^{-2k-4}|\mathcal{S}|\,. \tag{B.3}$$

By the approximate LRH, for each $S \in \mathcal{S}$, there is $\boldsymbol{w}_S \in \mathbb{R}^m$ and a function $h_S \in L^2(\mathrm{Unif}\{+1,-1\}^d)$ such that

$$h_S(\boldsymbol{x}) = \mathrm{AND}_S(\boldsymbol{x}) - \boldsymbol{w}_S \cdot \varphi(\boldsymbol{x}), \text{ and } \|h_S\| \le 2^{-k-2}\,.$$

We use this to construct a new embedding for each variable $\psi(\boldsymbol{x})$ that can *exactly* represent all of the ANDs in $\mathcal{S}$ with low norm. This embedding is given by the concatenation:

$$\psi(\boldsymbol{x}) := (\varphi(\boldsymbol{x}), \frac{1}{\tau}[h_S(\boldsymbol{x})]_{S\in\mathcal{S}}) \in \mathbb{R}^{m+|\mathcal{S}|}.$$

We also define

$$\boldsymbol{v}_S = (\boldsymbol{w}_S, \tau \boldsymbol{e}_S) \in \mathbb{R}^{m+|\mathcal{S}|},$$

so that

$$\boldsymbol{v}_S \cdot \psi(\boldsymbol{x}) = \boldsymbol{w}_S \cdot \varphi(\boldsymbol{x}) + h_S(\boldsymbol{x}) = \mathrm{AND}_S(\boldsymbol{x}).$$

For each $S \in \mathcal{S}$, we have

$$\|\boldsymbol{v}_S\|^2 = \|\boldsymbol{w}_S\|^2 + \tau^2 \le 2\tau^2$$

and we also have

$$\mathbb{E}_{\boldsymbol{x}}[\|\psi(\boldsymbol{x})\|^2] = \mathbb{E}_{\boldsymbol{x}}[\|\varphi(\boldsymbol{x})\|^2] + \sum_{S\in\mathcal{S}} \mathbb{E}_{\boldsymbol{x}}[h_S(\boldsymbol{x})^2/\tau^2] \le \mathbb{E}_{\boldsymbol{x}}[\|\varphi(\boldsymbol{x})\|^2] + 2^{-2k-4}|\mathcal{S}|/\tau^2$$

Thus the representation $\psi(\boldsymbol{x})$ satisfies the $\sqrt{2}\tau$-LRH, and by Lemma B.2 we know that

$$2\tau^2(\mathbb{E}_{\boldsymbol{x}}[\|\varphi(\boldsymbol{x})\|^2] + 2^{-2k-4}|\mathcal{S}|/\tau^2) \ge 2^{-2k-2}|\mathcal{S}|\,,$$

which implies

$$\tau^2 \mathbb{E}_{\boldsymbol{x}}[\|\varphi(\boldsymbol{x})\|^2] \ge 2^{-2k-4}|\mathcal{S}|/\tau^2\,,$$

which is the claim in (B.3) that we had to show. □

B.2 Composing distillation with agnostic distillation

We fill out the details for Remark 4.4 here.

First, note that two distillations compose in a natural way.

Lemma B.5 (Composing distillation)**.** *Suppose that $\mathcal{F}$ can be (ϵ_1, δ_1)-distilled into $\mathcal{G}$ in s_1 samples and t_1 time, and suppose also that $\mathcal{G}$ can be (ϵ_2, δ_2)-distilled into $\mathcal{H}$ in s_2 samples and t_2 time. Then $\mathcal{F}$ can be $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-distilled into $\mathcal{H}$ in $\max(s_1, s_2)$ samples and $t_1 + t_2$ time.*

Proof. Immediate by composing the two distillation algorithms. □

However, as pointed out in Remark 4.4, two agnostic distillations do not compose. On the other hand, distillation composes with agnostic distillation, as we point out here.

Lemma B.6 (Composing distillation with agnostic distillation)**.** *Suppose that $\mathcal{F}$ can be (ϵ_1, δ_1)-distilled into $\mathcal{G}$ in s_1 samples and t_1 time, and suppose also that $\mathcal{G}$ can be agnostically (ϵ_2, δ_2)-distilled into $\mathcal{H}$ in s_2 samples and t_2 time. Then $\mathcal{F}$ can be agnostically $(2\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-distilled into $\mathcal{H}$ in $\max(s_1, s_2)$ samples and $t_1 + t_2$ time.*

Proof. Compose the distillation from $\mathcal{F}$ to $\mathcal{G}$ with the agnostic distillation from $\mathcal{G}$ to $\mathcal{H}$. Let $f \in \mathcal{F}$ be the input, $g \in \mathcal{G}$ be the intermediate step, and $h \in \mathcal{H}$ be the output. By a union bound, with probability $\delta_1 + \delta_2$,

$$\begin{aligned}
\operatorname{error}_{\mathcal{D}}(h; f) &\leq \operatorname{error}_{\mathcal{D}}(h; g) + \operatorname{err}_{\mathcal{D}}(g; f) \\
&\leq \operatorname{error}_{\mathcal{D}}(h; g) + \epsilon_1 \\
&\leq \min_{\tilde{h} \in \mathcal{H}} \operatorname{error}_{\mathcal{D}}(\tilde{h}; g) + \epsilon_2 + \epsilon_1 \\
&\leq \min_{\tilde{h} \in \mathcal{H}} \operatorname{error}_{\mathcal{D}}(\tilde{h}; f) + 2\epsilon_1 + \epsilon_2 \,.
\end{aligned}$$

□