# Generalization on the Unseen, Logic Reasoning and Degree Curriculum

Emmanuel Abbe [1 2] Samy Bengio [2] Aryo Lotfi [1] Kevin Rizk [1]

## Abstract

This paper considers the learning of logical (Boolean) functions with focus on the *generalization on the unseen (GOTU)* setting, a strong case of out-of-distribution generalization. This is motivated by the fact that the rich combinatorial nature of data in certain reasoning tasks (e.g., arithmetic/logic) makes representative data sampling challenging, and learning successfully under GOTU gives a first vignette of an 'extrapolating' or 'reasoning' learner. We then study how different network architectures trained by (S)GD perform under GOTU and provide both theoretical and experimental evidence that for a class of network models including instances of Transformers, random features models, and diagonal linear networks, a *min-degree-interpolator* is learned on the unseen. We also provide evidence that other instances with larger learning rates or mean-field networks reach leaky min-degree solutions. These findings lead to two implications: (1) we provide an explanation to the *length generalization problem* (e.g., Anil et al. 2022); (2) we introduce a curriculum learning algorithm called *Degree-Curriculum* that learns monomials more efficiently by incrementing supports.

## 1. Introduction

Neural networks trained by stochastic gradient descent (SGD) have proved to be a powerful learning paradigm when there is enough representative data about the distribution to be learned, specifically in applications involving images or text where there is also a good understanding of the relevant architectures.

There is now an increasing interest in tackling tasks involving more 'reasoning' components, which depart from classical perception tasks of images and texts. While such tasks remain vaguely defined, a list that we consider here under this class is given by: (1) arithmetic and algebra (Saxton et al., 2019; Lewkowycz et al., 2022), (2) synthetic tasks such as PVR (Zhang et al., 2021) and LEGO (Zhang et al., 2022), (3) visual reasoning such as CLEVR (Johnson et al., 2017), (4) physical reasoning such as Phyre (Bakhtin et al., 2019), (5) algorithmic data such as CLRS (Veličković et al., 2022) and reasoning on graphs (Mahdavi et al., 2022).

One common trademark of these tasks is that the input space is usually of discrete/combinatorial nature, and consequently, the data may not necessarily lay on a low dimensional manifold that is well sampled. In various cases, the input space may even have a variable length. This combinatorial nature is already present in text, but it is further amplified in, say, arithmetic since most symbol combinations could a priori represent a valid input (in contrast to text). Further, the target function in such tasks may rely on a large composition of logical steps or mathematical operations that require to be jointly learned. Therefore, in such reasoning tasks, the setting with abundant representative data seems less prominent. This motivates us to focus on a strong out-of-distribution (OOD) generalization setting.

For instance, when learning arithmetic or logic functions on a training set with a bounded length or bounded number of truth assignments, how would the neural network generalize on more general input assignments (this is a case of length generalization)? When training a neural network to learn a Boolean formula, such as a voting scheme on data from a polarized cohort of voters, how does the network generalize to an unpolarized cohort?

We thus consider the problem of learning functions with a holdout domain where part of the distribution support is barely/never seen at training, and with target functions that are Boolean to capture the discrete and combinatorial nature of various reasoning tasks (e.g., arithmetic, decision trees, logical circuits). Learning successfully under holdout gives a first vignette that the learner is operating with a certain amount of 'reasoning' or 'extrapolation' since memorization is voided on the unseen domain.

### 1.1. Our main contributions

1. We lay down some basic principles of stronger generalization requirements that rely on the 'generalization

[1]EPFL [2]Apple. Correspondence to: Aryo Lotfi <aryo.lotfi@epfl.ch>.

*Proceedings of the 40th International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

on the unseen (GOTU)' performance metric, defined as a strong case of OOD generalization (Section 2), setting a benchmark for 'extrapolating' or 'reasoning' solutions on the considered tasks.

2. We study how standard neural network architectures trained by (S)GD perform on the GOTU metric, in particular, which solutions are learned on the unseen domain for such architectures:
   (i) we prove two theoretical results showing that for a class of network models including random features model (Theorem 3.8) and deep diagonal linear networks (Theorem 3.11), a *min-degree-interpolator (MD interpolator)* is learned on the unseen;
   (ii) we show experimental results (Section 4) supporting that Transformers tend to also have the *min-degree bias (MD bias)* towards min-degree solutions.

   The MD interpolator is defined as the interpolator of minimal *degree-profile*, i.e., the Boolean function interpolating the training data and having a Fourier-Walsh transform whose energy concentrates on basis elements of lowest possible degree. Connections to algebraic geometry are given in Appendix C in order to characterize how MD interpolators can be constructed from the 'vanishing ideal' of the seen data. We also point out that very large learning rates or other architectures (such as mean-field networks) can exhibit leaky MD bias (i.e., assigning larger mass on higher-degree monomials); see Appendix B.2.

3. Using these, we obtain two additional results:
   (i) we provide a formal explanation (Theorem 5.1) to the 'length generalization problem' discussed in (Anil et al., 2022) (for the case of bounded weight vectors, also related to (Zhang et al., 2022));
   (ii) we turn the min-degree bias into an asset to accelerate learning by introducing a curriculum learning algorithm called 'Degree-Curriculum' (Algorithm 1), which successively increases the input complexity with respect to Hamming weights in order to incrementally learn the monomials support (see Section 5.2).

## 2. Generalization on the Unseen

The classical setting of statistical learning theory requires the control of three error pillars for the generalization of a learning model: (1) the approximation error (depending on the properties/richness of the model class), (2) the estimation error (depending on the properties/richness of the training set), (3) the optimization error (depending on the properties/richness of the training algorithm).

In some of the recent deep learning applications for computer vision and natural language processing, the richness of the training set, the size of the model and its alignment with the data, as well as the computational power, make the three pillars well controlled. The recent success of large language models (LLM) and scaling laws are perfect examples of this phenomenon (Alabdulmohsin et al., 2022).

As mentioned in the introduction, the type of data occurring in reasoning tasks is slightly different due to the richness and combinatorial nature of the data. To better cope with this challenge, we propose in this paper to depart from the classical generalization objectives described with the three pillars. We focus instead upfront on distribution shift and, more specifically, a strong case of OOD generalization where part of the distribution domain is almost/completely unseen at training but used at testing (in particular, prohibiting any memorization scheme).

Of course, on the unseen domain, all bets are off for generalization: one cannot hope for an algorithm trained on a given data domain to perform well on a larger data domain without any incentive to do so. Yet various algorithms will have various implicit biases on the unseen and thus produce various solutions on the unseen. Understanding this 'bias on the unseen' for different network architectures and Boolean target functions is the objective of this paper.

We start by redefining the generalization error when the train and test distribution are not necessarily the same.

**Definition 2.1.** Let $X_1, \dots, X_m$ be samples drawn i.i.d. under $\mu_1$ and labeled by a target function $f$, and let $\tilde{f}$ be the function learned by a learning algorithm. The algorithm has $(\mu_1, \mu_2, m, \epsilon)$-generalization (for loss $\ell$) if $\mathbb{E}_{X^m \sim \mu_1^{\otimes m}, X_{m+1} \sim \mu_2}[\ell(\tilde{f}_{X^m}(X_{m+1}), f(X_{m+1}))] \leq \epsilon$. In other words, the algorithm is trained under distribution $\mu_1$ and tested under distribution $\mu_2$, producing $\epsilon$-test-loss with sample complexity $m$.

Now we focus on a special case of interest, a strong case where we essentially see all the data on some part of the domain but miss another part. Naturally, we will next study a 'soft version' of this metric, where both in-distribution and out-of-distribution generalization are considered, but this strong case is already rich and insightful.

**Definition 2.2** (Generalization on the Unseen)**.** Consider a given sample space $\Omega$. During training, part of $\Omega$ is not sampled, and we call this the unseen domain (or the holdout set) $\mathcal{U}$. At testing, however, we sample from the full set $\Omega$. This represents a special case of the previous definition where $\mu_1 = \mu|_{\Omega \setminus \mathcal{U}}$ and $\mu_2 = \mu|_{\Omega}$ for some $\mu$.

We now further specify the setting: we assume that the training error is 0 on the training set $\Omega \setminus \mathcal{U}$, e.g., seeing all the samples in $\Omega \setminus \mathcal{U}$, and define the generalization on the unseen (GOTU) for an algorithm $\tilde{f}$ and target function $f$ as

$$GOTU(f, \tilde{f}, \mathcal{U}) = \mathbb{E}_{X \sim_U \mathcal{U}}[\ell(\tilde{f}_{\Omega \setminus \mathcal{U}}(X), f(X))], \qquad (1)$$

where $\sim_U \mathcal{U}$ indicates uniform sampling from $\mathcal{U}$. Notice we only sample on $\mathcal{U}$ at testing because we assumed zero training error and considered the whole $\Omega \setminus \mathcal{U}$ as the training set.

A few remarks are in order:

- GOTU is a special case of OOD and distribution shift setting that is extremal in the sense that it completely gives access to part of the distribution domain and completely omits the complement. Since we consider rich enough models to interpolate the data, the 'statistical' and 'approximation' pillars of the learning problem are removed (there may still be randomness used by the learning algorithm, thus statistical analysis may still be relevant). The problem thus turns into a pure optimization problem where the central object of study is the implicit bias of the learning algorithm on the unseen. Note that this is not exactly the same implicit bias as studied in the setting of overparametrized models (Soudry et al., 2017; Gunasekar et al., 2017; 2018b; Arora et al., 2019; Razin & Cohen, 2020; Chizat & Bach, 2020; Moroshko et al., 2020) as here we have the distribution shift and investigate the behavior of the equivalence class of interpolators on the unseen $\mathcal{U}$.
- In some experiments, we replace the 'perfect' training data on the seen domain with a 'large' sampling on the seen domain. We defined the GOTU in the extreme case to simplify the number of parameters to track and to allow for cleaner theorem statements, but there could also be a sampling rate on $\Omega \setminus \mathcal{U}$; this is left for future research. Also, we assume a uniform prior here because this is a natural first case for arithmetic/logic tasks, but this could also be relaxed.
- We will consider different subsets $\mathcal{U}$ in the applications. We are sometimes interested in $\mathcal{U}$'s for which the data invariances or equivariances could give hope to learn. This is further specified with the next definition.

**Definition 2.3.** A function $f : \Omega \to \mathbb{R}$ is (1) $G$-invariant or invariant under the group action $G$ on $\Omega$ if $f(gx) = f(x)$ for all $g \in G$, $x \in \Omega$; (2) $G_{i,o}$-equivariant or equivariant under the action $G_{i,o}$ if $f(g_i(x)) = g_o(f(x))$ for all $(g_i, g_o) \in G_{i,o}$ and $x \in \Omega$.

As stated earlier, we cannot expect algorithms to generalize on the unseen domain by themselves. However, we can hope that certain training algorithms will catch invariances/equivariances and thus extrapolate. For example consider the parity function on $d$ bits defined as $f(x_1, \ldots, x_d) = x_1 x_2 \cdots x_d$. This function is permutation-invariant (group $G = S_d$). In particular, if one uses a model favoring permutation symmetries, one may not have to see all inputs that are permutation equivalent. There has been a series of works designing layers/architectures that are equivalent under a prespecified family of actions (e.g., all permutations) (Ravanbakhsh et al., 2017; Zaheer et al., 2017; Hartford et al., 2018). More recently, (Zhou et al., 2020) proposes a method to learn invariances in a multi-task setting using meta-learning. An example of an equivariant Boolean function would be the majority function on $\{+1, -1\}^d$, $d$ odd, with the action of global bit flipping on the input and the output (since the majority is reversed if all the bits are flipped). Thus a holdout on vectors of dual-weight could again be handled by a model having such an equivariance. Note that we are also interested in cases where these equi/in-variances are not present in the target, to understand what solutions neural nets favor on the unseen.

# 3. Results

We consider $f : \Omega \to \mathbb{R}$ with $\Omega = \{\pm 1\}^d$. We introduce some preliminary material on Boolean functions in the next part and then state our results.

## 3.1. Preliminaries

**Fourier-Walsh transform.** Any function $f : \{\pm 1\}^d \to \mathbb{R}$ can be expressed as $f(x) = \sum_{T \in [d]} \hat{f}(T) \chi_T(x)$, where $\chi_T(x) = \prod_{i \in T} x_i$ are the monomials and $\hat{f}(T) = \mathbb{E}_{X \sim_U \{\pm 1\}^d} [\chi_T(X) f(X)]$ are the coefficients. For example, the majority function on 3 bits can be written as $\mathrm{Maj}(x_1, x_2, x_3) = \frac{1}{2}(x_1 + x_2 + x_3 - x_1 x_2 x_3)$.

**Unseen domain and vanishing ideals.** We now introduce the unseen domain $\mathcal{U}$. First, consider the canonical holdout, when a bit is frozen during training, e.g., $x_i = 1$ and $\mathcal{U} = \{x \in \{\pm 1\}^d : x_i = -1\}$. In this case, one can see that any function of the form $f(x) + (1 - x_i)\Delta(x)$ ($\Delta(x)$ is arbitrary) is an equivalent interpolator on the training data. For general unseen domain $\mathcal{U} \subseteq \Omega = \{\pm 1\}^n$, there exist polynomials $v_1(x), \cdots, v_k(x)$ such that $x \in \Omega \setminus \mathcal{U} \iff v_1(x) = \ldots = v_k(x) = 0$ (see Appendix C). Consequently, all solutions of the form $f(x) + \Delta_1(x) v_1(x) + \cdots + \Delta_k(x) v_k(x)$ are equivalent at training. This is the quotient space of $f$ under the vanishing ideal defined by $\Omega \setminus \mathcal{U}$. We refer to Appendix C for more details on this relation to algebraic geometry.

We now define measures of complexity relevant to us.

**Definition 3.1** (Degree)**.** For a function $f : \{\pm 1\}^d \to \mathbb{R}$, the degree $\deg(f)$ refers to the maximum degree of the monomials present in the Fourier-Walsh transform of $f$.

**Definition 3.2** (Degree profile)**.** For $f : \{\pm 1\}^d \to \mathbb{R}$, we define the degree-profile of $f$, $\mathrm{DegP}(f) \in \mathbb{R}^{d+1}$ such that $\mathrm{DegP}(f)_i = \sum_{T \subseteq [d], |T| = d+1-i} \hat{f}(T)^2$ for $1 \leq i \leq d+1$. Furthermore, we consider lexicographic ordering on these vectors, i.e., $\mathrm{DegP}(f) < \mathrm{DegP}(g)$ iff $\exists i$ $\mathrm{DegP}(f)_i < \mathrm{DegP}(g)_i$ and $\mathrm{DegP}(f)_j = \mathrm{DegP}(g)_j$ $1 \leq j < i$. For example, the degree-profile of $\mathrm{Maj}(x_1, x_2, x_3)$ is

$(1/4, 0, 3/4, 0)$, while its degree is 3.

Note that the degree-profile is a stronger notion of degree, i.e., $\deg(f) < \deg(g) \implies \mathrm{DegP}(f) < \mathrm{DegP}(g)$.

**Definition 3.3** (Min-degree interpolators)**.** Consider a target function $f$ and unseen domain $\mathcal{U}$. The set of interpolators is defined as $\mathcal{F}_{\text{int}}(f, \mathcal{U}) = \{g : \{\pm1\}^d \to \mathbb{R} \mid g(x) = f(x), \forall x \in \mathcal{U}^c\}$, where $\mathcal{U}^c := \Omega \setminus \mathcal{U}$ is the seen domain. We call an interpolator a *min-degree interpolator (MD interpolator)* of $(f, \mathcal{U})$ (or of $\{x, f(x)\}_{x \in \mathcal{U}^c}$) if it is an element of $\mathcal{F}_{\text{int}}(f, \mathcal{U})$ that minimizes the degree-profile with respect to the lexicographic order. This means that no part of the Fourier-Walsh expansion of the interpolator could be replaced with a lower-degree alternative and still interpolate.

For example, consider the case of 'canonical holdout' where we always have $x_1 = 1$ at training, i.e., $\mathcal{U} = \{x \in \{\pm1\}^d : x_1 = -1\}$, and target function $x_1x_2 + x_1x_3x_4$. Here, both $x_1x_2 + x_3x_4$ and $x_2 + x_3x_4$ are of degree 2 but only $x_2 + x_3x_4$ is an MD interpolator since $x_1x_2$ in the first function is replaceable with the lower-degree $x_2$. Further, note that there may be multiple interpolators having minimal max-degree rather than degree-profile. For example, consider the unseen domain induced by $x_i = x_j$ and target $f(x) = x_i + x_j$. Then $2x_i$ and $x_i + x_j$ are both interpolators with minimal max-degree, but only $x_i + x_j$ is an interpolator with a minimal degree-profile. In fact, the MD interpolator is always unique (if $f_1$ and $f_2$ are interpolators with the same degree-profile, then $\frac{f_1+f_2}{2}$ is an interpolator with a smaller degree-profile unless $f_1 = f_2$.)

## 3.2. Main theoretical results

We show that certain models have a min-degree implicit bias on the unseen. We start by giving another example.

### 3.2.1. Result preview from an example

Consider trying to learn the majority target function on 3 voters $x_1, x_2, x_3$ having the following data distribution: voters 1 and 2 never vote both negatively, i.e., $(x_1, x_2)$ is never $(-1, -1)$ in the training data. Now train a neural network to learn the target on such a training data distribution (with only 3 variables, one will quickly see all sequences satisfying the required condition; this is to simplify the example, in our results, we consider higher dimensional versions of such examples). Since we always have $(x_1, x_2) \neq (-1, -1)$, it must be the case that $(1 - x_1)(1 - x_2) = 0$ (this ensures that either $x_1$ or $x_2$ must be equal to 1). Thus, the functions $f(x)$ or $f(x) + \Delta(x)(1 - x_1)(1 - x_2)$ (for any arbitrary $\Delta$) are equivalent on the training data. One can thus wonder which $\Delta$ function will a neural network trained by (S)GD converge to. There is no reason to expect that it will converge to $\Delta = 0$; so can we characterize which $\Delta$ will occur?

Our main results show that —(i) provably for random features model or diagonal linear networks in the linear case (two architectures that we can analyze rigorously), and (ii) empirically for Transformers — (S)GD will converge to a $\Delta$ that makes $f(x) + \Delta(x)(1 - x_1)(1 - x_2)$ having the lowest 'degree-profile' (see Definition 3.2), which in the above majority example is obtained as follows: first expand the target in the basis of multivariate monomials, $\mathrm{Maj}(x_1, x_2, x_3) = (x_1 + x_2 + x_3 - x_1x_2x_3)/2$, then find $\Delta(x)$ that makes $(x_1 + x_2 + x_3 - x_1x_2x_3)/2 + \Delta(x)(1 - x_1)(1 - x_2)$ having the least $\ell_2$ mass on the highest degree monomials, i.e., in this case, $\Delta(x) = x_3/2$, giving $(x_1 + x_2 + x_3 - x_1x_2x_3)/2 + \Delta(x)(1 - x_1)(1 - x_2) = (x_1 + x_2 + 2x_3 - x_1x_3 - x_2x_3)/2$ which is degree 2 rather than 3 (see Figure 10 for numerical experiments). This paper describes what are the general mathematical concepts behind this specific example: (i) Fourier-Walsh Boolean analysis, (ii) the notion of vanishing ideal, and (iii) minimal degree-profile interpolators and the implicit bias of neural networks towards them.

### 3.2.2. General case

Our first result is on learning sparse Boolean functions with the random features model.

**Definition 3.4.** We consider a $P$-dimensional latent function $h : \{\pm1\}^P \to \mathbb{R}$ embedded in ambient dimension $d$. More precisely, we consider learning $f : \{\pm1\}^d \to \mathbb{R}$ such that $f(x) = h(x_{i_1}, \ldots, x_{i_P})$. We further denote $I = \{i_1, \ldots, i_P\}$ and $x_I = (x_{i_1}, \ldots, x_{i_P})$. We also assume that some specific combinations of $x_I$ are not present in the training samples, i.e., $x_I \notin \mathcal{U}^* \subset \{\pm1\}^P$ and define the unseen domain as $\mathcal{U} = \{x \in \{\pm1\}^d \mid x_I \in \mathcal{U}^*\}$.

Note that considering sparse functions enables us to define the unseen domain properly and differentiate between the unseen domain (where there are minimal structures) and unseen data (for example when there is uniform sampling).

Our first result is for the random features (RF) model (Rahimi & Recht, 2007). The RF model was initially introduced to approximate kernels and enhance the time complexity of kernel methods (Rahimi & Recht, 2007). RF models can also be viewed as approximations of neural networks in the NTK regime (Jacot et al., 2018; Ghorbani et al., 2019; Mei & Montanari, 2022). In this paper, we take the latter view on them as well, with the following formulation.

**Definition 3.5** (Random features model)**.** Consider $x \in \mathbb{R}^d$ as the input; we define random features model with $N$ random features as

$$f_{\mathrm{RF}}(x; a, w, b) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} a_i \sigma(\langle w_i, x \rangle + b_i), \tag{2}$$

where $a_i \in \mathbb{R}$ are the trainable parameters, $\sigma$ is the activation function, and $w_i, b_i \sim \mathcal{N}(0, \frac{1}{d})^{\otimes d} \otimes \mathcal{N}(0, \frac{1}{d})$ are the random weights and biases. We use $\phi_i(x) := \sigma(\langle w_i, x \rangle + b_i)$ as a shorthand notation for the $i$-th feature.

The following activation property strengthens the condition presented in (Abbe et al., 2022c).

**Definition 3.6** (Strongly expressive)**.** We call a continuous activation function $\sigma : \mathbb{R} \to \mathbb{R}$ strongly expressive up to $P$ if (A1) $\sigma$ satisfies upper bound $\mathbb{E}_{g\sim\mathcal{N}(0,2)}[\sigma(g)^4] < \infty$; and (A2) $\forall T \subseteq [d], |T| \le P$ $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2] = \Omega_d(d^{-|T|})$, where $\hat{\phi}_{w,b}(T) := \mathbb{E}_x[\sigma(\langle w, x\rangle + b)\chi_T(x)]$ is the Fourier coefficient of $T$ in the random feature created by $w, b$.

As will be proven in Lemma A.1, property (A1) implies $\mathbb{E}[\hat{\phi}_{w,b}(T)^2] = O(d^{-|T|})$ for $|T| = O_d(1)$. Therefore, the second condition (A2) is ensuring that the model is able to strongly express degree $k \le P$ monomials.

We note that $\hat{\phi}_{w,b}(T)^2$ has been studied in (Abbe et al., 2022c) as the initial alignment (INAL) between monomial $\chi_T(x)$ and $\phi_{w,b}(x)$. Indeed, based on Lemma A.2. of (Abbe et al., 2022c), the following conditions give us a family of strongly expressive activation functions.

**Lemma 3.7.** *Any continuous polynomially-bounded function $\sigma$ such that its first $P$ coefficients in the Hermite expansion are non-zero is strongly expressive up to $P$.*

For example, polynomial activation functions such as $(1+x)^k$ are strongly expressive up to $k$.

**Theorem 3.8.** *Let $f : \{\pm 1\}^d \to \mathbb{R}$ be a $P = O_d(1)$-sparse function to be learned in the GOTU setting (Definition 3.4) by a random features model with parameters $(N, \sigma, a, b, w)$ (Definition 3.5) with a strongly expressive activation function. As $N$ diverges, the random features model can interpolate the training data with high probability. Furthermore, defining $f_{\mathrm{RF}}^{d,N}(\mathcal{U})$ to be the interpolating solution minimizing $\|a\|_2$ (i.e., the solution reached by gradient descent/flow starting from $a = 0$ under $\ell_2$ loss), we have w.h.p.*

$$f_{\mathrm{RF}}^{d,N}(\mathcal{U}) \overset{N\to\infty}{\to} \mathrm{MinDegInterp}(f, \mathcal{U}) + \epsilon_d \qquad (3)$$

*where $\mathrm{MinDegInterp}(f, \mathcal{U})$ is the min-degree interpolator on the training data $\{x, f(x)\}_{x\in\mathcal{U}^c}$ and $\epsilon_d$ is a function on $P$ variables that tends pointwise to 0 as d diverges. (We refer to the above as a 'min-degree bias' or 'MD bias'.)*

*Proof Sketch.* In Lemma A.1, we show that random features generated by a strongly expressive $\sigma$ have in general a decaying degree-profile with $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2] = \Theta(d^{-|T|})$ for $|T| \le P$. We then investigate the interpolators in the Fourier-Walsh basis and show that the minimality condition of $\|a\|_2$ is equivalent to learning the minimal degree-profile interpolator since high-degree monomials are less expressed in the features and consequently larger $\|a\|$'s are required to capture them. The full proof relies on concentration results and Boolean Fourier analysis and is given in Appendix A.

*Remark* 3.9 (Other activation functions). Note that Theorem 3.8 does not hold for any arbitrary activation function. For example, if $\sigma(z) = z^2$, one can easily see that $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(x)(\{i\})^2], \mathbb{E}_{w,b}[\hat{\phi}_{w,b}(x)(\{i,j\})^2] \in \Theta_d(d^{-2})$, and hence degree 1 monomials have no priority over degree 2 monomials. An important case is the ReLU activation. Results of (Abbe et al., 2022c) show that for the ReLU activation and $|T| \le P$, we have

$$\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2] = \begin{cases} \Omega(d^{-|T|}) & |T| \text{ even or } |T| = 1 \\ \Omega(d^{-|T|-1}) & \text{otherwise} \end{cases}. \qquad (4)$$

Consequently, the min-degree bias still exists, but in a weaker form. For further discussion and experiments on ReLU activation refer to Appendix A.

In the experiments, we show that having the sparsity assumption may not be necessary in some cases, and the min-degree bias can be observed for small values of $d$ and $N$ as well. Furthermore, we show that the min-degree bias goes beyond the random features and NTK models; see Section 4.

We next move to a theorem on deep diagonal linear neural networks where we will be able to analyze non-linear dynamics for gradient flow. Note that in the case of linear functions, replacing a degree-1 variable $x_k$ with the degree-0 variable 1 is the only case of lower degree bias. In other words, we consider the case that unseen data is $\mathcal{U} = \{x \mid x_k = -1\}$ (referred to canonical holdout in (Abbe et al., 2022a)). We show that diagonal linear neural networks learn the min-degree interpolator with a leakage factor that vanishes as their initialization scale is small enough or as their depth is large enough. We now define diagonal linear neural networks with bias.

**Definition 3.10** (Diagonal linear neural network with bias)**.** We define a diagonal linear neural network (DLNN) with bias as an extension of diagonal neural networks, where there is only one parameter for bias at the last layer. I.e.,

$$\theta = (b, w_1^{(1)}, \dots, w_d^{(1)}, \dots, w_1^{(L)}, \dots, w_d^{(L)}),$$

$$f_{\mathrm{NN}}(x_1, \dots, x_d; \theta) = b + \sum_{i=1}^{d} \left( \prod_{l=1}^{L} w_i^{(l)} \right) x_i,$$

where $\theta$, $d$, and $L$ represent the model's parameters, input dimension, and depth, respectively.

**Theorem 3.11.** *Let $f : \{\pm 1\}^d \to \mathbb{R}$ be a linear function, i.e., $f(x_1, \cdots, x_d) = \hat{f}(\emptyset) + \sum_{i=1}^{d} \hat{f}(\{i\})x_i$. Consider learning this function using gradient flow on a diagonal neural network (where depth $L \ge 2$) while the $k$-th component is frozen at training (the canonical holdout setting with $\mathcal{U} = \{x \in \{\pm 1\}^d \mid x_k = -1\}$). For any $\epsilon > 0$, there exists an $\alpha_{max}$ (increasing with $L$) such that if all the model's parameters are initialized i.i.d. under the uniform distribution $U(-\alpha, \alpha)$ for any $0 < \alpha \le \min\{\alpha_{\max}, \frac{1}{2}\}$, then, with probability 1, the training loss converges to 0, and the coefficient of the learned function $f_{\mathrm{NN}}$ on the high-degree monomial $x_k$ is less than $\epsilon$, i.e., $\hat{f}_{\mathrm{NN}}(\{k\}) \le \epsilon$.*

*Proof Sketch.* We prove this theorem by analyzing the trajectory of gradient flow on the parameters. Primarily, we show the convergence of the model. Note that $\hat{f}_{\mathrm{NN}}(\{k\}) \leq \epsilon$ is equivalent to $x_k$ being ignored by the neural network, i.e., the frozen variable $x_k$ not contributing to the bias learned by the neural network. We pursue the proof in two steps. As the first step, we show there exists a time $T_\epsilon$ such that the bias is almost learned by the bias parameter and the rest of the parameters and the role of $x_k = 1$ are still small (note that this point is close to a saddle). For the second step, we show that the contribution of $x_k = 1$ to the bias will not change much throughout the training process.

*Remark* 3.12. Note that with the assumptions of Theorem 3.11, the generalization error of the model becomes[1]

$$GOTU(f, f_{\mathrm{diag}}, \mathcal{U} = \{x : x_k = -1\}) = 4\mathrm{Inf}_k(f) + O(\epsilon),$$

where $\mathrm{Inf}_k(f) = \hat{f}(\{k\})^2$ is the Boolean influence of the $k$-th bit (O'Donnell, 2014). This confirms the empirical observations of (Abbe et al., 2022a) on fully connected linear neural networks. Indeed, we expect our proof to generalize to fully connected linear neural networks. Assuming small enough initialization, one can show that the bias parameter of the last layer would learn the bias of the target function while the rest of the parameters do not move much, which is the first step of the proof. The second step, showing that the contribution to the bias remains almost the same after this point, requires more precise analysis since the network's learning of weights and biases are closely coupled.

## 4. Experiments

In this section, we present our experimental results on the min-degree bias of neural networks.[2] We have used four architectures for our experiments: a multi-layer perceptron (MLP) with 4 hidden layers, the random features model (Definition 3.5), Transformers (Vaswani et al., 2017), and 2-layer neural network with mean-field parametrization (Mei et al., 2018). By doing this, we consider a spectrum of models covering lazy regimes, active/feature learning regimes, and models of practical interest. For the Transformer, ±1 bits are first encoded using an encoding layer and then passed to the Transformer; while for the rest of the architectures, binary vectors are directly used as the input.

For each experiment, we generate all binary sequences in $\mathcal{U}^c = \{\pm 1\}^d \setminus \mathcal{U}$ for training.[3] We then train models under the $\ell_2$ loss. We employ Adam (Kingma & Ba, 2014) optimizer for the Transformer model and mini-batch SGD for the rest of the architectures. We also use moderate learning rates as learning rate can affect the results (refer to Appendix B.2). During training, we evaluate the coefficients of the function learned by the neural network using $\hat{f}_{\mathrm{NN}}(T) = \mathbb{E}_{x\sim_U\{\pm1\}^d}[\chi_T(x) f_{\mathrm{NN}}(x)]$ to understand which interpolating solution has been learned by the model. Moreover, each experiment is repeated 10 times and averaged results are reported. For more information on the setup of experiments, hyperparameter sensitivity analysis, and additional experiments refer to Appendix B.

Here, we consider the following 3 functions and unseen domains on input dimension 15. Dimension 15 is used as a large dimension where the training data can be generated explicitly but has otherwise no specific meaning (Appendix B provides other instances). The first function is an example of degree-2 where the unseen domain induces a degree-1 MD interpolator. The second example is the classic degree-2 parity or XOR function. The third example is such that the function is symmetric under cyclic permutations while its MD interpolator is not, in order to test whether certain models would favor symmetric interpolators. We consider other examples such as the majority function in Appendix B. Let:

1. $f_1(x) = x_0x_1 - 1.25x_1x_2 + 1.5x_2x_0$ and $\mathcal{U}_1 = \{x_0x_1x_2 = -1\}$. In this case, we have $x_0x_1 = x_2$, $x_1x_2 = x_0$, and $x_2x_0 = x_1$ at training, hence the MD interpolator is $\tilde{f}_1(x) = x_2 - 1.25x_0 + 1.5x_1$.
2. $f_2(x) = x_0x_1$ and $\mathcal{U}_2 = \{(x_0, x_1) = (-1, -1)\}$. Note that the MD interpolator is $\tilde{f}_2(x) = x_1 + x_0 - 1$ for the seen domain.
3. $f_3(x) = x_0x_1x_2 + x_1x_2x_3 + \cdots + x_{13}x_{14}x_0 + x_{14}x_0x_1$ and $\mathcal{U}_3 = \{(x_0, x_1, x_2) = (-1, -1, -1)\}$. In this case, the MD interpolator is given by $\tilde{f}_3(x) = (x_0x_1 + x_1x_2 + x_2x_0 - x_0 - x_1 - x_2 + 1) + x_1x_2x_3 + \cdots + x_{13}x_{14}x_0 + x_{14}x_0x_1$.

We generally obtain that the Transformer exhibits a strong MD bias. The solutions learned by the Transformer for $f_1$, $f_2$, $f_3$ are shown in Figure 1. It can be seen that these are very close to the MD interpolator in all cases. The other models, however, display a 'leaky' MD bias where higher degree monomials are still captured along with the lower degree ones. Particularly, Figure 2 shows a mean-field model and an MLP having such leaky MD biases. Note that the RF model in Figure 2 has a small leakage as well, simply caused by the ambient dimension being $d = 15$ and not diverging as in Theorem 3.8. In Appendix B.2, we discuss the effect that large learning rates may increase the leakage.

## 5. Further Implications

We now discuss some of the consequences of the min-degree bias. First, we explain why the MD bias makes some length

[1] The factor 4 is removed if we consider the half-quadratic loss and GOTU on the full space.

[2] Code: https://github.com/aryol/GOTU

[3] In practice, one can generate a large enough number of samples so that the function is learned well on the training distribution.

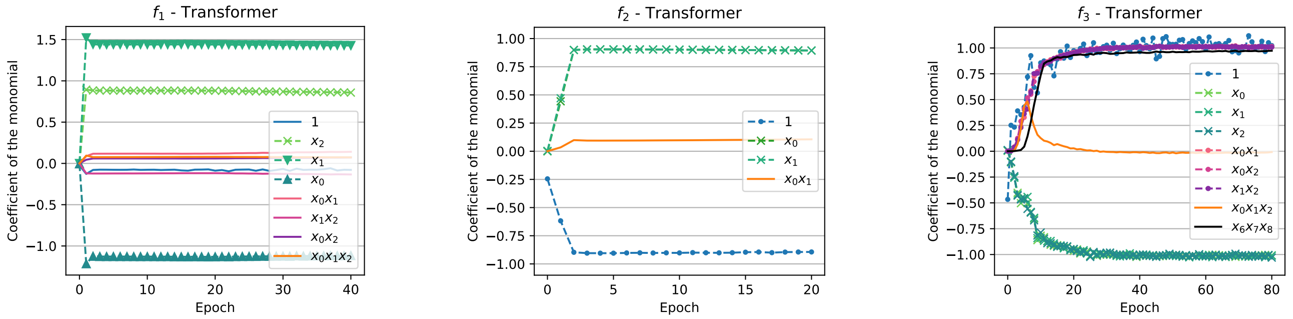

*Figure 1.* Target functions $f_1$, $f_2$, and $f_3$ learned by the Transformer (model details in Appendix B). Note that in all of the cases the Transformer model learns a solution very close to the min-degree interpolator. More precisely, the coefficients of $x_0x_1, x_1x_2, x_2x_0$ in the left plot ($f_1$), the coefficient of $x_0x_1$ in the middle plot ($f_2$), and the coefficient of $x_0x_1x_2$ in the right plot ($f_3$) are close to zero.

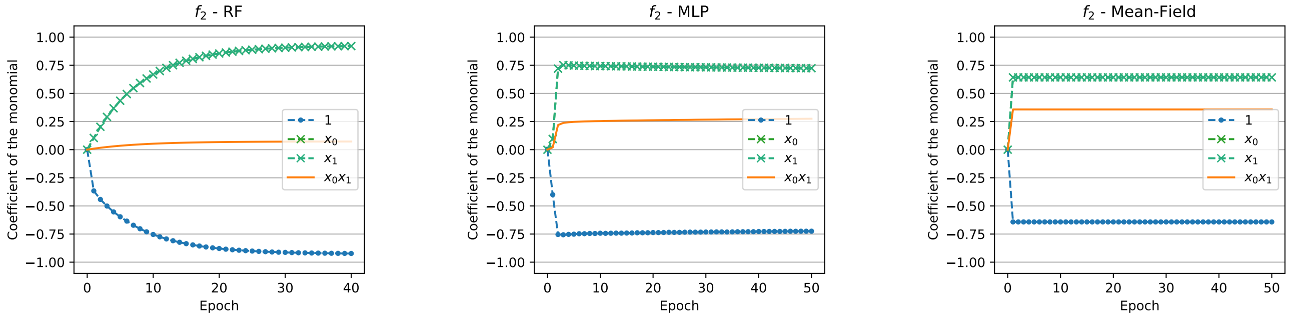

*Figure 2.* $f_2(x_0, \ldots, x_{14}) = x_0x_1$ learned by the RF, MLP, and mean-field models while samples satisfying $(x_0, x_1) = (-1, -1)$ are withheld during training. Consequently, $x_0x_1$ (solid orange line) is replaceable by $x_0 + x_1 - 1$ (dashed lines). The MLP and mean-field models learn a leaky min-degree interpolator with the $x_0x_1$ coefficient bounded away from 0. The RF model learns the min-degree interpolator with a small leakage since the ambient dimension is $d = 15$; this leakage disappears as $d$ increases as stated in Theorem 3.8.

generalization problems difficult. Second, we show how to turn the MD bias into a strategy for curriculum learning and enable an improved sample complexity.

### 5.1. Length generalization

Several recent works on the reasoning of neural networks evaluate whether neural networks are able to generalize when the length of the problem is increased, and it is often found that neural networks struggle with length generalization (Zhang et al., 2022; Anil et al., 2022). For example, consider learning the parity problem $\mathrm{parity}(x_1, \ldots, x_d) = x_1x_2\cdots x_d$ on $x_i = \pm 1$. Two variants of this task can be considered: (1) the number of bits, $d$, is increased during test, and (2) $d$ is the same during training and test; however, during training, only samples with a bounded number of $-1$'s are observed, i.e., the radius $r$ Hamming ball $B_r := \{x \in \{\pm 1\}^d \mid \#_{-1}(x) \leq r\}$ (note that $+1$ is the identity element in this setting). Anil et al. (2022) show that both of these variants capture the notion and difficulty of length generalization.[4] Here, we focus on the latter variant which falls under our GOTU setting.

**Theorem 5.1.** *Consider a Boolean function $f : \{\pm 1\}^d \to \mathbb{R}$. Then (i) there exists a unique function $f_r : \{\pm 1\}^d \to \mathbb{R}$ such that $\forall x \in B_r, f_r(x) = f(x)$ and $\deg(f_r) \leq r$; (ii) when $f$ is a parity function (monomial) of degree $k \leq d$, the $\ell_2$-test-loss of the MD interpolator is larger than $\binom{k-1}{r}^2$.*

We defer the proof to Appendix A. Now consider learning the parity function $x_1x_2\cdots x_d$ where training samples have $r$ or less $-1$ coordinates, i.e., training samples belong to $B_r$. Using the previous theorem, there is a degree $r$ alternative to $x_1x_2\cdots x_d$. Note that when such a low-degree alternative exists, assuming the min-degree bias, the model will learn this alternative instead of the full function of degree $d$. This explains why in this case neural networks cannot generalize when the length is increased. We conduct an experiment to evaluate this, where we learn the full parity function on

[4] We train our model directly on the parity function; whereas (Anil et al., 2022) uses large language models and fine-tunes parity tasks on them. In this sense, our approach is closer to (Zhang et al., 2022) which also trains models on their synthetic task.

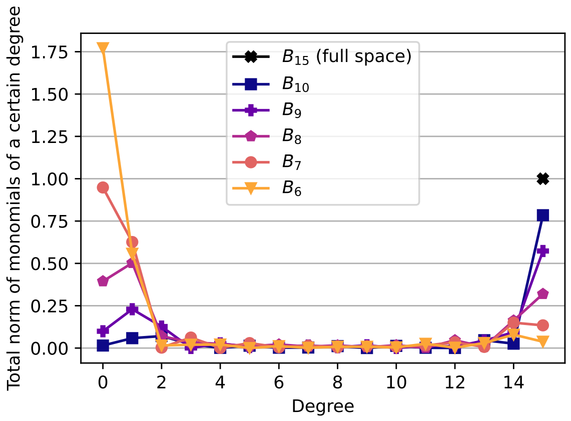

*Figure 3.* Learning full parity function in dimension $d = 15$ in the length generalization setting with inputs in $B_6, B_7, B_8, B_9, B_{10}$ and $B_{15}$ (full space) respectively, with an MLP (model details in Appendix B). X-axis: degree-profile component, Y-axis: degree-profile value, i.e., $\sum_{T:|T|=x} \hat{f}_{\mathrm{NN}}(T)^2$. As the length of training samples is decreased, the coefficient of the full parity gets smaller and the coefficients of low-degree monomials get larger.

15 bits using the MLP model trained on different lengths. Figure 3 shows that we learn more of lower degree terms and less of the full parity term as we train on shorter lengths.

### 5.2. Curriculum learning

The bias of neural networks towards min-degree solutions can also be utilized to boost the learning via a curriculum learning (Bengio et al., 2009) algorithm. We propose to train models by increasing the 'complexity' of training samples with respect to the input Hamming weight, i.e., $B_{r_1} \subseteq B_{r_2} \subseteq \ldots \subseteq B_{r_k}$ where $B_r$ is the Hamming ball of radius $r$. Training a model on samples included in $B_r$ with $r < d$ produces biased inputs compared to the uniform distribution. It has been shown that learning parities with GD on biased inputs is easier for various architectures (Malach et al., 2021; Daniely & Malach, 2020). In particular, the biasedness of the input distribution can be viewed as converting a monomial on non-centered inputs to a staircase on centered inputs as discussed in (Abbe et al., 2021). Moreover, (Abbe et al., 2022b) shows that the sample complexity for learning staircases is significantly reduced compared to that of monomials of matching degree. In particular, a layer-wise analysis shows that the hidden neurons in the first layer detect the support of a parity function under biased inputs, allowing for the fitting of the target function with the second layer if enough neuron diversity is available. One can thus attempt to bootstrap this approach and progressively climb the support (and degree) of the target function by training successively the network on increasing balls. We now develop this approach into a general curriculum algorithm.

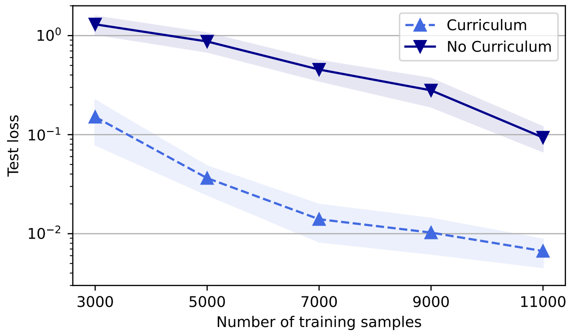

*Figure 4.* Generalization loss on the 16-parity function for different numbers of samples with and without the Degree-Curriculum Algorithm.

**Algorithm 1** Degree-Curriculum algorithm

```
Input: Training samples $S = \{(x_i, y_i)\}_{i=1}^m$; Curriculum
$B_{r_1} \subset B_{r_2} \subset \ldots \subset B_{r_k} = B_d$; Loss threshold $\epsilon$
for $i = 1$ to $k$ do
  $S_{r_i} \coloneqq \{(x, y) \in S | x \in B_{r_i}\}$ (samples in $B_{r_i}$)
  initialize train loss $= 1 + \epsilon$.
  while train loss $> \epsilon$ do
    train model with SGD on $S_{r_i}$
    update train loss
  end while
end for
```

Note that at the $i$-th step of Algorithm 1, all the training samples belong to $B_{r_i}$. Thus, for models obeying the MD bias on the unseen, the model learns the MD interpolator of degree at most $r_i$. Further, if the sampling set $S$ is such that $B(r_i) \cap S$ contains enough degree $r_i$ elements, the MD interpolator is of degree $r_i$ — see Theorem 5.1. If one then takes $r_i = r_{i-1} + 1$, the new MD interpolator has monomials at step $i - 1$ that are contained in those at step $i$, as in the learning of a merged staircase function (Abbe et al., 2022b) (and a lower leap function more generally if one takes a leap in the curriculum degrees). Thus, for a parity target, the Degree-Curriculum algorithm learns the support sets incrementally as for the implicit staircase function.

We evaluate the Degree-Curriculum algorithm on learning full parity function $x_0 x_1 \cdots x_{15}$ with an MLP. More precisely, for the same training set and hyperparameters, we once train the MLP with normal SGD and once with the proposed Degree-Curriculum algorithm. We choose curriculum $B_4, B_8, B_{12}, B_{16}$ and loss threshold $\epsilon = 0.001$. The results are depicted in Figure 4 (also see Figure 11 for the same experiment in dimension 30).

In Algorithm 1, it is assumed that the training set is given with the random access model. We can also consider a variant with the query access model, where at step $i$, training samples are queried directly from $B_{r_i}$ (or some distribution). In the former case, the probability of a sample belonging to $B_r$ is small for small values of $r$ (e.g., $r = o_d(d)$). We thus expect the Degree-Curriculum algorithm under the query access model to be more efficient in that regard. In a concurrent work (Cornacchia & Mossel, 2023), the benefit of using a query model with a biased sample distribution before a denser distribution to learn parities is also investigated. Particularly, an improvement in the number of GD iterations has been proved using 1-step gradient arguments.

Note that in the Boolean setting and for the parity functions, $+1$ is the identity element. Thus, the number of $-1$'s used in the Degree-Curriculum algorithm can also be viewed as the length of the inputs. Interestingly, some works in the natural processing domain have used the length of the sentences (possibly along with other properties) to design their curriculum strategy (Spitkovsky et al., 2010; Zaremba & Sutskever, 2014; Kocmi & Bojar, 2017; Platanios et al., 2019). Finally, we can naturally extend the Degree-Curriculum algorithm to non-Boolean settings using the same principle as above: *Build curriculum sets $\{\tilde{B}_i\}$ of 'increased complexity' in order to have a path of learned functions on support sets $\{\mathcal{S}^{(i)}\}$ that are as tightly nested as possible (e.g., staircases or low-leap functions (Abbe et al., 2022b)), with the target function at last*.

## 6. Related Literature

Given the deployment of machine learning models in the real world, out-of-distribution generalization is a critical aspect of machine learning that has been extensively studied both in theory (Ben-David et al., 2006; Mansour et al., 2009; Redko et al., 2020) and in practice (Gulrajani & Lopez-Paz, 2020; Miller et al., 2021; Wiles et al., 2022). Our work considers an extreme case of distribution shift in which part of the domain is entirely unseen during the training, and thus OOD generalization is only possible if the target function has special structures (e.g., being compositional or having in/equi-variances) and the model captures those structures. OOD generalization and the ability to extrapolate have also been used as proxies for measuring the reasoning capabilities of neural networks (Saxton et al., 2019; Zhang et al., 2021; Csordás et al., 2021; Zhang et al., 2022) as these models are prone to memorization of training samples (Carlini et al., 2019; Feldman & Zhang, 2020; Kandpal et al., 2022; Carlini et al., 2022; Zhang et al., 2021) or learning undesirable shortcuts (Zhang et al., 2022). A special case is length generalization (Zaremba & Sutskever, 2014; Lake & Baroni, 2018; Hupkes et al., 2020; Zhang et al., 2022; Anil et al., 2022), i.e., generalization to the input lengths beyond what is seen during the training. In this paper, we provided an explanation for the length generalization problem in the simple instance of parity functions (Anil et al., 2022).

It has been shown that training with gradient descent imposes particular implicit regularization on the solutions found by the models such as sparsity (Moroshko et al., 2020), norm minimization (Bartlett et al., 2021), and margin maximization (in linear classification setting) (Soudry et al., 2017). This implicit regularization (or implicit bias) of neural networks trained with gradient-based algorithms has been used to explain the generalization of (often overparametrized) models (Bartlett et al., 2021). These results depend on the optimizer (Gunasekar et al., 2018a) and model (Gunasekar et al., 2018b) and are usually proven for simple models such as linear models (Soudry et al., 2017; Yun et al., 2020; Jacot et al., 2021) including diagonal linear neural networks (Gunasekar et al., 2018b; Moroshko et al., 2020) as studied in this paper. Our result for the random feature model builds upon the implicit bias toward solutions with minimum norm (Bartlett et al., 2021). Related to us is also the spectral bias (Xu et al., 2019; Rahaman et al., 2019) stating that neural networks, when learning a function in continuous settings, capture the lower frequency components faster (note that degree in Boolean functions plays a similar role to the frequency). In this paper, we develop a related insight in the Boolean setting by introducing the notion of degree-profile and showing the min-degree implicit bias for several models theoretically and empirically.

## 7. Conclusions and Future Directions

In this paper, we put forward the concept of generalization on the unseen (GOTU) and considered the learning of Boolean functions. We showed that various network architectures have a bias toward the min-degree interpolator, with theoretical results for the RF and DLNN, and experimental results for Transformers. We also found empirically that for large learning rates or for other models such as mean-field networks, a leaky version of the MD bias takes place.

We showed that the MD bias can be utilized in a curriculum learning algorithm where the training takes place on sets of increasing complexity. We also demonstrated that the MD bias can impede the learning of symmetric solutions and can make length generalization difficult.

The min-degree bias is a form of Occam's razor chosen by GD-trained neural nets, where the 'simplicity' is measured by the 'degree-profile'. However, this might not be a desirable form of razor for various reasoning tasks. We believe that other forms promoting symmetries, compositionality, or more generally minimum description length (MDL) may often be more suitable. The next natural steps are thus to correct this min-degree bias. We propose here some directions to pursue: (1) architecture design promoting symmetries or compositionality, (2) hyperparameter tuning (e.g., learning rates, scale), (3) data augmentation and multitasking, (4) MDL-like regularization at training.

## References

Abbe, E., Boix-Adsera, E., Brennan, M., Bresler, G., and Nagaraj, D. The staircase property: How hierarchical structure can guide deep learning, NeurIPS, 2021.

Abbe, E., Bengio, S., Cornacchia, E., Kleinberg, J., Lotfi, A., Raghu, M., and Zhang, C. Learning to reason with neural networks: Generalization, unseen data and boolean measures. *arXiv preprint arXiv:2205.13647*, 2022a.

Abbe, E., Boix-Adsera, E., and Misiakiewicz, T. The merged-staircase property: a necessary and nearly sufficient condition for SGD learning of sparse functions on two-layer neural networks, COLT, 2022b.

Abbe, E., Cornacchia, E., Hazla, J., and Marquis, C. An initial alignment between neural network and target is needed for gradient descent to learn, 2022c. URL https://arxiv.org/abs/2202.12846.

Alabdulmohsin, I., Neyshabur, B., and Zhai, X. Revisiting neural scaling laws in language and vision. *arXiv preprint arXiv:2209.06640*, 2022.

Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. Exploring length generalization in large language models. *arXiv preprint arXiv:2207.04901*, 2022.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization, 2019. URL https://arxiv.org/abs/1905.13655.

Bakhtin, A., van der Maaten, L., Johnson, J., Gustafson, L., and Girshick, R. Phyre: A new benchmark for physical reasoning. *Advances in Neural Information Processing Systems*, 32, 2019.

Bartlett, P. L., Montanari, A., and Rakhlin, A. Deep learning: a statistical viewpoint. *Acta numerica*, 30:87–201, 2021.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, volume 267, 2019.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

Chizat, L. and Bach, F. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, pp. 1305–1338. PMLR, 2020.

Cornacchia, E. and Mossel, E. A mathematical model for curriculum learning. *arXiv preprint arXiv:2301.13833*, 2023.

Cox, D., Little, J., and OShea, D. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.

Csordás, R., Irie, K., and Schmidhuber, J. The devil is in the detail: Simple tricks improve systematic generalization of transformers. *arXiv preprint arXiv:2108.12284*, 2021.

Daniely, A. and Malach, E. Learning parities with neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20356–20365. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/eaae5e04a259d09af85c108fe4d7dd0c-Paper.pdf.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dummit, D. S. and Foote, R. M. *Abstract algebra*, volume 3. Wiley Hoboken, 2004.

Feldman, V. and Zhang, C. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. Limitations of lazy training of two-layers neural network. *Advances in Neural Information Processing Systems*, 32, 2019.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

Gunasekar, S., Woodworth, B., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization, 2017. URL https://arxiv.org/abs/1705.09280.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry, 2018a. URL https://arxiv.org/abs/1802.08246.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks, 2018b. URL https://arxiv.org/abs/1806.00468.

Hartford, J., Graham, D., Leyton-Brown, K., and Ravanbakhsh, S. Deep models of interactions across sets. In *International Conference on Machine Learning*, pp. 1909–1918. PMLR, 2018.

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67: 757–795, 2020.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Jacot, A., Ged, F., Şimşek, B., Hongler, C., and Gabriel, F. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint arXiv:2106.15933*, 2021.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Kandpal, N., Wallace, E., and Raffel, C. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pp. 10697–10707. PMLR, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kocmi, T. and Bojar, O. Curriculum learning and minibatch bucketing in neural machine translation. *arXiv preprint arXiv:1707.09533*, 2017.

Lake, B. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pp. 2873–2882. PMLR, 2018.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

Mahdavi, S., Swersky, K., Kipf, T., Hashemi, M., Thrampoulidis, C., and Liao, R. Towards better out-of-distribution generalization of neural algorithmic reasoning tasks. *ArXiv*, 2211.00692, 2022. URL https://arxiv.org/abs/2211.00692.

Malach, E., Kamath, P., Abbe, E., and Srebro, N. Quantifying the benefit of using differentiable learning over tangent kernels. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7379–7389. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/malach21a.html.

Mansour, Y., Mohri, M., and Rostamizadeh, A. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009.

Mei, S. and Montanari, A. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

Mei, S., Montanari, A., and Nguyen, P.-M. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33): E7665–E7671, 2018.

Miller, J. P., Taori, R., Raghunathan, A., Sagawa, S., Koh, P. W., Shankar, V., Liang, P., Carmon, Y., and Schmidt, L. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *International Conference on Machine Learning*, pp. 7721–7735. PMLR, 2021.

Möller, H. M. and Buchberger, B. The construction of multivariate polynomials with preassigned zeros. In *European Computer Algebra Conference*, pp. 24–31. Springer, 1982.

Moroshko, E., Gunasekar, S., Woodworth, B., Lee, J. D., Srebro, N., and Soudry, D. Implicit bias in deep linear classification: Initialization scale vs training accuracy, 2020. URL https://arxiv.org/abs/2007.06738.

O'Donnell, R. *Analysis of Boolean Functions*. Cambridge University Press, 2014. doi: 10.1017/CBO9781139814782.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Platanios, E. A., Stretcu, O., Neubig, G., Poczos, B., and Mitchell, T. M. Competence-based curriculum learning for neural machine translation. *arXiv preprint arXiv:1903.09848*, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Equivariance through parameter-sharing. In *International conference on machine learning*, pp. 2892–2901. PMLR, 2017.

Razin, N. and Cohen, N. Implicit regularization in deep learning may not be explainable by norms, 2020. URL https://arxiv.org/abs/2005.06398.

Redko, I., Morvant, E., Habrard, A., Sebban, M., and Bennani, Y. A survey on domain adaptation theory: learning bounds and theoretical guarantees. *arXiv preprint arXiv:2004.11829*, 2020.

Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data, 2017. URL https://arxiv.org/abs/1710.10345.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. From baby steps to leapfrog: How “less is more” in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 751–759, 2010.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veličković, P., Badia, A. P., Budden, D., Pascanu, R., Banino, A., Dashevskiy, M., Hadsell, R., and Blundell, C. The clrs algorithmic reasoning benchmark. *arXiv preprint arXiv:2205.15659*, 2022.

Wiles, O., Gowal, S., Stimberg, F., Rebuffi, S.-A., Ktena, I., Dvijotham, K. D., and Cemgil, A. T. A fine-grained analysis on distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Dl4LetuLdyK.

Xu, Z.-Q. J., Zhang, Y., Luo, T., Xiao, Y., and Ma, Z. Frequency principle: Fourier analysis sheds light on deep neural networks, 2019.

Yun, C., Krishnan, S., and Mobahi, H. A unifying view on implicit bias in training linear neural networks. *arXiv preprint arXiv:2010.02501*, 2020.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Zaremba, W. and Sutskever, I. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

Zhang, C., Raghu, M., Kleinberg, J. M., and Bengio, S. Pointer value retrieval: A new benchmark for understanding the limits of neural network generalization. *ArXiv*, abs/2107.12580, 2021.

Zhang, Y., Backurs, A., Bubeck, S., Eldan, R., Gunasekar, S., and Wagner, T. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022.

Zhou, A., Knowles, T., and Finn, C. Meta-learning symmetries by reparameterization. *arXiv preprint arXiv:2007.02933*, 2020.

# A. Proofs

## A.1. Proofs for the random features model

We start by proving a lemma showing that for strongly expressive activation functions each random feature is low-degree in the sense that the high-degree monomials have small coefficients in the Fourier-Walsh expansion of the random features.

**Lemma A.1** (Random features are low-degree)**.** *Consider random features generated by an activation function that is strongly expressive up to $P = O_d(1)$, i.e., $\phi_{w,b}(x) = \sigma(\langle w, x\rangle + b)$ where $w_i, b \sim \mathcal{N}(0, \frac{1}{d})$ are the random weights and bias. We have the following additional properties:*

*A3.* $\forall T \subseteq [d]$ $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2]$ *exists and* $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2] = \Theta(d^{-|T|})$ *for* $|T| \leq P$*;*

*A4.* $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)\hat{\phi}_{w,b}(T')] = 0$ *for* $T \neq T'$*; and*

*A5.* $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2] = 0 \iff \hat{\phi}_{w,b}(T) = 0 \;\forall\; w, b,$

*where* $\hat{\phi}_{w,b}(T)$ *is the coefficient of monomial* $T$ *in random feature* $\phi_{w,b}(x)$*.*

*Proof.* For property (A3), consider all subsets of $[d] = \{1, \ldots, d\}$ with size $k \leq P$: $T_1, T_2, \ldots, T_{\binom{d}{k}}$. Note that due to the symmetry, we have $\mathbb{E}_{w,b}[\hat{\phi}(T_1)^2] = \cdots = \mathbb{E}_{w,b}[\hat{\phi}(T_{\binom{d}{k}})^2]$. Moreover, we have

$$\binom{n}{k}\mathbb{E}_{w,b}[\hat{\phi}(T_i)^2] = \sum_{i=1}^{\binom{d}{k}} \mathbb{E}_{w,b}[\hat{\phi}(T_i)^2] = \mathbb{E}_{w,b}[\sum_{i=1}^{\binom{d}{k}} \hat{\phi}(T_i)^2] \leq \mathbb{E}_{w,b}[\sum_{T\subseteq[d]} \hat{\phi}(T)^2] = \mathbb{E}_{w,b}[\mathbb{E}_x[\phi(x)^2]] \tag{5}$$

$$= \mathbb{E}_x[\mathbb{E}_{w,b}[\sigma(\langle w, x\rangle + b)^2]] = \mathbb{E}_{g\sim\mathcal{N}(0,\frac{d+1}{d})}[\sigma(g)^2], \tag{6}$$

where in Equation 5 we used Parseval's identity. By assumption (A1) on the function we know that $\mathbb{E}_{g\sim\mathcal{N}(0,2)}[\sigma(g)^4]$ is finite. Thus, $\mathbb{E}_{g\sim\mathcal{N}(0,2)}[\sigma(g)^2]^2$ is also finite and consequently $\mathbb{E}_{g\sim\mathcal{N}(0,\frac{d+1}{d})}[\sigma(g)^2]^2$ can be upper bounded independently of $d$, which proves the existence part. Furthermore, $\mathbb{E}_{w,b}[\hat{\phi}(T_i)^2] = O_d(\binom{d}{k}^{-1}) = O_d(d^{-k})$, where we used $k \leq P = O_d(1)$. Now by property (A2), we can conclude that $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2] = \Theta(d^{-|T|})$ for $|T| \leq P$.

For property (A4), assuming $T \neq T'$ take $i \in T\Delta T'$. Without loss of generality suppose $i \in T, i \notin T'$. For weight vector $w$, we flip the sign of the $i$-th coordinate and denote the resulting vector by $w_{-i}$. Now note that $\mathbb{E}_x[\sigma(\langle w, x\rangle + b)\chi_T(x)] = -\mathbb{E}_x[\sigma(\langle w_{-i}, x\rangle + b)\chi_T(x)]$ and $\mathbb{E}_x[\sigma(\langle w, x\rangle + b)\chi_{T'}(x)] = \mathbb{E}_x[\sigma(\langle w_{-i}, x\rangle + b)\chi_{T'}(x)]$. Hence, $\hat{\phi}_{w,b}(T)\hat{\phi}_{w,b}(T') = -\hat{\phi}_{w_{-i},b}(T)\hat{\phi}_{w_{-i},b}(T')$ and $\mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)\hat{\phi}_{w,b}(T')] = 0$.

Note that the last property is a consequence of the continuity assumption on the activation function. □

Now we can prove Theorem 3.8.

***Proof of Theorem 3.8.*** First, recall the set of all interpolating solutions on the training set $\mathcal{U}^c$ as

$$\mathcal{F}_{\text{int}}(f_{\text{target}}, \mathcal{U}) = \{f : \{\pm 1\}^d \to \mathbb{R} \mid f(x) = f_{\text{target}}(x) \;\forall x \in \mathcal{U}^c\}.$$

Note that a solution given by $a_1, \ldots, a_N$ is interpolating if and only if $\frac{1}{\sqrt{N}}\sum_{i=1}^N a_i\phi_i(x) \in \mathcal{F}_{\text{int}}$.

Moreover, we study the features and solutions in the Fourier-Walsh basis. First, we index all possible monomials, i.e., $\{T_1, T_2, \ldots, T_{2^d}\} = 2^{\{1,2,3,\ldots,d\}}$ and $\chi_{T_i}(x) = \prod_{j\in T_i} x_j$. Further, we define the coefficient of monomial $T_j$ in the $i$-th feature as $\hat{\phi}_i(T_j) := \mathbb{E}_x[\phi_i(x)\chi_{T_j}(x)]$ and $F \in \mathbb{R}^{2^d\times N}$ as the matrix of features in the Fourier expansion, i.e., $F_{i,j} = \frac{1}{\sqrt{N}}\hat{\phi}_j(T_i)$. Using this notation, $a$ corresponds to an interpolating solution if and only if

$$\exists g \in \mathcal{F}_{\text{int}} \;\; Fa = \hat{g}, \tag{7}$$

where $\hat{g}$ represents function $g$ in the Fourier-Walsh basis. Furthermore, note that

$$(FF^T)_{i,j} = \sum_{k=1}^{N} (\frac{1}{\sqrt{N}}\hat{\phi}_k(T_i))(\frac{1}{\sqrt{N}}\hat{\phi}_k(T_j)) = \frac{1}{N}\sum_{k=1}^{N}\hat{\phi}_k(T_i)\hat{\phi}_k(T_j). \tag{8}$$

Note that weights and biases of the features are sampled i.i.d., therefore, as $N \to \infty$, $(FF^T)_{i,j}$ converges to $\mathcal{N}(\mathbb{E}_w[\hat{\phi}_w(T_i)\hat{\phi}_w(T_j)], N^{-1}\mathrm{Var}_w[\hat{\phi}_w(T_i)\hat{\phi}_w(T_j)])$ in distribution, due to the central limit theorem (CLT). Note that for the CLT to hold, the variances have to be finite which holds because of property (A1). More specifically, $\mathbb{E}_{g\sim\mathcal{N}(0,2)}[\sigma(g)^4]$ is finite, and hence, $\mathbb{E}_{g\sim\mathcal{N}(0,\frac{d+1}{d})}[\sigma(g)^4]$ is finite. Moreover,

$$\infty > \mathbb{E}_{g\sim\mathcal{N}(0,\frac{d+1}{d})}[\sigma(g)^4] = \mathbb{E}_{w,b}[\mathbb{E}_x[\sigma(\langle w,x\rangle + b)^4]] \geq \mathbb{E}_{w,b}[\mathbb{E}_x[\sigma(\langle w,x\rangle + b)^2]^2] \tag{9}$$

$$= \mathbb{E}_{w,b}[(\sum_{T\subseteq[d]}\hat{\phi}_{w,b}(T)^2)^2] \geq \mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T_i)^2\hat{\phi}_{w,b}(T_j)^2] \ \ \forall i,j, \tag{10}$$

where we used Parseval's identity from Equation (9) to Equation (10). We define $\Phi \in \mathbb{R}^{2^d\times 2^d}$ as a shorthand notation as

$$\Phi_{i,j} = \mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T_i)\hat{\phi}_{w,b}(T_j)] = \begin{cases} 0 & i \neq j \\ \mathbb{E}[\hat{\phi}_{w,b}(T_i)^2] & i = j \end{cases}, \tag{11}$$

where we have used properties (A3) and (A4).

### A.1.1. Existence of interpolating solutions

Now, we show that an interpolating solution exists with high probability. Particularly, take any interpolator $g$ that only depends on the latent variables $x_{i_1}, \ldots, x_{i_P}$ and we show that $\hat{g}$ is in the image of $F$ w.h.p. and hence being an interpolating solution given Equation (7). Consider monomials such as $T$ for which $\forall w, b \ \hat{\phi}_{w,b}(T) = 0$. Due to properties (A2) and (A5), we know that such $T$'s satisfy $\deg(T) > P$, hence their corresponding rows are both zero in $F$ and in $\hat{g}$. We remove these rows from $F$ and $\hat{g}$ and call the new ones $\tilde{F}$ and $\tilde{g}$. We also remove corresponding rows and columns from $\Phi$ and denote the new matrix by $\tilde{\Phi}$.

Note $Fa = \hat{g} \iff \tilde{F}a = \tilde{g}$, therefore to prove that $\hat{g} \in \mathrm{Image}(F)$ its enough to show that $\tilde{F}$ is full row-rank, or equivalently, $\tilde{F}\tilde{F}^T$ is full rank. Note that $\tilde{F}\tilde{F}^T$ converges to $\tilde{\Phi}$ almost surely. Note that $\tilde{\Phi}$ is a diagonal matrix such that all elements on the diagonal are positive as all zero-entries of the diagonal are already removed by property (A5). Therefore $\tilde{\Phi}$ is full rank and $\tilde{F}\tilde{F}^T$ becomes full rank almost surely as $N \to \infty$. This concludes the proof of the existence of interpolators.

### A.1.2. Learning the min degree-profile interpolating solution

Now, we investigate the interpolating solution found by the model. Note that we are interested in the interpolating solution with the minimum norm $\|a\|_2$ (which is the solution found by GD starting from $a = 0$). Consider an interpolating solution $g \in \mathcal{F}_{\text{int}}$. The interpolator $g$ is found by the model if and only if $Fa = \hat{g}$, where $\hat{g}$ is the Fourier expansion of $g$ written in the vector form. Moreover, note that the $a$ satisfying $Fa = \hat{g}$ with the minimum norm $\|a\|_2$ is $a_g^* = F^\dagger\hat{g}$, where $F^\dagger$ is the Moore-Penrose pseudo-inverse. Therefore, we have

$$\|a_{\mathrm{RF}}\|_2^2 = \min_{g\in\mathcal{F}_{\text{int}},\hat{g}\in\mathrm{Im}(F)} \|F^\dagger\hat{g}\|_2^2 \implies g_{\mathrm{RF}} = \arg\min_{g\in\mathcal{F}_{\text{int}},\hat{g}\in\mathrm{Im}(F)} \|F^\dagger\hat{g}\|_2^2. \tag{12}$$

Now note that we have

$$\|F^\dagger\hat{g}\|_2^2 = \|F^T(FF^T)^\dagger\hat{g}\|_2^2 = \hat{g}^T(FF^T)^\dagger FF^T(FF^T)^\dagger\hat{g} \tag{13}$$

We know that $FF^T$ almost surely converges to $\Phi$, which is a diagonal matrix. Moreover, by property (A5), we know that the zero elements on the diagonal of $\Phi$ correspond to zero rows of $F$, and hence zero entries of $g$ since $g \in \mathrm{Im}(F)$. Thus, we can say that $(FF^T)^\dagger$ and $\|F^\dagger\hat{g}\|^2$ converge to $\Phi^\dagger$ and $g^T\Phi^\dagger g$ as $N \to \infty$ w.h.p. Furthermore, since $g \in \mathrm{Im}(F)$, zero entries on diagonal $\Phi$ (or $\Phi^\dagger$) correspond to zero entries of $g$, thus, we also have

$$g_{\mathrm{RF}} = \arg\min_{g\in\mathcal{F}_{\text{int}},\hat{g}\in\mathrm{Im}(F)} \|F^\dagger\hat{g}\|_2^2 \xrightarrow{N\to\infty(a.s.)} \arg\min_{g\in\mathcal{F}_{\text{int}},\hat{g}\in\mathrm{Im}(F)} g^T\Phi^\dagger g. \tag{14}$$

Also note that

$$g^T \Phi^\dagger g = \sum_{T \subseteq [d]: \mathbb{E}_{w,b}[\hat{\phi}(T)^2] \neq 0} \hat{g}(T)^2 \mathbb{E}_{w,b}[\hat{\phi}(T)^2]^{-1}. \tag{15}$$

We now focus on interpolators minimizing the quantity introduced in Equation (15). First, note that these interpolators do not have any monomials having a variable other than latent variables $\{x_{i_1}, \ldots, x_{i_p}\}$, i.e., all of the learned monomials would be in $2^{\{x_{i_1}, \ldots, x_{i_p}\}}$. To see this, consider an interpolating solution $g$ containing such monomials, $T_1, \ldots, T_m \not\subseteq I_P = \{i_1, \ldots, i_P\}$. For simplicity, we use the notation $x = (x_{I_P}, x_{[d] \setminus I_P})$ to differentiate between latent variables and the rest of the bits. Now define

$$g_I((x_{I_P}, x_{[d] \setminus I_P})) \coloneqq 2^{-(d-P)} \sum_{x_{[d] \setminus I_P} \in \{\pm 1\}^{d-P}} g(x). \tag{16}$$

Note that $g_I((x_{I_P}, x_{[d] \setminus I_P}))$ is independent of $x_{[d] \setminus I_P}$. Therefore $g_I(x) = g(x)$ for all the training samples. Moreover, note that

$$\hat{g}_I(T) = \begin{cases} \hat{g}(T) & T \subseteq I_P \\ 0 & o.w. \end{cases}, \tag{17}$$

which shows that $g_I \Phi^\dagger g_I < g \Phi^\dagger g$ unless $g = g_I$. Note that if $\mathbb{E}_{w,b}[\hat{\phi}(T)^2] = 0$ for some $T$, then $\hat{g}(T) = 0$, since we are considering the solution learned by the RF model and $\hat{g} \in \mathrm{Im}(F)$. In sum, the function learned by the RF model converges to an interpolator that only contains the latent coordinates, as $N \to \infty$ w.h.p. Note that $\mathbb{E}[\hat{\phi}_{w,b}(T)^2]$ is the same for all $T$ of the same size due to symmetry, we denote this shared quantity by $\hat{\phi}_{|T|,d}$. Now, we revisit Equation (15), for the functions defined on latent coordinates $I_P$, we have

$$g^T \Phi^\dagger g = \sum_{T \subseteq [d]: \mathbb{E}_{w,b}[\hat{\phi}(T)^2] \neq 0} \hat{g}(T)^2 \mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2]^{-1} \tag{18}$$

$$= \sum_{T \subseteq I_P} \hat{g}(T)^2 \mathbb{E}_{w,b}[\hat{\phi}_{w,b}(T)^2]^{-1} = \sum_{i=0}^{P} ( \sum_{T \subseteq I_P : |T| = i} \hat{g}(T)^2) \hat{\phi}_{|T|,d}^{-1}. \tag{19}$$

Note that since $\sigma$ is strongly expressive up to $P$, we have $\hat{\phi}_{k,d}^{-1} = \Theta(d^k)$. Putting this along Equation (19) shows that the solution of the RF model converges to $\mathrm{MinDegInterp} + \epsilon_d$ almost surely as $N \to \infty$, where $\epsilon_d$ is a vanishing function (w.r.t. $d$) on the latent coordinates, which concludes the proof. □

### A.1.3. RF model with ReLU activation

In this part, we study the random features model equipped with the ReLU activation function. Here, we mostly rely on the results of (Abbe et al., 2022c). First, following proposition B.1 of (Abbe et al., 2022c), we note that for every odd $k \geq 3$, the coefficient of $k$-th Hermite polynomial in the Hermite expansion of $\mathrm{ReLU}$ is zero. On the other hand, this coefficient is non-zero for $k = 1$ and any even $k$. Consequently, following Lemma A.2 of (Abbe et al., 2022c), for monomials $\chi_T$ and $|T| \leq P = O_d(1)$ we have

$$\mathbb{E}_{w,b}[\mathbb{E}_x[\mathrm{ReLU}(\langle w, x \rangle + b) \chi_T(x)]^2] = \mathbb{E}_{w,b}[\hat{\phi}_{w,b,\mathrm{ReLU}}(T)] = \begin{cases} \Omega(d^{-|T|}) & |T| \textit{ even or } |T| = 1 \\ \Omega(d^{-(|T|+1)}) & o.w. \end{cases}, \tag{20}$$

where $\hat{\phi}_{w,b,\mathrm{ReLU}}(T)$ is the coefficient of monomial $T$ in random feature created by the weights and bias $w, b$ and the ReLU activation. Informally, Equation (20) indicates that odd monomials with degrees larger than one are not strongly expressed in the random features when ReLU is used as the activation function. Nonetheless, note that as in Lemma A.1, we can still deduce that $\mathbb{E}_{w,b}[\hat{\phi}_{w,b,\mathrm{ReLU}}(T)] = O(d^{-|T|})$ for $|T| \leq P = O_d(1)$. This upper bound along with the lower bounds obtained in Equation (20) and the minimization problem of Equation (19) indicate that the random features model with ReLU activation would replace degree 2 or $2k + 1$ monomials with lower degree monomials if possible. However, it might not replace degree $2k + 2$ monomials with degree $2k + 1$ monomials for $k \geq 1$. We further illustrate this with an experiment.

We consider learning $f_3(x_0, \ldots, x_{14}) = x_0 x_1 x_2 + x_0 x_3 x_4 x_5$ under the unseen domain $\mathcal{U} = \{x \in \{\pm 1\}^{14} | x_0 = -1\}$. Note that in this case, the min-degree interpolator is $x_1 x_2 + x_3 x_4 x_5$. However, for the ReLU activation, we know that $x_3 x_4 x_5$ would not necessarily be preferred to $x_0 x_3 x_4 x_5$ since the $\deg(x_3 x_4 x_5) = 3$ is odd. In Figure 5, we compare the solution learned by the RF model with ReLU and polynomial activation (here $(1+x)^6$). It can be seen that the polynomial activation learns the MD interpolator, whereas the RF with the ReLU activation function only learns the lower-degree monomial for the odd monomial and not for the even one.

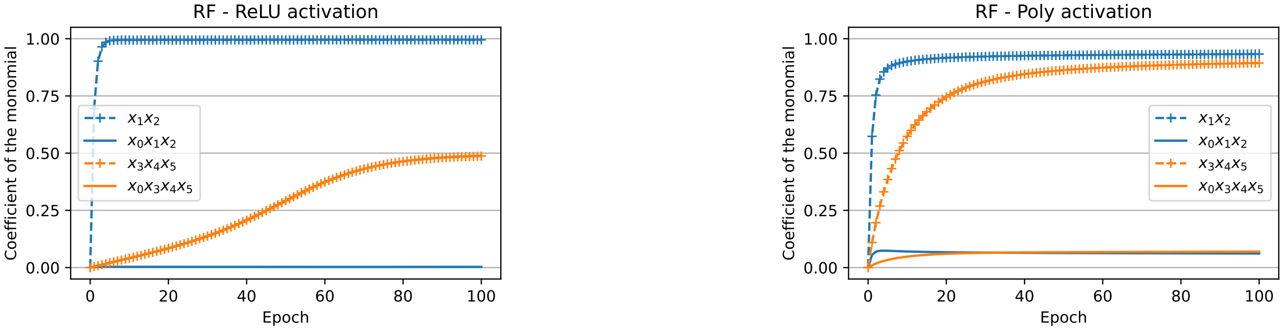

*Figure 5.* Target function $f_3(x_0,\dots,x_{14}) = x_0x_1x_2 + x_0x_3x_4x_5$ being learned by random features models under $\mathcal{U} = \{x_0 = -1\}$. The RF model with strongly expressive activation (here $(1+x)^6$) learns the min-degree interpolator (right), while the min-degree bias of the RF model with ReLU activation depends on the degree of monomials being even or odd (left). More precisely, the RF model does not prefer degree $2k+1$ monomial to degree $2k+2$ monomial for $k \geq 1$. Note that for the RF with ReLU activation (left), the coefficients of $x_3x_4x_5$ and $x_0x_3x_4x_5$ are equal and hence overlap.

### A.2. Proof for diagonal linear neural networks Theorem 3.11

Here, we present the proof of Theorem 3.11.

*Proof.* We denote parameters at time $t$ by $\theta(t)$. Also, we consider the training under half $\ell_2$ loss, to simply remove the 2 factor from gradients. Consider the (half) $\ell_2$ loss function for a training sample $x$, we have

$$L(\theta(t), x, f) = \frac{1}{2}\left(f_{\mathrm{NN}}(x) - f(x)\right)^2 \tag{21}$$

$$= \frac{1}{2}\left(\left(b - \hat{f}(\emptyset)\right) + \sum_{i=1}^{d}\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)x_i\right)^2. \tag{22}$$

Moreover, we know every component of the training sample is sampled from $\mathrm{Rad}(\frac{1}{2})$, except the frozen bit which is set to $x_k = 1$. We denote this uniform distribution by $U_{-k}^{d-1}$. Given this, the expected loss of the training set can be calculated as follows

$$\begin{aligned}
\mathbb{E}_{U_{-k}^{d-1}}[L(\theta(t), x, f)] &= \frac{1}{2}\mathbb{E}_{U_{-k}^{d-1}}\left[\left(\left(b - \hat{f}(\emptyset)\right) + \sum_{i=1}^{d}\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)x_i\right)^2\right] \\
&= \frac{1}{2}\mathbb{E}_{U_{-k}^{d-1}}\left[\left(\left((b + \prod_{l=1}^{L} w_k^{(l)}) - (\hat{f}(\emptyset) + \hat{f}(\{k\}))\right) + \sum_{i \neq k}\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)x_i\right)^2\right] \\
&= \frac{1}{2}\left((b + \prod_{l=1}^{L} w_k^{(l)}) - (\hat{f}(\emptyset) + \hat{f}(\{k\}))\right)^2 + \frac{1}{2}\sum_{i \neq k}^{d}\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)^2,
\end{aligned} \tag{23}$$

where we have used Parseval's theorem (O'Donnell, 2014) to get the last equation. For simplicity, we define $B := \hat{f}(\emptyset) + \hat{f}(\{k\})$ and $B_{\mathrm{NN}} := b + \prod_{l=1}^{L} w_k^{(l)}$ as the total bias of the target function and the neural network respectively. We know the gradient flow (GF) of the parameters of the neural network is given by

$$\dot{\theta} = -\nabla_\theta \mathbb{E}_{U_{-k}^{d-1}}[L(\theta(t), x, f)]. \tag{24}$$

Therefore, using (23), we can derive the gradient flow for each of the parameters as below

$$\dot{b} = -\nabla_b \mathbb{E}_{U^{d-1}_{-k}}[L(\theta(t), x, f)] = -(b + \prod_{l=1}^{L} w_k^{(l)}) + (\hat{f}(\emptyset) + \hat{f}(\{k\})) = -(B_{\text{NN}} - B), \tag{25}$$

$$\begin{aligned} \dot{w}_k^{(l)} &= -\nabla_{w_k^{(l)}} \mathbb{E}_{U^{d-1}_{-k}}[L(\theta(t), x, f)] = -((b + \prod_{j=1}^{L} w_k^{(j)}) + (\hat{f}(\emptyset) + \hat{f}(\{k\}))) \prod_{j \neq l}^{L} w_k^{(j)} \\ &= -\prod_{j \neq l}^{L} w_k^{(j)} (B_{\text{NN}} - B), \end{aligned} \tag{26}$$

$$\forall i \neq k,\ \ \dot{w}_i^{(l)} = -\nabla_{w_i^{(l)}} \mathbb{E}_{U^{d-1}_{-k}}[L(\theta(t), x, f)] = -\left(\prod_{j=1}^{L} w_i^{(j)} - \hat{f}(\{i\})\right) \prod_{j \neq l}^{L} w_i^{(j)}. \tag{27}$$

Using the above, we can derive the balancedness property of the neural network, i.e.,

$$\frac{d}{dt}(w_k^{(l)})^2 = 2 w_k^{(l)} \dot{w}_k^{(l)} = -2 \prod_{j=1}^{L} w_k^{(j)} (B_{\text{NN}} - B) = 2 w_k^{(l')} \dot{w}_k^{(l')} = \frac{d}{dt}(w_k^{(l')})^2, \tag{28}$$

$$\forall i \neq k,\ \ \frac{d}{dt}(w_i^{(l)})^2 = 2 w_i^{(l)} \dot{w}_i^{(l)} = -2 \left(\prod_{j=1}^{L} w_i^{(j)} - \hat{f}(\{i\})\right) \prod_{j=1}^{L} w_i^{(j)} = 2 w_i^{(l')} \dot{w}_i^{(l')} = \frac{d}{dt}(w_i^{(l')})^2. \tag{29}$$

Therefore, $\forall i\ \ (w_i^{(l)})^2 - (w_i^{(l')})^2$ is constant during training. Using this property, we can show that most of the model's parameters are always bounded away from 0 during training. To see this, fix an index $i \in [d]$. Let $j_i^* = \text{argmin}_{j \in [L]} |w_i^{(j)}(0)|$. Furthermore, define

$$c_i := \min_{j \neq j_i^* \in [L]} (w_i^{(j)}(0))^2 - (w_i^{(j_i^*)}(0))^2 \geq 0. \tag{30}$$

Since the model parameters are initialized randomly using the uniform distribution, we can say that $c_i > 0$ with probability 1. Now, due to the balancedness property, we know that

$$\forall j \neq j_i^*,\ \ (w_i^{(j)}(t))^2 - (w_i^{(j_i^*)}(t))^2 = (w_i^{(j)}(0))^2 - (w_i^{(j_i^*)}(0))^2 \geq c_i \implies (w_i^{(j)}(t))^2 \geq c_i + (w_i^{(j_i^*)}(t))^2 \geq c_i. \tag{31}$$

Now we are able to show the convergence of the model. To begin with, note that

$$\frac{d}{dt}(\prod_{l=1}^{L} w_k^{(l)}) = \sum_{l=1}^{L} \dot{w}_k^{(l)} \prod_{j \neq l} w_k^{(j)} = -\left(\sum_{l=1}^{L} (\prod_{j \neq l}^{L} w_k^{(j)})^2\right) (B_{\text{NN}} - B), \tag{32}$$

$$\forall i \neq k,\ \ \frac{d}{dt}(\prod_{l=1}^{L} w_i^{(l)}) = \sum_{l=1}^{L} \dot{w}_i^{(l)} \prod_{j \neq l} w_i^{(j)} = -\left(\sum_{l=1}^{L} (\prod_{j \neq l}^{L} w_i^{(j)})^2\right) \left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right). \tag{33}$$

Now, first, we consider an index $i \neq k$. We have

$$\begin{aligned} \frac{d}{dt}\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)^2 &= 2\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right) \frac{d}{dt}\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right) \\ &= -2\left(\sum_{l=1}^{L} (\prod_{j \neq l}^{L} w_i^{(j)})^2\right) \left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)^2. \end{aligned} \tag{34}$$

Now using (31), we can say

$$\left(\sum_{l=1}^{L} (\prod_{j \neq l}^{L} w_i^{(j)})^2\right) \geq (\prod_{j \neq j_i^*}^{L} w_i^{(j)})^2 \geq c_i^{L-1} > 0. \tag{35}$$

Therefore, we have

$$\frac{d}{dt}\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)^2 = -2\left(\sum_{l=1}^{L}(\prod_{j\neq l}^{L} w_i^{(j)})^2\right)\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)^2 \le -2c_i^{L-1}\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)^2, \quad (36)$$

which shows

$$\left(\prod_{l=1}^{L} w_i^{(l)}(t) - \hat{f}(\{i\})\right)^2 \le \left(\prod_{l=1}^{L} w_i^{(l)}(0) - \hat{f}(\{i\})\right)^2 e^{-2c_i^{L-1}t}; \quad (37)$$

in other words, $\left(\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\})\right)^2$ goes to 0 exponentially fast in time, $t$. Finally, we make the same analysis for $(B_{\mathrm{NN}} - B)^2$. We have

$$\begin{aligned}
\frac{d}{dt}(B_{\mathrm{NN}} - B)^2 &= \frac{d}{dt}\left((b + \prod_{l=1}^{L} w_k^{(l)}) - B)\right)^2 = 2\left((b + \prod_{l=1}^{L} w_k^{(l)}) - B)\right)\frac{d}{dt}\left(b + \prod_{l=1}^{L} w_k^{(l)}\right) \\
&= 2\left((b + \prod_{l=1}^{L} w_k^{(l)}) - B)\right)\left(-(B_{\mathrm{NN}} - B) - \left(\sum_{l=1}^{L}(\prod_{j\neq l}^{L} w_k^{(j)})^2\right)(B_{\mathrm{NN}} - B)\right) \\
&= -2(B_{\mathrm{NN}} - B)^2\left(1 + \left(\sum_{l=1}^{L}(\prod_{j\neq l}^{L} w_k^{(j)})^2\right)\right) \le -2(B_{\mathrm{NN}} - B)^2.
\end{aligned}$$

The last equation shows that

$$(B_{\mathrm{NN}}(t) - B)^2 \le (B_{\mathrm{NN}}(0) - B)^2 e^{-2t}, \quad (38)$$

i.e., $(B_{\mathrm{NN}}(t) - B)^2$ converges to 0 exponentially fast in $t$ as well. Equations (23), (37), and (38) show that

$$L(\theta(t), x, f) \le L(\theta(0), x, f)e^{-ct}, \quad (39)$$

where $c = 2\min(1, \min(\{c_i\}_{i\neq k})^{L-1})$; hence, loss converges to zero exponentially fast in time (however, it is still initialization-dependent).

As shown in (38), the bias of neural network, converges like $(B_{\mathrm{NN}}(t) - B)^2 \le (B_{\mathrm{NN}}(0) - B)^2 e^{-2t}$. We denote $R := |\hat{f}(\emptyset) + \hat{f}(\{k\})| + 1 > |B_{\mathrm{NN}}(0) - B|$. Now notice that if $t \ge T_\epsilon := \frac{\log \frac{64R^2}{\epsilon^2}}{2}$, then we have

$$(B_{\mathrm{NN}}(t) - B)^2 \le (B_{\mathrm{NN}}(0) - B)^2 e^{-2t} \le R^2 e^{-\log \frac{64R^2}{\epsilon^2}} = \frac{\epsilon^2}{64}. \quad (40)$$

We now show the growth of $\prod_{l=1}^{L} w_k^{(l)}$ is comparatively slower, and therefore, it will not capture the bias fast enough and will remain small during the entire training process. More precisely, we first bound $\prod_{l=1}^{L} w_k^{(l)}$ at the beginning of training ($t \le T_\epsilon$). We define $m = \mathrm{argmax}_{l\in[L]}|w_k^{(l)}(0)|$. Again, by balancedness property, we know it will remain the largest during training, i.e.,

$$|w_k^{(i)}| = \sqrt{(w_k^{(i)})^2} \le \sqrt{(w_k^{(m)})^2} \le |w_k^{(m)}|. \quad (41)$$

Now note that

$$\begin{aligned}
\frac{d}{dt}(w_k^{(m)})^2 &= -2\prod_{j=1}^{L} w_k^{(j)}(B_{\mathrm{NN}}(t) - B) \le 2|\prod_{j=1}^{L} w_k^{(j)}(B_{\mathrm{NN}}(t) - B)| \\
&\le 2|w_k^{(m)}|^L |B_{\mathrm{NN}}(t) - B| \\
&\le 2((w_k^{(m)})^2)^{\frac{L}{2}}|B_{\mathrm{NN}}(0) - B| = 2((w_k^{(m)})^2)^{\frac{L}{2}} R, \quad (42)
\end{aligned}$$

where in the last line we used the fact that $(B_{\mathrm{NN}}(t) - B)^2$ is decreasing. Now, we provide a bound for $|w_k^{(m)}|$. First, we consider the case that $L = 2$. In this case, we have

$$\frac{d}{dt}(w_k^{(m)})^2 \le 2((w_k^{(m)})^2)R \Longrightarrow (w_k^{(m)}(t))^2 \le (w_k^{(m)}(0))^2 e^{2Rt}, \quad (43)$$

where we used Gronwall's lemma in the last equation. It also shows

$$\prod_{l=1}^{L} w_k^{(l)}(t) \le w_k^{(m)}(t)^L = w_k^{(m)}(t)^2 \le (w_k^{(m)}(0))^2 e^{2Rt}. \tag{44}$$

Now, we consider the case that $L > 2$. In this case, we also have (this could be considered as an extension of Gronwall's lemma, note that $w_k^{(m)} > 0$)

$$\frac{d}{dt}(w_k^{(m)})^2 \le 2((w_k^{(m)})^2)^{\frac{L}{2}} R \Longrightarrow \tag{45}$$

$$\frac{d}{dt}((w_k^{(m)})^2)^{1-\frac{L}{2}} = -(\frac{L}{2}-1)((w_k^{(m)})^2)^{-\frac{L}{2}} \frac{d}{dt}(w_k^{(m)})^2 \ge -(L-2)R, \tag{46}$$

using the above we have

$$(w_k^{(m)}(t)^2)^{1-\frac{L}{2}} - (w_k^{(m)}(0)^2)^{1-\frac{L}{2}} = \int_0^t \frac{d}{dt}(w_k^{(m)}(t)^2)^{1-\frac{L}{2}} \ge -(L-2)Rt \Longrightarrow \tag{47}$$

$$w_k^{(m)}(t)^2 \le \frac{1}{(|w_k^{(m)}(0)|^{2-L} - (L-2)Rt)^{\frac{1}{\frac{L}{2}-1}}} \qquad t < \frac{|w_k^{(m)}(0)|^{2-L}}{(L-2)R}, \tag{48}$$

hence, we have

$$\prod_{l=1}^{L} w_k^{(l)}(t) \le (w_k^{(m)}(t)^2)^{\frac{L}{2}} \le \frac{1}{(|w_k^{(m)}(0)|^{2-L} - (L-2)Rt)^{\frac{L}{L-2}}} \qquad t < \frac{|w_k^{(m)}(0)|^{2-L}}{(L-2)R}. \tag{49}$$

Now we consider each of these bounds at $t = T_\epsilon$. First, for $L = 2$, we have

$$\prod_{l=1}^{L} w_k^{(l)}(t) \le (w_k^{(m)}(0))^2 e^{2Rt} = (w_k^{(m)}(0))^2 e^{2RT_\epsilon}, \tag{50}$$

which is upper bounded by $\frac{\epsilon}{8}$ if $(w_k^{(m)}(0))^2 \le \alpha^2 \le \alpha_{max}^2 = \frac{\epsilon}{8e^{2RT_\epsilon}}$. Now, we consider the bound for deeper networks, $L > 2$, at time $t = T_\epsilon$. We want to bound $\prod_{l=1}^{L} w_k^{(l)}(t)$ by $\frac{\epsilon}{8}$. Using (49) this will happen if we have

$$\frac{1}{(|w_k^{(m)}(0)|^{2-L} - (L-2)RT_\epsilon)^{\frac{L}{L-2}}} \le \frac{\epsilon}{8} \iff (L-2)RT_\epsilon + (\frac{8}{\epsilon})^{\frac{L-2}{L}} \le |w_k^{(m)}(0)|^{2-L}, \tag{51}$$

which will happen if $|w_k^{(m)}(0)| \le \alpha_{\max} \coloneqq ((L-2)RT_\epsilon + (\frac{8}{\epsilon})^{\frac{L-2}{L}})^{\frac{1}{2-L}}$.

So we proved for small enough initializations, there exists a time, $T_\epsilon$, where

$$|b(T_\epsilon) + \prod_{l=1}^{L} w_k^{(l)}(T_\epsilon) - B| \le \frac{\epsilon}{8}, \tag{52}$$

$$|\prod_{l=1}^{L} w_k^{(l)}(T_\epsilon)| \le \frac{\epsilon}{8}, \tag{53}$$

$$|b(T_\epsilon) - B| \le |b(T_\epsilon) + \prod_{l=1}^{L} w_k^{(l)}(T_\epsilon) - B| + |\prod_{l=1}^{L} w_k^{(l)}(T_\epsilon)| \le \frac{2\epsilon}{8}. \tag{54}$$

We now show that this picture will not change much during the rest of the training process. To see this, note that $|B_{\mathrm{NN}}(t) - B|$ is always decreasing over time and is continuous. Therefore, $B_{\mathrm{NN}}(t) - B$ cannot change the sign (since changing the sign means that the variable had become equal to 0 at some time, which is contrary to the fact that its absolute value is decreasing). Considering equations (25) and (32) we can conclude that both $b(t) - B$ and $\prod_{l=1}^{L} w_k^{(l)}(t)$ are either increasing or decreasing during the whole training. First, consider both of them is increasing. For $t > T_\epsilon$, we have

$$|\prod_{l=1}^{L} w_k^{(l)}(t) + b(t) - B| \leq |\prod_{l=1}^{L} w_k^{(l)}(T_\epsilon) + b(T_\epsilon) - B| \leq \frac{\epsilon}{8} \Longrightarrow \tag{55}$$

$$\frac{-\epsilon}{8} \leq \prod_{l=1}^{L} w_k^{(l)}(T_\epsilon) \leq \prod_{l=1}^{L} w_k^{(l)}(t) \leq \frac{\epsilon}{8} - (b(t) - B) \leq \frac{\epsilon}{8} - (b(T_\epsilon) - B) \leq \frac{3\epsilon}{8} \Longrightarrow |\prod_{l=1}^{L} w_k^{(l)}(t)| \leq \frac{3\epsilon}{8}, \tag{56}$$

$$|b(t) - B| \leq |\prod_{l=1}^{L} w_k^{(l)}(t) + b(t) - B| + |\prod_{l=1}^{L} w_k^{(l)}(t)| \leq \frac{4\epsilon}{8}. \tag{57}$$

The case for both functions being decreasing is also similar. This shows that $f_{\mathrm{NN}}(\{k\}) < \epsilon$ during the entire training. Now we can study GOTU loss for $t \geq T_\epsilon$ using Parseval's theorem as follows:

$$\begin{aligned} GOTU(f, f_{\mathrm{NN}}, \{x_k = -1\}) &= \left( ((b - \prod_{l=1}^{L} w_k^{(l)}) - (\hat{f}(\emptyset) - \hat{f}(\{k\})))^2 + \sum_{i \neq k}^{d} (\prod_{l=1}^{L} w_i^{(l)} - \hat{f}(\{i\}))^2 \right) && (58) \\ &= \left( ((b - B) - \prod_{l=1}^{L} w_k^{(l)} + 2\hat{f}(\{k\}))^2 + O(e^{-ct}) \right) && (59) \\ &= 4\hat{f}(\{k\})^2 + O_t(e^{-ct}) + O_\epsilon(\epsilon), && (60) \end{aligned}$$

which proves the theorem. Note that if we consider half $\ell_2$ loss for the entire population $\Omega$ the loss becomes $\hat{f}(\{k\})^2 + O_t(e^{-ct}) + O_\epsilon(\epsilon)$. □

*Remark* A.2 (Initialization of bias variable). Note that the analysis is independent of the initialization of the bias variable (as long as it satisfies a simple bound such as $|b(0)| \leq \frac{1}{2}$).

*Remark* A.3 (Effect of depth). The current theorem proves that the low-degree solution is learned when the initialization scale is small enough. To see the effect of depth, we prove that $\alpha_{\max}$ found in this proof is increasing by depth, $L$. In other words, if we have deeper networks, we can use larger initializations and still have the generalization error close to the Boolean influence.

*Proof for Remark A.3.* Consider $L \geq 3$. We know that $\alpha_{\max} := ((L-2)RT_\epsilon + (\frac{8}{\epsilon})^{\frac{L-2}{L}})^{\frac{1}{2-L}}$. For simplicity define

$$P := RT_\epsilon, \tag{61}$$

$$Q := \frac{8}{\epsilon} > e^3 \text{ (we assume this)}, \tag{62}$$

$$g(x) := (xP + Q^{\frac{x}{x+2}})^{\frac{-1}{x}}. \tag{63}$$

Now note that $\alpha_{\max} = g(L-2)$. Therefore, we need to prove $g(x)$ is increasing for $x \geq 1$. To see this, note that

$$\frac{d}{dx}\frac{x}{x+2} = \frac{2}{(x+2)^2}, \tag{64}$$

$$\frac{d}{dx} Q^{\frac{x}{x+2}} = Q^{\frac{x}{x+2}} (\ln Q) \frac{2}{(x+2)^2}, \tag{65}$$

$$\frac{d}{dx} \ln(xP + Q^{\frac{x}{x+2}}) = \frac{P + Q^{\frac{x}{x+2}} (\ln Q) \frac{2}{(x+2)^2}}{xP + Q^{\frac{x}{x+2}}}, \tag{66}$$

$$\frac{d}{dx} \frac{-\ln(xP + Q^{\frac{x}{x+2}})}{x} = \frac{-x \frac{P + Q^{\frac{x}{x+2}} (\ln Q) \frac{2}{(x+2)^2}}{xP + Q^{\frac{x}{x+2}}} + \ln(xP + Q^{\frac{x}{x+2}})}{x^2}. \tag{67}$$

Therefore, $\frac{d}{dx}\frac{-\ln(xP+Q^{\frac{x}{x+2}})}{x} \geq 0$, iff

$$\ln(xP+Q^{\frac{x}{x+2}}) \geq x\frac{P+Q^{\frac{x}{x+2}}(\ln Q)\frac{2}{(x+2)^2}}{xP+Q^{\frac{x}{x+2}}} \iff (xP+Q^{\frac{x}{x+2}})\ln(xP+Q^{\frac{x}{x+2}}) \geq xP+Q^{\frac{x}{x+2}}(\ln Q)\frac{2x}{(x+2)^2}, \tag{68}$$

which holds because

$$xP\ln(xP+Q^{\frac{x}{x+2}}) \geq xP\ln(Q^{\frac{1}{3}}) \geq xP, \tag{69}$$

$$Q^{\frac{x}{x+2}}\ln(xP+Q^{\frac{x}{x+2}}) \geq Q^{\frac{x}{x+2}}\ln(Q)\frac{x}{x+2} \geq Q^{\frac{x}{x+2}}(\ln Q)\frac{2x}{(x+2)^2}. \tag{70}$$

Therefore, $\exp(\frac{-\ln(xP+Q^{\frac{x}{x+2}})}{x}) = g(x)$ is increasing. Finally, we have to compare $\alpha_{\max}$ for depths 2 and 3. Note that for depth two $\alpha_{\max}(2) = \sqrt{\frac{\epsilon}{8}}e^{-RT_\epsilon} = \sqrt{\frac{1}{Q}}e^{-P}$ while for depth three, we have $\alpha_{\max}(3) = \frac{1}{P+\sqrt[3]{Q}}$. Therefore, we have

$$\frac{1}{\alpha_{\max}(2)} = e^P\sqrt{Q} \geq (P+1)\sqrt{Q} \geq P+\sqrt[3]{Q} = \frac{1}{\alpha_{\max}(3)},$$

which gives the desired result. □

### A.3. Proof for Theorem 5.1

*Proof.* First, we prove the existence and uniqueness of such low-degree interpolators. Afterward, we consider it explicitly for parity functions.

Note that we know there are no $r+1$ bits which are all equal to $-1$ in $B_r$. Therefore, for any $r+1$ indices, we have $(x_{i_1}-1)\cdots(x_{i_{r+1}}-1) = 0$. Consequently, each $x_{i_1}\cdots x_{i_{r+1}}$ can be replaced by a degree $r$ polynomial. Now consider the Fourier-Walsh expansion of $f(x)$. By applying the previous identity, one can replace all monomials in the Fourier-Walsh expansion of $f(x)$ with degree $r$ (or less) alternatives, while the value of the function on $B_r$ does not change.

Now we prove the uniqueness. Consider all monomials $\chi_T(x)$ where $|T| \leq r$. There are in total $\binom{d}{0}+\binom{d}{1}+\cdots+\binom{d}{r} = |B_r|$ of such monomials and consider all functions given by these monomials $f_a(x) = \sum_{i=1}^{|B_r|} a_i\chi_{T_i}(x)$. Note that for each $x_j \in B_r$ $1 \leq j \leq |B_r|$, $f_a(x_j) = \sum_{i=1}^{|B_r|} a_i\chi_{T_i}(x_j)$. In other words, $f_a(x_j)$ is a linear combination of $a_i$'s, i.e., $(f_a(x_1),\ldots,f_(x_{|B_r|}))^T = M(a_1,\ldots,a_{|B_r|})^T$, where $M_{i,j} = \chi_{T_j}(x_i)$. Now note that we have proven that any function can be written in this way, i.e., $\mathrm{rank}(M) = |B_r|$ showing that $\dim(\ker(M)) = 0$ and hence the uniqueness.

Now, we particularly study the case of monomials. Without loss of generality, consider degree $k > r$ monomial $\mathrm{parity}_k(x) := x_1x_2\cdots x_k$. We claim that

$$\begin{aligned} f_r(x) &:= 1 + \sum_{1\leq i\leq k}(x_i-1) + \sum_{1\leq i<j\leq k}(x_i-1)(x_j-1) + \cdots + \sum_{1\leq i_1<i_2<\cdots<i_r\leq k}(x_{i_1}-1)\cdots(x_{i_r}-1) \\ &= 1 + \sum_{T\subseteq[k]:|T|=1}\prod_{i\in T}(x_i-1) + \cdots + \sum_{T\subseteq[k]:|T|=r}\prod_{i\in T}(x_i-1) \end{aligned} \tag{71}$$

is the the unique low-degree equivalent of $\mathrm{parity}_k$ on $B_r$, i.e., $\mathrm{parity}_k(x) = f_r(x)$ $\forall x \in B_r$. To see this, take any $x \in B_r$. Define $s(x)$ as the number of $-1$ bits in $x_1,\cdots,x_k$, i.e., $s(x) := |\{x_i = -1 | 1 \leq i \leq k\}|$. Note that $0 \leq s(x) \leq k$ and $\mathrm{parity}_k(x) = (-1)^{s(x)}$. Furthermore, we have

$$\forall 1 \leq i \leq r \quad \sum_{T\subseteq[k]:|T|=i}\prod_{j\in T}(x_j-1) = (-2)^i\binom{s(x)}{i}. \tag{72}$$

Therefore,

$$f_r(x) = 1 + \sum_{T\subseteq[k]:|T|=1}\prod_{i\in T}(x_i-1) + \cdots + \sum_{T\subseteq[k]:|T|=r}\prod_{i\in T}(x_i-1) \tag{73}$$

$$= 1 + (-2)^1\binom{s(x)}{1} + \cdots + (-2)^i\binom{s(x)}{i} + \cdots + (-2)^r\binom{s(x)}{r} = (1-2)^s = (-1)^{s(x)} = \mathrm{parity}_k(x), \tag{74}$$

where we used the fact that $s(x) \leq r$. Now we consider the constant term (i.e., bias) of $f_r(x)$. Indeed notice that the constant in $f_r(x)$ is given by

$$\hat{f}_r(\emptyset) = 1 - \binom{k}{1} + \binom{k}{2} - \cdots + (-1)^r \binom{k}{r}. \tag{75}$$

It can easily be proven that the above constant is equal to $(-1)^r \binom{k-1}{r}$ by induction on $r$. Note that it is clear for $r = 1$ and the induction step from $r$ to $r + 1$ is given by

$$1 - \binom{k}{1} + \cdots + (-1)^r \binom{k}{r} + (-1)^{r+1} \binom{k}{r+1} = (-1)^r \binom{k-1}{r} + (-1)^{r+1} \binom{k}{r+1} \tag{76}$$

$$= (-1)^{r+1} (\binom{k}{r+1} - \binom{k-1}{r}) = (-1)^{r+1} \binom{k-1}{r+1}. \tag{77}$$

Therefore, by Parseval's identity we have

$$\mathbb{E}_x[(\text{parity}_k(x) - f_r(x))^2] > \hat{f}_r(\emptyset)^2 = \binom{k-1}{r+1}^2, \tag{78}$$

which proves the lower bound. Note that we ignored other Fourier-Walsh coefficients for the lower bound above.

□

# B. Experiment details and additional experiments

## B.1. Experiment details

### B.1.1. Architectures

We use MLP, Transformer (Vaswani et al., 2017), mean-field (Mei et al., 2018) and random features model (Definition 3.5) for experiments. Here, we describe them in detail.

- **MLP.** The MLP model is a fully connected network consisting of 4 hidden layers of sizes $512, 1024, 512, 64$. The ReLU activation function is used for all layers except the output layer. Moreover, the standard initialization of PyTorch has been followed, i.e., the weights of each layer are initialized with $U(\frac{-1}{\sqrt{\text{dim}_{\text{in}}}}, \frac{1}{\sqrt{\text{dim}_{\text{in}}}})$ where $\text{dim}_{\text{in}}$ is the input dimension of the layer.

- **Transformer.** We have employed the encoder part of Transformer networks which are widely used in computer vision (Dosovitskiy et al., 2020) and language modeling (Raffel et al., 2019). First, all binary $\pm 1$ bits are encoded into a 256-dimensional vector using a shared embedding layer. Afterward, the embedded input is passed through 12 transformer layers. Finally, a linear layer is used to compute the output of the model. Moreover, the size of MLP hidden layers is set to 256, and 6 heads are used for the self-attention blocks.

- **Mean-field.** We also use a two-layer neural network in the mean-field parametrization. More precisely, following (Abbe et al., 2022b), $f_{\text{MF}}(x) = \frac{1}{N}\sum_{i=1}^{N} a_i \sigma(\langle w_i, x \rangle + b_i)$, where $a_i \sim U(-1, 1)$ and $w_i, b_i \sim U(\frac{-1}{\sqrt{d}}, \frac{1}{\sqrt{d}})^{\otimes d} \otimes U(\frac{-1}{\sqrt{d}}, \frac{1}{\sqrt{d}})$. We use ReLU as the activation function and set the number of neurons to $N = 2^{16}$. Note that with this formulation, one has to take large values for the learning rate, e.g., 100 or 1000.

- **Random features model.** Following Definition 3.5, we have used $f_{\text{RF}} = \sum_{i=1}^{N} a_i \sigma(\langle w_i, x \rangle + b_i)$ as the parametrization of the RF model. Moreover, we initialize $a_i = 0$ and $w_i, b_i \sim \mathcal{N}(0, \frac{1}{d})^{\otimes d} \otimes \mathcal{N}(0, \frac{1}{d})$ where $d$ is the input dimension. We also use $N = 2^{13}$ random features for our experiments. We have used the ReLU activation function for almost all of the experiments. We have only used polynomial activation $(1 + x)^6$ for the experiment comparing RF models with the ReLU activation and polynomial activation (Figure 5).

### B.1.2. Procedure

The implementation of experiments has been done using the PyTorch framework (Paszke et al., 2019). Additionally, the experiments were executed on NVIDIA A100 GPUs and the experiments took around 60 GPU hours in total (excluding the selection of hyperparameters). Now we discuss the training procedures.

**Length generalization and main experiments.** We first explain the experiments of the main experiment section and also experiments for the length generalization. For each function $f : \{\pm 1\}^d \to \mathbb{R}$ and unseen domain $\mathcal{U}$, we generate all binary vectors in $\mathcal{U}^c = \{\pm 1\}^d \setminus \mathcal{U}$ for the training set. Consequently, we usually take small values of $d$ for the experiments. Our main motivation for doing so is to eliminate the randomness generated by the sampling of training examples and also to assume the in-distribution generalization. Nonetheless, we believe the min-degree bias still holds when training examples are sampled randomly as is illustrated in the experiments included in this appendix.

We then train our models. For the Transformer, we have used Adam (Kingma & Ba, 2014) optimizer with batch size $256$. For the RF models, we have used mini-batch SGD with a batch size of $256$. Also, for the rest of the architectures, SGD with batch size $64$ has been used. We did not observe any significant difference in the results of experiments by varying the batch sizes. We generally selected the learning rates per model (and task) by the stability of the training and the speed of convergence. We have included more details about the learning rate in Appendix B.2. We also set the number of training epochs large enough that the loss of models is always less than $10^{-2}$. We also note that Transformers usually learn the target function in a few epochs, reaching a loss of the order of $10^{-4}$. After that, the training becomes unstable in some instances. Indeed note that Transformers are usually trained with learning rate schedulers. However, we did not use any learning rate schedulers for simplicity and instead opted for early stopping to avoid unstable phases of training. Note that our main objective is to demonstrate the min-degree bias of neural networks and not to optimize any performance metric. As a result, we did not focus on hyperparameter tuning in these experiments. Generally, hyperparameters used for our experiments are available in our code.

Finally, we track the coefficient of different monomials, i.e., $\hat{f}_{\mathrm{NN}}(T) = \mathbb{E}_x[\chi_T(x) f_{\mathrm{NN}}(x)]$ during the training. We have also repeated each experiment for $10$ different seeds and reported the averages. Particularly, we have also drawn $95\%$-CI in Figures 4, 6, and 11, but we did not draw CI for other experiments to keep the plots more readable.

**Curriculum learning experiments.** In contrast to other experiments, there is no unseen domain in these experiments. Also here we draw a fixed number of samples uniformly from $\{\pm 1\}^d$. We train the MLP model with the same training set, learning rate, and batch size, once with normal mini-batch SGD and once with Degree-Curriculum (Algorithm 1). Therefore, everything between the Degree-Curriculum algorithm and the normal training is the same. We use Adam optimizer for these experiments as we found it to be faster than plain SGD. Moreover, we selected the learning rate based on the results of the normal training and then used the same learning rate for the Degree-Curriculum algorithm to have a fair comparison. Finally, we compare the average generalization loss between the two algorithms.

### B.2. Sensitivity to learning rate

We noticed that the min-degree bias of some architectures such as MLPs depends on the learning rate. More precisely, we observed that smaller learning rates promote the min-degree bias and larger learning rates increase the leakage of the models. Here, we demonstrate the effect of the learning rate with an example. Consider learning $f_2(x_0, \ldots, x_{14}) = x_0 x_1$ under unseen domain $\mathcal{U}_2 = \{(x_0, x_1) = (-1, -1)\}$. In this case, the min-degree interpolator is $x_0 + x_1 - 1$. Nonetheless, any $\alpha(x_0 x_1) + (1 - \alpha)(x_0 + x_1 - 1)$ is also a valid interpolator where $\alpha$ shows the leakage of the interpolator. We tried learning $f_2$ under $\mathcal{U}_2$ with an MLP and varied the learning rate; the results are depicted in Figure 6.

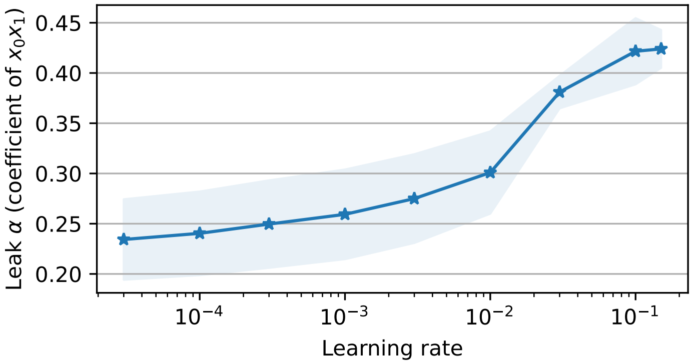

*Figure 6.* Leakage of the interpolators learned by the MLP model trained with different learning rates. Larger learning rates weaken the min-degree bias and lead to higher leaks.

It can be seen that larger learning rates cause higher leaks in the models. We note that training becomes more unstable with larger learning rates to the point that the model cannot be trained with learning rates larger than $0.2$. Also notice that $\alpha < 0.5$ in all cases, hence, the min-degree alternatives are still dominant. In general, in our experiments, we tried to select moderate values for the learning rate to ensure that the optimization process is stable. Nonetheless, we never used learning rate below $10^{-5}$ for Adam and we usually set learning rate between $10^{-4}$ to $10^{-3}$ for SGD. Exact hyperparameters for different experiments are available in our code.

### B.3. Additional experiments

Here, we complete the experiments presented in Section 4 and also provide an additional experiment for the Degree-Curriculum. First, we report results of other architectures on $(f_1, \mathcal{U}_1) = (x_0x_1 - 1.25x_1x_2 + 1.5x_1x_2, \{x_0x_1x_2 = -1\})$ and $(f_3, \mathcal{U}_3) = (x_0x_1x_2 + \cdots + x_{13}x_{14}x_0 + x_{14}x_0x_1, \{(x_0, x_1, x_2) = (-1, -1, -1)\})$. Figures 7 and 8 present the results for function $f_1$ and $f_3$ respectively. In consistency with our results in Section 4, we can see the MLP and mean-field models learn leaky min-degree solutions, while the leakage for the random features model is negligible.

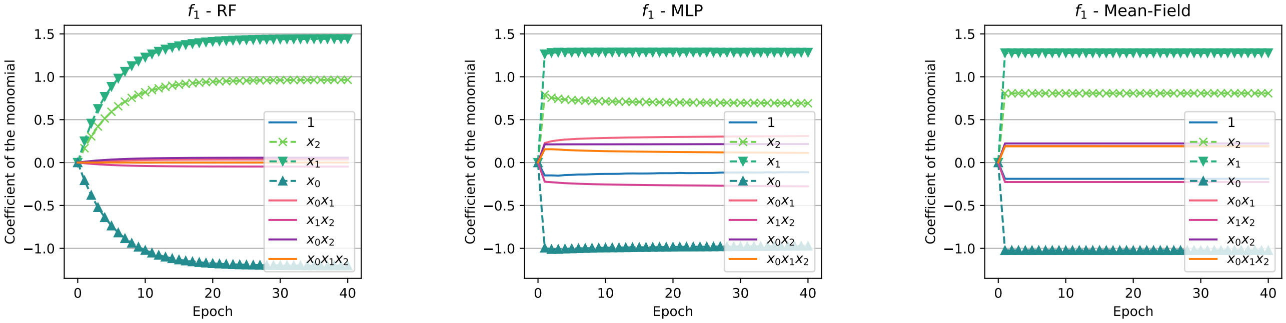

*Figure 7.* $f_1(x_0, \dots, x_{14}) = x_0x_1 - 1.25x_1x_2 + 1.5x_1x_2$ learned by RF, MLP, and mean-field models under unseen domain $\mathcal{U}_1\{x_0x_1x_2 = -1\}$. In this case, the MD interpolator is $x_2 - 1.25x_0 + 1.5x_0$. The solid lines and dashed lines present the coefficients of the original function and the coefficients of the MD interpolator respectively. The plots indicate that the mean-field model and MLP model both learn leaky MD interpolators. The RF model has a smaller leak which is caused by the ambient dimension $d = 15$ being small.

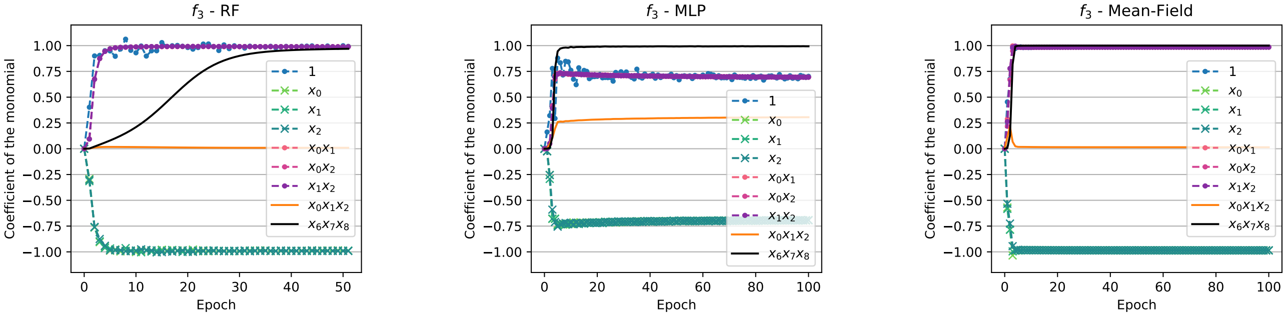

*Figure 8.* $f_3(x_0, \dots, x_{14}) = x_0x_1x_2 + \cdots + x_{13}x_{14}x_0 + x_{14}x_0x_1$ learned by RF, MLP, and mean-field models while samples satisfying $(x_0, x_1, x_2) = (-1, -1, -1)$ are not seen during training. In this case, $x_0x_1x_2$ (orange solid line) is replaceable by $x_0x_1 + x_1x_2 + x_2x_0 - x_0 - x_1 - x_2 + 1$ (dashed lines). It can be seen that the RF model learns the MD interpolator, while both the MLP and mean-field models learn the MD interpolator with a leakage.

Next, we try the second experiment of the paper in a larger ambient dimension to see if the ambient dimension has an effect on the leakage of the models. More specifically, we use ambient dimension $d = 40$ and consider learning $f_2(x_0, x_1, \dots, x_{39}) = x_0x_1$ under unseen domain $\{(x_0, x_1) = (-1, -1)\}$. In this case, the MD interpolator is again $x_0 + x_1 - 1$. For this experiment, we can not generate all $2^{39}$ elements of $\mathcal{U}^c$, thus, we only use $2^{15}$ samples uniformly drawn from $\mathcal{U}^c$. We also use the same number of samples for the estimation of Fourier-Walsh coefficients. The results are depicted in Figure 9. For the random features model, it can be seen that the leakage is reduced compared to Figure 2 where the ambient dimension is 15. On the other hand, the leakage of other models has remained the same, which shows that the sparsity and ambient dimension do not affect them. This is indeed consistent with our expectations as we know models such as the mean-field are able to perform feature-learning and learn the support of sparse Boolean functions (Abbe et al., 2022b).

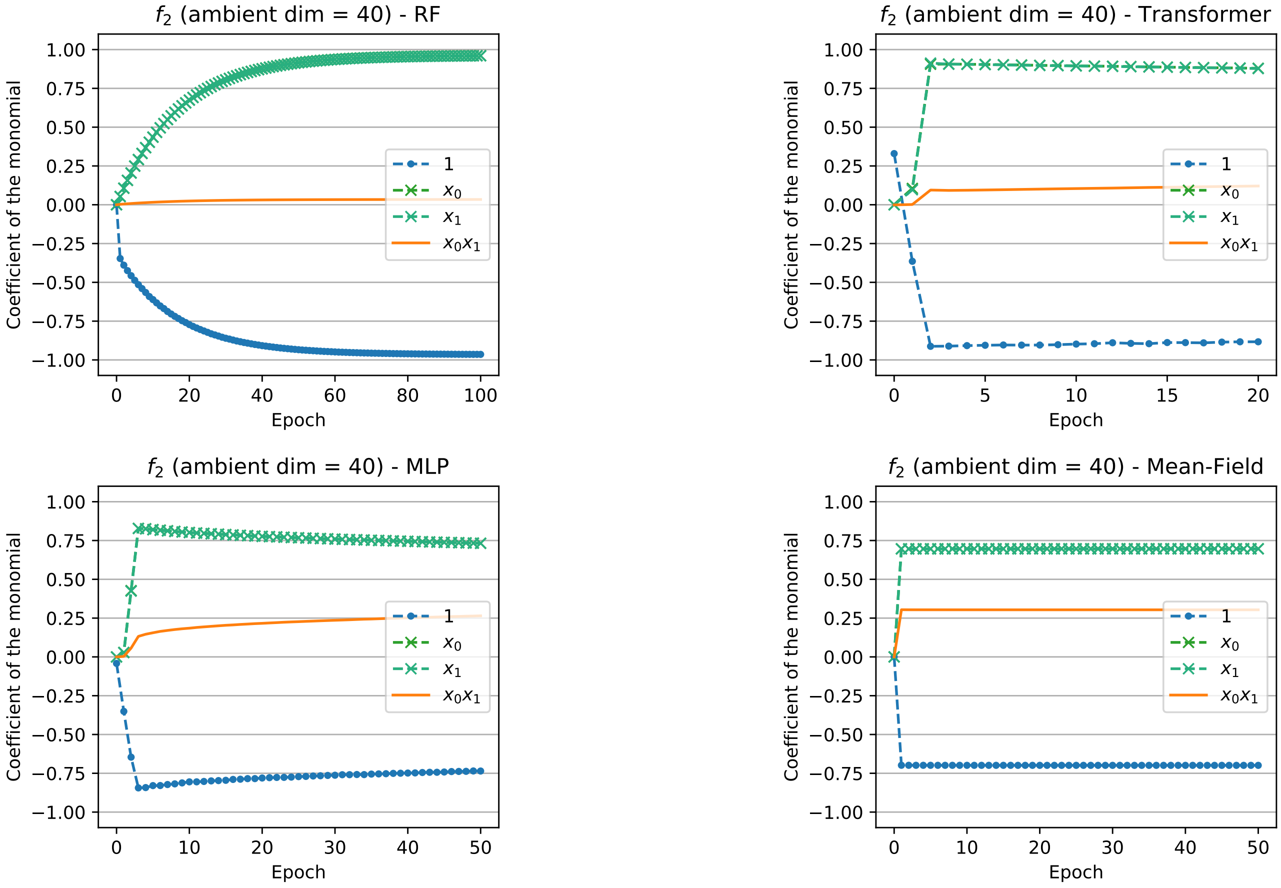

*Figure 9.* $f_2(x_0, x_1, \ldots, x_{39}) = x_0x_1$ learned by the RF, Transformer, MLP, and mean-field models while training samples satisfy $(x_0, x_1) \neq (-1, -1)$. Consequently, $x_0x_1$ (solid orange line) is replaceable by $x_0 + x_1 - 1$ (dashed lines). The Transformer, MLP, and mean-field models learn leaky solutions and the leakage is very similar to Figure 2 where the ambient dimension is 15. In contrast, the leak of the RF model is decreased in comparison to Figure 2.

Further, we consider the majority function on 3 bits embedded in a 40-dimensional ambient space, i.e., $f_4(x_0, x_1, \ldots, x_{39}) = \mathrm{Maj}(x_0, x_1, x_2) = \frac{1}{2}(x_0 + x_1 + x_2 - x_0x_1x_2)$ under the unseen domain $\mathcal{U}_4 = \{x \in \{\pm 1\}^d | (x_0, x_1) = (-1, -1)\}$. Note that in this case $x_0x_1x_2$ can be replaced with $x_0x_2 + x_1x_2 - x_2$ which leads to MD interpolator being equal to $\frac{1}{2}(x_0 + x_1 + 2x_2 - x_0x_2 - x_1x_2)$. Similar to the previous experiments, we trained the RF, MLP, mean-field, and Transformer on this instance. For this example, we do not generate the whole $\mathcal{U}^c$, and instead, we use $2^{15}$ samples. This number is still large enough that gives the generalization on the seen domain. The results of this experiment are presented in Figure 10. Note that in this case, the original target function is more symmetric than the MD interpolator. Nonetheless, none of the models are able to recover the more symmetric function.

Lastly, we present the curriculum experiment for the full parity function in dimension 30, i.e., $f(x_0, \ldots, x_{29}) = x_0x_1 \cdots x_{29}$, in Figure 11 to complete the experiment presented in Figure 4. Similar to the experiment presented in Figure 4, for the same training set and hyperparameters, we once train the MLP with normal SGD and once with the proposed Degree-Curriculum algorithm with curriculum $B_1 \subset B_2 \subset \cdots \subset B_{29} \subset B_{30}$ (leap 1 curriculum) and loss threshold $\epsilon = 0.001$. It can be seen that the curriculum method can reduce the sample complexity of the task.

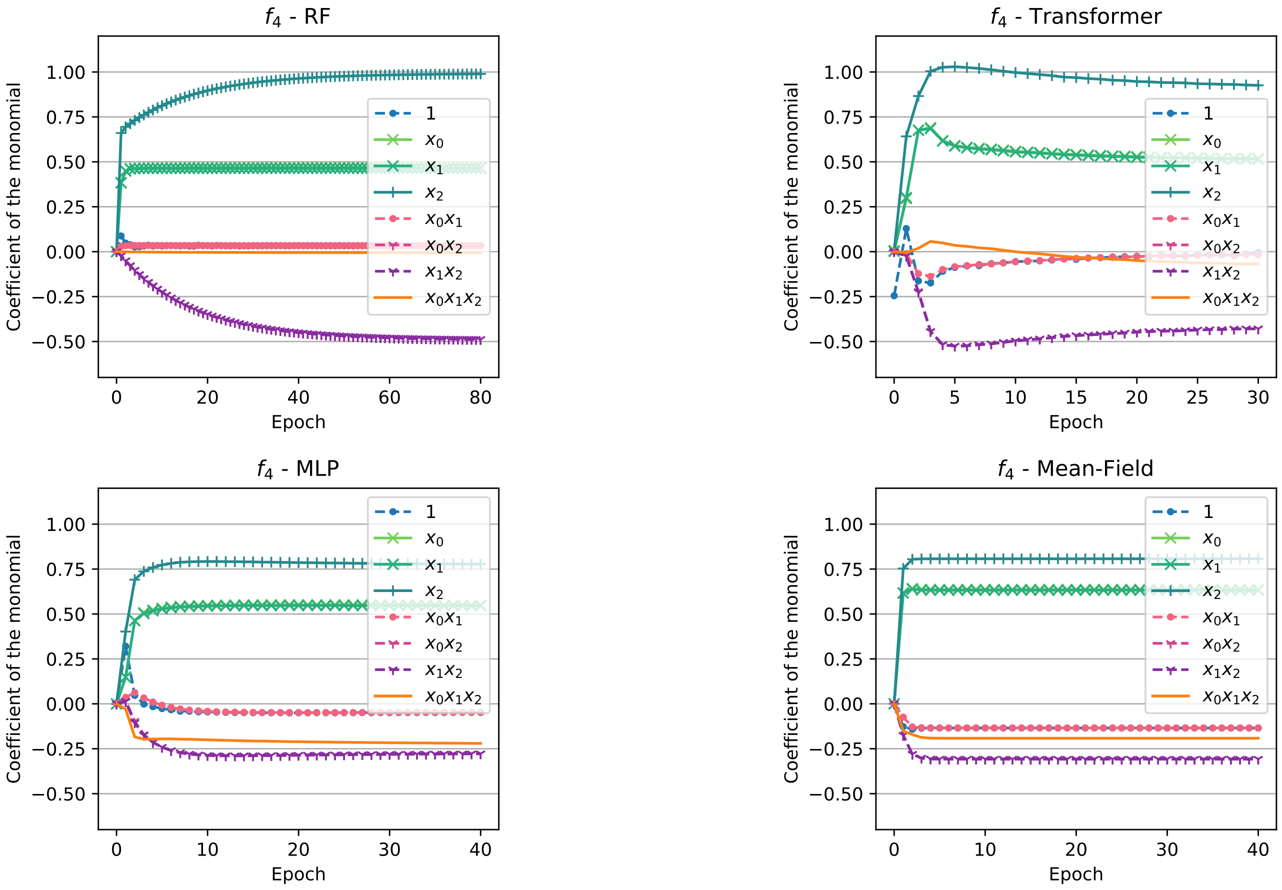

*Figure 10.* $f_4(x_0, x_1, \ldots, x_{39}) = \mathrm{Maj}(x_0, x_1, x_2) = \frac{1}{2}(x_0 + x_1 + x_2 - x_0x_1x_2)$ learned by the RF, Transformer, MLP, and mean-field models while samples satisfying $(x_0, x_1) = (-1, -1)$ are excluded from training. In this case, $x_0x_1x_2$ (orange solid line) is replaceable by $x_0x_2 + x_1x_2 - x_2$. As expected, the RF learns the MD interpolator, while other models have leakage.

## C. Vanishing ideals

In this section, we discuss the connection between unseen domains in Boolean settings and algebraic geometry and vanishing ideals. We refer interested readers to (Dummit & Foote, 2004) for broader coverage of this topic. First, we recall some basic properties of rings and fields. A ring is a set with two binary operations, the addition $+$ and the multiplication $*$ where $*$ may not have an inverse. A field is a ring such that all nonzero elements have an inverse. For example, $\mathbb{Z}$ with addition and multiplication is a ring but not a field. Whereas $\mathbb{R}$ and $\mathbb{C}$ are examples of fields. Here we will mostly work with polynomial rings with $d$ variables. Note that $\mathbb{R}[x_1, x_2, x_3, \cdots, x_d]$ is of special interest to us since any Boolean function $f : \{\pm 1\}^d \to \mathbb{R}$ can be represented by a polynomial $p(x) \in \mathbb{R}[x_1, x_2, x_3, \cdots, x_d]$ thanks to its Fourier-Walsh expansion. Particularly, we focus on polynomial rings $R = \mathbb{K}[x_1, x_2, \cdots x_d]$ where $\mathbb{K}$ is a field. We start by recalling a few definitions.

**Definition C.1** (Ideal)**.** Let $R$ be a commutative ring. $I \subseteq R$ is an ideal if

- $(I, +)$ is a group, and
- for all $r \in R$ and $i \in I$ we have $ri \in I$.

Having defined ideals, note that ideals can be generated from a $G \subseteq R$.

**Definition C.2.** Consider a commutative ring $R$ and let $G \subseteq R$. The ideal generated by $G$ denoted by $\langle G \rangle$ is the smallest

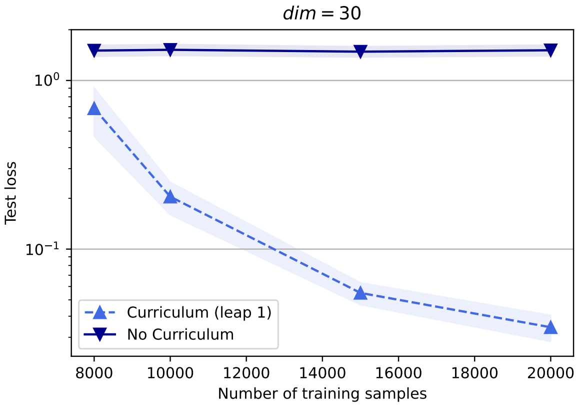

*Figure 11.* Test loss on the full parity function in dimension 30 for different training set sizes with and without the Degree-Curriculum algorithm. We note that the MLP model trained without curriculum was not able to learn the full parity function in dimension 30 for the given sample sizes (and even up to $10^5$ samples), in contrast to the same model trained with Degree-Curriculum.

ideal that contains $G$. Particularly, if $G = \{g_1, g_2, \cdots, g_n\}$ is finite, we have

$$\langle G\rangle = \langle g_1, g_2, \cdots, g_n\rangle = \{\sum_{i=1}^{n} r_i g_i | \forall r_1, r_2, \ldots, r_n \in R\}. \tag{79}$$

For example, for $R = \mathbb{R}[x_1, x_2]$, we have $\langle x_1 - 1, x_1x_2 + 5\rangle = \{p(x_1 - 1) + q(x_1x_2 + 5)|p, q \in \mathbb{R}[x_1, x_2]\}$. Another important notion is the notion of quotients, which is similar to the modulo operator. The following definition will make it more rigorous.

**Definition C.3** (Quotient)**.** Let $R$ be a commutative ring and $I$ an ideal of $R$. Quotient $R/I$ is defined as elements of the form $r + I$ with $r \in R$ such that $r + I = r' + I$ if $r - r' \in I$. Furthermore, for any for $r + I, r' + I \in R/I$, addition $+$ and multiplication $\cdot$ for $R/I$ are defined as

- $(r + I) + (r' + I) = r + r' + I$, and
- $(r + I) \cdot (r' + I) = rr' + I$.

Also, for $r' \in R/I$, any element $r \in R$ satisfying $r' = r + I$ is called a representative of $r'$. $R/I$ as defined above is indeed a ring.

Consider the following ideal $I_\Omega = \langle x_1^2 - 1, x_2^2 - 1, \cdots, x_d^2 - 1\rangle$ of $\mathbb{R}[x_1, x_2, x_3, \cdots, x_d]$ for Boolean functions. Note that for each binary bit $x_i$ we have $x_i^2 - 1 = 0$. Therefore, the Fourier-Walsh transform is a bijection between $\mathbb{R}[x_1, x_2, x_3, \cdots, x_d]/I_\Omega$ and the set of Boolean functions.

Now we are ready to define vanishing ideals. Given a set of points $S \subseteq \mathbb{K}^d$ where $\mathbb{K}$ is a field, we are interested in the set of polynomials that are zero on $S$. In the case of generalization on the unseen domain $\mathcal{U} \subseteq \Omega$, we are interested in the functions that vanish on $\mathcal{U}^c = \Omega \setminus \mathcal{U}$, as they are 0 on the training set and give a class of interpolators on $\Omega$.

**Definition C.4** (Vanishing ideals)**.** For a field $\mathbb{K}$ and $S \subseteq \mathbb{K}^d$, vanishing ideal $I(S)$ of $S$ is defined as

$$I(S) := \{f \in \mathbb{K}[x_1, \ldots, x_d]|f(x) = 0 \text{ for all } x \in S\}.$$

Note that $I_\Omega$ is indeed the vanishing ideal of $\Omega$, i.e., $I(\Omega) = I_\Omega = \langle x_1^2 - 1, x_2^2 - 1, \cdots, x_d^2 - 1\rangle$. Furthermore, for any $S \subseteq \{\pm 1\}^d$, we have $I_\Omega \subseteq I(S)$ and thus $I(S)$ can be written as $I(S) = \langle v_1, v_2, \cdots, v_n\rangle + I_\Omega$ for some $n \in \mathbb{N}$ and Boolean functions $v_i$. For example, consider canonical holdout $\mathcal{U} = \{x \in \{\pm 1\}^d | x_1 = -1\}$; in this case we get $I(\mathcal{U}^c) = \langle x_1 - 1\rangle + I_\Omega$. We could also do an ‘inverse operation’: given an ideal or set of functions, find all the points which are zero under the elements of the ideal.

**Definition C.5.** For a field $\mathbb{K}$ and $G \subseteq R = \mathbb{K}[x_1, \ldots, x_d]$, and $I = \langle G \rangle$, we define $V(G) = V(I)$ as

$$V(G) = V(I) = \{x \in \mathbb{K}^d | f(x) = 0 \text{ for all } f \in I\}.$$

Therefore, operations $V$ and $I$ give us a way to transfer some algebraic properties to geometric properties. What we defined could be seen as part of the theory of classical algebraic geometry. In algebraic geometry, we are interested in the following type of sets $S$:

**Definition C.6.** A set $S \subseteq \mathbb{K}^d$ is called an affine variety, if there exists some ideal $I$ such that $V(I) = S$.

In our case, all the $S$ are affine varieties as they are finite. For more details about algebraic geometry, please refer to (Cox et al., 2013).

Now given an $S \subseteq \Omega$, the following lemma gives us a recipe to find $I(S)$.

**Lemma C.7.** *For S and W two affine varieties, we have that $I(S \cup W) = I(S) \cap I(W)$. Also, for $x = (i_1, i_2, \cdots i_d) \in \mathbb{K}^d$, we have that $I(x) = m_x = \langle x_1 - i_1, x_2 - i_2, \cdots x_d - i_d \rangle$, where $m_x$ is a maximal ideal.*

Since in our case $S$ is finite, one can apply this lemma a multitude of times to find $I(S)$. Moreover, this ideal only vanishes on the elements of $S$ and not on any other element in $\Omega \setminus S$.

**Example C.8.** Suppose we work with $d = 2$, and we only allow the set $V := \{(-1, -1), (1, 1)\}$. We will have that $I(V) := \langle x_1 + 1, x_2 + 1 \rangle \cap \langle x_1 - 1, x_2 - 1 \rangle$. By doing the calculations or using an algebra program (e.g., SageMath) we find that

$$I(V) := \langle x_1 - x_2 \rangle + I_\Omega.$$

So in general, for a certain $S \subseteq \Omega$, we would like to express, $I(S)$ as $\langle v_1, \ldots v_n \rangle + I_\Omega$ for some desirable Boolean functions $v_1, v_2, v_3, \ldots, v_n$. In fact, there are known algorithms that find a basis for an ideal (Möller & Buchberger, 1982).

Before relating what we have defined to unseen domains, note that in our case the conditions only depend on a subset of the variables. Without loss of generality, suppose our conditions only depend on the first $k$ coordinates. Mathematically, that means $\mathcal{U} = \mathcal{U}_k \times \{-1, 1\}^{d-k}$, where $\mathcal{U}_k \subseteq \{-1, 1\}^k$. Hence, we have $\mathcal{U}^c = \mathcal{U}_k^c \times \{-1, 1\}^{d-k}$. The following lemma allows us to calculate $I(\mathcal{U}^c)$ based on $I(\mathcal{U}_k^c)$.

**Lemma C.9.** *Suppose $\mathcal{U}^c = \mathcal{U}_k^c \times \{-1, 1\}^{d-k}$ for some $\mathcal{U}_k^c \subseteq \{-1, 1\}^k$, if $I(\mathcal{U}_k^c) = \langle v_1, v_2, \cdots, v_n \rangle + \langle x_1^2 - 1, \cdots + x_k^2 - 1 \rangle$ for Boolean functions $v_1, v_2, \cdots, v_n$, we have*

$$I(\mathcal{U}^c) = \langle v_1, v_2, \cdots, v_n \rangle + I_\Omega.$$

Now having defined the vanishing ideals and quotients, we explain how they relate to our setting. In our setting, we are given $\mathcal{U} \subset \Omega = \{-1, 1\}^d$ representing the unseen domain, and a Boolean function $f$ that we wish to learn, which could be seen as an element of $R = \mathbb{R}[x_1, \ldots, x_d]$. As we finish training, we will converge to a solution $f_{\text{sol}}$ which is an interpolator of $f$ on $\mathcal{U}^c$. This means that $f - f_{\text{sol}}$ vanishes on $\mathcal{U}^c$ and so $f - f_{\text{sol}} \in I(\mathcal{U}^c)$. Hence, $f + I(\mathcal{U}^c) = f_{\text{sol}} + I(\mathcal{U}^c)$, which means that $f_{\text{sol}}$ is a representative of the class $f + I(\mathcal{U}^c)$ in the ring $R/I(\mathcal{U}^c)$. Here we are interested in the minimum degree-profile interpolator, and our goal is to classify given a $f$ and $\mathcal{U}$, the minimum degree-profile representatives of $f + I(\mathcal{U}^c)$ in the ring $R/I(\mathcal{U}^c)$. This gives us an overview of how our settings can be related to algebraic notions.

### C.1. Minimum degree-profile interpolators

We are generally interested to find the minimum degree-profile interpolators. One way to do this is as follows: given a Boolean function $f : \{-1, 1\}^d \to \mathbb{R}$ which we suppose depends only on variables $x_1, \ldots, x_P$ for some integer $P$ and an unseen set $\mathcal{U} \subseteq \Omega$, we find Boolean functions $v_1, \ldots, v_n$ which only depend on the variables $x_1, \ldots, x_P$ such that $I(\Omega \setminus \mathcal{U}) = \langle v_1, v_2, \ldots, v_n \rangle + I_\Omega$. We know that minimum degree interpolator $f_{\text{MDI}}$ is of the form

$$f_{\text{MDI}} = f + g_1 v_1 + \ldots g_n v_n,$$

for some Boolean functions $g_1, \ldots, g_n$. Now note that if we look at the equation above through the lens of Fourier-Walsh expansion, we realize that coefficients of $f_{\text{MDI}}$ are linear combinations of Fourier coefficients of $g_1, \ldots, g_n$. One can utilize this structure to minimize the elements of the degree-profile one by one since each element of the degree-profile is a quadratic expression in Fourier coefficients of $g_1, \ldots, g_n$. Therefore, one can solve these second-degree optimization problems to calculate the unique MD interpolator.

The process presented above is quite long, but there are some cases for which it is easier to find the minimum degree-profile interpolator. We will present some examples below.

**Example C.10** (Generalized canonical holdout)**.** Given a point in $\{-1,1\}^k$ i.e., $i = (i_1, \cdots, i_k) \in \{-1,-1\}^k$, for $\mathcal{U} = (\{-1,1\}^k \setminus \{i\}) \times \{-1,1\}^{d-k}$ and for any Boolean function $f$, the minimum degree-profile interpolator can be found as follows: we first notice that $I(\Omega \setminus \mathcal{U}) = \langle x_1 - i_1, \cdots, x_k - i_k \rangle + I_\Omega$. And so given $f$, the minimum degree-profile interpolator corresponds to $f_{\mathrm{MDI}}(x_1, \cdots x_k, x_{k+1}, \cdots x_d) = f(i_1, \cdots i_k, x_{k+1}, \cdots x_d)$.

**Example C.11.** For $\mathcal{U} = \{(-1,-1), (1,1)\} \times \{-1,1\}^{d-2}$ and for any Boolean function $f$, the MD interpolator can be computed by noticing that $I(\Omega \setminus \mathcal{U}) = \langle x_1 + x_2 \rangle + I_\Omega$. Hence, given an $f$ and in order to find the MD interpolator one should replace any $x_1$ found by $\frac{1}{2}(x_1 - x_2)$ and all $x_2$ by $\frac{1}{2}(x_2 - x_1)$.

Here is another case where it is easy to find the MD interpolator. We further present a proof for why it is the MD interpolator in this case.

**Lemma C.12.** *Let $i = (i_1, \cdots i_k) \in \{-1,1\}^k$ be any point. For any Boolean function $f$ and $\mathcal{U} = i \times \{-1,1\}^{d-k}$, we have that $f$ has a minimum degree-profile interpolator given by replacing all $x_1 x_2 \cdots x_k$ found by another function $g'(x_1, \ldots, x_k)$ which can be determined.*

This is an expected result, we provide nonetheless a formal proof.

*Proof.* We have $I = I(\Omega \setminus \mathcal{U}) = \langle (x_1 - i_1)(x_2 - i_2) \cdots (x_k - i_k) \rangle + I_\Omega$. By expanding $(x_1 - i_1)(x_2 - i_2) \cdots (x_k - i_k)$, we get an expression of the form

$$x_1 x_2 \cdots x_k + g(x_1, x_2, \ldots, x_k)$$

with $g(x_1, x_2, \ldots, x_k)$ containing all the possible monomials consisting of $x_1, \ldots, x_k$ of degree strictly less than $k$ with coefficients being 1 or $-1$. Consider the polynomial $f_{\mathrm{MDI}}$, by replacing all the $x_1 x_2 \cdots x_k p(x_{k+1}, \cdots x_d)$ that appears in $f$ by $-g(x_1, x_2, \cdots, x_k) p(x_{k+1}, \cdots x_d)$. We claim that $f_{\mathrm{MDI}}$ is the minimum degree-profile interpolator. In fact, suppose that this is not the case, so there exists a polynomial $q$ such that

$$(f_{\mathrm{MDI}} + q(x_1 - i_1)(x_2 - i_2) \cdots (x_k - i_k)) \text{ modulo } I_\Omega$$

is not equal to and has a lower degree-profile than $f_{\mathrm{MDI}}$. For this to happen, we need at least one monomial of $f_{\mathrm{MDI}}$ to be (partly) replaced by the same degree or lower degree alternatives. Among all such monomials, we consider the highest degree one, $\chi_M = \prod_{i \in M} x_i$. We assume that $M = T \cup R$ such that $T \subseteq [k]$ and $R \cap [k] = \emptyset$. Note that monomials that contained $x_1 x_2 \cdots x_k$ are already replaced, hence, $T \neq [k]$. We write $q$ in the form of $q(x) = s(x_1, \ldots, x_k)\chi_R + q'(x)$ where $q'(x)$ does not contain any monomial of the form $\chi_R \chi_{T'}$ for $T' \subseteq [k]$. Note that by our assumption $q(x)(x_1 - i_1) \cdots (x_k - i_k)$, and thus, $s(x_1, \ldots, x_k)\chi_R(x_1 - i_1) \cdots (x_k - i_k)$ must have generated $\beta \chi_M$, for some $\beta \neq 0$. Note that $\chi_M$ is the highest degree monomial (partly) replaced by $q$. Thus, $\chi_{[k] \cup R}$, is not be generated by $q$, otherwise the degree-profile would have been increased. In other words, $s(x_1, \ldots, x_k)\chi_R(x_1 - i_1) \cdots (x_k - i_k)$ must have generated $\beta \chi_M$ ($\beta \neq 0$) and not $\chi_{[k] \cup R}$. We now show that such thing is impossible and reach a contradiction. Assume that $s(x_1, \ldots, x_k) = \sum_{U \subseteq [k]} \alpha_U \chi_U$. Notice we can remove the $\chi_R$ part from the question. Thus, we can consider the equivalent statement that $s(x_1, \ldots, x_k)(x_1 - i_1) \cdots (x_k - i_k)$ does not generate $x_1 \cdots x_k$ while it generates $\beta \chi_T$. Now we compute the coefficients of $\chi_T$ and $\chi_{[k]} = x_1 \cdots x_k$ in $s(x_1, \ldots, x_k)(x_1 - i_1) \cdots (x_k - i_k)$. We have

$$\begin{aligned}
s(x)(x_1 - i_1) \cdots (x_k - i_k) &= s(x)\Big(\sum_{V \subseteq [k]} (-1)^{k-|V|} \chi_V \prod_{n \in [k] \setminus V} i_n\Big) \\
&= \Big(\sum_{U \subseteq [k]} \alpha_U \chi_U\Big)\Big(\sum_{V \subseteq [k]} (-1)^{k-|V|} \chi_V \prod_{n \in [k] \setminus V} i_n\Big) \\
&= \Big(\sum_{U \subseteq [k]} \alpha_U (-1)^{|U|} \prod_{n \in U} i_n\Big)\chi_{[k]} + \cdots + \Big(\sum_{U \subseteq [k]} \alpha_U (-1)^{k-|U \Delta T|} \prod_{n \in [k] \setminus (T \Delta U)} i_n\Big)\chi_M + \cdots,
\end{aligned} \tag{80}$$

and the coefficient of $\chi_M$ is $\sum_{U\subseteq[k]} \alpha_U (-1)^{k-|U\Delta T|} \prod_{n\in[k]\setminus(T\Delta U)} i_n = \beta$. Using $[k]\setminus(U\Delta T) = U\Delta([k]\setminus T)$, we have

$$\beta = \sum_{U\subseteq[k]} \alpha_U (-1)^{k-|U\Delta T|} \prod_{n\in[k]\setminus(T\Delta U)} i_n = \sum_{U\subseteq[k]} \alpha_U (-1)^{k-|U|-|T|} \prod_{n\in([k]\setminus T)\Delta U} i_n \tag{81}$$

$$= ((-1)^{k-|T|} \prod_{n\in[k]\setminus T} i_n)(\sum_{U\subseteq[k]} \alpha_U (-1)^{|U|} \prod_{n\in U} i_n) = 0, \tag{82}$$

thus, $\beta = 0$ which is a contradiction, showing that it is not possible to reduce the degree-profile. □