A Mathematical Model for Curriculum Learning for Parities

Elisabetta Cornacchia [1] **Elchanan Mossel** [2]

Abstract

Curriculum learning (CL) - training using samples that are generated and presented in a meaningful order - was introduced in the machine learning context around a decade ago. While CL has been extensively used and analysed empirically, there has been very little mathematical justification for its advantages. We introduce a CL model for learning the class of k-parities on d bits of a binary string with a neural network trained by stochastic gradient descent (SGD). We show that a wise choice of training examples involving two or more product distributions, allows to reduce significantly the computational cost of learning this class of functions, compared to learning under the uniform distribution. Furthermore, we show that for another class of functions - namely the 'Hamming mixtures' - CL strategies involving a bounded number of product distributions are not beneficial.

1. Introduction

Several experimental studies have shown that humans and animals learn considerably better if the learning materials are presented in a curated, rather than random, order (Elio & Anderson, 1984; Ross & Kennedy, 1990; Avrahami et al., 1997; Shafto et al., 2014). This is broadly reflected in the educational system of our society, where learning is guided by an highly organized curriculum. This may involve several learning steps: with easy concepts introduced at first and harder concepts built from previous stages.

Inspired by this, (Bengio et al., 2009) formalized a *curriculum learning (CL)* paradigm in the context of machine learning and showed that for various learning tasks it provided improvements in both the training speed and the performance obtained at convergence. This seminal paper inspired many subsequent works, that studied curriculum learning strategies in various application domains, e.g. computer vision (Sarafianos et al., 2017; Dong et al., 2017), computational biology (Xiong et al., 2021), auto-ML (Graves et al., 2017), natural language modelling (Shi et al., 2013; Zaremba & Sutskever, 2014; Shi et al., 2015; Campos, 2021). While extensive empirical analysis of CL strategies have been carried out, there is a lack of theoretical analysis. In this paper, we make progress in this direction.

A stylized family of functions that is known to pose computational barriers is the class of k-parities over d bits of a binary string. In this work we focus on this class. To define this class: for each subset S of coordinates, the parity over S is defined as $+1$ if the number of negative bits in S is even, and -1 otherwise, i.e. $\chi_S(x) := \prod_{i\in S} x_i$, $x_i \in \{\pm 1\}$. The class of k-parities contains all χ_S such that $|S| = k$ and it has cardinality $\binom{d}{k}$. Learning k-parities requires learning the support of χ_S by observing samples $(x, \chi_S(x))$, $x \in \{\pm 1\}^d$, with the knowledge of the cardinality of S being k. This requires finding the right target function among the $\binom{d}{k}$ functions belonging to the class.

Learning parities is always possible, and efficiently so, by specialized methods (e.g. Gaussian elimination over the field of two elements). Moreover, ((Abbe & Sandon, 2020)) showed that there exists a neural net that learns parities of any degree if trained by SGD with small batch size. However, this is a rather unconventional net. In fact, under the uniform distribution, parities are not efficiently learnable by population queries with any polynomially small noise. The latter can be explained as follows. Assume we sample our binary string uniformly at random, i.e. for each $i \in \{1, ..., d\}$, $x_i \sim \mathrm{Rad}(1/2)$[1]. Then, the covariance between two parities $\chi_S, \chi_{S'}$ is given by:

$$\mathbb{E}_{x\sim \mathrm{Rad}(1/2)^{\otimes d}} [\chi_S(x)\chi_{S'}(x)] = \begin{cases} 1 & \text{if } S = S', \\ 0 & \text{if } S \neq S', \end{cases}$$

where $x \sim \mathrm{Rad}(1/2)^{\otimes d}$ denotes the product measure such that $x_i \overset{iid}{\sim} \mathrm{Rad}(1/2)$, $i \in \{1, ..., d\}$. More abstractly, a parity function of k bits is uncorrelated with *any* function of $k-1$ or less bits. This property makes parities hard to learn for any progressive algorithm, such as gradient descent. Indeed, when trying to learn the set of relevant

[1]Institute of Mathematics, EPFL, Lausanne, Switzerland [2]Department of Mathematics and IDSS, MIT, US. Correspondence to: Elisabetta Cornacchia <elisabetta.cornacchia@epfl.ch>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1] $z \sim \mathrm{Rad}(p)$ if $\mathbb{P}(z = 1) = 1 - \mathbb{P}(z = -1) = p$.

features, a learner cannot know how close its progressive guesses are to the true set. In other words, all wrong guesses are indistinguishable, which suggests that the learner might have to perform exhaustive search among all the $\binom{d}{k}$ sets.

The hardness of learning unbiased parities - and more in general any classes of functions with low cross-correlations - with gradient descent has been analysed e.g. in (Abbe & Sandon, 2020), where the authors show a lower bound on the computational complexity of learning low cross-correlated classes with gradient-based algorithms with bounded gradient precision. For k-parities, this gives a computational lower bound of $d^{\Omega(k)}$ for any architecture and initialization.

However, if we look at different product distributions, then the inner product of a monomial and a component x_i that is inside and outside the support becomes distinguishable. Suppose the inputs are generated as $x \sim \mathrm{Rad}(p)^{\otimes d}$, for some $p \in (0,1)$. Then the covariance between χ_S and $\chi_{S'}$ is:

$$\mathbb{E}_{x\sim \mathrm{Rad}(p)^{\otimes d}}\Big[(\chi_S(x) - \mathbb{E}[\chi_S(x)]) \cdot (\chi_{S'}(x) - \mathbb{E}[\chi_{S'}(x)])\Big] = \mu_p^{2k-|S\cap S'|} - \mu_p^{2k},$$

where we denoted by $\mu_p := \mathbb{E}_{z\sim \mathrm{Rad}(p)}[z] = 2p-1$. This implies that if for instance $|p-0.5| > 0.1$, just computing correlations with each bit, will recover the parity with complexity linear in d and exponential in k. If we choose $p = 1 - 1/k$, say, we can get a complexity that is linear in d and polynomial in k. Moreover, the statements above hold even for parities with random noise.

This may lead one to believe that learning biased parities is easy for gradient descent based methods for deep nets. Indeed, (Malach et al., 2021) showed that biased parities are learnable by SGD on a differentiable model consisting of a linear predictor and a fixed module implementing the parity. However, if we consider fully connected networks, as our experiments show (Figure 1), while gradient descent for a p far from a half converges efficiently to zero training loss, the learned function actually has *non-negligible error* when computed with respect to the uniform measure. This is intuitively related to the fact that, by concentration of measure, there are essentially no examples with Hamming weight[2] close to $d/2$ in the training set sampled under $\mathrm{Rad}(p)^{\otimes d}$, and therefore it is not reasonable to expect for a general algorithm like gradient descent on fully connected networks (that does not know that the target function is a parity) to learn the value of the function on such inputs.

We thus propose a more subtle question: Is it possible to generate examples from different product distributions and present them in a specific order, in such a way that the error with respect to the unbiased measure becomes negligible?

[2]The Hamming weight of $x \in \{\pm 1\}^d$ is: $H(x) = \sum_{i=1}^d 1(x_i = 1)$.

As we mentioned, training on examples sampled from a biased measure is not sufficient to learn the parity under the unbiased measure. However, it does identify the support of the parity. Our curriculum learning strategy is the following: We initially train on inputs sampled from $\mathrm{Rad}(p)^{\otimes d}$ with p close to 1, then we move (either gradually or by sharp steps) towards the unbiased distribution $\mathrm{Rad}(1/2)^{\otimes d}$. We show that this strategy allows to learn the k-parity problem with a computational cost of $d^{O(1)}$ with SGD on the hinge loss or on the *covariance loss* (see Def. 3.2). In our proof, we consider layer-wise training (similarly to e.g. (Malach et al., 2021; Malach & Shalev-Shwartz, 2020; Barak et al., 2022)) and the result is valid for any (even) k and d.

As we mentioned earlier, the failure of learning parities under the uniform distribution from samples coming from a different product measure is due to concentration of Hamming weight. This leads us to consider a family of functions that we call *Hamming mixtures*. Given an input x, the output of a Hamming mixture is a parity of a subset S of the coordinates, where the subset S depends on the Hamming weight of x (see Def. 2.4). Our intuition is based on the fact that given a polynomial number of samples from, say, the $p = 1/4$ biased measure, it is impossible to distinguish between a certain parity χ_S and a function that is χ_S, for x's whose Hamming weight is at most $3/8d$, and a different function χ_T, for x's whose Hamming weight is more than $3/8d$, for some T that is disjoint from S. In other words, a general algorithm does not know whether there is consistency between x's with different Hamming weight. We show a lower bound for learning Hamming mixtures with curriculum strategies that do not allow to get enough samples with relevant Hamming weight.

Of course, curriculum learning strategies with enough learning steps allow to obtain samples from several product distributions, and thus with all relevant Hamming weights. Therefore, we expect that CL strategies with unboundedly many learning steps will be able to learn the Hamming mixtures.

While our results are restricted to a limited and stylized setting, we believe they may open new research directions. Indeed, we believe that our general idea of introducing correlation among subsets of the input coordinates to facilitate learning, may apply to more general settings. We discuss some of these future directions in the conclusion section of the paper.

Importantly, we remark that a limitation of the curriculum strategy presented in this paper is that it requires an oracle that provides labeled samples from arbitrary product measures. However, in applications one usually has a fixed dataset and would like to select samples in a suitable order, to facilitate learning. We leave to future work the analysis of a setting where curriculum and non-curriculum have a common sampling distribution.

Contributions. Our contributions are the following.

1. We propose and formalize a mathematical model for curriculum learning;
2. We prove that our curriculum strategy allows to learn k-parities with SGD with the hinge loss or with the covariance loss on a two-layers fully connected network with a computational cost of $d^{O(1)}$;
3. We empirically verify the effectiveness of our curriculum strategy for a set of fully connected architectures and parameters;
4. We propose a class of functions - the *Hamming mixtures* - that is provably not learnable by some curriculum strategies with finitely many learning steps. We conjecture that a *continuous* curriculum strategy (see Def. 2.6) may allow to significantly improve the performance for learning such class of functions.

1.1. Related Work

Learning parities on uniform inputs. Learning k-parities over d bits requires determining the set of relevant features among $\binom{d}{k}$ possible sets. The statistical complexity of this problem is thus $\theta(k\log(d))$. The computational complexity is harder to determine. k-parities can be solved in $d^{O(1)}$ time by specialized algorithms (e.g. Gaussian elimination) that have access to at least d samples. In the statistical query (SQ) framework (Kearns, 1998) - i.e. when the learner has access only to noisy queries over the input distribution - k-parities cannot be learned in less then $\Omega(d^k)$ computations. (Abbe & Sandon, 2020; Shalev-Shwartz et al., 2017) showed that gradient-based methods suffer from the same SQ computational lower bound if the gradient precision is not good enough. On the other hand, (Abbe & Sandon, 2020) showed that one can construct a very specific network architecture and initialization that can learn parities beyond this limit. This architecture is however far from the architectures used in practice. (Barak et al., 2022) showed that SGD can learn sparse k-parities with SGD with batch size $d^{\theta(k)}$ on a small network. Moreover, they empirically provide evidence of 'hidden progress' during training, ruling out the hypothesis of SGD doing random search. (Andoni et al., 2014) showed that parities are learnable by a $d^{\theta(k)}$ network. The problem of learning *noisy* parities (even with small noise) is conjectured to be intrinsically computationally hard, even beyond SQ models (Alekhnovich, 2003).

Learning parities on non-uniform inputs. Several works showed that when the input distribution is not the $\mathrm{Unif}\{\pm1\}^d$, then neural networks trained by gradient-based methods can efficiently learn parities. (Malach et al., 2021) showed that biased parities are learnable by SGD on a differentiable model consisting of a linear predictor and fixed module implementing the parity. (Daniely & Malach, 2020) showed that sparse parities are learnable on a two layers network if the input coordinates outside the support of the parity are uniformly sampled and the coordinates inside the support are correlated. To the best of our knowledge, none of these works propose a curriculum learning model to learn parities under the uniform distribution.

Curriculum learning. *Curriculum Learning (CL)* in the context of machine learning has been extensively analysed from the empirical point of view (Bengio et al., 2009; Wang et al., 2021; Soviany et al., 2022). However, theoretical works on CL seem to be more scarce. In (Saglietti et al., 2022) the authors propose an analytical model for CL for functions depending on a sparse set of relevant features. In their model, easy samples have low variance on the irrelevant features, while hard samples have large variance on the irrelevant features. In contrast, our model does not require knowledge of the target task to select easy examples. In (Weinshall et al., 2018; Weinshall & Amir, 2020) the authors analyse curriculum learning strategies in convex models and show an improvement on the speed of convergence of SGD. In contrast, our work covers an intrinsically non-convex problem. Some works also analysed variants of CL: e.g. self-paced CL (SPCL), i.e. curriculum is determined by both prior knowledge and the training process (Jiang et al., 2015), implicit curriculum, i.e. neural networks tend to consistently learn the samples in a certain order (Toneva et al., 2018). To a different purpose, (Abbe et al., 2021a; 2022a) analyse staircase functions - sum of nested monomials of increasing degree - and show that the hierarchical structure of such tasks guides SGD to learn high degree monomials. Moreover, (Refinetti et al., 2022; Kalimeris et al., 2019) show that SGD learns functions of increasing complexity during training. In a concurrent work (Abbe et al., 2023), the authors propose a curriculum learning algorithm (named 'Degree Curriculum') that consists of training on Boolean inputs of increasing Hamming weight, and they empirically show that it reduces the sample complexity of learning parities on small input dimension. However, the paper does not include a theoretical analysis of such curriculum.

2. Definitions and Main Results

We define a curriculum strategy for learning a general Boolean target function. We will subsequently restrict our attention to the problem of learning parities or mixtures of parities. For brevity, we denote $[d] = \{1, ..., d\}$. Assume that the network is presented with samples $(x, f(x))$, where $x \in \{\pm1\}^d$ is a Boolean vector and $f : \{\pm1\}^d \to \mathbb{R}$ is a target function that generates the labels. We consider a neural network $\mathrm{NN}(x;\theta)$, whose parameters are initialized at random from an initial distribution P_0, and trained by

stochastic gradient descent (SGD) algorithm, defined by:

$$\theta^{t+1} = \theta^t - \gamma_t \frac{1}{B} \sum_{i=1}^{B} \nabla_{\theta^t} L(\theta^t, f, x_i^t), \tag{1}$$

for all $t \in \{0, ..., T-1\}$, where L is an almost surely differentiable loss-function, γ_t is the learning rate, B is the batch size and T is the total number of training steps. For brevity, we write $L(\theta^t, f, x) := L(\mathrm{NN}(.; \theta^t), f, x)$. We assume that for all $i \in [B]$, $x_i^t \overset{iid}{\sim} \mathcal{D}^t$, where $\mathcal{D}^t$ is a step-dependent input distribution supported on $\{\pm 1\}^d$. We define our curriculum learning strategy as follows. Recall that $z \sim \mathrm{Rad}(p)$ if $\mathbb{P}(z = 1) = 1 - \mathbb{P}(z = -1) = p$.

Definition 2.1 (r-steps curriculum learning (r-CL))**.** For a fixed $r \in \mathbb{N}$, let $T_1, ...T_r \in \mathbb{N}$ and $p_1, ..., p_r \in [0, 1]$. Denote by $\bar{p} := (p_1, ..., p_r)$ and $\bar{T} := (T_1, ..., T_{r-1})$. We say that a neural network $\mathrm{NN}(x; \theta^t)$ is trained by SGD with a r-CL$(\bar{T}, \bar{p})$ if θ^t follows the iterations in (1) with:

$$\begin{aligned}
&\mathcal{D}^t = \mathrm{Rad}(p_1), && 0 < t \le T_1,\\
&\mathcal{D}^t = \mathrm{Rad}(p_2), && T_1 < t \le T_2,\\
&\cdots\\
&\mathcal{D}^t = \mathrm{Rad}(p_r), && T_{r-1} < t \le T.
\end{aligned}$$

We say that r is the number of *curriculum steps*.

We assume r to be independent on T, in order to distinguish the r-CL from the *continuous*-CL (see Def. 2.6 below). We hypothesize that r-CL may help to learn several Boolean functions, if one chooses appropriate r and $\bar{p}$. However, in this paper we focus on the problem of learning unbiased k-parities. For such class, we obtained that choosing $r = 2$, a wise $p_1 \in (0, 1/2)$ and $p_2 = 1/2$ brings a remarkable gain in the computational complexity, compared to the standard setting with no curriculum. An interesting future direction would be studying the optimal r and $\bar{p}$. Before stating our Theorem, let us clarify the generalization error that we are interested in. As mentioned before, we are interested in learning the target over the uniform input distribution.

Definition 2.2 (Generalization error)**.** We say that SGD on a neural network $\mathrm{NN}(x; \theta)$ learns a target function $f : \{\pm 1\}^d \to \mathbb{R}$ with r-CL$(\bar{T}, \bar{p})$ up to error ϵ, if it outputs a network $\mathrm{NN}(x; \theta^T)$ such that:

$$\mathbb{E}_{x \sim \mathrm{Rad}(1/2)^{\otimes d}} \left[L(\theta^T, f, x) \right] \le \epsilon, \tag{2}$$

where L is any loss function such that $\mathbb{E}_{x \sim \mathrm{Rad}(1/2)^{\otimes d}}[L(f, f, x)] = 0$.

We state here our main theoretical result informally. We refer to Section 3.1 for the formal statement with exact exponents and remarks.

Theorem 2.3 (Main positive result, informal)**.** *There exists a 2-CL strategy such that a 2-layer fully connected network of $d^{O(1)}$ size trained by SGD with batch size $d^{O(1)}$ can learn any k-parities (for k even) up to error ϵ in at most $d^{O(1)}/\epsilon^2$ iterations.*

Let us analyse the computational complexity of the above. At each step, the number of computations performed by a 2-layer fully connected network is given by:

$$(dN + N) \cdot B, \tag{3}$$

where d is the input size, N is the number of hidden neurons and B is the batch size. Multiplying by the total number of steps and substituting the bounds from the Theorem we get that we can learn the k-parity problem with a 2-CL strategy in at most $d^{O(1)}$ total computations. Specifically, $O(1)$ denotes quantities that do not depend on k or on d, and the statement holds also for large k, d. We prove the Theorem in two slightly different settings, see Section 3.1.

One may ask whether the r-CL strategy is beneficial for learning general target tasks (i.e. beyond parities). While we do not have a complete picture to answer this question, we propose a class of functions for which some r-CL strategies are not beneficial. We call those functions the *Hamming mixtures*, and we define them as follows.

Definition 2.4 ((S,T,ϵ)-Hamming mixture)**.** For $\epsilon \in [0, 1]$, $S, T \in [d]$, we say that $G_{S,T,\epsilon} : \{\pm 1\}^d \to \mathbb{R}$ is a (S,T,ϵ)-Hamming mixture if

$$G_{S,T,\epsilon}(x) := \chi_S(x) 1(H(x) \le \epsilon d) + \chi_T(x) 1(H(x) > \epsilon d),$$

where $H(x) := \sum_{i=1}^{d} 1(x_i = 1)$ is the Hamming weight of x, $\chi_S(x) := \prod_{i \in S} x_i$ and $\chi_T(x) := \prod_{i \in T} x_i$ are the parity functions over set S and T respectively.

The intuition of why such functions are hard for some r-CL strategies is the following. Assume we train on samples $(x, G_{S,T,\epsilon}(x))$, with S, T disjoint and $\epsilon \in (0, 1/2)$. Assume that we use a 2-CL strategy and we initially train on samples $x \sim \mathrm{Rad}(p)^{\otimes d}$ for some $p < \epsilon$. If the input dimension d is large, then the Hamming weight of x is with high probability concentrated around pd (e.g. by Hoeffding's inequality). Thus, in the first part of training the network will see, with high probability, only samples of the type $(x, \chi_S(x))$, and it will not see the second addend of $G_{S,T,\epsilon}$. When we change our input distribution to $\mathrm{Rad}(1/2)^{\otimes d}$, the network will suddenly observe samples of the type $(x, \chi_T(x))$. Thus, the pre-training on p will not help determining the support of the new parity χ_T (in some sense the network will "forget" the first part of training). This intuition holds for all r-CL such that $p_1, ..., p_{r-1} < \epsilon$. We state our negative result for Hamming mixtures here informally, and refer to Section 4 for a formal statement and remarks.

Theorem 2.5 (Main negative result, informal)**.** *For each r-CL strategy with r bounded, there exists a Hamming mixture*

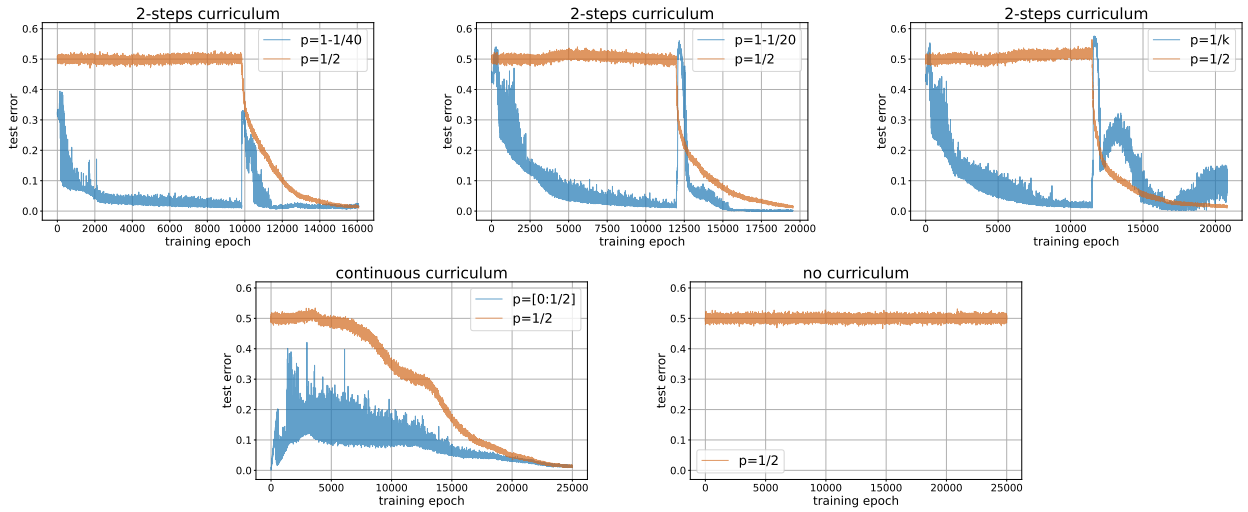

Figure 1. Learning 20-parities with 2-steps curriculum, with initial bias $p_1 = 39/40$ (top-left), $p_1 = 19/20$ (top-center), $p_1 = 1/20$ (top-right), with continuous curriculum (bottom-left) and with no curriculum (bottom-right). In all plots, we use a 2-layers ReLU MLP with batch size 1024, input dimension 100, and 100 hidden units.

$G_{S,T,\epsilon}$ that is not learnable by any fully connected neural network of $\mathrm{poly}(d)$ *size and permutation-invariant initialization trained by the noisy gradient descent algorithm (see Def. 4.1) with* $\mathrm{poly}(d)$ *gradient precision in* $\mathrm{poly}(d)$ *steps.*

Inspired by the hardness of Hamming mixtures, we define another curriculum learning strategy, where, instead of having finitely many discrete curriculum steps, we gradually move the bias of the input distribution during training from a starting point p_0 to a final point p_T. We call this strategy a *continuous*-CL strategy.

Definition 2.6 (Continuous curriculum learning (C-CL))**.** Let $p_0, p_T \in [0,1]$. We say that a neural network $\mathrm{NN}(x;\theta^t)$ is trained by SGD with a C-CL(p_0, p_T, T) if θ^t follows the iterations in (1) with:

$$\mathcal{D}^t = \mathrm{Rad}\left(p_0 + t \cdot \frac{p_T - p_0}{T}\right) \qquad t \in [T]. \tag{4}$$

We conjecture that a well chosen C-CL might be beneficial for learning any Hamming mixture. A positive result for C-CL and comparison between r-CL and C-CL are left for future work.

3. Learning Parities

3.1. Theoretical Results

Our goal is to show that the curriculum strategy that we propose allows to learn k-parities with a computational complexity of $d^{O(1)}$. We prove two different results. In the first one, we consider SGD on the hinge loss and prove that a network with $\theta(d^2)$ hidden units can learn the k-parity problem in $d^{O(1)}$ computations, if trained with a well chosen 2-CL strategy. Let us state our first Theorem.

Theorem 3.1 (Hinge Loss)**.** *Let k, d be both even integers, such that $k \le d/2$. Let $\mathrm{NN}(x;\theta) = \sum_{i=1}^{N} a_i \sigma(w_i x + b_i)$ be a 2-layers fully connected network with activation $\sigma(y) := \mathrm{Ramp}(y)$ (as defined in (9)) and $N = \tilde{\theta}(d^2 \log(1/\delta))$[3]. Consider training $\mathrm{NN}(x;\theta)$ with SGD on the hinge loss with batch size $B = \tilde{\theta}(d^{10}/\epsilon^2 \log(1/\delta))$. Then, there exists an initialization, a learning rate schedule, and a 2-CL strategy such that after $T = \tilde{\theta}(d^6/\epsilon^2)$ iterations, with probability $1 - 3\delta$, SGD outputs a network with generalization error at most ϵ.*

For our second Theorem, we consider another loss function, that is convenient for the analysis, namely the *covariance loss*, for which we give a definition here.

Definition 3.2 (Covariance loss)**.** Let $f : \mathcal{X} \to \mathbb{R}$ be a target function and let $\hat{f} : \mathcal{X} \to \mathbb{R}$ be an estimator. Let

$$\mathrm{cov}(f, \hat{f}, x, P_{\mathcal{X}}) := \\ := \left(f(x) - \mathbb{E}_{x' \sim P_{\mathcal{X}}}[f(x')]\right) \cdot \left(\hat{f}(x) - \mathbb{E}_{x' \sim P_{\mathcal{X}}}[\hat{f}(x')]\right),$$

where $P_{\mathcal{X}}$ is an input distribution supported in $\mathcal{X}$. We define the covariance loss as

$$L_{\mathrm{cov}}(f, \hat{f}, x, P_{\mathcal{X}}) := \max\{0, 1 - \mathrm{cov}(f, \hat{f}, x, P_{\mathcal{X}})\}.$$

Remark 3.3. We will consider optimization over the covariance loss through SGD with large batch size ($B = \tilde{\theta}(d^2 k^3)$). At each step, we use the batch to estimate first the inner expectations (i.e. $E_x[f(x)]$ and $E_x[\mathrm{NN}(x;\theta^t)]$) and then the

[3] $\tilde{\theta}(d^c) = \theta(d^c \cdot \mathrm{poly}(\log(d)))$, for all $c \in \mathbb{R}$.

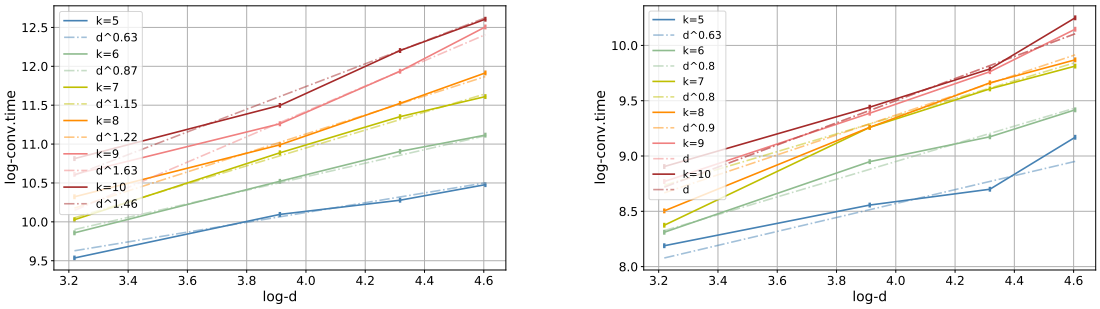

Figure 2. Convergence time for different values of d, k. Left: we take $p_1 = 1/16$ and a 2-layers ReLU architecture with with $h = 2^k$ hidden units. Right: we take $p_1 = 1 - \frac{1}{2k}$ and a 2-layers ReLU architecture with $h = d$ hidden units.

gradients. The expectation of the labels (i.e. $E_x[f(x)]$) does not need to be estimated at each training step and could be estimated once per curriculum step. One could also use part of the batch at each step to estimate the inner expectations and part of the batch to estimate the gradients.

We show that SGD on the covariance loss can learn the k-parity problem in $d^{O(1)}$ computations using a network with only $O(k)$ hidden units. The reduction of the size of the network, compared to the hinge loss case, allows to get a tighter bound on the computational cost, see Remark 3.5.

Theorem 3.4 (Covariance Loss)**.** *Let k, d be integers, such that $k \leq d$ and k even. Let $\mathrm{NN}(x;\theta) = \sum_{i=1}^{N} a_i \sigma(w_i x + b_i)$ be a 2-layers fully connected network with activation $\sigma(y) := \mathrm{ReLU}(y)$ and $N = \tilde{\theta}(k)$. Consider training $\mathrm{NN}(x;\theta)$ with SGD on the covariance loss with batch size $B = \tilde{\theta}(d^2 k^3/\epsilon^2 \log(1/\delta))$. Then, there exists an initialization, a learning rate schedule, and a 2-CL strategy such that after $T = \tilde{\theta}(k^4/\epsilon^2)$ iterations, with probability $1-3\delta$, SGD outputs a network with generalization error at most ϵ.*

The proofs of Theorem 3.1 and Theorem 3.4 follow a similar outline. Firstly, we prove that training the first layer of the network on one batch of size $d^{O(1)}$ sampled from a biased input distribution (with appropriate bias), allows to recover the support of the parity. We then show that training the second layer on the uniform distribution allows to achieve the desired generalization error under the uniform distribution. We refer to Appendices A and B for restatements of the Theorems and their full proofs.

Remark 3.5. Let us look at the computational complexity given by the two Theorems. Theorem 3.1 tells that we can learn k-parities in $dNB + (T-1)N = \tilde{\theta}(d^{19})$ computations. We remark that our result holds also for large k (we however need to assume k, d even and $k \leq d/2$, for technical reasons). On the other hand, Theorem 3.4 tells that we can learn k-parities in $\tilde{\theta}(d^3 k^8)$, which is much lower than the bound given by Theorem 3.1. Furthermore, the proof holds for all $k \leq d$. The price for getting this tighter bound is the use of a loss that (to the best of our knowledge) is not common in the machine learning literature, and that is particularly convenient for our analysis.

Remark 3.6. We remark that our proofs extend to the gradient descent model with bounded gradient precision, used in (Abbe & Sandon, 2020), with gradient precision bounded by $d^{O(1)}$. Thus, for large k, d, our result provides a separation to their $d^{\Omega(k)}$ computational lower bound for learning k-parities under the uniform distribution with no curriculum.

Remark 3.7. Let us comment on the p_1 (i.e. the bias of the initial distribution) that we used. In both Theorems we take p_1 close to 1. In Theorem 3.1 we take $p_1 \approx 1 - \theta(1/d)$, and the proof is constructed specifically for this value of p_1. In Theorem 3.4, the proof holds for any $p_1 \in (1/2, 1)$ and the asymptotic complexity in d does not depend on the specific choice of p_1. However, to get $\mathrm{poly}(k)$ complexity we need to take $p_1 = 1 - \theta(1/k)$, while we get $\exp(k)$ complexity for all $p_1 = \theta_{d,k}(1)$.

Our theoretical analysis captures a fairly restricted setting: in our proofs we use initializations and learning schedules that are convenient for the analysis. We conduct experiments to verify the usefulness of our CL strategy in more standard settings of fully connected architectures.

3.2. Empirical Results

In all our experiments we use fully connected ReLU networks and we train them by SGD on the square loss [4].

In Figure 1, we compare different curriculum strategies for learning 20-parities over 100 bits, with a fixed architecture, i.e. a 2-layer ReLU network with 100 hidden units. We run a 2-steps curriculum strategy for 3 values of p_1, namely $p_1 = 39/40, 19/20, 1/20$. In all the 2-CL experiments we train on the biased distribution until convergence, and then we move to the uniform distribution. We observe that

[4] Code: https://github.com/ecornacchia/Curriculum-Learning-for-Parities

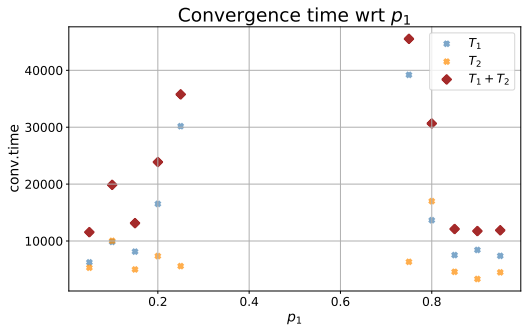

Figure 3. Convergence time with respect to the initial bias p_1. We compute the convergence time for learning a 10-parity over 100 bits with a 2-layer ReLU network. We omitted all points with convergence time above $100,000$.

training with an initial bias of $p_1 = 39/40$ allows to learn the 20-parity in $16,000$ epochs. One can see that during the first part of training (on the biased distribution), the test error under the uniform distribution stays at $1/2$ (orange line), and then drops quickly to zero when we start training on the uniform distribution. This trend of hidden progress followed by a sharp drop has been already observed in the context of learning parities with SGD in the standard setting with no-curriculum (Barak et al., 2022). Here, the length of the 'hidden progress' phase is controlled by the length of the first phase of training. Interestingly, when training with continuous curriculum, we do not have such hidden progress and the test error under the uniform distribution decreases slowly to zero. With no curriculum, the network does not achieve non-trivial correlation with the target in $25,000$ epochs.

In Figure 2 we study the convergence time of a 2-CL strategy on a 2-layers ReLU network for different values of the input dimension (d) and size of the parity (k). We take two slightly different settings. In the plot on the left, we take a fixed initial bias $p_1 = 1/16$ and $h = 2^k$ hidden units. On the right we take $p_1 = 1 - \frac{1}{2k}$ initial bias and an architecture with $h = d$ hidden units. The convergence time is computed as $T_1 + T_2$, where T_1 and T_2 are the number of steps needed to achieve training error below 0.01 in the first and second part of training, respectively. We compute the convergence time for $k = 5, 6, 7, 8, 9, 10$ and $d = 25, 50, 75, 100$, and for each k we plot the convergence time with respect to d in log-log scale. Each point is obtained by averaging over 10 runs. We observe that for each k, the convergence time scales (roughly) polynomially as d^{c_k}, with c_k varying mildly with k.

In Figure 3, we study the convergence time of a 2-CL strategy for different values of the initial bias p_1. We consider the problem of learning a 10-parity over 100 bits with a 2-layers ReLU network with $h = 100$ hidden units. As before, we computed the convergence time as $T_1 + T_2$, where T_1 and T_2 are the number of steps needed to achieve training error below 0.01 in the first and second part of training, respectively. We ran experiments for $p_1 = 0.001, 0.05, 0.1, 0.15, ..., 0.95, 0.999$. We omitted from the plot any point for which the convergence time exceeded $100,000$ iterations: these correspond to p_1 near $1/2$ and $p_1 = 0.001, 0.999$. Each point is obtained by averaging over 10 runs. We observe that the convergence time is smaller for p_1 close to 0 or to 1. Moreover, T_2 has modest variations across different p_1's.

4. Learning Hamming Mixtures

In this section we consider the class of functions defined in Def. 2.4 and named Hamming mixtures. We consider a specific descent algorithm, namely the noisy GD algorithm with batches (used also in (Abbe & Sandon, 2020; Abbe et al., 2021b)). We give a formal definition here of noisy GD with curriculum.

Definition 4.1 (Noisy GD with CL)**.** Consider a neural network $\mathrm{NN}(.;\theta)$, with initialization of the weights θ^0. Given an almost surely differentiable loss function, the updates of the noisy GD algorithm with learning rate γ_t and gradient range A are defined by

$$\theta^{t+1} = \theta^t - \gamma_t \left(\mathbb{E}_{x^t} [\nabla_{\theta^t} L(\theta^t, f, x^t)]_A + Z^t \right), \quad (5)$$

where for all $t \in \{0, ..., T-1\}$, Z^t are i.i.d. $\mathcal{N}(0, \tau^2)$, for some τ, and they are independent from other variables, $x^t \sim \mathcal{D}^t$, for some time-dependent input distribution $\mathcal{D}^t$, f is the target function, from which the labels are generated, and by $[.]_A$ we mean that whenever the argument is exceeding A (resp. $-A$) it is rounded to A (resp. $-A$). We call A/τ the *gradient precision*. In the noisy-GD algorithm with r-CL, we choose $\mathcal{D}^t$ according to Def. 2.1.

Let us state our hardness result for learning Hamming mixtures with r-CL strategies with r bounded.

Theorem 4.2. *Assume the network observes samples generated by $G_{S,T,\epsilon}(x)$ (see Def. 2.4), where $|S| = k_S$, $|T| = k_T$ such that $k_S, k_T = o(\sqrt{d})$, and $|S \cap T| = 0$. Then, for any r-CL$(\bar{T}, \bar{p})$ with r bounded and $p_r = 1/2$, there exists an ϵ such that the noisy GD algorithm with r-CL$(\bar{T}, \bar{p})$ (as in (5)) on a fully connected neural network with $|\theta|$ weights and permutation-invariant initialization, after T training steps, outputs a network $\mathrm{NN}(x, \theta^T)$ such that*

$$\left| \mathbb{E}_{x \sim \mathrm{Rad}(1/2)^{\otimes d}} \left[G_{S,T,\epsilon}(x) \cdot \mathrm{NN}(x; \theta^T) \right] \right| \leq \frac{AT\sqrt{|\theta|}}{\tau} \left(\frac{1}{d^{k_T/2}} + e^{-d\delta^2} \right) + \frac{2k_S k_T}{d} + O(d^{-2}),$$

where A, τ are the gradient range and the noise level in the noisy-GD algorithm and δ is a constant.

The proof uses an SQ-like lower bound argument for noisy GD, in a similar flavour of (Abbe et al., 2022b; Abbe & Boix-Adsera, 2022). We refer to Appendix C for the full proof.

Remark 4.3. In Theorem 4.2, the neural network can have any fully connected architecture and any activation such that the gradients are well defined almost everywhere. The initialization can be from any distribution that is invariant to permutations of the input neurons.

For the purposes of $\mathbb{E}\left[G_{S,T,\epsilon}(x)\cdot \mathrm{NN}(x;\theta^T)\right]$, it is assumed that the neural network outputs a guess in $\{\pm 1\}$. This can be done with any form of thresholding, e.g. taking the sign of the value of the output neuron.

Remark 4.4. One can remove the $\frac{2k_Sk_V}{d}$ term in the right hand side by further assuming e.g. that set S is supported on the first $d/2$ coordinates and set V on the last $d/2$ coordinates. This also allows to weaken the assumption on the cardinality of S and V. We formalize this in the following Corollary.

Corollary 4.5. *Assume the network observes samples generated by $G_{S,V,\epsilon}(x)$, where $S\subseteq\{1,...,d/2\}$, and $V\subseteq\{d/2+1,...,d\}$ (where we assumed d to be even for simplicity). Denote $k_V=|V|$. Then, for any r-CL$(\bar{T},\bar{p})$ with r bounded and $p_r=1/2$, there exists an ϵ such that the noisy GD algorithm with r-CL$(\bar{T},\bar{p})$ (as in (5)) on a fully connected neural network with $|\theta|$ weights and permutation-invariant initialization, after T training steps, outputs a network $\mathrm{NN}(x,\theta^{(T)})$ such that*

$$\left|\mathbb{E}_{x\sim\mathrm{Rad}(1/2)^{\otimes d}}\left[G_{S,V,\epsilon}(x)\cdot\mathrm{NN}(x;\theta^{(T)})\right]\right| \leq \frac{2AT\sqrt{|\theta|}}{\tau}\left(\binom{d/2}{k_V}^{-1/2}+e^{-d\delta^2}\right),$$

for some $\delta>0$.

The proof of Corollary 4.5 is deferred to Appendix D.

Theorem 4.2 states failure at the weakest form of learning, i.e. achieving correlation better than guessing in the asymptotic of large d. More specifically, it tells that if the network size, the number of training steps and the gradient precision (i.e. A/τ) are such that $\frac{AT\sqrt{|\theta|}}{\tau}=o(d^{-k_T/2})$, then the network achieves correlation with the target under the uniform distribution of $o_d(1)$. Corollary 4.6 follows immediately from the Theorem.

Corollary 4.6. *Under the assumptions of Theorem 4.2, if $k_T=\omega_d(1)$ (i.e. k_T grows with d), $|\theta|, A/\tau, T$ are all polynomially bounded in d, then*

$$\left|\mathbb{E}_{x\sim\mathrm{Rad}(1/2)^{\otimes d}}\left[G_{S,T,\epsilon}(x)\cdot\mathrm{NN}(x;\theta^T)\right]\right|=o_d(1), \quad (6)$$

i.e. in $\mathrm{poly}(d)$ computations the network will fail at weak-learning $G_{S,T,\epsilon}$.

We conjecture that if we take instead a C-CL strategy with an unbounded number of curriculum steps, we can learn efficiently (i.e. in $\mathrm{poly}(d)$ time) any $G_{S,T,\epsilon}$ (even with $k_T=\omega_d(1)$ and for any ϵ). Furthermore, we believe this conjecture to hold for any bounded mixture, i.e. any function of the type:

$$\sum_{m=1}^{M}\chi_{S_m}(x)1(\epsilon_{m-1}d\leq H(x)<\epsilon_m d), \quad (7)$$

with $S_1,...,S_M$ being distinct sets of coordinates, $0=\epsilon_0<\epsilon_1...<\epsilon_M\leq 1$, and M bounded.

5. Conclusion and Future Work

In this work, we mainly focused on learning parities and Hamming mixtures with r-CL strategies with bounded r. Some natural questions arise, for instance: does the depth of the network help? What is the optimal number of curriculum steps for learning parities? We leave to future work the analysis of C-CL with unboundedly many curriculum steps and the comparison between r-CL and C-CL. In the previous Section, we also raised a conjecture concerning the specific case of Hamming mixtures.

Furthermore, we believe that our results can be extended to more general families of functions. First, consider the set of k-Juntas, i.e., the set of functions that depend on k out of d coordinates. This set of functions contains the set of k-parities so it is at least as hard to learn. Moreover, as in the case of parities, Juntas are correlated with each of their inputs for generic p, see e.g. (Mossel et al., 2004). So it is natural to expect that curriculum learning can learn such functions in time $d^{O(1)}2^{O(k)}$ (the second term is needed since there is a doubly exponential number of Juntas on k bits). In this work we propose to learn parities using a mixture of product distributions, but there are other ways to correlate samples that may be of interest. For example, some works in PAC learning showed that, even for the uniform measure, samples that are generated by a random walk often lead to better learning algorithms (Bshouty et al., 2005; Arpe & Mossel, 2008). Do such random walk based algorithms provide better convergence for gradient based methods?

We further believe that a similar idea to the one presented in this paper can be applied to product distributions with orthogonal basis (such as Hermite monomials for the i.i.d. standard Gaussian distribution or spherical harmonics for the uniform distribution over a sphere). These basis elements are no longer orthogonal under biased distributions, and we anticipate that the footprints of our proof would extend to these scenarios. However, in real-world datasets, input coordinates are often not i.i.d., and each coordinate may depend on multiple other coordinates. Nevertheless, we are hopeful that in certain real-world datasets it may be

possible to identify easy and hard samples by means of the variance of the input coordinates (i.e. $\frac{1}{d-1}\sum_{i=1}^{d}(x_i-\bar{x})^2$ for $x \in \mathbb{R}^d$). For instance, consider a task where a learner is required to identify a small object in an image (e.g. a 'stop' signal or a traffic light). In each image, the learner has to identify the relevant subset of coordinates and, intuitively, this is easier in images where the background is plain (samples with low variance) than in images where the background is noisy (samples with large variance).

To conclude, we remark that an important limitation of the curriculum strategy presented in this paper is that it requires an oracle that provides labeled samples from arbitrary product measures. However, in applications one usually has a fixed dataset and would like to select samples in a suitable order, to facilitate learning. It would be an interesting future direction to consider settings where curriculum and non-curriculum have a common sampling distribution.

Acknowledgement

This work was supported in part by the Simons-NSF Collaboration on the Theoretical Foundations of Deep Learning (deepfoundations.ai). It started while E.C. was visiting the MIT Institute for Data, Systems, and Society (IDSS) under the support of the collaboration grant. E.M is also partially supported by the Vannevar Bush Faculty Fellowship award ONR-N00014-20-1-2826 and by a Simons Investigator Award in Mathematics (622132).

References

Abbe, E. and Boix-Adsera, E. On the non-universality of deep learning: quantifying the cost of symmetry. *arXiv preprint arXiv:2208.03113*, 2022.

Abbe, E. and Sandon, C. On the universality of deep learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20061–20072, 2020.

Abbe, E., Boix Adsera, E., Brennan, M., Bresler, G., and Nagaraj, D. The staircase property: How hierarchical structure can guide deep learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021a.

Abbe, E., Kamath, P., Malach, E., Sandon, C., and Srebro, N. On the power of differentiable learning versus PAC and SQ learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021b.

Abbe, E., Adsera, E. B., and Misiakiewicz, T. The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks. In *Conference on Learning Theory*, pp. 4782–4887. PMLR, 2022a.

Abbe, E., Cornacchia, E., Hazla, J., and Marquis, C. An initial alignment between neural network and target is needed for gradient descent to learn. In *International Conference on Machine Learning*, pp. 33–52. PMLR, 2022b.

Abbe, E., Bengio, S., Lotfi, A., and Rizk, K. Generalization on the unseen, logic reasoning and degree curriculum. *arXiv preprint arXiv:2301.13105*, 2023.

Alekhnovich, M. More on average case vs approximation complexity. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pp. 298–307. IEEE, 2003.

Andoni, A., Panigrahy, R., Valiant, G., and Zhang, L. Learning polynomials with neural networks. In *International conference on machine learning*, pp. 1908–1916. PMLR, 2014.

Arpe, J. and Mossel, E. Agnostically learning juntas from random walks. *arXiv preprint arXiv:0806.4210*, 2008.

Avrahami, J., Kareev, Y., Bogot, Y., Caspi, R., Dunaevsky, S., and Lerner, S. Teaching by examples: Implications for the process of category acquisition. *The Quarterly Journal of Experimental Psychology Section A*, 50(3): 586–606, 1997.

Barak, B., Edelman, B. L., Goel, S., Kakade, S., Malach, E., and Zhang, C. Hidden progress in deep learning: Sgd learns parities near the computational limit. *arXiv preprint arXiv:2207.08799*, 2022.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Bshouty, N. H., Mossel, E., O'Donnell, R., and Servedio, R. A. Learning dnf from random walks. *Journal of Computer and System Sciences*, 71(3):250–265, 2005.

Campos, D. Curriculum learning for language modeling. *arXiv preprint arXiv:2108.02170*, 2021.

Daniely, A. and Malach, E. Learning parities with neural networks. *Advances in Neural Information Processing Systems*, 33:20356–20365, 2020.

Dong, Q., Gong, S., and Zhu, X. Multi-task curriculum transfer deep learning of clothing attributes. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 520–529. IEEE, 2017.

Elio, R. and Anderson, J. R. The effects of information order and learning mode on schema abstraction. *Memory & cognition*, 12(1):20–30, 1984.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. PMLR, 2017.

Jiang, L., Meng, D., Zhao, Q., Shan, S., and Hauptmann, A. G. Self-paced curriculum learning. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

Kalimeris, D., Kaplun, G., Nakkiran, P., Edelman, B., Yang, T., Barak, B., and Zhang, H. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

Kearns, M. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

Malach, E. and Shalev-Shwartz, S. Computational separation between convolutional and fully-connected networks. *arXiv preprint arXiv:2010.01369*, 2020.

Malach, E., Kamath, P., Abbe, E., and Srebro, N. Quantifying the benefit of using differentiable learning over tangent kernels. In *International Conference on Machine Learning*, pp. 7379–7389. PMLR, 2021.

Mossel, E., O'Donnell, R., and Servedio, R. A. Learning functions of k relevant variables. *Journal of Computer and System Sciences*, 69(3):421–434, 2004.

Refinetti, M., Ingrosso, A., and Goldt, S. Neural networks trained with sgd learn distributions of increasing complexity. *arXiv preprint arXiv:2211.11567*, 2022.

Ross, B. H. and Kennedy, P. T. Generalizing from the use of earlier examples in problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 16(1):42, 1990.

Saglietti, L., Mannelli, S. S., and Saxe, A. An analytical theory of curriculum learning in teacher–student networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2022(11):114014, 2022.

Sarafianos, N., Giannakopoulos, T., Nikou, C., and Kakadiaris, I. A. Curriculum learning for multi-task classification of visual attributes. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2608–2615, 2017.

Shafto, P., Goodman, N. D., and Griffiths, T. L. A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive psychology*, 71:55–89, 2014.

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

Shalev-Shwartz, S., Shamir, O., and Shammah, S. Failures of gradient-based deep learning. In *International Conference on Machine Learning*, pp. 3067–3075. PMLR, 2017.

Shalev-Shwartz, S. et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Shi, Y., Larson, M., and Jonker, C. M. K-component recurrent neural network language models using curriculum learning. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 1–6. IEEE, 2013.

Shi, Y., Larson, M., and Jonker, C. M. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*, 33(1):136–154, 2015.

Soviany, P., Ionescu, R. T., Rota, P., and Sebe, N. Curriculum learning: A survey. *International Journal of Computer Vision*, pp. 1–40, 2022.

Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Wang, X., Chen, Y., and Zhu, W. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Weinshall, D. and Amir, D. Theory of curriculum learning, with convex loss functions. *Journal of Machine Learning Research*, 21(222):1–19, 2020.

Weinshall, D., Cohen, G., and Amir, D. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*, pp. 5238–5246. PMLR, 2018.

Xiong, Y., He, X., Zhao, D., Tian, T., Hong, L., Jiang, T., and Zeng, J. Modeling multi-species rna modification through multi-task curriculum learning. *Nucleic acids research*, 49(7):3719–3734, 2021.

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

A. Proof of Theorem 3.1

Theorem A.1 (Theorem 3.1, restatement)**.** *Let k, d be both even integers, such that $k \leq d/2$. Let $\mathrm{NN}(x;\theta) = \sum_{i=1}^{N} a_i \sigma(w_i x + b_i)$ be a 2-layers fully connected network with activation $\sigma(y) := \mathrm{Ramp}(y)$ (as defined in (9)) and $N \geq (d+1)(d-k+1)\log((d+1)(d-k+1)/\delta)$. Consider training $\mathrm{NN}(x;\theta)$ with SGD on the hinge loss with batch size $B \geq (8\zeta^2 N^2)^{-1}\log(\frac{Nd+N}{\delta})$, with $\zeta \leq \frac{\epsilon\mu^k}{24(d+1)^2(d-k+1)^2 N}$ and $\mu = \sqrt{1-\frac{1}{2(d-k)}}$. Then, there exists an initialization and a learning rate schedule, and a 2-CL strategy such that after $T \geq \frac{64}{\epsilon^2}(d-k+1)^3(d+1)N$ iterations, with probability $1-3\delta$ SGD outputs a network with generalization error at most ϵ.*

A.1. Proof Setup

We consider a 2-layers neural network, defined as:

$$\mathrm{NN}(x;\theta) = \sum_{i=1}^{N} a_i \sigma(w_i x + b_i), \tag{8}$$

where N is the number of hidden units, $\theta = (a, b, w)$ and $\sigma := \mathrm{Ramp}$ denotes the activation defined as:

$$\mathrm{Ramp}(x) = \begin{cases} 0 & x \leq 0, \\ x & 0 < x \leq 1, \\ 1 & x > 1 \end{cases} . \tag{9}$$

Without loss of generality, we assume that the labels are generated by $\chi_{[k]}(x) := \prod_{i=1}^{k} x_i$. Indeed, SGD on fully connected networks with permutation-invariant initialization is invariant to permutation of the input neurons, thus our result will hold for all $\chi_S(x)$ such that $|S| = k$. Our proof scheme is the following:

1. We train only the first layer of the network for one step on data $(x_i, \chi_{[k]}(x_i))_{i\in[B]}$ with $x_i \sim \mathrm{Rad}(p)^{\otimes d}$ for $i \in [B]$, with $p = \frac{1}{2}\sqrt{1-\frac{1}{2(d-k)}} + \frac{1}{2}$;
2. We show that after one step of training on such biased distribution, the target parity belongs to the linear span of the hidden units of the network;
3. We subsequently train only the second layer of the network on $(x_i, \chi_{[k]}(x_i))_{i\in[B]}$ with $x_i \sim \mathrm{Rad}(1/2)^{\otimes d}$ for $i \in [B]$, until convergence;
4. We use established results on convergence of SGD on convex losses to conclude.

We train our network with SGD on the hinge loss. Specifically, we apply the following updates, for all $t \in \{0, 1, ..., T-1\}$:

$$\begin{aligned} w_{i,j}^{t+1} &= w_{i,j}^{t} - \gamma_t \frac{1}{B}\sum_{s=1}^{B} \nabla_{w_{i,j}^t} L(\theta^t, \chi_{[k]}, x_s^t), \\ a_i^{t+1} &= a_i^t - \xi_t \frac{1}{B}\sum_{s=1}^{B} \nabla_{a_i^t} L(\theta^t, \chi_{[k]}, x_s^t) + c_t, \\ b_i^{t+1} &= \lambda_t \left(b_i^t + \psi_t \frac{1}{B}\sum_{s=1}^{B} \nabla_{b_i^t} L(\theta^t, \chi_{[k]}, x_s^t) \right) + d_t, \end{aligned} \tag{10}$$

where $L(\theta^t, \chi_{[k]}, x) = \max\{0, 1 - \chi_{[k]}(x)\,\mathrm{NN}(x;\theta^t)\}$. Following the 2-steps curriculum strategy introduced above, we set

$$x_s^0 \overset{iid}{\sim} \mathrm{Rad}\,(p)^{\otimes d} \qquad \forall s \in [B], \tag{11}$$

$$x_s^t \overset{iid}{\sim} \mathrm{Rad}\,(1/2)^{\otimes d} \qquad \forall t \geq 1, s \in [B], \tag{12}$$

where $p = \frac{1}{2}\sqrt{1 - \frac{1}{2(d-k)}} + \frac{1}{2}$. For brevity, we denote $\mu := 2p - 1 = \sqrt{1 - \frac{1}{2(d-k)}}$. We set the parameters of SGD to:

$$\gamma_0 = \mu^{-(k-1)}2N, \qquad \gamma_t = 0 \qquad \forall t \geq 1, \tag{13}$$

$$\xi_0 = 0, \qquad \xi_t = \frac{\epsilon}{2N} \qquad \forall t \geq 1, \tag{14}$$

$$\psi_0 = \frac{N}{\mu^k}, \qquad \psi_t = 0 \qquad \forall t \geq 1, \tag{15}$$

$$c_0 = -\frac{1}{2N}, \qquad c_t = 0 \qquad \forall t \geq 1, \tag{16}$$

$$\lambda_0 = (d+1), \qquad \lambda_t = 1 \qquad \forall t \geq 1, \tag{17}$$

$$d_0 = 0, \qquad d_t = 0 \qquad \forall t \geq 1, \tag{18}$$

and we consider the following initialization scheme:

$$\begin{aligned} &w_{i,j}^{(0)} = 0 \qquad \forall i \in [N], j \in [d]; \\ &a_i^{(0)} = \frac{1}{2N} \qquad \forall i \in [N]; \\ &b_i^{(0)} \sim \text{Unif}\left\{\frac{b_{lm}}{d+1} + \frac{1}{2} : l \in \{0, ..., d\}, m \in \{-1, ..., d-k\}\right\}, \end{aligned} \tag{19}$$

where we define

$$b_{lm} := -d + 2l - \frac{1}{2} + \frac{m+1}{d-k}. \tag{20}$$

Note that such initialization is invariant to permutations of the input neurons. We choose such initialization because it is convenient for our proof technique. We believe that the argument may generalize to more standard initialization (e.g. uniform, Gaussian), however this would require more work and it may not be a trivial extension.

A.2. First Step: Recovering the Support

As mentioned above, we train our network for one step on $(x_i, \chi_{[k]}(x_i))_{i\in[B]}$ with $x_i \sim \text{Rad}(p)^{\otimes d}$.

Population gradient at initialization. Let us compute the population gradient at initialization. Since we set $\xi_0 = 0$, we do not need to compute the initial gradient for a. Note that at initialization $|\,\text{NN}(x; \theta^0)| < 1$. Thus, the initial population gradients are given by

$$\forall j \in [k], i \in [N] \qquad G_{w_{i,j}} = -a_i \mathbb{E}_{x\sim\text{Rad}(p)^{\otimes d}}\left[\prod_{l\in[k]\setminus j} x_l \cdot 1(\langle w_i, x\rangle + b_i \in [0,1])\right] \tag{21}$$

$$\forall j \notin [k], i \in [N] \qquad G_{w_{i,j}} = -a_i \mathbb{E}_{x\sim\text{Rad}(p)^{\otimes d}}\left[\prod_{l\in[k]\cup j} x_l \cdot 1(\langle w_i, x\rangle + b_i \in [0,1])\right] \tag{22}$$

$$\forall i \in [N] \qquad G_{b_i} = -a_i \mathbb{E}_{x\sim\text{Rad}(p)^{\otimes d}}\left[\prod_{l\in[k]} x_l \cdot 1(\langle w_i, x\rangle + b_i \in [0,1])\right] \tag{23}$$

Lemma A.2. *Initialize a, b, w according to (19). Then,*

$$\forall j \in [k], \qquad G_{w_{i,j}} = -\frac{\mu^{k-1}}{2N}; \tag{24}$$

$$\forall j \notin [k], \qquad G_{w_{i,j}} = -\frac{\mu^{k+1}}{2N}; \tag{25}$$

$$G_{b_i} = -\frac{\mu^k}{2N}. \tag{26}$$

Proof. If we initialize according to (19), we have $\langle w_i, x\rangle + b_i \in [0,1]$ for all i. The results holds since $\mathbb{E}_{x\sim\text{Rad}(p)^{\otimes d}}[\chi_S(x)] = \mu^{|S|}$. □

Effective gradient at initialization.

Lemma A.3. *Let*

$$\hat{G}_{w_{i,j}} := \frac{1}{B}\sum_{s=1}^{B} \nabla_{w_{i,j}^0} L(\theta^0, \chi_{[k]}, x_s^t) \tag{27}$$

$$\hat{G}_{b_i} := \frac{1}{B}\sum_{s=1}^{B} \nabla_{b_i^0} L(\theta^0, \chi_{[k]}, x_s^t) \tag{28}$$

be the effective gradients at initialization. If $B \geq (8\zeta^2 N^2)^{-1}\log(\frac{Nd+N}{\delta})$*, then with probability* $1-2\delta$*,*

$$\|\hat{G}_{w_{i,j}} - G_{w_{i,j}}\|_\infty \leq \zeta, \tag{29}$$

$$\|\hat{G}_{b_i} - G_{b_i}\|_\infty \leq \zeta, \tag{30}$$

where $G_{w_{i,j}}, G_{b_i}$ *are the population gradients.*

Proof. We note that $\mathbb{E}[\hat{G}_{w_{i,j}}] = G_{w_{i,j}}$, $\mathbb{E}[\hat{G}_{b_i}] = G_{b_i}$, and $|\hat{G}_{w_{i,j}}|, |\hat{G}_{b_i}| \leq \frac{1}{2N}$

$$\mathbb{P}_{x\sim \mathrm{Rad}(p)^{\otimes d}}\left(|\ \hat{G}_{w_{i,j}} - G_{w_{i,j}}\ | \geq \zeta\right) \leq 2\exp\left(-8\zeta^2 N^2 B\right) \leq \frac{2\delta}{Nd+N}, \tag{31}$$

$$\mathbb{P}_{x\sim \mathrm{Rad}(p)^{\otimes d}}\left(|\ \hat{G}_{b_i} - G_{b_i}\ | \geq \zeta\right) \leq 2\exp\left(-8\zeta^2 N^2 B\right) \leq \frac{2\delta}{Nd+N}. \tag{32}$$

The result follows by union bound. □

Lemma A.4. *Let*

$$w_{i,j}^{(1)} = w_{i,j}^{(0)} - \gamma_0 \hat{G}_{w_{i,j}} \tag{33}$$

$$b_i^{(1)} = \lambda_0\left(b_i^{(0)} - \psi_0 \hat{G}_{b_i}\right) \tag{34}$$

$$\tag{35}$$

If $B \geq (8\zeta^2 N^2)^{-1}\log(\frac{Nd+N}{\delta})$*, with probability* $1-2\delta$

i) For all $j \in [k]$*,* $i \in [N]$*,* $|w_{i,j}^{(1)} - 1| \leq \frac{2N\zeta}{\mu^{k-1}}$*;*

ii) For all $j \notin [k]$*,* $|w_{i,j}^{(1)} - (1 - \frac{1}{2(d-k)})| \leq \frac{2N\zeta}{\mu^{k-1}}$ *;*

iii) For all $i \in [N]$*,* $|b_i^{(1)} - (d+1)(b_i^{(0)} - \frac{1}{2})| \leq \frac{N(d+1)\zeta}{\mu^k}$.

Proof. We apply Lemma A.4:

i) For all $j \in [k]$, $i \in [N]$, $|\hat{w}_{i,j}^{(1)} - 1| = \gamma_0|\hat{G}_{w_{i,j}} - G_{w_{i,j}}| \leq \frac{2N\zeta}{\mu^{k-1}}$;

ii) For all $j \notin [k]$, $i \in [N]$, $|\hat{w}_{i,j} - (1 - \frac{1}{2(d-k)})| = \gamma_0|\hat{G}_{w_{i,j}} - G_{w_{i,j}}| \leq \frac{2N\zeta}{\mu^{k-1}}$;

iii) For all $i \in [N]$,

$$|\hat{b}_i^{(1)} - (d+1)(b_i^{(0)} - \frac{1}{2})| = |\lambda_0(b_i^{(0)} + \psi_0\hat{G}_{b_i}) - \lambda_0(b_i^{(0)} + \psi_0 G_{b_i})| \tag{36}$$

$$\leq |\lambda_0| \cdot |\psi_0| \cdot |\hat{G}_{b_i} - G_{b_i}| \tag{37}$$

$$\leq \frac{N(d+1)\zeta}{\mu^k}. \tag{38}$$

□

Lemma A.5. *If $N \geq (d+1)(d-k+1)\log((d+1)(d-k+1)/\delta)$, then with probability $1-\delta$, for all $l \in \{0,...,d\}$, and for all $m \in \{-1,...,d-k\}$ there exists i such that $b_i^{(0)} = \frac{b_{lm}}{d+1} + \frac{1}{2}$.*

Proof. The probability that there exist l, m such that the above does not hold is

$$\left(1 - \frac{1}{(d+1)(d-k+1)}\right)^N \leq \exp\left(-\frac{N}{(d+1)(d-k+1)}\right) \leq \frac{\delta}{(d+1)(d-k+1)}. \tag{39}$$

The result follows by union bound. □

Lemma A.6. *Let $\sigma_{lm}(x) = \text{Ramp}\left(\sum_{j=1}^d x_j - \frac{1}{2(d-k)}\sum_{j>k} x_j + b_{lm}\right)$, with b_{lm} given in (20). If $B \geq (8\zeta^2 N^2)^{-1}\log(\frac{Nd+N}{\delta})$ and $N \geq (d+1)(d-k+1)\log((d+1)(d-k+1)/\delta)$, with probability $1-3\delta$, for all l, m there exists i such that*

$$\left|\sigma_{lm}(x) - \text{Ramp}\left(\sum_{j=1}^d \hat{w}_{i,j}^{(1)} x_j + \hat{b}_i^{(1)}\right)\right| \leq 3N(d+1)\zeta\mu^{-k}. \tag{40}$$

Proof. By Lemma A.5, with probability $1-\delta$, for all l, m there exists i such that $b_i^{(0)} = \frac{b_{lm}}{d+1} + \frac{1}{2}$. For ease of notation, we replace indices $i \mapsto (lm)$, and denote $\hat{\sigma}_{lm}(x) = \text{Ramp}\left(\sum_{j=1}^d w_{lm,j}^{(1)} x_j + b_{lm}^{(1)}\right)$. Then, by Lemma A.4 with probability $1-2\delta$,

$$\begin{aligned}
|\sigma_{lm}(x) - \hat{\sigma}_{lm}(x)| &\leq \left|\sum_{j=1}^k (w_{lm,j}^{(1)} - 1)x_j + \sum_{j=k+1}^d \left(w_{lm,j}^{(1)} - \left(1 - \frac{1}{2(d-k)}\right)\right) x_j + b_{lm}^{(1)} - b_{lm}\right| && (41)\\
&\leq k2N\zeta\mu^{-(k-1)} + (d-k)2N\zeta\mu^{-(k-1)} + N(d+1)\zeta\mu^k && (42)\\
&\leq 3N(d+1)\zeta\mu^{-k}. && (43)
\end{aligned}$$

□

Lemma A.7. *There exists a^* with $\|a^*\|_\infty \leq 4(d-k)$ such that*

$$\sum_{l=0}^d \sum_{m=-1}^{d-k} a_{lm}^* \sigma_{lm}(x) = \chi_{[k]}(x). \tag{44}$$

Proof. Recall, that we assumed d, k even and recall that

$$\sigma_{lm}(x) = \text{Ramp}\left(\sum_{j=1}^d x_j - \frac{1}{2(d-k)}\sum_{j>k} x_j + b_{lm}\right), \tag{45}$$

where $b_{lm} = -d + 2l - \frac{1}{2} + \frac{m+1}{d-k}$ for $l \in [d], m \in \{-1,...,d-k+1\}$ and $\text{Ramp}(x) = \begin{cases} 0 & x \leq 0, \\ x & 0 < x \leq 1, \\ 1 & x > 1 \end{cases}$.

Let $\sum_{j=1}^d x_j = d - 2t$, where t is the total number of -1, and similarly let $-\sum_{j=k+1}^d x_j = (d-k) - 2s$, where s is the number of $+1$ outside the support of the parity $\chi_{[k]}(x)$.We have,

$$\sigma_{lm}(x) = \text{Ramp}\left(2(l-t) + \frac{m+1-s}{d-k}\right). \tag{46}$$

We take

$$a^*_{lm} = (-1)^l(-1)^m 2(d-k) \qquad \forall l \in [d], m = -1, \tag{47}$$
$$a^*_{lm} = (-1)^l(-1)^m 4(d-k) \qquad \forall l \in [d], m \in \{0, 1, ..., d-k-2\}, \tag{48}$$
$$a^*_{lm} = (-1)^l(-1)^m 3(d-k) \qquad \forall l \in [d], m = d-k-1, \tag{49}$$
$$a^*_{lm} = (-1)^l(-1)^m (d-k) \qquad \forall l \in [d], m = d-k, \tag{50}$$

Note that for all $l < t$,

$$2(l-t) + \frac{m+1-s}{d-k} \leq -2 + \frac{d-k+1}{d-k} \leq 0, \tag{51}$$

thus, $\sigma_{lm}(x) = 0$ for all m. Moreover, for all $l > t$,

$$2(l-t) + \frac{m-s+1}{d-k} \geq 2 - \frac{d-k}{d-k} = 1. \tag{52}$$

Thus, $\sigma_{lm}(x) = 1$ for all m and

$$\sum_{m=-1}^{d-k} a^*_{lm}\sigma_{lm}(x) = \sum_{m=-1}^{d-k} a^*_{lm} = 0. \tag{53}$$

If $l = t$,

$$\begin{aligned}
&\sum_{m=-1}^{d-k} a^*_{tm}\sigma_{tm}(x) \\
&\quad = (-1)^t(d-k)\left[\sum_{m=0}^{d-k-2} 4(-1)^m \operatorname{Ramp}\left(\frac{m+1-s}{d-k}\right) - 3\operatorname{Ramp}\left(\frac{d-k-s}{d-k}\right) + \operatorname{Ramp}\left(\frac{d-k+1-s}{d-k}\right)\right] \\
&\quad = (-1)^t\left[\sum_{m=s}^{d-k-2} 4(-1)^m (m+1-s)_+ - 3(d-k-s)_+ + (d-k+1-s)_+\right] \\
&\quad = (-1)^t(-1)^s.
\end{aligned}$$

Since we assumed d, k even, $(-1)^s = \prod_{i=k+1}^d x_i$. Moreover, observe that $\chi_{[k]}(x) = \prod_{i=k+1}^d x_i \cdot \prod_{i=1}^d x_i$. Thus,

$$\sum_{lm} a^*_{lm}\sigma_{lm}(x) = (-1)^t(-1)^s = \chi_{[k]}(x). \tag{54}$$

□

Lemma A.8. *Let $f^*(x) = \sum_{l,m} a^*_{lm}\sigma_{lm}(x)$ and let $\hat{f}(x) = \sum_{l,m} a^*_{lm}\hat{\sigma}_{lm}(x)$, with $\sigma_{lm}, \hat{\sigma}_{lm}$ defined in Lemma B.4 and a^* defined in Lemma A.7. If $B \geq (8\zeta^2N^2)^{-1}\log(\frac{Nd+N}{\delta})$ and $N \geq (d+1)(d-k+1)\log((d+1)(d-k+1)/\delta)$, with probability $1-3\delta$ for all x,*

$$L(\hat{f}, f^*, x) \leq (d+1)^2(d-k+1)^2 12N\zeta\mu^{-k}. \tag{55}$$

Proof.

$$|f^*(x) - \hat{f}(x)| = \left|\sum_{l,m} a^*_{l,m}(\sigma_{lm}(x) - \hat{\sigma}_{lm}(x)\right| \tag{56}$$
$$\leq d(d-k+1)\|a^*\|_\infty \sup_{lm} |\sigma_{lm}(x) - \hat{\sigma}_{lm}(x)| \tag{57}$$
$$\leq (d+1)^2(d-k+1)^2 12N\zeta\mu^{-k}. \tag{58}$$

Thus,

$$(1 - f(x)f^*(x))_+ \leq |1 - f(x)f^*(x)| \tag{59}$$
$$= |f^{*^2}(x) - f(x)f^*(x)| \tag{60}$$
$$= |f^*(x)| \cdot |f^*(x) - f(x)| \leq (d+1)^2(d-k+1)^2 12N\zeta\mu^{-k}, \tag{61}$$

which implies the result. □

A.3. Second Step: Convergence

To conclude, we use an established result on convergence of SGD on convex losses (see e.g. (Shalev-Shwartz et al., 2012; Shalev-Shwartz & Ben-David, 2014; Daniely & Malach, 2020; Malach & Shalev-Shwartz, 2020; Barak et al., 2022)).

Theorem A.9. *Let $\mathcal{L}$ be a convex function and let $a^* \in \operatorname{argmin}_{\|a\|_2 \leq \mathcal{B}} \mathcal{L}(a)$, for some $\mathcal{B} > 0$. For all t, let $\alpha^{(t)}$ be such that $\mathbb{E}\left[\alpha^{(t)} \mid a^{(t)}\right] = -\nabla_{a^{(t)}} \mathcal{L}(a^{(t)})$ and assume $\|\alpha^{(t)}\|_2 \leq \rho$ for some $\rho > 0$. If $a^{(0)} = 0$ and for all $t \in [T]$ $a^{(t+1)} = a^{(t)} + \gamma\alpha^{(t)}$, with $\gamma = \frac{\mathcal{B}}{\rho\sqrt{T}}$, then :*

$$\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(a^{(t)}) \leq \mathcal{L}(a^*) + \frac{\mathcal{B}\rho}{\sqrt{T}}. \tag{62}$$

Let $\mathcal{L}(a) := \mathbb{E}_{x \sim \mathrm{Rad}(1/2)^{\otimes d}}\left[L((a, b^{(1)}, w^{(1)}), \chi_{[k]}, x)\right]$. Then, $\mathcal{L}$ is convex in a and for all $t \in [T]$,

$$\alpha^{(t)} = -\frac{1}{B}\sum_{s=1}^{B} \nabla_{a^{(t)}} L((a^{(t)}, b^{(1)}, w^{(1)}), \chi_{[k]}, x) \tag{63}$$
$$= -\frac{1}{B}\sum_{s=1}^{B} \sigma(w^{(1)}x + b^{(1)}). \tag{64}$$

Thus, recalling $\sigma = \mathrm{Ramp}$, we have $\|\alpha^{(t)}\|_2 \leq \sqrt{N}$. Let a^* be as in Lemma A.7. Clearly, $\|a^*\|_2 \leq 4(d-k+1)^{3/2}(d+1)^{1/2}$. Moreover, $a^{(1)} = 0$. Thus, we can apply Theorem A.9 with $\mathcal{B} = 4(d-k+1)^{3/2}(d+1)^{1/2}$, $\rho = \sqrt{N}$ and obtain that if

1. $N \geq (d+1)(d-k+1)\log((d+1)(d-k+1)/\delta)$;
2. $\zeta \leq \frac{\epsilon\mu^k}{24(d+1)^2(d-k+1)^2N}$;
3. $B \geq (8\zeta^2N^2)^{-1}\log(\frac{Nd+N}{\delta})$;
4. $T \geq \frac{64}{\epsilon^2}(d-k+1)^3(d+1)N$.

then, with probability $1 - 3\delta$ over the initialization

$$\mathbb{E}_{x \sim \mathrm{Rad}(1/2)^{\otimes d}}\left[\min_{t \in [T]} L\left(\theta^{(t)}, \chi_{[k]}, x\right)\right] \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon. \tag{65}$$

Remark A.10. We assume $k \leq d/2$ to avoid exponential dependence of ζ (and consequently of the batch size and of the computational complexity) in d. Indeed, if $k \leq d/2$, then,

$$\mu^k = \left(1 - \frac{1}{2(d-k)}\right)^{k/2} \geq \left(1 - \frac{1}{d}\right)^{d/4} \sim e^{-1/4}. \tag{66}$$

B. Proof of Theorem 3.4

Theorem B.1 (Theorem 3.4, restatement)**.** *Let k, d be integers, such that $d \geq k$ and k even. Let $\mathrm{NN}(x;\theta) = \sum_{i=1}^{N} a_i\sigma(w_ix + b_i)$ be a 2-layers fully connected network with activation $\sigma(y) := \mathrm{ReLU}(y)$ and $N \geq (k+1)\log(\frac{k+1}{\delta})$. Consider training $\mathrm{NN}(x;\theta)$ with SGD on the covariance loss with batch size $B \geq (2\zeta^2)^{-1}\log(\frac{dN}{\delta})$, with $\zeta \leq \frac{\epsilon(\mu^{k-1} - \mu^{k+1})}{64k^2N} \cdot \left(1 + \frac{d-k}{k}\right)^{-1}$, for some $\mu \in (0,1)$. Then, there exists an initialization, a learning rate schedule, and a 2-CL strategy such that after $T \geq \frac{64k^3N}{\epsilon^2}$ iterations, with probability $1 - 3\delta$ SGD outputs a network with generalization error at most ϵ.*

B.1. Proof Setup

Similarly as before, we consider a 2-layers neural network, defined as $\mathrm{NN}(x;\theta) = \sum_{i=1}^{N} a_i \sigma(w_i x + b_i)$, where N is the number of hidden units, $\theta = (a, b, w)$ and $\sigma := \mathrm{ReLU}$. Our proof scheme is similar to the previous Section. Again, we assume without loss of generality that the labels are generated by $\chi_{[k]}(x) := \prod_{i=1}^{k} x_i$. We assume k to be even. We train our network with SGD on the covariance loss, defined in Def. 3.2. We use the same updates as in (10) with:

$$x_s^0 \overset{iid}{\sim} \mathrm{Rad}(p)^{\otimes d} \qquad \forall s \in [B], \tag{67}$$

$$x_s^t \overset{iid}{\sim} \mathrm{Rad}(p)^{\otimes d} \qquad \forall t \geq 1, s \in [B], \tag{68}$$

for some $p \in (1/2, 1)$. We denote $\mu := 2p - 1$. We set the parameters to:

$$\gamma_0 = 16N(\mu^{k-1} - \mu^{k+1})^{-1} k^{-1}, \qquad \gamma_t = 0 \quad \forall t \geq 1, \tag{69}$$

$$\xi_0 = 0, \qquad \xi_t = \frac{\epsilon}{8N} \quad \forall t \geq 1, \tag{70}$$

$$\psi_0 = 0, \qquad \psi_t = 0 \quad \forall t \geq 1, \tag{71}$$

$$c_0 = -1, \qquad c_t = 0 \quad \forall t \geq 1, \tag{72}$$

$$\lambda_0 = 1, \qquad \lambda_t = 1 \quad \forall t \geq 1, \tag{73}$$

$$d_0 = -1, \qquad d_t = 0 \quad \forall t \geq 1, \tag{74}$$

and we consider the following initialization scheme:

$$\begin{aligned} w_{i,j}^{(0)} &= 0 \qquad \forall i \in [N], j \in [d]; \\ a_i^{(0)} &= \frac{1}{16N} \qquad \forall i \in [N]; \\ b_i^{(0)} &\sim \mathrm{Unif}\left\{ \frac{2(i+1)}{k} : i \in \{0, 1, ..., k\} \right\}. \end{aligned} \tag{75}$$

B.2. First Step: Recovering the Support

Population gradient at initialization. At initialization, we have $|\mathrm{NN}(x;\theta^0)| < \frac{1}{4}$, thus

$$\left| \mathrm{cov}(\chi_{[k]}, \theta^0, x, \mathrm{Rad}(p)^{\otimes d}) \right| < 1. \tag{76}$$

The initial gradients are therefore given by:

$$\forall i, j \qquad G_{w_{i,j}} = -a_i \mathbb{E}_{x \sim \mathrm{Rad}(p)^{\otimes d}} \left[\Big(\prod_{l \in [k]} x_l - \mu^k \Big) \cdot \Big(x_j 1(\langle w_i, x \rangle + b_i > 0) - \mathbb{E} x_j 1(\langle w_i, x \rangle + b_i > 0) \Big) \right] \tag{77}$$

$$\forall i \in [N] \qquad G_{b_i} = -a_i \mathbb{E}_{x \sim \mathrm{Rad}(p)^{\otimes d}} \left[\Big(\prod_{l \in [k]} x_l - \mu^k \Big) \cdot \Big(1(\langle w_i, x \rangle + b_i > 0) - \mathbb{E} 1(\langle w_i, x \rangle + b_i > 0) \Big) \right] \tag{78}$$

If we initialize a, b, w according to (75). Then,

$$\forall j \in [k], \qquad G_{w_{i,j}} = -\frac{\mu^{k-1} - \mu^{k+1}}{16N}; \tag{79}$$

$$\forall j \notin [k], \qquad G_{w_{i,j}} = 0; \tag{80}$$

$$G_{b_i} = 0. \tag{81}$$

Effective gradient at initialization.

Lemma B.2. *Let*

$$w_{i,j}^{(1)} = w_{i,j}^{(0)} - \gamma_0 \hat{G}_{w_{i,j}} \tag{82}$$

$$b_i^{(1)} = \lambda_0 \left(b_i^{(0)} - \psi_0 \hat{G}_{b_i} \right) + d_0, \tag{83}$$

$$\tag{84}$$

where $\hat{G}_{w_{i,j}}, \hat{G}_{b_i}$ are the gradients estimated from the initial batch. Then, with probability $1-2\delta$, if $B \geq (2\zeta^2)^{-1} \log\left(\frac{dN}{\delta}\right)$,

i) For all $j \in [k]$, $i \in [N]$, $|w_{i,j}^{(1)} - \frac{1}{k}| \leq \frac{\zeta 16N}{k(\mu^{k-1}-\mu^{k+1})}$;

ii) For all $j \notin [k]$, $|w_{i,j}^{(1)}| \leq \frac{\zeta 16N}{k(\mu^{k-1}-\mu^{k+1})}$;

iii) For all $i \in [N]$, $b_i^{(1)} = b^{(0)} - 1$

Proof. By Lemma A.4 ,if $B \geq (2\zeta^2)^{-1} \log\left(\frac{dk}{\delta}\right)$, then for all $j \in [k], i \in [N]$, $|\hat{G}_{w_{i,j}} - G_{w_{i,j}}| \leq \zeta$. Thus,

i) For all $j \in [k]$, $i \in [N]$, $|w_{i,j}^{(1)} - \frac{1}{k}| = \gamma_0 |\hat{G}_{w_{i,j}} - G_{w_{i,j}}| \leq \frac{\zeta 16N}{k(\mu^{k-1}-\mu^{k+1})}$;

ii) For all $j \in [k]$, $i \in [N]$, $|w_{i,j}^{(1)}| = \gamma_0 |\hat{G}_{w_{i,j}} - G_{w_{i,j}}| \leq \frac{\zeta 16N}{k(\mu^{k-1}-\mu^{k+1})}$;

iii) follows trivially. □

Lemma B.3. *If $N \geq (k+1)\log\left(\frac{k+1}{\delta}\right)$, with probability $1-\delta$ for all $i \in \{0, ..., k\}$ there exists $l \in [N]$ such that $b_l^{(0)} = \frac{2(i+1)}{k}$.*

Proof. The probability that there exists i such that the above does not hold is

$$\left(1 - \frac{1}{k+1}\right)^N \leq \exp\left(-\frac{N}{k+1}\right) \leq \frac{\delta}{k+1}. \tag{85}$$

The result follows by union bound. □

Lemma B.4. *Let $\sigma_i(x) := \mathrm{ReLU}\left(\frac{1}{k}\sum_{j=1}^k x_j + b_i\right)$, with $b_i = -1 + \frac{2(i+1)}{k}$. Then, with probability $1-3\delta$, for all $i \in \{0, ..., k\}$ there exists $l \in [N]$ such that*

$$\left|\sigma_i(x) - \mathrm{ReLU}\left(\sum_{j=1}^d w_{l,j}^{(1)} x_j + b_l^{(1)}\right)\right| \leq \frac{\zeta 16N}{\mu^{k-1}-\mu^{k+1}} \cdot \left(1 + \frac{d-k}{k}\right). \tag{86}$$

Proof. By Lemma B.3 and Lemma B.2, with probability $1-3\delta$, for all i there exists l such that $b_l^{(1)} = -1 + \frac{2(i+1)}{k}$, and

$$\left|\sigma_i(x) - \mathrm{ReLU}\left(\sum_{j=1}^d w_{l,j}^{(1)} x_j + b_l^{(1)}\right)\right| \leq \left|\sum_{j=1}^k \left(\frac{1}{k} - w_{l,j}^{(1)} x_j\right)\right| + \left|\sum_{j=k+1}^d w_{l,j}^{(1)} x_j\right| \tag{87}$$

$$\leq \frac{\zeta 16N}{\mu^{k-1}-\mu^{k+1}} + \frac{\zeta 16N(d-k)}{(\mu^{k-1}-\mu^{k+1})k} \tag{88}$$

$$= \frac{\zeta 16N}{\mu^{k-1}-\mu^{k+1}} \cdot \left(1 + \frac{d-k}{k}\right). \tag{89}$$

□

Lemma B.5. *There exist a^* with $\|a^*\|_\infty \le 2k$ such that*

$$\sum_{i=0}^{k} a_i^* \sigma_i(x) = \chi_{[k]}(x). \tag{90}$$

Proof. We assume k to be even. Let $\sum_{j=1}^{k} x_j = k - 2t$, where $t := |\{i : x_i = -1, i \in [k]\}|$. Thus,

$$\sigma_i(x) = \mathrm{ReLU}\left(\frac{2(i+1-t)}{k}\right). \tag{91}$$

We choose

$$a_i^* = (-1)^i 2k \qquad \forall i \in \{0, 1, ..., k-2\}, \tag{92}$$

$$a_i^* = (-1)^i \frac{3}{2}k \qquad i = k-1, \tag{93}$$

$$a_i^* = (-1)^i \frac{1}{2}k \qquad i = k. \tag{94}$$

One can check that with these a_i^* the statement holds. □

Lemma B.6. *Let $f^*(x) = \sum_{i=0}^{k} a_i^* \sigma_i(x)$ and let $\hat{f}(x) = \sum_{i=0}^{k} a_i^* \hat{\sigma}_i(x)$, with $\sigma_i(x)$ defined above and $\hat{\sigma}_i(x) := \mathrm{ReLU}(\sum_{j=1}^{d} w_{i,j}^{(1)} x_j + b_i^{(1)})$. Then, with probability $1 - 3\delta$ for all x,*

$$(1 - f(x)f^*(x))_+ \le \frac{32k^2\zeta N}{\mu^{k-1} - \mu^{k+1}} \cdot \left(1 + \frac{d-k}{k}\right), \tag{95}$$

where $(z)_+ := \max\{0, z\}$.

Proof.

$$|f^*(x) - \hat{f}(x)| = \Big| \sum_i a_i^* (\sigma_i(x) - \hat{\sigma}_i(x) \Big| \tag{96}$$

$$\le k\|a^*\|_\infty \sup_i |\sigma_i(x) - \hat{\sigma}_i(x)| \tag{97}$$

$$\le \frac{32k^2\zeta N}{\mu^{k-1} - \mu^{k+1}} \cdot \left(1 + \frac{d-k}{k}\right). \tag{98}$$

Thus,

$$(1 - f(x)f^*(x))_+ \le |1 - f(x)f^*(x)| \tag{99}$$

$$= |f^{*^2}(x) - f(x)f^*(x)| \tag{100}$$

$$= |f^*(x)| \cdot |f^*(x) - f(x)| \le \frac{32k^2\zeta N}{\mu^{k-1} - \mu^{k+1}} \cdot \left(1 + \frac{d-k}{k}\right). \tag{101}$$

□

B.3. Second Step: Convergence

We apply Theorem A.9 with $\mathcal{L}(a) := \mathbb{E}_{x \sim \mathrm{Rad}(1/2)^{\otimes d}} \left[L_{\mathrm{cov}}((a, b^{(1)}, w^{(1)}), \chi_{[k]}, x)\right]$, $\rho = 2\sqrt{N}$, $\mathcal{B} = 2k\sqrt{k}$. We get that if

1. $N \ge (k+1)\log(\frac{k+1}{\delta})$;
2. $\zeta \le \frac{\epsilon(\mu^{k-1} - \mu^{k+1})}{64k^2 N} \cdot \left(1 + \frac{d-k}{k}\right)^{-1}$;

3. $B \geq (2\zeta^2)^{-1} \log(\frac{dN}{\delta})$;
4. $T \geq \frac{64k^3N}{\epsilon^2}$.

then with probability $1 - 3\delta$ over the initialization,

$$\mathbb{E}_{x\sim \mathrm{Rad}(1/2)^{\otimes d}} \left[\min_{t\in[T]} L_{\mathrm{cov}} \left(\chi_{[k]}, \theta^t, x, \mathrm{Rad}(1/2)^{\otimes d} \right) \right] \leq \epsilon. \tag{102}$$

Remark B.7. We remark that if $\mu = \theta_{d,k}(1)$, then ζ decreases exponentially fast in k, and as a consequence the batch size and the computational cost grow exponentially in k. If however we choose $\mu = 1 - 1/k$, then we get $\zeta = 1/\operatorname{poly}(k)$ and, as a consequence, the batch size and the computational cost grow polynomially in k.

C. Proof of Theorem 4.2

Let us consider $r = 2$. The case of general r follows easily. Let us state the following Lemma.

Lemma C.1. *Let $x \sim \mathrm{Rad}(p)^{\otimes d}$ and let $H(x) := \sum_{i=1}^d 1(x_i = 1)$ be the Hamming weight of $H(x)$. Assume $\epsilon < p < \epsilon'$ for some $\epsilon, \epsilon' \in [0, 1/2]$, then,*

$$\mathbb{P}_{x\sim \mathrm{Rad}(p)^{\otimes d}} \left(H(x) \geq \epsilon' d \right) \leq 2 \exp\left(-(\epsilon' - p)^2 d \right); \tag{103}$$

$$\mathbb{P}_{x\sim \mathrm{Rad}(p)^{\otimes d}} \left(H(x) \leq \epsilon d \right) \leq 2 \exp\left(-(p - \epsilon)^2 d \right). \tag{104}$$

Proof of Lemma C.1. We apply Hoeffding's inequality with $\mathbb{E}[H(x)] = pd$ and $\sum_{i=1}^d x_i = d - 2H(x)$:

$$\mathbb{P}(|H(x) - pd| \geq td) \leq 2 \exp\left(-(t - p)^2 d \right). \tag{105}$$

□

Take ϵ such that $|p_1 - \frac{1}{2}| > \epsilon$, and consider the following algorithm:

1. Choose a set $V \subseteq [d]$ uniformly at random among all subsets of $[d]$ of cardinality k_S;
2. Take a fully connected neural network $\mathrm{NN}(x; \psi)$ with the same architecture as $\mathrm{NN}(x; \theta)$ and with initialization $\psi^0 = \theta^0$;
3. Train $\mathrm{NN}(x; \psi)$ on data $(x, \chi_V(x))$, with $x \sim \mathrm{Rad}(p_1)^{\otimes d}$;
4. Train until convergence the pre-trained network $\mathrm{NN}(x; \psi)$ with initialization ψ^{T_1} on data $(x, \chi_T(x))$, with $x \sim \mathrm{Rad}(1/2)^{\otimes d}$.

The result holds by the following two Lemmas.

Lemma C.2. *If $V = S$, $\mathrm{TV}(\theta^T; \psi^T) \leq \frac{AT\sqrt{|\theta|}}{\tau} \exp(-d\delta^2)$, where $\delta = \min\{|\epsilon - p_1|, |1/2 - \epsilon|\}$ and TV denotes the total variation distance between the law of θ^T and ψ^T.*

Proof. Clearly, $\mathrm{TV}(\theta^0; \psi^0) = 0$. Then, using subadditivity of TV

$$\mathrm{TV}(\theta^T; \psi^T) \leq \sum_{t=1}^T \mathrm{TV}(\theta^t; \psi^t | \{Z^i\}_{i\leq t-2}) \tag{106}$$

$$= \sum_{t=1}^T \mathrm{TV}(\gamma(g_{\theta^{t-1}} + Z^{t-1}); \gamma(g_{\psi^{t-1}} + Z^{t-1}) | \{Z_i\}_{i\leq t-1}), \tag{107}$$

where $g_{\theta^{t-1}}, g_{\psi^{t-1}}$ denote the population gradients in θ^{t-1} and ψ^{t-1}, respectively. Then, recalling that the Z^{t-1} are Gaussians, we get

$$\mathrm{TV}(\theta^T;\psi^T) \overset{(a)}{\leq} \sum_{t=1}^{T} \frac{1}{2\tau\gamma} \|\gamma g_{\theta^{t-1}} - \gamma g_{\psi^{t-1}}\|_2 \tag{108}$$

$$\overset{(b)}{\leq} \sum_{t=1}^{T_{r-1}} \frac{1}{2\tau\gamma} \cdot 2\sqrt{|\theta|}A\gamma \cdot \mathbb{P}(H(x) \geq \epsilon d) + \sum_{t=T_{r-1}}^{T} \frac{1}{2\tau\gamma} \cdot 2\sqrt{|\theta|}A\gamma \cdot \mathbb{P}(H(x) \leq \epsilon d) \tag{109}$$

$$\overset{(c)}{\leq} \frac{AT\sqrt{|\theta|}}{\tau} \exp(-d\delta^2). \tag{110}$$

In (a) we applied the formula for the TV between Gaussian variables with same variance. In (b) we used that each gradient is in $[-A, A]$ and that during the first part of training, for all x with $H(x) < \epsilon$, the two gradients are the same, and similarly in the second part of training for all $H(x) > \epsilon d$. In (c) we applied Lemma C.1. □

We apply Theorem 3 from (Abbe & Sandon, 2020), which we restate here for completeness.

Theorem C.3 (Theorem 3 in (Abbe & Sandon, 2020))**.** *Let $\mathcal{P}_k$ be the set of k-parities over d bits. Noisy-GD on any neural network of size $|\theta|$ and any initialization, after T steps of training under the uniform distribution, outputs a network such that*

$$\frac{1}{|\mathcal{P}_k|} \sum_{f \in \mathcal{P}_k} |\mathbb{E}_{x \sim \mathrm{Rad}(1/2)^{\otimes d}} \left[\mathrm{NN}(x;\theta^T) \cdot f(x)\right]| \leq \frac{T\sqrt{|\theta|}A}{\tau} \cdot d^{-k/2}. \tag{111}$$

In our case, this implies:

$$\mathbb{E}_V |\mathbb{E}_x \left[\mathrm{NN}(x;\psi^T) \cdot G_{S,T,\epsilon}(x)\right]| \leq \frac{(T - T_{r-1})\sqrt{|\theta|}A}{\tau d^{k_T/2}} \tag{112}$$

To conclude our proof, note that:

$$\mathbb{E}_V |\mathbb{E}_x \left[\mathrm{NN}(x,\psi^T) \cdot G_{S,T,\epsilon}(x)\right]| = \mathbb{E}_V \left[\left|\mathbb{E}_x \left[\mathrm{NN}(x,\psi^T) \cdot G_{S,T,\epsilon}(x)\right]\right| \,|\, \left|V \cap T\right| = 0\right] \mathbb{P}(|V \cap T| = 0) \tag{113}$$

$$+ \mathbb{E}_V \left[\left|\mathbb{E}_x \left[\mathrm{NN}(x,\psi^T) \cdot G_{S,T,\epsilon}(x)\right]\right| \,|\, \left|V \cap T\right| > 0\right] \mathbb{P}(|V \cap T| > 0). \tag{114}$$

One can check that,

$$\mathbb{P}(|V \cap T| > 0) = 1 - \frac{2k_S k_T}{d} + O(d^{-2}). \tag{115}$$

Moreover, by symmetry, for any V such that $|V \cap T| = 0$, the algorithm achieves the same correlation (to see this, one find an appropriate permutation of the input neurons and use invariance of GD on fully connected networks with permutation-invariant initialization). Thus,

$$\mathbb{E}_V \left[\left|\mathbb{E}_x \left[\mathrm{NN}(x,\psi^T) \cdot G_{S,T,\epsilon}(x)\right]\right| \,|\, \left|V \cap T\right| = 0\right] = \mathbb{E}_V \left[\left|\mathbb{E}_x \left[\mathrm{NN}(x,\psi^T) \cdot G_{S,T,\epsilon}(x)\right]\right| \,|\, V = S\right]. \tag{116}$$

By Lemma C.2,

$$|\mathbb{E}_x \left[\mathrm{NN}(x;\theta^T) \cdot G_{S,T,\epsilon}(x)\right]| \leq \mathbb{E}_V \left[\left|\mathbb{E}_x \left[\mathrm{NN}(x,\psi^T) \cdot G_{S,T,\epsilon}(x)\right]\right| \,|\, V = S\right] + \frac{AT\sqrt{|\theta|}}{\tau} \exp(-d\delta^2) \tag{117}$$

$$\leq \frac{(T - T_{r-1})\sqrt{|\theta|}A}{\tau d^{k_T/2}} + \frac{AT\sqrt{|\theta|}}{\tau} \exp(-d\delta^2) + \frac{2k_S k_T}{d} + O(d^{-2}). \tag{118}$$

The argument for general r holds by taking ϵ such that $|p_l - \frac{1}{2}| > \epsilon$ for all $l \in [r-1]$ and by replacing step 3 of the algorithm above with the following:

3. Train $\mathrm{NN}(x;\psi)$ on data $(x, \chi_V(x))$ using a $(r-1)$-CL$((T_1, ..., T_{r-1}), (p_1, ..., p_{r-1}))$ strategy.

D. Proof of Corollary 4.5

We use the same proof strategy of Theorem 4.2: specifically, we use the same algorithm for training a network $\mathrm{NN}(x;\psi)$ with the same architecture as $\mathrm{NN}(x;\theta)$, so that Lemma C.2 still holds. We import Theorem 3 from (Abbe & Sandon, 2020) in the following form.

Theorem D.1 (Theorem 3 in (Abbe & Sandon, 2020))**.** *Let $\mathcal{F}$ be the set of k-parities over set $\{d/2+1,...,d\}$. Noisy-GD on any neural network $\mathrm{NN}(x;w)$ of size $|w|$ and any initialization, after T steps of training on samples drawn from the uniform distribution, outputs a network such that*

$$\frac{1}{|\mathcal{F}|}\sum_{f\in\mathcal{F}}\left|\mathbb{E}_{x\sim\mathrm{Rad}(1/2)^{\otimes d}}\left[\mathrm{NN}(x;w^{(T)})\cdot f(x)\right]\right|\leq\frac{T\sqrt{|w|}A}{\tau}\cdot\binom{d/2}{k}^{-1/2}. \tag{119}$$

Similarly as before, Theorem D.1 and Lemma C.1 imply:

$$\mathbb{E}_V\left|\mathbb{E}_{x\sim\mathrm{Rad}(1/2)^{\otimes d}}\left[\mathrm{NN}(x;\psi^{(T)})\cdot G_{S,V,\epsilon}(x)\right]\right|\leq\frac{(T-T_{r-1})\sqrt{|\theta|}A}{\tau}\cdot\binom{d/2}{k_V}^{-1/2}+\exp(-d\delta^2), \tag{120}$$

where by $\mathbb{E}_V$ we denote the expectation over set V sampled uniformly at random from all subsets of $\{d/2+1,...,d\}$ of cardinality k_V. Since both the initialization and noisy-GD on fully connected networks are invariant to permutation of the input neurons, for any $V\subseteq\{d/2+1,...,d\}$, the algorithm achieves the same correlation. Thus, applying Lemma C.2:

$$\left|\mathbb{E}_x\left[\mathrm{NN}(x;\theta^{(T)})\cdot G_{S,V,\epsilon}(x)\right]\right|\leq\frac{(T-T_{r-1})\sqrt{|\theta|}A}{\tau}\cdot\binom{d/2}{k_V}^{-1/2}+\frac{2AT\sqrt{|\theta|}}{\tau}\exp(-d\delta^2). \tag{121}$$