# Resource-Efficient Methods in Machine Learning

Kiran Vodrahalli

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2022

# Abstract

Resource-Efficient Methods in Machine Learning

Kiran Vodrahalli

In this thesis, we consider resource limitations on machine learning algorithms in a variety of settings. In the first two chapters, we study how to learn nonlinear model classes (monomials and neural nets) which are structured in various ways – we consider sparse monomials and deep neural nets whose weight-matrices are low-rank respectively. These kinds of restrictions on the model class lead to gains in **resource efficiency** – sparse and low-rank models are computationally easier to deploy and train. We prove that sparse nonlinear monomials are easier to learn (smaller sample complexity) while still remaining computationally efficient to both estimate and deploy, and we give both theoretical and empirical evidence for the benefit of novel nonlinear initialization schemes for low-rank deep networks. In both cases, we showcase a *blessing of nonlinearity* – sparse monomials are in some sense easier to learn compared to a linear class, and the prior state-of-the-art linear low-rank initialization methods for deep networks are inferior to our proposed nonlinear method for initialization. To achieve our theoretical results, we often make use of the theory of *Hermite polynomials* – an orthogonal function basis over the Gaussian measure. In the last chapter, we consider **resource limitations in an online streaming setting**. In particular, we consider how many data points from an oblivious adversarial stream we must store from one pass over the stream to output an additive approximation to the Support Vector Machine (SVM) objective, and prove stronger lower bounds on the memory complexity.

# Table of Contents

# List of Figures

# List of Tables

## Acknowledgements

First and foremost, I would like to thank my advisors Alex Andoni and Daniel Hsu for their support and guidance over the past five years. They have both been very generous with their time, ideas, and feedback. From Daniel I learned to be more precise with my ideas, to interrogate claims more deeply, and a taste for what a good paper looks like. Alex taught me the importance of perseverance on a problem but also when to cut my losses, how to break down a problem to its essence and to strategize how to prove complicated claims, and also to maintain a lighter heart when certain research avenues do not pan out. Of course, these lessons are just a sample of many more things I learned from both of them, and I am certain the impact of being their student will resonate as I continue to grow throughout my research career.

I would next like to thank Christos Papadimitriou for his mentorship throughout my Ph.D. We worked together with Mihalis Yannakakis on a paper which does not show up in this thesis but that I am particularly proud of: *The Platform Design Problem.* I learned a great deal of lessons from Christos throughout my Ph.D. on how to conduct impactful research, but perhaps more importantly, on how to view life as well. I greatly enjoyed TA-ing for his inaugural class on Computation and the Brain at Columbia, and also appreciate his joining my thesis committee.

I would also like to thank my final two committee members, Rocco Servedio and Mahesh Sathiamoorthy, for taking the time to witness my defense, ask insightful questions, and for giving me helpful feedback on my thesis.

Now I would like to take the chance to thank my mentors from my undergraduate and master's degrees at Princeton. First, without the encouragement and mentorship of Sanjeev Arora

and Ken Norman, my undergraduate and master's degree advisors, I would never have had the opportunity to begin my Ph.D. degree. I also have to thank Christiane Fellbaum for mentoring me throughout my undergraduate degree and getting me interested in machine learning research (via natural language understanding) in the first place. I also would like to thank Ramon van Handel for sending me a very influential email during my junior year at Princeton which convinced me to focus on mathematics for my major degree as opposed to trying to do my own major – it is primarily thanks to him that I ended up taking courses that eventually pointed me in the direction of a Ph.D. in theoretical machine learning. Finally, I would like to thank Elad Hazan for greatly encouraging me and giving me confidence that I could succeed in a research career.

From June 2021 to February 2022, I was able to intern at Google Brain with Rakesh Shivanna and Mahesh Sathiamoorthy on Ed Chi's team – I greatly appreciated the opportunity, and fortuitously, we were able to include the results of that internship in Chapter 3 of this thesis. I greatly appreciated their mentorship and help throughout the internship.

I also greatly enjoyed the collaborations I had with my co-authors in the past five years, including with Misha Khodak, Nikunj Saunshi, Sanjeev Arora, Alex Andoni, Rishabh Dudeja, Daniel Hsu, Brandon Araki, Xiao Li, Thomas Leech, Cristian-Ioan Vasile, Mark Donahue, Jonathan DeCastro, J. Micah Fry, Daniela Rus, Mathias Lécuyer, Riley Spahn, Roxana Geambasu, Tim Roughgarden, Christos Papadimitriou, Mihalis Yannakakis, Rakesh Shivanna, Mahesh Sathiamoorthy, Sagar Jain, Ed Chi, and Jon Schneider. Special thanks are owed to Rishabh Dudeja, who coined the phrase "blessing of nonlinearity" that ended up being a cornerstone of this thesis. I would also like to particularly thank Tim Roughgarden for his mentorship and time, and for serving on my candidacy exam committee.

I have also taken many classes over the years from which I learned a great deal. In chronological order, I would like to thank Michael Damron, Arvind Narayanan, Josh Hug, Mark Braverman, Elias Stein, Ramon van Handel, Sanjeev Arora, Emmanuel Abbé, Elad Hazan, Sebastian Seung, Ken Norman, Barbara Engelhardt, Peter Ramadge, Assaf Naor, Samory Kpotufe, Yuxin Chen, Daniel Hsu, Alex Andoni, Ilya Razenshteyn, Michael Collins, Shipra

Agrawal, Ming Yuan, Nakul Verma, Christos Papadimitriou, and Tim Roughgarden.

My Ph.D. experience would have been very different without the camaraderie of my fellow Ph.D students and postdocs at Columbia. I learned many things during reading groups and discussions from, in no particular order, Rishabh Dudeja, Geelon So, Giannis Karamanolakis, Clayton Sanford, Carolina Zheng, Ana Stoica, William Brown, Sandip Sinha, Chris Tosh, Peilin Zhong, Kevin Shi, Ji Xu, Navid Ardeshir, Ilya Razenshteyn, Shunhua Jiang, Hengjie Zhang, Negev Nosatzki, Miranda Christ, Maryam Bahrani, Dan Mitropolsky, Shivam Nadimpalli, Eric Neyman, Binghui Peng, Tim Randolph, Arnold Filtser, Emmanouil Vlatakis, Kira Goldner, Chin Ho Lee, Marshall Ball, and Erik Waingarten.

I also have many friends external to Columbia's research community to thank for many engaging research discussions over the years. In particular, I would like to thank Minoru Araki, Siddhartho Bhattacharya, Jeremy Cohen, Ghassen Jerfel, Misha Khodak, Holden Lee, Katherine Lee, Anand Natarajan, Nikunj Saunshi, Jon Schneider, Elizabeth Yang, Angela Zhou, and my brother Kailas.

Finally, special thanks are owed to my partner Amy, who has been a huge support for me throughout my academic career, as well as to my parents, who both dedicated selfless amounts of time and energy raising me and ensuring I received an excellent education. I greatly appreciate everything they have done for me, and I could not have done it without them.

# Dedication

To Amy and my parents.

# Chapter 1: Introduction

## 1.1 Resource Limitations in Modern Machine Learning

In this thesis, we consider resource limitations on machine learning algorithms in various settings. In modern machine learning, models in practice have increasingly large size and are trained on increasingly large amounts of data. Another practical limitation of modern machine learning systems is the reality that it is desirable to use "real-time" modern data streams [1]. In particular, this constraint means that

1. Machine learning systems should make real-time predictions.
2. It is easy to incorporate new data and update models continually.
3. Data is streaming – we may only get one pass over the data stream unless we expend significant resources on data storage.

These requirements mean that it is necessary for the deployed models to be fast to evaluate, and also that our models must cope with *online streaming settings*. Motivated by the difficulties and opportunities associated with this practice and the desire for *resource-efficient* methods, my work has focused on

1. *Sparse Nonlinear Models*: What are the statistical and computational limits for training models with simple descriptions (e.g. sparse or low-rank models), and what algorithms attain those limits?
2. *Low-Rank Deep Models*: Low-rank deep networks are a popular compressed parameterization of deep models which replace weight matrices with products of low rank matrices, and thereby speed up training and learning. Can we come up with initialization schemes which improve the performance of low-rank models?

3. *Memory in the Streaming Data Setting*: How much data from a stream must we store to accurately train non-smooth, strongly convex models?

### 1.1.1 Resource-Efficient Structured Model Classes

In the first two chapters, we study how to learn nonlinear model classes (monomials and neural nets) which are structured in various ways – we consider sparse monomials and deep neural nets whose weight-matrices are low-rank respectively. These kinds of restrictions on the model class lead to gains in *resource efficiency* – sparse and low-rank models are computationally easier to deploy and train. We prove that sparse nonlinear monomials are easier to learn (smaller sample complexity) while still remaining computationally efficient to both estimate and deploy, and we give both theoretical and empirical evidence for the benefit of novel nonlinear initialization schemes for low-rank deep networks. In both cases, we showcase a *blessing of nonlinearity* – sparse monomials are in some sense easier to learn compared to a linear class, and the prior state-of-the-art linear low-rank initialization methods for deep networks are inferior to our proposed nonlinear method for initialization. To achieve our theoretical results, we often make use of the theory of *Hermite polynomials* – an orthonormal function basis over the Gaussian measure.

### 1.1.2 Resource Limitations in Streaming Settings

In the last chapter, we consider *resource limitations in online streaming settings*. In particular, we consider how many data points from an oblivious adversarial stream we must store from one pass over the stream in order to output an additive approximation to the Support Vector Machine (SVM) objective. By proving stronger lower bounds on the memory complexity, we partially characterize the limitations of deploying streaming algorithms for non-smooth, strongly convex objectives to low-memory devices.

## 1.2 Our Contributions

### 1.2.1 Blessings of Non-Linearity

In Chapter 2, we follow the development from [2] and study the problem of learning a real-valued function of correlated variables. Solving this problem is of interest since many classical learning results apply only in the case of learning functions of random variables that are independent. We show how to recover a high-dimensional, sparse monomial model from Gaussian examples with sample complexity that is poly-logarithmic in the total number of variables and polynomial in the number of relevant variables. In particular, we design an attribute-efficient algorithm for learning the function $f(x) = \prod_{i \in S} x_i^{\beta_i}$, where $x \sim \mathcal{D}_x = \mathcal{N}(0, \Phi)$, that uses sample size $n = O(k^2 \cdot \text{poly}(\log(p), \log(k)))$ and runs in $\text{poly}(n, p, k)$ time. In particular, the algorithm exactly recovers the set $S$ and exponents $\beta_i$ with high probability. The algorithm does not have access to $\Phi$, and indeed, the sample size may be too small to learn it accurately. Our algorithm is based on a transformation of the variables—taking their logarithm—followed by a sparse linear regression procedure, which is statistically and computationally efficient. While this transformation is commonly used in applied non-linear regression, its statistical guarantees have never been rigorously analyzed. We prove that the sparse regression procedure succeeds even in cases where the original features are highly correlated and fail to satisfy the standard assumptions required for sparse linear regression. In particular, the nonlinearity of the monomial function – via the nonlinear logarithmic transform of the input features required for the reduction to sparse linear regression – destroys correlations in the original features and allows a key property of the problem instance, the *restricted minimum eigenvalue*, to always be strictly positive (even when the non-transformed restricted minimum eigenvalue is zero), which in turns yield sample complexity guarantees which depend only polylogarithmically on the dimension of the input features and polynomially on the sparsity. This correlation-destroying property of the nonlinear transform we apply in our algorithm is therefore a *blessing of nonlinearity*.

In Chapter 3, we follow the development from [3] and propose a novel nonlinear low-rank

initialization framework for training low-rank deep neural networks – networks where the weight parameters are re-parameterized by products of two low-rank matrices. The most successful prior existing approach, spectral initialization, draws a sample from the initialization distribution for the full-rank setting and then optimally approximates the full-rank initialization parameters in the Frobenius norm with a pair of low-rank initialization matrices via singular value decomposition. Our method is inspired by the insight that approximating the *function* corresponding to each layer is more important than approximating the parameter values – in particular, we modify the spectral initialization objective to take into account *layerwise nonlinearities*. By training ResNet and EfficientNet models [4, 5] on ImageNet [6], we empirically demonstrate that our nonlinear low-rank approximation objective outperforms spectral initialization, and moreover that optimizing the nonlinear objective *at initialization* positively correlates with improved post-training generalization error. We provably demonstrate that there is a significant gap between these two approaches for ReLU networks, particularly as the desired rank of the approximating weights decreases and as the layer width increases when the width is super-linear in dimension. Combining these theoretical results with our empirical results, we therefore identify rank, width, and dimension settings for which a different blessing of nonlinearity applies. We also provide practical heuristic algorithms to solve the layerwise function approximation problem which are no more expensive than the existing spectral initialization approach.

### 1.2.2 Stronger Memory Lower Bounds in Streaming Non-Smooth Optimization

In Chapter 4, we strengthen memory lower bounds on the space complexity of one-pass streaming $\ell_2$-regularized primal SVM, providing an essentially tight space lower bound of $\Theta\left(1/\sqrt{\epsilon}\right)$ in 1 dimension, where $\epsilon$ is the approximation error of the optimization problem. This result resolves an open problem on the space complexity of streaming SVM in 1 dimension, and also clearly applies in 2+ dimensions, and improves the 2-dimensional lower bound as well. Previously, no lower bounds were known for 1 dimension, and the best known lower bound for 2-dimensions was $\Omega(\epsilon^{-1/4})$. Notably, our 1-dimensional lower bound holds for $\lambda = \Theta(1)$, where $\lambda$ is the $\ell_2$

regularization coefficient – previous lower bounds for $d < O\left(\log(1/\epsilon)\right)$ required $\lambda = \Theta(\text{poly}(\epsilon))$.

We attain the 1-dimensional result via a novel reduction to the communication complexity problem of Augmented Index. Given an instance of Augmented Index (see Definition 1.3.14), we need to construct a 1-Dimensional Streaming SVM problem instance – this step consists of constructing a dataset $\mathbf{D}$ and a regularization parameter $\lambda \in \mathbb{R}^+$ (Definition 4.1.2). However, there are several restrictions on how we may construct the dataset. In particular, we must construct the dataset with Alice and Bob and the restrictions implicit in the information they have available – that is, Alice must encode the information from her bit string into one part of the dataset, and Bob must encode his index into the other part of the dataset, only using information from the first $i - 1$ bits of the bitstring. This setup obeys the reduction from communication problems outlined in Section 1.3.5. Finally, we show that knowledge of an $\epsilon$-optimal point for the Streaming SVM problem instance solves the Augmented Index instance, completing the reduction and the lower bound.

## 1.3 Preliminaries

### 1.3.1 Notation

In general, we use the notation $[n]$ to refer to the list $[1, 2, \ldots n]$. Also, given an ordered set, list, or vector $X$, we will denote by $X_k$ the $k^{th}$ element of $X$ according to the natural ordering (for instance, in the case of a vector $x \in \mathbb{R}^d$, we would index the dimensions from 1 to $d$ in increasing order and refer to $x_i$ as the value of $x$ for coordinate $i$). If there is no natural ordering present from the definition of $X$, assume a lexicographic ordering of the set elements with ties broken arbitrarily in a fixed manner from the start. We refer to $\mathbb{Z}^+, \mathbb{R}^+$ to denote positive integers and real-valued numbers respectively, $\|\cdot\|_p$ to refer to the $\ell_p$ norm

$$\|x\|_p = \left(\sum_{j=1}^{d} |x_i|^p\right)^{1/p}$$

for $x \in \mathbb{R}^d$, and $d_{\ell_2}$ to refer to the $\ell_p$ distance $d_{\ell_p}(x, y) = \|x - y\|_p$.

### 1.3.2 Hermite Polynomials

In Chapters 2 and 3, we work with assumptions of Gaussian data (and in Chapter 3, this assumption also reflects the practical use case as well). Consequently, to achieve our theoretical results, we make use of the theory of Hermite polynomials – an orthonormal polynomial family over the Gaussian measure. Hermite polynomials can intuitively be thought of as Fourier analysis with respect to the Gaussian measure, and allow for the simplification of various objectives involved in learning models over Gaussian data, which makes the analysis easier. In this section we introduce the main concepts involved in Hermite analysis, and take the definition, required basic facts, and development from [7].

**Definition 1.3.1** (Gaussian Space)**.** Let $f, g : \mathbb{R} \to \mathbb{R}$ be Borel functions, and define the inner product

$$\left\langle f, g \right\rangle := \quad \int_{-\infty}^{\infty} f(x) \cdot g(x) \cdot \varphi(x) dx \quad = \quad \mathbb{E}_{x \sim \mathcal{N}(0,1)} \left[ f(x) \cdot g(x) \right],$$

where

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \cdot \exp\left( -\frac{x^2}{2} \right)$$

is the Gaussian density. Then, $L_2(\mathcal{N}(0,1))$ is the space of Borel functions that satisfy

$$\langle f, f \rangle = \mathbb{E}_{x \sim \mathcal{N}(0,1)} \left[ f(x)^2 \right] < \infty.$$

**Definition 1.3.2** (Hermite Polynomials)**.** We define the Hermite polynomials, indexed by the non-negative integers, as the result of applying the Gram-Schmidt orthonormalization process to the standard polynomial basis $\{1, x, x^2, \dots\}$ with respect to the inner product defined by Gaussian space (see Definition 1.3.1), and then normalizing so that the Gaussian space norm for each polynomial is 1:

$$H_0(a) = 1, \quad H_1(a) = a, \quad H_2(a) = \frac{a^2 - 1}{\sqrt{2}}, \quad H_3(a) = \frac{a^3 - 3a}{\sqrt{6}}, \quad \dots$$

In particular, this family of functions is an orthonormal polynomial basis for $L_2\left(\mathcal{N}(0,1)\right)$. We refer to these polynomials as the *orthonormal Hermite basis functions*.

**Definition 1.3.3** (Properties of the Hermite Orthonormal Basis (from [7]))**.** The Hermite basis is an orthonormal basis over $L_2(\mathcal{N}(0,1))$. In particular, we can write for $f \in L_2\left(\mathcal{N}(0,1)\right)$

$$f(a) = \sum_{\ell=0}^{\infty} c_\ell H_\ell(a),$$

where $H_\ell(a)$ is the $\ell^{th}$ orthonormal Hermite basis function (Definition 1.3.2), and $c_\ell = \mathbb{E}_{a\sim\mathcal{N}(0,1)}[f(a)H_\ell(a)]$. The $\ell^{th}$ Hermite basis function satisfies the following relation:

$$H_\ell(a) := \frac{1}{\sqrt{\ell!}} \frac{(-1)^\ell}{\varphi(a)} \frac{d^\ell}{da^\ell}\varphi(a).$$

We also have the recurrence relation

$$H_{\ell+1}(a) = \frac{1}{\sqrt{\ell+1}}\left(aH_\ell(a) - \frac{d}{da}H_\ell(a)\right)$$

and derivative formula

$$\frac{d}{da}H_\ell(a) = \sqrt{\ell}H_{\ell-1}(a).$$

The following important lemma (see [7]) provides a rule for calculating $\mathbb{E}_{a,a'}\left[f(a)f(a')\right]$ when $a, a'$ are correlated standard Gaussian random variables.

**Lemma 1.3.4.** *Let $a, a'$ be standard Gaussian random variables with correlation $\rho$. Then, we have*

$$\mathbb{E}_{a,a'}[H_\ell(a)H_{\ell'}(a')] = \begin{cases} \rho^\ell & \text{if } \ell = \ell' \\ 0 & \text{otherwise.} \end{cases}$$

*and*

$$\begin{aligned}\mathbb{E}_{a,a'}[f(a)f(a')] &= \sum_{\ell,\ell'=1}^{\infty} c_\ell c_{\ell'} \mathbb{E}_{a,a'}[H_\ell(a)H_{\ell'}(a')] \\ &= \sum_{\ell=0}^{\infty} c_\ell^2 \rho^\ell,\end{aligned}$$

*where $c_\ell, c_{\ell'}$ are the real-valued coefficients of $f$ in the Hermite orthonormal basis.*

### 1.3.3 Compressed Sensing

In Chapter 2, we make use of results from the theory of compressed sensing, since we will reduce our non-linear sparse recovery problem into a linear sparse recovery problem to which the existing theory of compressed sensing applies. In this section, we review these useful results, following the development of [8].

The main problem in compressed sensing is to recover an $s$-sparse vector $w \in \mathbb{R}^p$ from observations of the form $Aw + \eta = b$ where $A \in \mathbb{R}^{n\times p}$ is the sensing matrix, $w \in \mathbb{R}^p$ is the signal vector, $\eta \in \mathbb{R}^n$ is the measurement noise, and $b \in \mathbb{R}^n$ is the observation vector. A commonly-used estimator is the Lasso [9]: for $\vartheta > 0$, a Lasso estimate is defined to be $\hat{w}_{\text{Lasso}}(\vartheta) \in \left\{\arg\min_{u\in\mathbb{R}^p} \frac{1}{2n}\|Au - b\|_2^2 + \vartheta\|u\|_1\right\}$. Notably, the set defined above can have cardinality greater than one when when $\text{rank}(A) < p$, which for instance occurs when $p > n$ [10]. This estimator succeeds in recovering $w$ under certain conditions on $A$. One such condition is the restricted eigenvalue condition introduced by [11] which we review here.

**Definition 1.3.5.** For $T \subset [p]$ and $q_0 > 0$, define $C(q_0, T) := \{v \in \mathbb{R}^p : \|v\|_2 = 1, \|v_{T^c}\|_1 \leq q_0\|v_T\|_1\}$. $T$ is commonly taken to be the non-zero support $S$ of the sparse vector to recover. We say the $(q_0, T, A)$*-restricted eigenvalue condition (REC)* is satisfied by matrix $A \in \mathbb{R}^{n\times p}$ if $\tilde{\lambda}(q_0, T, A) := \min_{v\in C(q_0,T)} \frac{1}{n}\|Av\|_2^2 > 0$. When $q_0$ and $T$ are apparent from context and $|T| = s$, we will simply write $\tilde{\lambda}(s, A)$.

To certify that a REC is satisfied by $A$, it suffices to consider a sparse minimal eigenvalue of

$A^T A$.

**Definition 1.3.6.** For $s \in \mathbb{N}$, the *s-sparse minimal eigenvalue* of matrix $M$ is defined by $\lambda_{\min}(s, M) := \min_{\|v\|_2=1, \|v\|_0 \leq s} v^T M v$.

It is easy to see that the constraint set in the above definition is equivalent to $C(s, 0)$. Hence, a matrix $A$ satisfies $(s, q_0)$-REC with $\tilde{\lambda}(s, q_0) \geq \lambda_{\min}\left(s, A^T A/n\right)$ whenever this latter quantity is positive.

The following well-known result about the performance of the estimator $\hat{w}_{\text{Lasso}}(\vartheta)$ is due to [11]; the specific form we state is taken from [8].

**Theorem 1.3.7.** *Consider the model $Aw + \eta = b$, and suppose the support $S$ of $w \in \mathbb{R}^p$ has size $k$, and the measurement matrix $A \in \mathbb{R}^{n \times p}$ satisfies $(q_0, S, A)$-REC with $q_0 = 3$. For any $\vartheta > 0$ such that $\vartheta \geq (2/n)\|A^T \eta\|_\infty$, the Lasso estimate $\hat{w}_{\text{Lasso}}(\vartheta)$ satisfies*

$$\|w - \hat{w}_{\text{Lasso}}(\vartheta)\|_2 \leq \frac{3\vartheta\sqrt{k}}{\tilde{\lambda}(k, 3, S, A)}.$$

### 1.3.4 Convex Optimization

In this section we introduce some concepts from convex optimization used in Chapter 4. In Chapter 4, we consider the online optimization of a strongly convex function, and require several structural properties of strongly convex functions in order to prove our memory lower bounds for streaming strongly convex optimization. We reproduce some standard definitions from convex optimization theory (see for instance [12]).

**Definition 1.3.8** (Sub-Gradient Set)**.** Consider a function $f : \mathbb{R}^d \to \mathbb{R}$. We say a vector $g \in \mathbb{R}^d$ is a *subgradient* of $f$ at $x \in \mathbf{dom}(f)$ if for all $y \in \mathbf{dom}(f)$,

$$f(y) \geq f(x) + g^\top \left(y - x\right).$$

We refer to the set of all such $g$ as $\partial f(x)$. If $f$ is convex and differentiable, then $\left|\partial f(x)\right| = 1$, and the unique element of $\partial f(x)$ is named $\nabla f(x)$. If $f$ is non-convex and differentiable and if the

subgradient exists at $x$, we also use the same definitions. If $f$ does not satisfy the condition that $|\partial f(x)| = 1$ for all $x \in \mathbf{dom}(f)$, we say it is *non-smooth.* Here $\mathbf{dom}(f)$ refers to the domain of $f$.

**Definition 1.3.9** (Convex Function)**.** Consider a function $f : \mathbb{R}^d \to \mathbb{R}$. We say $f$ is *convex* if $|\partial f(x)| > 0$ for all points $x \in \mathbf{dom}(f)$ and if it satisfies

$$f(y) \geq f(x) + g^\top \left(y - x\right)$$

for any pair of points $x, y \in \mathbf{dom}(f)$ and any $g \in \partial f(x)$.

**Definition 1.3.10** (Strongly Convex Function)**.** Consider a function $f : \mathbb{R}^d \to \mathbb{R}$. We say $f$ is *$\lambda$-strongly convex* if $|\partial f(x)| > 0$ for all points $x \in \mathbf{dom}(f)$ and if it satisfies the relation

$$f(y) \geq f(x) + g^\top \left(y - x\right) + \frac{\lambda}{2} \left\|y - x\right\|_2^2$$

for any pair of points $x, y \in \mathbf{dom}(f)$ and any $g \in \partial f(x)$.

**Definition 1.3.11** (Smooth Convex Function)**.** Consider a function $f : \mathbb{R}^d \to \mathbb{R}$. We say $f$ is *$L$-smooth* if it is differentiable at every point on its domain, and satisfies the relation

$$f(y) \leq f(x) + \nabla f(x)^\top \left(y - x\right) + \frac{L}{2} \left\|y - x\right\|_2^2$$

for any pair of points $x, y \in \mathbf{dom}(f)$.

**Definition 1.3.12** (Sub-Gradient Optimality Condition for Strongly Convex Functions)**.** Consider a strongly convex function $f : \mathbb{R}^d \to \mathbb{R}$. We say $x^*$ is the unique minimizer of $f$ if

$$0 \in \partial f(x^*).$$

### 1.3.5 Communication Complexity

In this section we introduce some concepts from communication complexity used in Chapter 4. Communication complexity lower bounds are often useful tools in proving memory lower bounds for one-pass streaming algorithms, which is the task we are concerned with in Chapter 4. We present the definition of randomized communication complexity, the Augmented Index communication problem, and a lower bound on its communication complexity (as developed in [13]).

**Definition 1.3.13** (One-Way Two-Party Randomized Communication Protocol)**.** In two-party communication, there are two parties, Alice and Bob. Alice has an input $\mathbf{a} \in \{0,1\}^a$, and Bob has an input $\mathbf{b} \in \{0,1\}^b$. Neither knows the other's input. Alice and Bob want to compute a Boolean function $f : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}$ of $\mathbf{a}$ and $\mathbf{b}$. A *randomized one-way communication protocol* is a message $M(\mathbf{a})$ from Alice to Bob that enables Bob to compute $f(\mathbf{a},\mathbf{b})$ accurately based only on the information $M(\mathbf{a}),\mathbf{b}$ with probability at least $3/4$. The *randomized communication complexity* of a given function $f$ is a lower bound on the number of bits Alice must send Bob in order for Bob to successfully compute $f$ with probability at least $3/4$.

**Definition 1.3.14** (Index and Augmented Index)**.** Augmented Index is a one-way communication problem between Alice and Bob. Alice has a bit string $\mathbf{a} := a_1 \ldots a_n$ of length $n$ with $a_i \in \{0,1\}$, and Bob has an index $i \in [n]$. Bob wants to know the value of $a_i$. A successful communication protocol of size $m$ has Alice send $m$ words to Bob, who must then output the correct value of $a_i$. This is the basic Index problem. In Augmented Index, Bob receives not only index $i$, but also the values of $a_1, \ldots, a_{i-1}$. Both communication problems have a randomized communication complexity lower bound of $\Omega(n)$ [13] – that is, Alice must send at least a constant fraction of all her bits to Bob.

Now we elaborate upon the connection between communication lower bounds for communication protocols and memory lower bounds for one-pass streaming problems.

**Definition 1.3.15** (One-Pass Streaming Problem)**.** In one-pass streaming, we receive parts of the description of the problem specification one at a time in some arbitrary order: call these parts the

*stream* $x_1, \ldots, x_n$ – here the positive integer $n$ is not known in advance. Then, we wish to compute some function $g(x_1, \ldots, x_n)$ having only seen one pass over the stream. This problem is trivially solvable if we store the whole stream – we are interested in how small a fraction of the stream we must store to still be able to reliably compute $g$ with success probability at least $3/4$ over any randomness in both the stream and the algorithm we use. Here, the term "reliably" has different meanings depending on the function $g$ – one common instantiation is to compute the value of $g$ within some additive error $\epsilon > 0$, in the case where $g$ may be real-valued.

To prove lower bounds for one-pass streaming problems, the general strategy we implement is to construct an instance of the problem the streaming algorithm is supposed to solve given a one-way communication problem. Then, Alice and Bob can use the one-pass streaming algorithm to solve their communication problem by implementing the following steps:

1. Alice runs the streaming algorithm on her input $\mathbf{a}$.
2. Alice sends the memory state of the streaming algorithm to Bob.
3. Bob finishes running the streaming algorithm, initialized to Alice's sent memory state, on his input $\mathbf{b}$, and outputs the answer.

Thus, if a one-pass streaming algorithm for the streaming problem succeeds with memory $m$, then a one-way communication protocol with communication $m$ exists as well.

# Chapter 2: Attribute-Efficient Learning of Monomials over Highly-Correlated Variables

In this chapter, we study resource limitations from the perspective of resource efficiency in machine learning – in particular, we consider a nonlinear sparse model class. A sparse model for a high-dimensional input only depends on a small number of the dimensions in the input, and thus one can hope for a tamer dependence on the dimension for the sample complexity and the computational complexity of learning. Sparsity is also often desirable from the standpoint of storage and deployment, given their smaller description complexity. In this chapter, we achieve a better understanding of when it is possible to learn certain classes of sparse non-linear models. The following development is based on the paper [14], which was co-authored with Alexandr Andoni, Rishabh Dudeja, and Daniel Hsu and was accepted to Algorithmic Learning Theory 2019.

## 2.1 Problem Statement

This section presents the formal learning problem, and introduces technical tools and notations used in our algorithm and analysis.

We consider the following canonical problem in learning theory. We observe $n$ features-response pairs $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^p \times \mathbb{R}$ drawn i.i.d. from the following model:

$$x_i \sim \mathcal{D}_x, \quad y_i = f(x_i).$$

Here, $\mathcal{D}_x$ is some distribution on $\mathbb{R}^p$. The goal is to design an algorithm to accurately estimate the unknown function $f$ with small sample complexity ($n$) and small run-time. Moreover, the unknown function $f$ may depend on only $k$ out of the $p$ features, with $k \ll p$. This models the problem of feature selection in machine learning and statistics. In this situation, a reasonable goal

is to design algorithms that are *attribute-efficient*—that is, require $n = \text{poly}(\log(p), k)$ samples and $\text{poly}(n, p, k)$ run-time. While there is a long line of work studying this problem, most existing work has one or more of the following limitations:

1. Many existing results provide algorithms and hardness results when the features are Boolean, i.e., $\mathcal{D}_x$ is supported $\{0,1\}^p$ or $\{-1,+1\}^p$. These results, however, do not necessarily reflect the difficulty or qualities of the learning problem when the features are real-valued, which is common in many practical settings.
2. A long line of work in compressed sensing and high-dimensional statistics assumes $f$ is a (sparse) linear function, but does not extend to non-linear functions.
3. To the best of our knowledge, all existing work for real-valued attributes and non-linear functions $f$ assumes that $\mathcal{D}_x$ is a product measure, for example a standard normal $\mathcal{D}_x = \mathcal{N}(0, I_p)$ [15].

In particular, the question of attribute-efficient learning is not well understood even for simple classes of non-linear functions and some canonical non-product measures. In this work, we address this gap by considering the problem of learning sparse monomials in the noiseless setting under the Gaussian measure. In particular, we assume:

$$\mathcal{D}_x = \mathcal{N}(0, \Phi), \quad f(x) = \prod_{i \in S} x_i^{\beta_i}.$$

For simplicity, we assume that covariance matrix $\Phi$ satisfies $\Phi_{i,i} = 1$ for all $i$, since we can rescale the features to have unit variance. The $\beta_i \in \mathbb{N} \cup \{0\}$ are degrees of each of the relevant variables $S \subseteq \{1, \ldots, p\}$, and $|S| = k$. Even in this simple setup, a number of standard approaches fail to give an algorithm that runs in $\text{poly}(n, p, k)$ time and has $\text{poly}(\log(p), k)$ sample complexity.

1. One natural approach is to expand the feature space by constructing all possible monomials of degree $\leq d$ and consisting of at most $k$ variables (there are at least $\Omega(p^k)$ such monomials) and using Empirical Risk Minimization. One expects this procedure to work with sample

size $n = O(\log(p^k)) = O(k \log(p))$, but the approach is computationally inefficient. Sparse regression [e.g., 9]) in the expanded feature space has similar sample complexity and run-time (and may require additional assumptions on the expanded design matrix). [16] analyze this approach when $\mathcal{D}_x$ is the uniform distribution on $\{-1, 1\}^p$ and $f$ is a sum of $s$ monomial terms and obtain a sample complexity of $O(ps^2)$ and a run-time of $O(2^p)$.

2. One can avoid explicit feature expansion by using the kernel trick. Kernel ridge regression is equivalent to $\ell_2$-penalized least squares in the expanded feature space, and can be solved in $\text{poly}(n, p, k)$ time. Standard analyses of kernel ridge regression imply that the sample complexity of this approach is proportional to the Rademacher complexity of linear classes with bounded $\ell_2$ norm in the expanded space [e.g., 17, Lemma 22]. Unfortunately, the latter quantity depends on the average squared norm of the feature vector in the expanded space, which in the Gaussian case scales like $\Omega(p^k)$. We also refer the reader to Theorem 2 of [18] for a precise analysis of the $L_2$ risk bound which makes the $p^{\Omega(d)}$ dependence explicit for kernel ridge regression when $f$ is a degree $d$ polynomial and $\mathcal{D}_x$ is supported on the unit sphere.

3. [15] describe an algorithm that learns a degree-$k$ polynomial with at most $s$ monomial terms under a product measure on $\mathbb{R}^p$, achieving a run-time and sample complexity of $\text{poly}(p, 2^k, s)$. There is a natural reduction of our problem to their setting: learn the matrix $\Phi$ and then apply a whitening transformation $\Phi^{-1/2}$ to the feature vectors. However, this reduction may convert a degree-$k$ monomial over the original features into a dense polynomial with $s = \Omega(p^k)$ terms over the new features.

We observe $n$ i.i.d. feature-response pairs $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^p \times \mathbb{R}$ from the following model:

$$x_i \sim \mathcal{N}(0, \Phi), \quad y_i = \prod_{j \in S} x_{i,j}^{\beta_j}, \tag{2.1}$$

where $S \subseteq [p] := \{1, \ldots, p\}$ is the set of relevant variables, and $\beta \in (\mathbb{N} \cup \{0\})^p$ is the vector of degrees (with $\beta_j \neq 0$ iff $j \in S$). The total degree of the monomial is $\|\beta\|_1 = \sum_{j \in S} \beta_j$. We say the monomial is $k$-sparse when $|S| = k$. (Our results also permit $\beta_j < 0$, but such a model would not

be a monomial.)

The attribute-efficient learning goal is to recover, with high probability, both $S$ and $\beta$ with sample size $n = \text{poly}(\log(p), k)$ and run-time $\text{poly}(n, p, k)$.

For simplicity, we assume that the features are standardized, so the feature variances satisfy $\Phi_{i,i} = 1$ for all $i \in [p]$. We also assume the cross-correlations satisfy

$$|\Phi_{i,j}| \leq 1 - \epsilon, \quad \forall i \neq j$$

for some $\epsilon > 0$. This latter assumption is necessary so that $\beta$ is identifiable. Indeed, if there are two perfectly correlated features, then it is impossible to distinguish them, in which case $\beta$ cannot be uniquely determined. These assumptions are not restrictive and still permit highly correlated features. In particular, the covariance matrix is permitted to be rank deficient, so some features can be linear combinations of others.

Restricting ourselves to this minimal assumption allows us to study the case where some of the data features are highly correlated and the population covariance matrix is low-rank, while excluding situations involving pairs of identical features.

## 2.2 Related Work

There are a large number of results on attribute-efficient learning under different assumptions on $\mathcal{D}_x$ and the target function $f$. We discuss representative results from each category.

*Learning with Boolean Features*

When $\mathcal{D}_x$ is supported on $\{0, 1\}^p$, learning monomials with positive integral degrees is the same as learning conjunctions. This class was shown to be PAC learnable by [19]. Furthermore, there also exists a computationally efficient and attribute-efficient learner due to [20]. When $\mathcal{D}_x$ is supported on $\{-1, +1\}^p$, then learning monomials with positive integral degrees corresponds to learning parities. Parity functions are PAC learnable using Gaussian elimination over $\mathbb{F}_2$ in time

$O(n^3)$ [21].

When a parity function involves only $k$ variables, a brute force search over all size-$k$ subsets of variables PAC learns $k$-sparse parities with an attribute-efficient sample complexity of $\text{poly}(\log(p), k)$ but has a run-time of $O(p^k)$. Finding an attribute-efficient algorithm with $\text{poly}(n, p, k)$ run-time is a long-standing open problem of [22]. Some notable improvements over the brute-force run-time include an attribute-efficient algorithm with run-time $O(p^{k/2})$ due to Dan Spielman [23], and an attribute-inefficient improper learner with sample complexity $n = O(p^{1-1/k})$ and run-time $O(p^4)$ for the noiseless case with an arbitrary distribution over $\{-1, +1\}^p$ due to [23]. Finally, an $O(p^{0.8k}\ \text{poly}\left(1/(1-2\eta)\right)$-time (but attribute-inefficient) algorithm of [24] learns parities in the noisy setting (where labels are flipped with probability $\eta$) under the uniform distribution.

*Average Case Analysis for Learning Parities*

The key bottleneck in avoiding the $p^{O(k)}$ dependence in run-time while learning $k$-sparse parities over the uniform distribution on $\{-1, +1\}^p$ in an attribute-efficient manner is that it is not clear how to decide if a feature is relevant or not without considering its interaction with every possible set of $k-1$ features. In light of this, [25] study the problem when $f$ is a DNF with $s$ terms ($k$-sparse parities are DNFs of size $s = 2^k$) and show that a natural greedy feature selection algorithm can learn such $f$ in time and sample complexity $\text{poly}(s, p)$ under a product distribution whose parameters are adversarially chosen and then randomly perturbed by a small amount. Similarly, [26] identify a property of the function $f$ called the unique sign property (USP) that facilitates learning. For functions $f$ defined on $\{-1, +1\}^p$ satisfies USP and depends on just $k$ features, their algorithm learns $f$ under the uniform distribution with run-time and sample complexity $\text{poly}(p, 2^k)$. In the spirit of smoothed analysis, they show the USP is satisfied when an adversarially chosen $k$-sparse function $f$ is perturbed by a small amount of random noise.

*Learning with Real-Valued Features*

When $\mathcal{D}_x$ is a product measure and the features are real-valued (for example, the uniform measure on $[-1, 1]^p$ or the standard Gaussian measure on $\mathbb{R}^p$), [15] consider the problem of learning sparse polynomials of degree $d$ that contain at most $s$ monomial terms with additive noise. They show a surprising result that in contrast to learning sparse parities with noise, it is possible to avoid a $p^d$ dependence in run-time. They design an algorithm with $\text{poly}(p, 2^d, s)$ sample complexity and run-time. At the heart of their approach are linear-time correlation tests that detect if a feature participates in the highest degree (lexicographically) monomial. Once they detect all features participating in the highest degree monomial, they remove it, and recurse on the residual polynomial. An interesting property of their algorithm is that it never looks at the signs of either the responses or the features. This highlights the fact that in the real-valued case the magnitudes of the observations contain valuable information (which was not present in the case of parities) that can be leveraged to design algorithms with sub-$O(p^d)$ run-time. The algorithm we propose has the same property. While the class of functions we can handle is smaller (1-sparse polynomials), we are able to handle extremely large correlations between features. In this highly-correlated setting, it is not immediately clear how to analyze the correlation tests proposed by [15]. Hence, we rely on a completely different technique: computing a log-transform of the responses and using sparse linear regression.

### 2.3 Main Results

**Our contributions.** We design an attribute-efficient algorithm for learning the function $f(x) = \prod_{i \in S} x_i^{\beta_i}$, where $x \sim \mathcal{D}_x = \mathcal{N}(0, \Phi)$, that uses sample size $n = O(k^2 \cdot \text{poly}(\log(p), \log(k)))$ and runs in $\text{poly}(n, p, k)$ time. In particular, the algorithm exactly recovers the set $S$ and exponents $\beta_i$ with high probability. The algorithm does not have access to $\Phi$, and indeed, the sample size may be too small to learn it accurately.

Our algorithm provably succeeds as long as $\max_{i \neq j} |\Phi_{i,j}| < 1$. This is, in a sense, the minimal

assumption on $\Phi$: if violated, this model is not even *identifiable*. To put this into context, it is instructive to contrast to the case when $f$ is a sparse *linear* function, under the same input distribution $x \sim \mathcal{N}(0, \Phi)$. For the latter problem, there is no known computationally efficient and attribute-efficient algorithm to estimate the set $S$ under similarly-weak assumptions on $\Phi$.

The key algorithmic technique is to apply a log-transform to the features and response, and reduce the problem to a sparse linear regression problem. While this is a commonly-used technique in applied statistics, to the best of our knowledge, it has not been rigorously analyzed before. We show that this log-transform is precisely what allows us to provably learn $f$ when it is a monomial. Specifically, we analyze how the covariance matrix changes after the log-transform, showing that the log-transform eliminates linear dependencies between two or more features – this "blessing of non-linearity" allows us to ensure a restricted strong convexity property always holds (see Figure 2.1). To again contrast with the case of learning sparse *linear* functions, the linear dependencies are precisely the obstacle for designing computationally-efficient and attribute-efficient algorithms.

In this section, we present our learning algorithm and its performance guarantees.

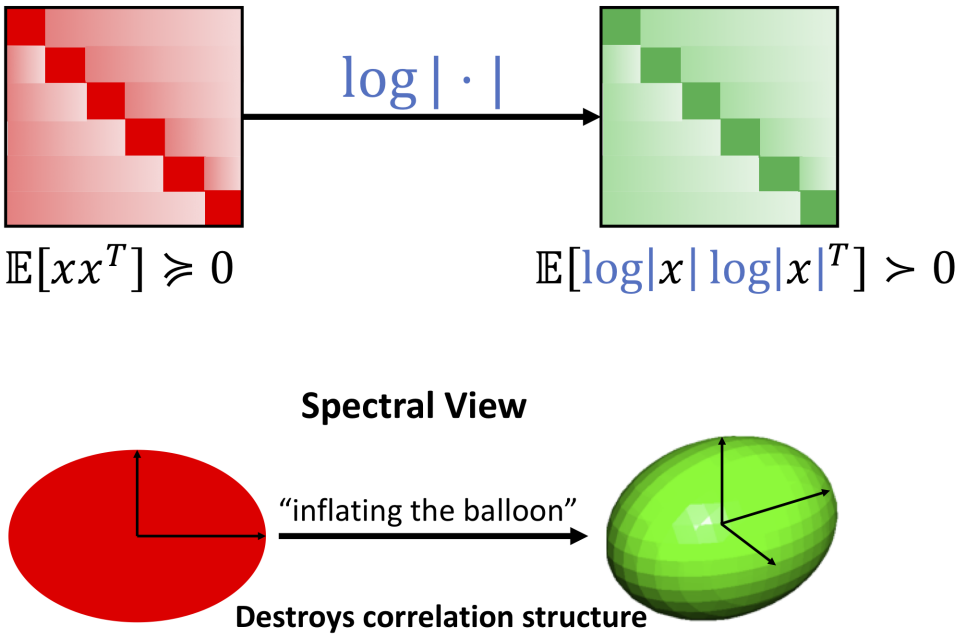



Figure 2.1: The log transform inflates the spectrum of the resulting data covariance matrix, ensuring we can lower bound the minimum restricted eigenvalue by a positive constant, leading to our sample complexity bound.

### 2.3.1 Algorithm

Our proposed attribute-efficient learning algorithm, given as Algorithm 1, is based on a log-transformation of the data, followed by sparse linear regression. For concreteness, we use Lasso [9] for the second step, although other sparse regression methods could also be used.

**Algorithm 1** Learn Sparse Monomial

**Require:** data matrix $X \in \mathbb{R}^{n \times p}$, responses $y \in \mathbb{R}^n$, regularization parameter $\vartheta > 0$

1: Apply $\log\left(|\cdot|\right)$ transformation to data and responses, element-wise: $\hat{X} \leftarrow \log\left(|X|\right)$ and $\hat{y} \leftarrow \log\left(|y|\right)$.
2: Solve Lasso optimization problem: $\hat{\beta} \leftarrow \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2n}\|\hat{X}\beta - y\|_2^2 + \vartheta\|\beta\|_1$.
3: Select variables: $\hat{S} \leftarrow \{j \in [p] : \hat{\beta}_j \neq 0\}$.
4: **return** $\hat{S}$ and $\hat{\beta}$.

The logarithm transformation is a folklore technique in applied statistics [see, e.g., 27] but, to the best of our knowledge, has not received a non-trivial theoretical analysis in a setting similar to ours. We compose the log-transform with absolute value in Algorithm 1 to ensure non-negativity.

We make two observations about the $\log\left(|\cdot|\right)$-transformation. First, it converts the monomial model in Eq. (2.1) to the following:

$$\log\left(|y_i|\right) = \sum_{j \in S} \beta_j \log\left(|x_{i,j}|\right). \tag{2.2}$$

Second, the transformation is only applicable to non-zero entries in the data matrix $X$ and response vector $y$. For the Gaussian data in our problem setup, all entries are non-zero almost surely.

So, after the transformation, the problem reduces to a linear sparse recovery problem, which can be efficiently solved using well-known techniques from compressed sensing under appropriate conditions on the design matrix (e.g., restricted eigenvalues).

The following simple proposition formalizes the reduction.

**Proposition 2.3.1.** *A unique solution $\hat{\beta}$ to the transformed model in Eq.* (2.2) *is the unique solution to the original model in Eq.* (2.1).

### 2.3.2 Performance Guarantees

Our approach to analyzing Algorithm 1 is based on applying the performance guarantee for Lasso from Theorem 1.3.7. Because we apply Lasso to data from the $\log\left(|\cdot|\right)$-transformed model in Eq. (2.2), we need to prove that REC is satisfied by $\hat{X} = \log\left(|X|\right)$. As noted before, it is sufficient to lower-bound $\tilde{\lambda}(k, 3, S, \hat{X}/\sqrt{n})$. This is the content of the following theorem.

**Theorem 2.3.2.** *Let $\delta \in (0, 1)$ be an arbitrary confidence parameter. Suppose the covariance matrix $\Phi$ satisfies $\Phi_{i,i} = 1$, $\forall i \in [p]$ and $\max_{i \neq j} |\Phi_{i,j}| < 1 - \epsilon$. Then, the $\log\left(|\cdot|\right)$-transformed design matrix $\hat{X} = \log\left(|X|\right)$ for $X$ taken from the model in Eq. (2.1) with true support $|S| = k$ satisfies*

$$\tilde{\lambda}\left(k, \frac{1}{\sqrt{n}}\hat{X}\right) \geq \frac{1}{5}\sqrt{\frac{\epsilon}{\log(16k) + 2}},$$

*with probability $1 - \delta$, provided that*

$$n \geq C \cdot \frac{k^2 \log(2k)}{\epsilon} \cdot \log^2\left(\frac{2p}{\delta}\right) \cdot \log^2\left(\frac{k \log(k)}{\epsilon} \log\left(\frac{2p}{\delta}\right)\right). \tag{2.3}$$

*In the above display, $C$ is a universal constant.*

Therefore, applying Theorem 1.3.7, we immediately get as a corollary the following performance guarantee for Algorithm 1.

**Corollary 2.3.3.** *Let $\delta \in (0, 1)$ be an arbitrary confidence parameter and $\vartheta$ be the regularization parameter. Suppose the covariance matrix $\Phi$ satisfies $\Phi_{i,i} = 1$ for all $i \in [p]$ and $\max_{i \neq j} |\Phi_{i,j}| < 1 - \epsilon$, and that the sample size $n$ satisfies the inequality in Eq. (2.3). For $X$ and $y$ taken from the model in Eq. (2.1) with $|S| = k$, Algorithm 1 returns $\hat{\beta}$ such that, with probability at least $1 - \delta$,*

$$\|\hat{\beta} - \beta\|_2 \leq 15\vartheta\sqrt{\frac{k(\log(16k) + 2)}{\epsilon}}.$$

*Remark* 2.3.4. We note that, as $\vartheta \to 0$, $\|\hat{\beta} - \beta\|_2 \to 0$, and hence, Algorithm 1 recovers $\beta$ exactly. Furthermore, in the limit $\vartheta \to 0$, Algorithm 1 is equivalent to the Basis Pursuit estimator [28] defined as:

$$\hat{\beta}^{\mathrm{BP}} = \arg\min_{v \in \mathbb{R}^p} \|v\|_1 \text{ subject to } \hat{X}v = \hat{y}.$$

In particular, this means that under the conditions of Corollary 2.3.3, the Basis Pursuit estimator satisfies

$$\hat{\beta}_j^{\mathrm{BP}} = 0 \quad \forall j \notin S, \quad \hat{\beta}_j^{\mathrm{BP}} = \beta_j \quad \forall j \in S.$$

*Remark* 2.3.5. Suppressing logarithmic factors in $p$ and $k$, the above result shows that Algorithm 1 succeeds in recovering the monomial with high probability with $\tilde{O}(k^2/\epsilon)$ samples.

*Remark* 2.3.6. If we observe data with multiplicative noise, that is,

$$y_i = e^{\eta_i} \cdot \prod_{j \in S} x_{i,j}^{\beta_j} \tag{2.4}$$

where $\eta_i \in \mathbb{R}$ is a zero-mean sub-gaussian noise (e.g., $\eta_i \sim \mathcal{N}(0, \sigma^2)$), then the $\log|\cdot|$ transform reduces our problem to a noisy compressed sensing problem. Hence we can still apply Theorems 1.3.7 and 2.3.2, as long as we set the parameter $\vartheta$ according to the noise level. If the sample size is large enough relative to the noise level, we can exactly recover the degrees by rounding $\hat{\beta}$ to nearest integers. The details are straightforward and omitted.

## 2.4 Proofs

### 2.4.1 Additional Notation and Terminology

Let $X = [x_1|\cdots|x_n]^T \in \mathbb{R}^{n \times p}$ be the data matrix, and let $y = [y_1|\cdots|y_n]^T \in \mathbb{R}^n$ be the vector of responses. Throughout, log denotes the natural logarithm, and applying log or absolute value

to a matrix or vector means these operations are taken element-wise. For any matrix $M$, we write $M^{(l)}$ to denote its $l$-th Hadamard power, so $M^{(l)}_{i,j} = M^l_{i,j}$. We define the following notations:

$$z := \log(|x|), \qquad \Sigma := \mathbb{E}_z[zz^T], \qquad \hat{\Sigma} := \frac{1}{n}\sum_{i=1}^{n} z^{(i)} z^{(i)^T}.$$

where $\log(|\cdot|)$ is applied element-wise and $z^{(i)}$ denotes the $i^{th}$ data point.

### 2.4.2 Restricted eigenvalues for the $\log\left(|\cdot|\right)$-transformed data

In this section, we present the main technical results used in the proof of Theorem 2.3.2. We define the following notations:

$$z := \log(|x|), \qquad \Sigma := \mathbb{E}_z[zz^T], \qquad \hat{\Sigma} := \frac{1}{n}\sum_{i=1}^{n} z^{(i)} z^{(i)^T}.$$

where $\log(|\cdot|)$ is applied elementwise and $z^{(i)}$ denotes the $i^{th}$ empirical data point. We also use $z_i$ to denote the $i^{th}$ feature of $z$. The proof of Theorem 2.3.2 involves three steps:

1. We first determine an explicit formula for the population covariance matrix $\Sigma$ given in Lemma 2.4.1.
2. We leverage this explicit formula to prove a lower bound on $\lambda_{\min}(\Sigma)$ and $\tilde{\lambda}(k, \Sigma^{1/2})$ in Theorem 2.4.2.
3. Finally, we show that $\tilde{\lambda}(k, \hat{\Sigma}^{1/2})$ concentrates around $\tilde{\lambda}(k, \Sigma^{1/2})$ in Lemma 2.4.5.

One of our main technical contributions is a lower bound on $\tilde{\lambda}_{\min}(k, \Sigma^{1/2})$ under very weak assumptions about the covariance matrix $\Phi$ of the original features, namely, $|\Phi_{i,j}| < 1 - \epsilon$ for any $i \neq j$. In particular, this holds even in cases where $\Phi$ is low-rank or $\Phi^{1/2}$ doesn't satisfy REC. Intuitively, this result holds because the logarithm, a highly non-linear operation, destroys the linear dependence structure of a low-rank matrix as long as no two features are perfectly correlated (which is anyway necessary for identifiability).

### 2.4.3 Properties of $\log\left(|\cdot|\right)$-transform

The following key lemma provides several useful properties of the $\log|\cdot|$ transform, culminating in an explicit and convenient expression for $\Sigma$ in terms of $\Phi$.

**Lemma 2.4.1.** *Let $x \sim \mathcal{N}(0, \Phi)$ where $\Phi_{i,i} = 1$ for all $i \in [p]$. Define $z = \log\left(|x|\right)$. Then:*

1. *The random variable $z_i$ has bounded variance, in particular,* $\mathrm{var}(z_i) = \pi^2/8$.
2. *The function $a \mapsto \log(|a|)$ admits the following expansion in the Hermite polynomial basis $\{H_l\}_{l\geq 0}$:*

$$\log(|a|) = \sum_{l=0}^{\infty} c_{2l} H_{2l}(a), \quad c_{2l} = \frac{(-1)^{l-1} 2^{l-1} (l-1)!}{\sqrt{(2l)!}}.$$

3. $\mathbb{E}[z_i z_j] = \sum_{l=0}^{\infty} c_{2l}^2 \Phi_{i,j}^{2l}$.
4. $\Sigma = c_0^2 \boldsymbol{1}_{p\times p} + \sum_{l=1}^{\infty} c_{2l}^2 \Phi^{(2l)}$, *where $\boldsymbol{1}_{p\times p}$ is the $p \times p$ matrix of all 1's.*

*Proof sketch.*

1. The challenge in calculating the variance of $z_i$ is that integrals involving log moments and the Gaussian measure are not analytically easy to work with. To get around this, we leverage the fact that for any non-negative random variable $a$ and any $m \in \mathbb{N}$,

$$\mathbb{E}_a[\log^m a] = \lim_{\nu\to 0} \tfrac{\mathrm{d}^m}{\mathrm{d}\nu^m} \mathbb{E}_a[a^\nu].$$

   When $a = |x_i|$, the RHS of the above expression is available in closed-form. (This is the "Replica Trick" from statistical physics [29].)

2. Since the Hermite polynomials form a complete orthonormal basis for $L_2(\mathcal{N}(0,1))$, we can compute $c_l$ by the integral:

$$c_l = \int_{-\infty}^{\infty} \log\left(|a|\right) \cdot H_l(a) \cdot \frac{e^{-a^2/2}}{\sqrt{2\pi}}\, \mathrm{d}a.$$

We calculate the above integral by-parts and by leveraging the recursive structure of Hermite Polynomials.

3. The rationale behind expanding the $\log(|a|)$ in the Hermite polynomial basis is that there is a clean formula between the correlation of Hermite polynomials applied to two correlated Gaussian random variables [see, e.g., 7]:

$$\mathbb{E}[H_l(x_i)H_m(x_j)] = \Phi_{i,j}^l \mathbb{1}_{\{l=m\}}.$$

Using this fact and the expansion of $\log|\cdot|$ gives us the expression for $\mathbb{E}[z_i z_j]$.

4. The formula for $\Sigma$ immediately follows given the general expression for $\Sigma_{i,j} = \mathbb{E}[z_i z_j]$.

See Appendix A.2 for a detailed proof. □

### 2.4.4 Restricted eigenvalues of population covariance matrices

**Theorem 2.4.2.** *Let $\Phi$ be any covariance matrix with $\Phi_{i,i} = 1$ for all $i$ and $|\Phi_{i,j}| < 1 - \epsilon$ for $i \neq j$, and let $\Sigma = \mathbb{E}_z[zz^T]$ for $z \sim \mathcal{N}(0, \Phi)$. The following inequalities hold.*

1. $\lambda_{\min}(\Sigma) \geq \dfrac{\pi^2}{8}\lambda_{\min}(\Phi)$.
2. $\tilde{\lambda}\left(k, \Sigma^{1/2}\right) \geq \displaystyle\sum_{\ell=1}^{\frac{1}{2}\left\lfloor \frac{\log(16k))}{\log(\frac{1}{1-\epsilon})} \right\rfloor} \frac{\tilde{\lambda}\left(k, [\Phi^{(2\ell)}]^{1/2}\right)}{5\ell^{3/2}} + \frac{2}{5}\sqrt{\frac{2\log((1-\epsilon)^{-1})}{\log\left(\frac{16k}{1-\epsilon}\right) + \max\{2, \log((1-\epsilon)^{-1})\}}}$.

*Remark* 2.4.3. If $\Phi$ already has a positive minimum eigenvalue, we automatically have a constant multiplicative factor improvement after applying the $\log\left(|\cdot|\right)$-transformation. But even if $\lambda_{\min}(\Phi) = 0$, we still obtain a positive lower bound on $\tilde{\lambda}(k, \Sigma)$.

*Remark* 2.4.4. In Appendix A.3 (specifically Theorem A.3.5), we also prove a simpler minimum eigenvalue lower bound of $\lambda_{\min}(\Sigma) \geq \Omega(\sqrt{\epsilon/\log(p)})$, which is similar to the lower bound on $\tilde{\lambda}(k, \Sigma^{1/2})$ except with $\log(k)$ replaced by $\log(p)$. The improvement in Theorem 2.4.2, which has no explicit dependence on the ambient dimension $p$, uses a restricted form of Gershgorin's Circle

Theorem (Lemma A.3.7). Using either lower bound is sufficient to obtain the sample complexity guarantees in Theorem 2.3.2, but the improved bound highlights the power of the $\log(|\cdot|)$-transformation and may be of independent interest.

*Proof sketch.* We recall the explicit expression for $\Sigma$ from Lemma 2.4.1:

$$\Sigma = c_0^2 \mathbf{1}_{p\times p} + \sum_{l=1}^{\infty} c_l^2 \Phi^{(l)}.$$

The definitions of $\lambda_{\min}(\cdot)$ and $\tilde{\lambda}(k,\cdot)$ imply both are superadditive:

$$\tilde{\lambda}(k, \Sigma^{1/2}) \geq \sum_{l=1}^{\infty} c_l^2 \tilde{\lambda}(k, [\Phi^{(l)}]^{1/2}).$$

We obtained the bound on $\lambda_{\min}(\Sigma)$ by applying a linear algebraic result from [30] which implies that $\lambda_{\min}(\Phi^{(l)}) \geq \lambda_{\min}(\Phi)$. As for the second expression, we split the infinite sum into two parts and apply a restricted version of the Gershgorin Circle Theorem to the second part (see Lemma A.3.7 in Appendix A.3.2). We then analyze how fast the coefficients $c_l$ of the remaining terms decay to 0. We refer the reader to Appendix A.3 for a complete proof. □

### 2.4.5 Analysis of the empirical covariance matrix

The last piece required to complete the proof Theorem 2.3.2 is a concentration result about $|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})|$. This is stated in the following lemma.

**Lemma 2.4.5.** *Let $\delta \in (0,1)$ be an arbitrary confidence parameter. With probability $1-\delta$,*

$$|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})| \leq Ck\left(\sqrt{\frac{(\log(3/\delta) + 2\log(p))}{n}} + \frac{\log^2(n)(\log(3/\delta) + 2\log(p))^2}{n}\right).$$

*In the above display $C$ is a universal constant.*

*Proof sketch.* We apply Theorem 4.2 of [31] after verifying that the log-transformed features $z_i$ are entry-wise sub-exponential. See Appendix A.4 for a detailed proof. □

Table 2.1: The results of our exact recovery simulation for various choices of sample size $n$ and sparsity $k$ over 100 independent trials.

| Estimated probability of exact recovery | | | | |
|---|---|---|---|---|
| | $k = 2$ | $k = 4$ | $k = 6$ | $k = 8$ |
| $n = 128$ | 0.99 | 0.76 | 0.01 | 0.00 |
| $n = 384$ | 1.00 | 1.00 | 0.97 | 0.22 |
| $n = 640$ | 1.00 | 1.00 | 1.00 | 0.88 |

## 2.5 Simulations

We conducted a simple simulation to evaluate the robustness of our procedure to small *additive* noise (which our analysis does not cover). The $p = 512$-dimensional feature vectors are $x_i \sim \mathcal{N}(0, \Phi)$ for a rank-$p/2$ covariance matrix $\Phi$ given by

$$\Phi := \begin{bmatrix} I & \sqrt{\frac{2}{p}}H \\ \sqrt{\frac{2}{p}}H & I \end{bmatrix}.$$

Above, $I$ is the $(p/2) \times (p/2)$ identity matrix, and $H$ is the $(p/2) \times (p/2)$ Hadamard matrix. The responses are $y_i = \prod_{j \in S} x_{i,j} + \eta_i$ for independent $\eta_i \sim \mathcal{N}(0, \sigma^2)$, where $\sigma = 10^{-3}$, and $S = \{1, \ldots, k/2, p/2 + 1, \ldots, p/2 + k/2\}$.

Algorithm 1 was implemented with a setting of $\vartheta = \vartheta(n, p, \sigma)$ as suggested by [11]. For different values of the cardinality $k = |S|$ and sample size $n$, we estimated the probability of exact recovery of $S$ on 100 independent trials:

The results suggest that our procedure tolerates some level of additive noise, but that the sample size may need to increase significantly with the sparsity level $k$. This is reasonable, as the signal-to-noise ratio decreases exponentially with $k$.

# Chapter 3: Nonlinear Initialization Methods for Low-Rank Neural Networks

In this chapter, we study resource limitations in machine learning from the perspective of resource-efficiency: By restricting the weight matrices of deep neural networks to be low-rank, both the forward and backward passes of the network become more computationally efficient to execute. We consider deep neural networks whose weight matrices are parameterized by products of two low-rank matrices, e.g. $W = UV^\top$ where $W \in \mathbb{R}^{m\times d}, U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{d\times r}$. Since both the evaluation and training of deep networks depends on matrix-vector products, the number of addition and multiplication operations in the evaluation of $Wx$ (corresponding to a fully-connected layer) decreases from $\Theta(d \cdot m)$ to $\Theta(r \cdot (d + m))$.

The work in this section was performed at Google Brain during a Research Internship lasting from June 2021 to February 2022 with Rakesh Shivanna, Maheswaran Sathiamoorthy, Sagar Jain, and Ed Chi. The paper [3] is currently in submission.

## 3.1 Problem Statement

Training deep networks is a canonical task in modern machine learning. Training and serving deep networks with very large parameter counts efficiently is of paramount importance in recent years, since significant performance gains on a variety of tasks are possible simply by scaling up parameter counts [32, 33, 34, 35, 36, 37]. Further theoretical evidence suggests that requiring the resulting learned networks to be smooth functions (and therefore, in some sense, robust to perturbations) entails even larger parameter counts [38].

However, there are significant computational difficulties with both training and deploying such large models. Most existing works focus on efficiently deploying trained deep networks and uses

a variety of approaches including sparsifying neural network weights [39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50], model distillation [51, 52], low-rank post-processing [53, 54, 55], and mixtures thereof [56]. Unfortunately, these methods do not address the question of efficient training.

We study the problem of learning low-rank neural networks in this chapter and ask:

1. *Are there better initialization schemes for low-rank deep neural networks that improve post-training generalization error?*
2. *What factors govern the theory of choosing a low-rank initialization method?*

## 3.2 Related Work

### 3.2.1 Training-Time Efficient Deep Networks

Another general approach to improving speed and memory at both train *and* inference time is to replace each weight matrix $W \in \mathbb{R}^{d \times m}$ with a computationally efficient representation which improves the training speed of standard gradient-based optimization methods, while ideally not losing out on the representation capacity too much. A variety of papers have taken this approach with varying techniques including fast Fourier transform-inspired sparse matrix decompositions, low-rank factorizations, orthogonal basis kernel approximations, and sketching-based approaches [57, 58, 59, 60, 61, 62, 63, 64, 65, 66]. In the sparse deep network literature, there also exist methods to perform pruning before or during training (to also get efficiency gains during training) [67, 68, 69, 48, 49, 70].

### 3.2.2 Low-Rank Factorized Networks

Given the broad spectrum of proposed methodologies, it may be unclear to practitioners which families of methods are most practical to use. Two recently popular approaches are unstructured sparse pruning and low-rank factorization. Low-rank methods do not require specialized hardware to convert smaller parameter counts into compute savings, unlike the sparse methods [71, 72, 73].

For a fully-connected layer, the basic idea of a low-rank layer (or a *factored layer*) is to parameterize the network weights $W \in \mathbb{R}^{d\times m}$ with a low-rank matrix product $UV^\top$, where $U \in \mathbb{R}^{d\times r}, V \in \mathbb{R}^{m\times r}$ and $r$ is the (user-selected) rank. Simple generalizations exist for other standard layers including convolution and attention layers [58]. Low-rank deep networks reduce parameter counts (thus saving memory) as well as the number of ops required for matrix-vector multiplication: $(d+m)\cdot r$ vs. $d \cdot m$.

[58] demonstrate that if one pays attention to proper initialization and regularization, low-rank methods outperform sparse pruning approaches in many domains, contrary to existing beliefs that sparse methods outperform low-rank methods in parameter count savings. In particular, a low-rank initialization scheme called *spectral initialization* is crucial to achieve better performance – initialization schemes are in general quite important for achieving good performance in neural network training [74, 75, 76, 77, 78, 79, 80, 81, 82]. Spectral initialization samples a full-rank matrix $W \in \mathbb{R}^{d\times m}$ from a known init distribution, factorizes $W$ as $A\Sigma^{1/2}, \Sigma^{1/2}B^\top$ via singular value decomposition (SVD), and initializes $U$ and $V^\top$ with these factors.

However, there are no explanations for why this approach, which approximates the full-rank weight parameters *at initialization*, yields improved performance. Thus, it is a natural next step to develop better theoretical understanding of the properties of the low-rank network learning problem, with the hope that it will aid us in finding improved methods for training low-rank versions of deep networks and in uncovering the principles of learning low-rank network approximations.

### 3.2.3 Low-Rank Approximation Theory

At the heart of the approach we take (see Problem 3.3.6 in Section 3.3.4) is the idea that we want to make an alternate (nonlinear) low-rank approximation to a matrix $W$. Low-rank approximation has been long studied in the theoretical computer science literature (see [83, 84] for thorough surveys of the topic). One particularly related line of work is the masked low-rank approximation literature [85]. The basic idea of masked low-rank approximation is that we want to find a low-rank $Y$ that minimizes $\left\|M \circ (W - Y)\right\|_F^2$, where $M$ is a mask applied as an elementwise product

(the Hadamard product). This problem captures many different problems studied in the literature under various structural assumptions on $M$ (for instance, the case where $M$ is a real-valued non-negative is known as weighted low-rank approximation, and was studied by [86]). [85] study the case where $M$ is a binary mask and provide bicriteria approximation guarantees since the problem is hard in general. It is interesting to consider the connection between the binary masked low-rank approximation problem of [85] and our problem, where we apply ReLU to *Gaussian samples* and mask only the *output*. It would be interesting to establish further connections between our problem setting and other low-rank approximation settings, perhaps by adopting our setup from Problem 3.3.6 but possibly changing the input distribution. It seems plausible that for some choices of input distribution, one could make the problem computationally hard. For more background on the low-rank approximation literature, see the discussion in [85].

### 3.3 Main Results

Our main contributions are as follows:

1. The identification of the *function approximation at initialization* framework (Definitions 3.3.4, 3.3.5) and the simpler (and parallelized) Nonlinear Low-Rank Approximation objective (**NLRA**) (Problem 3.3.6) for initializing low-rank networks (and it also applies to other structured network approximation schemes);
2. Practical algorithms (Algorithms 2, 3) and corresponding empirical results which validate that our layerwise low-rank initialization scheme works as an efficient drop-in replacement for the commonly used spectral initialization with improved performance (on average 0.3% top-1 accuracy gain on ImageNet [6], and as much as up to 1.2% accuracy improvement);
3. Empirical observations that taking nonlinearities of layers into account for the initialization scheme improves accuracy for lower-rank layers and larger input dimension and width;
4. Theoretical proof that in settings used in practice, the sub-optimality of the spectral initialization with respect to the **NLRA** objective grows with decreasing rank and increasing width

when width is super-linear in input dimension;

5. Empirical confirmation that optimization of the **NLRA** objective at initialization positively correlates with decreasing post-training test error, thereby also suggesting that function approximation at initialization is a more useful initialization approach than parameter approximation at initialization for downstream test error.

### 3.3.1 Additional Notation and Terminology

**Definition 3.3.1** (Deep Feed-forward Neural Network)**.** A *feed-forward neural network* of depth $L$ is a function $f : \mathbb{R}^d \to \mathbb{R}^m$ that is parameterized by a list of matrices $W_1, \ldots, W_L$ and a list of non-linearity functions $\sigma_i : \mathbb{R}^{m_i} \to \mathbb{R}^{m_i}$

$$f(x) := \sigma_L \left( W_L \sigma_{L-1} (W_{L-1} \sigma_1 (\ldots W_1 x)) \right),$$

where $x \in \mathbb{R}^d$, $W_i \in \mathbb{R}^{d_i \times m_i}$ for $d_i, m_i \in \mathbb{Z}^+$ for $i \in [L]$ and where we refer to $d_1$ as $d$ and $m_L$ as $m$.

One can impose further structure upon the matrices $W_i$ to recover more specific classes of neural network. For instance, the popular 2-*dimensional convolutional neural networks*, which are commonly used in computer vision, interpret the input $x \in \mathbb{R}^d$ as a 2-dimensional grid (corresponding to a picture, for instance), and require the matrices $W_i$ to satisfy a block-diagonal structure where each block is a sparse matrix corresponding to a 2-dimensional convolution across the grid. These blocks need not have the same dimensions.

It is common for the non-linearities $\sigma_i$ from Definition 3.3.1 to be the same for $i \in [L-1]$, and specified via a one-dimensional function which is then applied element-wise. In Chapter 3, we will often take the non-linearity to be the Rectified Linear Unit (ReLU), which is defined by $\sigma_{\text{ReLU}}(x) := \max(0, x)$ for $x \in \mathbb{R}$. The final non-linearity is chosen depending on the learning task – one common choice for $\sigma_L$ is the softmax operation:

**Definition 3.3.2** (Softmax Function)**.** The *softmax* is defined as a function from $\mathbb{R}^m \to \mathbb{R}^m$

$$\text{softmax}(z)[k] := \frac{\exp(z_k)}{\sum_{j=1}^{m} \exp(z_j)},$$

for $z \in \mathbb{R}^m$, yielding a probability distribution on the probability simplex $\Delta_{m-1}$ (the $\ell_1$ ball of radius 1 restricted to the positive orthant of $\mathbb{R}^m$).

The softmax layer is often used to convert neural net outputs into probabilities over multiple classes in the context of multi-class prediction tasks.

We let $\mathcal{D}$ be a common initialization distribution used for each weight matrix, typically defined as a product distribution over the entries of the matrix. We denote the Frobenius norm by $\|\cdot\|_F$.

The *singular value decomposition* (SVD) has $W = U\Sigma V^T$, with $U \in \mathbb{R}^{d\times m}, \Sigma \in \mathbb{R}^{m\times m}, V \in \mathbb{R}^{m\times m}$. $U$ and $V$ have orthogonal columns hand $\Sigma$ is a diagonal matrix. The rank-$r$ *compressed singular value decomposition* of a matrix $W \in \mathbb{R}^{d\times m}$, we write $W_r = U_r\Sigma_r V_r^\top$, where $U_r \in \mathbb{R}^{d\times r}, \Sigma_r \in \mathbb{R}^{r\times r}$, and $V_r \in \mathbb{R}^{m\times r}$, and where the columns of $U_r$ and $V_r$ are orthogonal and $\Sigma_r$ is a diagonal matrix. The $r$ coordinates that are chosen correspond to the top $r$ singular values of $W$. The uncompressed form of the rank-$r$ SVD appends additional 0s to the diagonal of $\Sigma_r$ and a set of orthonormal vectors to the columns of $U$ and $V$ (which are respectively orthonormal to the existing columns of $U$ and $V$). By default we use the compressed form unless otherwise noted. We also use $\text{Unif}([n])$ to denote the uniform distribution over the elements of $[n]$.

### 3.3.2 Function Approximation at Initialization

In this section, we propose a general framework for initializing low-rank deep networks given a full-rank initialization scheme. We also describe a special case of our framework which is provably efficient to implement when the non-linearity is ReLU and the target rank $r$ is a constant.

*Function Approximation at Initialization Framework*

The key idea behind our approach is to mimic the full-rank initialization distribution as closely as possible. [58] follows this principle to argue for the spectral initialization approach to initialize the low-rank weights: the idea is to match the weight matrices in Frobenius norm as closely as possible using SVD. More precisely:

**Definition 3.3.3** (Spectral Initialization)**.** Given a full-rank weight $W \in \mathbb{R}^{d\times m}$ initialized according to distribution $\mathcal{D}$ and a target rank $r$, *spectral initialization* is the following procedure:

1. Sample $W \sim \mathcal{D}$.
2. Factorize $U\Sigma V^\top = W$ via singular value decomposition, where $U \in \mathbb{R}^{d\times r}, \Sigma \in \mathbb{R}^{r\times r}, V \in \mathbb{R}^{m\times r}$.
3. Output factor initialization $\hat{U} = U\Sigma^{1/2}$, $\hat{V} = V\Sigma^{1/2}$.

In words, we initialize the factors which best approximate $W$, the sampled full-rank initialization, in Frobenius norm.

However, it is not clear that this metric for matching low-rank to full-rank functions accurately captures what is important in the approximation: Ultimately, we are initializing highly nonlinear functions, and various terms in the weights may be less important than others (in a post-training setting, [53] demonstrates the efficacy of taking nonlinearities into account). Thus, we consider a function-approximation viewpoint rather than a weight-approximation viewpoint at initialization:

**Definition 3.3.4** (Function Approximation at Initialization)**.** Given a full-rank weight distribution $\mathcal{D}$, we find a low-rank weight matrix of rank $r$ for initialization as follows:

1. Sample $W = [W_1, \ldots, W_k] \sim \mathcal{D}$ with $W_i \in \mathbb{R}^{d_i\times m_i}$.
2. Solve $\hat{U}, \hat{V} = \arg\min_{U,V} \mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left(f_{U,V}(x) - f_W(x)\right)^2\right]$, where $f$ denotes the deep network class we consider, and $\hat{U} = [\hat{U}_1, \ldots, \hat{U}_k]$ and $\hat{V} = [\hat{V}_1, \ldots, \hat{V}_k]$, where $\hat{U}_i \in \mathbb{R}^{d_i\times r}$ and $\hat{V}_i \in \mathbb{R}^{m_i\times r}$.
3. Use $\hat{U}, \hat{V}$ as the initialization for the low-rank weights.

Thus, we attempt to match the function values of the networks, rather than simply the weights, making our approach *non-linearity-aware*. Here we have chosen to measure the similarity of the network outputs over a standard Gaussian input distribution; however, this aspect can easily be modified to be samples over a particular distribution of interest.

Since this approach is essentially function approximation of the initialization network (rather than the trained network, as in other work), we refer to our general approach as *function approximation at initialization*. To implement this approach, one can optimize the low-rank parameters with a gradient-based method with some automatic differentiation framework like Tensorflow [87]. We now proceed to outline a simplification to this general approach which is more tractable and easier to use in practice.

### 3.3.3 Layerwise Function Approximation at Initialization

To make the approximation problem more tractable to solve in practice, we propose a simplification to the general approach: keep the relevance of the non-linearity, but instead approximate each layer separately rather than the entire network. This approach has the benefits of a) being a simpler problem to solve, and b) being embarrassingly parallel to distribute. Thus, we can obtain a significant speedup in the initialization method compared to the full function approximation at initialization approach.

We propose an empirical sample-based stochastic optimization approach for solving the problem using gradient methods:

**Definition 3.3.5** (Layerwise Function Approximation at Initialization (LFAI))**.** Given a deep neural network $f$, define the function corresponding to the $i^{\text{th}}$ layer with weights $W_i$ as $f_{W_i}$. Then, given a full-rank weight distribution $\mathcal{D}$, we find a low-rank weight matrix of rank $r$ for initialization as follows:

1. Sample $W_i \sim \mathcal{D}$;
2. Solve $\hat{U}_i, \hat{V}_i = \arg\min_{U_i, V_i} \mathbb{E}_{x \sim \mathcal{N}(0,I)} \left[ \left( f_{U_i,V_i}(x) - f_{W_i}(x) \right)^2 \right]$ for all layers $i$ in parallel;

3. Use $\hat{U}_i, \hat{V}_i$ as the low-rank initialization for layer $i$.

We can optimize the parameters directly using some gradient-based method over Gaussian[1] samples. In this case, we are essentially throwing out information about $W_i$, since we only access information about $W_i$ via samples which are fed into the gradient-based method. This algorithm is run in parallel across all layers of the network. We present the algorithm at a single layer in Algorithm 2.

**Algorithm 2** LFAI-Gradient

**Require:** $r < d < m$, sample access to full-rank init distribution $\mathcal{D}$ over $\mathbb{R}^{d\times m}$, iterative gradient method $\mathcal{A}$, number of samples $N$

1: Sample $W \sim \mathcal{D}$.
2: Sample $\{x_k\}_{k=1}^N$ i.i.d. from $\mathcal{N}(0, I_{d\times d})$.
3: Run $\mathcal{A}$ using gradient $\nabla_{U,V} \frac{1}{N} \sum_{k=1}^N \left[\left(f_{U,V}(x_k) - f_W(x_k)\right)^2\right]$ until convergence.
4: **Return:** $\hat{U} \in \mathbb{R}^{d\times r}$, $\hat{V} \in \mathbb{R}^{m\times r}$.

To further improve efficiency, we can feed the initialization algorithm some prior information about $W$ by initializing the low-rank weights with spectral initialization (Definition 3.3.3) – we call this step the "spectral warm start," and observe that it helps with learning empirically (see Section 3.4). We can think of this step as reducing the sample complexity required for optimizing only from samples $(x_k, f(x_k))$. The algorithm is presented in Algorithm 3.

**Algorithm 3** LFAI-WS-Gradient

**Require:** $r < d < m$, sample access to full-rank init distribution $\mathcal{D}$ over $\mathbb{R}^{d\times m}$, iterative gradient method $\mathcal{A}$, number of samples $N$

1: Sample $W \sim \mathcal{D}$.
2: Sample $\{x_k\}_{k=1}^N$ i.i.d. from $\mathcal{N}(0, I_{d\times d})$.
3: Compute rank-$r$ SVD $W = U_r \Sigma_r V_r^\top$.
4: Initialize $U_0 := U_r \Sigma_r^{1/2}$; $V_0 := V_r \Sigma_r^{1/2}$.
5: Run $\mathcal{A}$ using gradient $\nabla_{U,V} \frac{1}{N} \sum_{k=1}^N \left[\left(f_{U,V}(x_k) - f_W(x_k)\right)^2\right]$ until convergence.
6: **Return:** $\hat{U} \in \mathbb{R}^{d\times r}$, $\hat{V} \in \mathbb{R}^{m\times r}$.

[1] See Remark 3.4.1 for a discussion of other input distributions.

### 3.3.4 Nonlinear Low-Rank Approximation

In this section, we demonstrate settings in which we can expect the results of using layer-wise function approximation at initialization for the ReLU activation to be significantly different from using spectral initialization, and we also resolve an open theoretical question on the computational tractability of low-rank ReLU approximation. We defer all proofs to the appendix. Throughout this section, we will refer to $\sigma_{\text{ReLU}}$ as $\sigma$ for simplicity. First we instantiate the layer-wise function approximation objective for fully-connected layers:

**Problem 3.3.6** (Nonlinear Low-Rank Approximation (NLRA))**.** Consider the objective

$$\mathcal{R}(Y) := \mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left\|\sigma_{\text{ReLU}}(x^\top Y) - \sigma_{\text{ReLU}}(x^\top W)\right\|_2^2\right] \tag{3.1}$$

where $\sigma_{\text{ReLU}}(x) = \max(0, x)$ is the ReLU activation, $W, Y \in \mathbb{R}^{d\times m}$, and $W$ are fixed ground-truth full-rank weights. Let opt $:= \mathcal{R}(Y^*)$, where $Y^*$ is the argmin over matrices of rank $r$. Our goal is to give a computationally efficient algorithm for outputting a rank $r$ matrix $\hat{Y}$ such that $\mathcal{R}(\hat{Y}) <$ opt$+\epsilon$.

We will derive some structural properties of this objective which will allow us to lower bound the gap between the quality of the spectral solution (linear approximation) and the quality of the (optimal) nonlinear low-rank approximation with respect to this nonlinear error measure.

Using our analysis, we uncover some conditions on the full-rank matrix which yield a more significant gap. In particular, as the rank gets smaller or as the layer width increases, the gap between the initialization methods blows up with dimension (Theorem 3.3.9 and Corollaries B.3.7, B.3.9). It is also the case that full-rank matrices $W \in \mathbb{R}^{d\times m}$ with more approximately orthogonal columns yields a larger gap. As a technical tool, we prove a characterization of the nonlinear function approximation problem (Theorem 3.3.8), which applies specifically to one-hidden-layer ReLU networks (rather than easily invertible activations). We then exploit the properties of this characterization to prove Theorem 3.3.9. We defer all proofs and full theorem statements to Section B.3 of the Appendix. Despite the restriction to ReLU, we believe our results to be characteristic for other activation functions, as we empirically demonstrate for the Swish activation in Section 3.4.

These results only apply to the difference between SVD (the spectral solution) and the optimal nonlinear low-rank approximation. While this result is useful for understanding what properties of the full-rank matrix $W$ govern the extreme cases where the two initialization approaches are very similar or very different, these initialization results do not directly prove anything about downstream generalization error, except in the setting where the true optimal weights are close to the initialization distribution. Nevertheless, we empirically demonstrate the connection between good nonlinear low-rank approximation and improved downstream generalization error for deep low-rank models in Section 3.4.

### 3.3.5 Characterizing **NLRA** for the ReLU Activation

We prove a lower bound on the gap between the output of spectral initialization and our NLRA method for one-hidden-layer ReLU network. To achieve this bound, we further develop theory characterizing the optimal low-rank matrix for Problem 3.3.6 for the ReLU activation. We begin with a useful definition of a function that arises in our analysis. In the proofs of Theorems 3.3.8 and 3.3.9, we re-write the objective in Problem 3.3.6 by recognizing that the following well-known kernel shows up in the objective expression when the non-linearity is given by the ReLU function:

**Definition 3.3.7** (ReLU Kernel: $1^{st}$-Order Arc-Cosine Kernel)**.** The *first-order arc-cosine kernel* [88] is defined by $k(x, y) := \|x\|_2\|y\|_2 \cdot \sqrt{h}(\rho_{xy})$, where $\rho_{xy} := \frac{x^\top y}{\|x\|_2\|y\|_2}$ and $\sqrt{h}(\rho_{xy}) = \big(\sqrt{1-\rho_{xy}^2} + (\pi - \cos^{-1}(\rho_{xy}))\rho_{xy}\big)/\pi$.

Now we present a characterization of the NLRA problem for the ReLU activation. This characterization reveals more structure of the ReLU kernel which allows us to easily lower bound the sub-optimality of standard Frobenius low-rank approximation computed using SVD.

**Theorem 3.3.8** (ReLU SVD)**.** *Consider the goal of finding the optimal rank-r solution to the objective $\mathcal{R}(Y)$ in Problem 3.3.6 with known $W \in \mathbb{R}^{d\times m}$. Then, an equivalent form of the problem*

*is*

$$\max_{\substack{Y\in\mathbb{R}^{d\times m} \text{ is rank } r \\ \|Y_i\|_2=\|W_i\|_2\sqrt{h}(\rho_i) \\ \rho_i=\frac{Y_i^\top W_i}{\|Y_i\|_2\|W_i\|_2}}} \frac{1}{2}\sum_{i=1}^m \|W_i\|_2^2 \cdot h(\rho_i).$$

*where $W = U\Sigma V^\top$ with $U \in \mathbb{R}^{d\times d}, \Sigma \in \mathbb{R}^{d\times d}$ is diagonal, and $V \in \mathbb{R}^{m\times d}$, with $U^\top U = I_{d\times d}, V^\top V = I_{d\times d}$ and $V_i \in \mathbb{R}^d$ is the $i^{th}$ column of $V^\top$, and where $h$ is defined in Definition 3.3.7.*

*Proof.* First, we expand

$$\begin{aligned}
&\mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left\|\sigma(x^\top Y)-\sigma(x^\top W)\right\|_2^2\right] \\
&= \mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left\|\sigma(x^\top Y)\right\|_2^2\right] + \mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left\|\sigma(x^\top W)\right\|_2^2\right] \\
&- 2\mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\langle\sigma(x^\top Y),\sigma(x^\top W)\rangle\right] \\
&= C + \frac{1}{2}\|Y\|_F^2 - 2\mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\langle\sigma(x^\top Y),\sigma(x^\top W)\rangle\right] \\
&= C + \frac{1}{2}\|Y\|_F^2 - \sum_{i=1}^m \|W_i\|_2\|Y_i\|_2\cdot\sqrt{h}\left(\frac{Y_i^\top W_i}{\|W_i\|_2\|Y_i\|_2}\right) \\
&= C - \sum_{i=1}^m\left[\left(\|W_i\|_2\sqrt{h}(\rho_i)\right)\beta_i - \frac{1}{2}\beta_i^2\right]
\end{aligned} \tag{3.2}$$

where $C = \frac{1}{2}\|W\|_F^2$ is a constant independent of choice of $Y$, and letting $\rho_i = \frac{Y_i^\top W_i}{\|Y_i\|_2\|W_i\|_2}$ and $\beta_i = \|Y_i\|_2$, where $Y_i$ and $W_i$ are column $i$ of $Y, W$ respectively. In the above display, we used Lemma B.3.3 and Lemma B.2.3.

Now, we can re-write the minimization problem as a maximization problem:

$$\max_{\rho,\beta:Y\in\mathbb{R}^{d\times m} \text{ is rank } r} \sum_{i=1}^m\left[\left(\|W_i\|_2\sqrt{h}(\rho_i)\right)\beta_i - \frac{1}{2}\beta_i^2\right].$$

Note we can write this as an optimization problem over just $\rho \in \mathbb{R}^m$, since the choice of norm $\beta_i = \|Y_i\|_2$ is independent of the value of $\rho_i$. Since the objective as a function of $\beta_i$ is separable and concave in each term of the sum, we can easily solve the maximization problem by setting the

derivative to 0:

$$\beta_i^* = \|W_i\|_2 \sqrt{h}(\rho_i).$$

Plugging in this optimal choice of $\beta_i$ for any choice of $\rho$ (and remembering that we must correctly re-normalize later), we get the new objective:

$$\max_{\substack{\rho: Y \in \mathbb{R}^{d\times m} \text{ is rank } r \\ \|Y_i\|_2 = \|W_i\|_2 \sqrt{h}(\rho_i)}} \frac{1}{2} \sum_{i=1}^{m} \|W_i\|_2^2 \cdot h(\rho_i). \tag{3.3}$$

□

### 3.3.6 Lower Bounding the Gap: Frobenius Approximation vs. **NLRA**

We now use the developed theory to prove a lower bound on the sub-optimality of using the SVD to optimize the objective of Problem 3.3.6 when given $W$. We defer the proofs to the appendix. Our main characterization theorem is as follows:

**Theorem 3.3.9** (Lower Bound on Suboptimality of SVD for **NLRA**)**.** *Recall the objective $\mathcal{R}(Y)$ from Problem 3.3.6, where we require that $Y \in \mathbb{R}^{d\times m}$ is a rank-$r$ matrix. Let $W = U\Sigma V^\top \in \mathbb{R}^{d\times m}$ be the SVD of $W$. Define $\rho_\sigma^* \in \mathbb{R}^m$ as the correlations $\|\Lambda^* {D^*}^\top \hat{W}_i\|_2$ for column $i$ of $W$, where $\Lambda^* \in \mathbb{R}^{r\times r}$ and $D^* \in \mathbb{R}^{d\times r}$ are as in Lemma B.3.5 and $\hat{W}_i = W_i/\|W_i\|_2$. As shorthand, denote $Y(\rho)$ as the associated low-rank matrix for correlation vector $\rho \in \mathbb{R}^m$ (computed as described in Theorem 3.3.8). Denote $\rho_{\text{SVD}}^*$ to be the optimal correlations in the case where we pick $\Lambda^*$ to correspond to the top $r$ singular values, and $D^* = U$ as in SVD. Then, we have the following lower bound for the suboptimality of the SVD solution $Y_{\text{SVD}}$:*

$$\left(\mathcal{R}(Y_{\text{SVD}})\right) - \mathcal{R}(Y(\rho_\sigma^*))\right) \geq \frac{1}{2}\|w \odot \sqrt{h(\rho_{\text{SVD}}^*)} - \rho_{\text{SVD}}^*\|_2^2 \tag{3.4}$$

*where $h$ is defined in Definition B.1, where $w = \left[\|W_1\|_1, \quad \ldots, \|W_m\|_2\right]$ is a vector of column norms of $W$, and where $\odot$ is the element-wise product.*

Corollary 3.3.10 provides some intuition for Theorem 3.3.9:

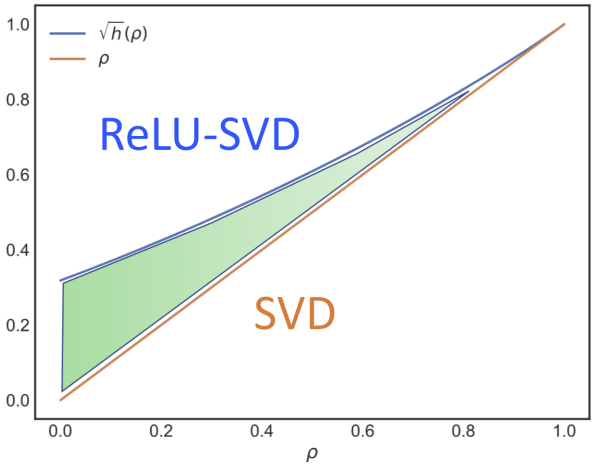
**Corollary 3.3.10** (Relationship between SVD and ReLU SVD)**.** *Suppose* $W = U\Sigma V^\top \in \mathbb{R}^{d\times m}$. *Consider the solution for rank-r ReLU SVD (given by* $Y^*$*) as described in Theorem 3.3.8. If* $h(\rho)$ *is replaced with* $\rho^2$*, and we always choose* $\Lambda^*$ *to correspond to the top r singular values of* $\Sigma$ *and* $D^*$ *to correspond to U, then* $Y^*$ *is the standard SVD solution.*

The intuition for this theorem's proof is given in Figure 3.1.

1. SVD maximizes **squared correlations $\rho^2$**.

2. ReLU-SVD maximizes squared **transformed** correlations $\sqrt{h(\rho)}^2$.

3. The gap generates the lower bound.




Correlation between Columns of Low-Rank Approximation and Full-Rank Matrices

Figure 3.1: Intuition for the proof of Theorem 3.3.9.

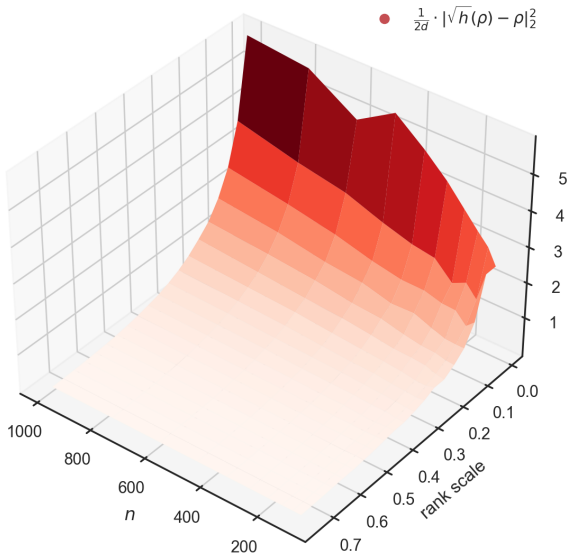



Figure 3.2: We plot the gap growth $\frac{1}{2d}\|\sqrt{h}(\rho) - \rho\|_2^2$ (see Theorem 3.3.9) for $W \in \mathbb{R}^{d \times m}$ with $m = n^{1.5}$ and $d = 0.2n$ with respect to the ReLU non-linearity. Note here the width is super-linear in dimension: $m = \Omega(n^{1+\epsilon})$ for all $\epsilon > 0$. The entries of ground truth matrix $W$ are drawn from $\mathcal{N}(0, 1)$, and then we normalize $\|W_i\|_2 = 1$, $\forall i \in [m]$. We observe that the gap increases with increasing dimension and decreasing rank scale. Note that the behavior demonstrated matches the theoretical predictions: the gap increases as $\Theta((d^{1/2} \cdot (1 - \sqrt{r/d})^2)$ as per Corollary 3.3.11.

Using the lower bound in Theorem 3.3.9, we can now characterize the conditions on the full-rank matrix $W \in \mathbb{R}^{d\times m}$, where the solution to Problem 3.3.6 and the SVD solution are significantly different. First, smaller correlations $\rho^*_{\text{SVD}}$ result in better solutions – this case roughly corresponds to the columns of $W$ being approximately orthogonal (Corollary B.3.7, Remark B.3.8). When $\max_i \rho^*_{\text{SVD}}(i) < 1$, we can prove that the sub-optimality gap grows as the width $m$ increases, and furthermore the sub-optimality gap is monotone non-decreasing as $r$ decreases (Corollary B.3.9). Finally, we prove a stronger version of Corollary B.3.9 under the assumption of uniform spherically distributed columns of $W \in \mathbb{R}^{d\times m}$, and recover the actual dependence on the rank scale $r/d$:

**Corollary 3.3.11** (Spherical Weights)**.** *Suppose the columns of $W \in \mathbb{R}^{d\times m}$ are drawn from the uniform distribution over the surface of the unit sphere*[2]*. Consider target rank $r \leq d$ and assume $m$ grows super-linearly in $d$, e.g. $m > \Omega(d)$. Now define*

$$L(x) := \left(\sqrt{1-x^2} - \arccos(x)\cdot x\right)^2.$$

*Let $c, C > 0$ be universal constants. Then, as $m \to \infty$, with probability at least $1-2\exp(-c\cdot C^2\cdot d)$, we have*

$$\frac{1}{m}\mathbb{E}_W\left[\mathcal{R}(Y_{\text{SVD}})) - \mathcal{R}(Y(\rho^*_\sigma))\right] \geq \frac{1}{2m}\mathbb{E}_W\left[\left\|\sqrt{h}(\rho^*_{\text{SVD}}) - \rho^*_{\text{SVD}}\right\|_2^2\right] \geq \frac{1}{2\pi^2}\cdot L\left(\sqrt{\frac{r}{d}}\right). \tag{3.5}$$

*We also have the following high probability bound as $m \to \infty$. Let $0 < t < 1 - \sqrt{r/d}$. Then*

$$\mathbb{P}_W\left(\frac{1}{m}\left(\mathcal{R}(Y_{\text{SVD}})) - \mathcal{R}(Y(\rho^*_\sigma))\right) \geq \frac{1}{2\pi^2}\cdot L\left(\sqrt{r/d}+t\right)\right) \geq 1 - 2\exp\left(-\Theta\left(m\cdot t\right)\right) - 2\exp(-c\cdot C^2\cdot d), \tag{3.6}$$

*which limits $\to 1-2\exp(-c\cdot C^2\cdot d)$ as $m\to\infty$ for any fixed $t$.*

[2]This assumption is reasonable, given the many similar initialization distributions used in practice (e.g., the He and Glorot initialization distributions [89, 90]). The same result holds for an appropriately scaled Gaussian vector initialization as well.

*Proof.* First,

$$\mathbb{E}_W\left[\mathcal{R}(Y_{\text{SVD}})) - \mathcal{R}(Y(\rho^*_\sigma))\right] \geq \frac{1}{2}\mathbb{E}_W\left[\left\|\sqrt{h}(\rho^*_{\text{SVD}}) - \rho^*_{\text{SVD}}\right\|_2^2\right] = \frac{1}{2}\mathbb{E}_W\left[\sum_{i=1}^{m}\left(\sqrt{h}(\rho^*_{\text{SVD}}(i)) - \rho^*_{\text{SVD}}(i)\right)^2\right],$$

since the inequality holds for every choice of $W$ inside the expectation, as we proved in Theorem 3.3.9. Then we can also write

$$\mathbb{E}_W\left[\mathcal{R}(Y_{\text{SVD}})) - \mathcal{R}(Y(\rho^*_\sigma))\right] \geq \frac{m}{2}\cdot\mathbb{E}_{W,i\sim\text{Unif}([m])}\left[\left(\sqrt{h}(\rho^*_{\text{SVD}}(i)) - \rho^*_{\text{SVD}}(i)\right)^2\right].$$

We will bound the latter term. Consider the gap function $f(\rho) = \left(\sqrt{h(\rho)} - \rho\right)^2$. This function is convex (Lemma B.3.10), so by Jensen's inequality we have that $\mathbb{E}_\rho\left[f(\rho)\right] \geq f(\mathbb{E}_\rho\left[\rho\right])$. Then, since $f$ is monotone non-increasing as a function of $\rho$ (Lemma B.3.10), we have that if we prove a bound $\mathbb{E}_\rho\left[\rho\right] \leq B_{\text{upper}}$, then

$$\mathbb{E}_\rho\left[f(\rho)\right] \geq f(\mathbb{E}_\rho\left[\rho\right]) \geq f(B_{\text{upper}}).$$

Thus, our next task is to upper bound $\mathbb{E}_{W,i\sim\text{Unif}([m])}\left[\rho^*_{\text{SVD}}(i)\right]$. From Theorem 3.3.9, we have

$$\rho^*_{\text{SVD}}(i) = \|\Sigma E_r V_i\|_2 / \|W_i\|_2$$

where $W = U\Sigma V^\top$ with $V_i \in R^d$ being a column of $V^\top$ and $E_r = \text{ diag }(\Lambda^*)$, where $\Lambda^* \in \{0,1\}^d$ is a binary vector with $r$ non-zero entries selecting the top $r$ singular vectors of $W$. Note $\rho^*_{\text{SVD}}(i) \leq 1$ since $\|W_i\|_2 = \|\Sigma V_i\|_2$. Thus, we have

$$\frac{\|\Sigma E_r V_i\|_2}{\|\Sigma V_i\|_2} = \frac{\|E_r \Sigma V_i\|_2}{\|\Sigma V_i\|_2} = \left\|E_r \frac{\Sigma V_i}{\|\Sigma V_i\|_2}\right\|_2$$

which holds since $\Sigma$ is diagonal.

Now, treating $W$ and its SVD as random variables, note that $W = U\Sigma V^\top$, whose columns are

uniformly distributed on the surface of the sphere, can be written as a matrix of $m$ i.i.d. Gaussian vectors drawn from $\mathcal{N}(0, I_{d\times d})$ with normalized columns, and that $U \in \mathbb{R}^{d\times d}$ is an orthogonal matrix. By the fact that the Gaussian is rotationally symmetric, multiplying by $U^\top$ does not change the distribution. Thus we find that $\Sigma V^\top \in R^{d\times m}$ is also a matrix of $m$ independent Gaussians distributed according to $\mathcal{N}(0, I_{d\times d})$.

Now note that $E_r$ is a random variable with dependencies on $\Sigma V_i$, since $\rho^*_{\text{SVD}}(i)$ is defined with the choice of the top $r$ singular vectors depending on the realization $W$.

Thus, we write

$$\mathbb{E}_{W,i\sim\text{Unif}([m])}\left[\rho^*_{\text{SVD}}(i)\right] = \mathbb{E}_{W,i\sim\text{Unif}([m])}\left[\left\|E_r\frac{\Sigma V_i}{\|\Sigma V_i\|_2}\right\|_2\right].$$

Since $W \in \mathbb{R}^{d\times m}$ is a matrix with sub-gaussian columns (see Definition 5.22 and Example 5.25 in [91]), the minimum and maximum singular vectors of $W$ satisfy the following upper and lower bounds with probability at least $1 - 2\exp\left(-c\cdot\epsilon^2\right)$, as per Theorem 5.39 in [91], where $C, c > 0$ are universal constants corresponding to the sub-gaussian norm of the (i.i.d.) columns of $W$ and $\epsilon > 0$:

$$\sqrt{m} - C\sqrt{d} - \epsilon \le \sigma_{\min}(W) \le \sigma_{\max}(W) \le \sqrt{m} + C\sqrt{d} + \epsilon.$$

Therefore, setting $\epsilon = C\sqrt{d}$, with probability at least $1 - 2\exp\left(-c\cdot C^2\cdot d\right)$, we have that

$$\frac{\sigma_{\max}(W)}{\sigma_{\min}(W)} \le \frac{\sqrt{m} + 2C\sqrt{d}}{\sqrt{m} - 2C\sqrt{d}}.$$

Since $m \to \infty$ and $m$ is super-linear in $d$, the ratio converges to 1, and the spectrum is uniform across all values. In this limiting case, $E_r$ can be chosen to be any selection of $r$ of the $d$ dimensions – thus, we can fix $E_r$ to have the first $r$ coordinates set to 1 and the rest to 0. Then, we can write

$$\mathbb{E}_{W,i\sim\text{Unif}([m])}\left[\left\|E_r\frac{\Sigma V_i}{\|\Sigma V_i\|_2}\right\|_2\right] \le \mathbb{E}_{i\sim\text{Unif}([m])}\left[\mathbb{E}_{u_i\sim\text{Unif}(\mathbb{S}^{d-1})}\left[\sqrt{u_i^\top E_r u_i}\right]\right],$$

where we used the fact that a normalized Gaussian random variable is also a random variable that is uniformly distributed over the surface of the sphere. Then, we have

$$\mathbb{E}_{i\sim\text{Unif}([m])}\left[\mathbb{E}_{u_i\sim\text{Unif}(\mathbb{S}^{d-1})}\left[\sqrt{u_i^\top E_r u_i}\right]\right] \leq \mathbb{E}_{i\sim\text{Unif}([m])}\left[\mathbb{E}_{u_i\sim\text{Unif}(\mathbb{S}^{d-1})}\left[\sqrt{u_i^\top E_r u_i}\right]\right],$$

where we used the monotonicity of the square root function. Since the square root is a concave function, we can apply Jensen's inequality again to get:

$$\mathbb{E}_{i\sim\text{Unif}([m])}\left[\mathbb{E}_{u_i\sim\text{Unif}(\mathbb{S}^{d-1})}\left[\sqrt{u_i^\top E_r u_i}\right]\right] \leq \sqrt{\mathbb{E}_{i\sim\text{Unif}([m])}\left[\mathbb{E}_{u_i\sim\text{Unif}(\mathbb{S}^{d-1})}\left[u_i^\top E_r u_i\right]\right]} = \sqrt{\frac{r}{d}},$$

where we apply Lemma B.3.11 and the fact that the distributions are identical for all $i \in [m]$ in the last step. Thus $B_{\text{upper}} = \sqrt{\frac{r}{d}}$, and the result directly follows by plugging this value in to our lower bound and expanding the definition of $\left(\sqrt{h}(\rho) - \rho\right)^2$.

Now we prove that our lower bound concentrates – that is, with high probability over the choice of $W$ with uniform spherical columns, our lower bound holds. We still have the lower bound

$$\mathcal{R}(Y_{\text{SVD}})) - \mathcal{R}(Y(\rho^*_\sigma)) \geq \frac{1}{2}\left\|\sqrt{h}(\rho^*_{\text{SVD}}) - \rho^*_{\text{SVD}}\right\|_2^2 = \frac{m}{2}\cdot\frac{1}{m}\sum_{i=1}^{m}\left(\sqrt{h}(\rho^*_{\text{SVD}}(i)) - \rho^*_{\text{SVD}}(i)\right)^2.$$

We can then apply the same Jensen and monotonicity arguments as above to bring the uniform expectation over $i \in [m]$ inside, so that we have a lower bound in terms of the *the average* $\rho^*_{\text{SVD}}(i)$ *over* $i \in [m]$:

$$\mathcal{R}(Y_{\text{SVD}})) - \mathcal{R}(Y(\rho^*_\sigma)) \geq \frac{m}{2}\cdot\left(\sqrt{h}\left(\mathbb{E}_{i\sim\text{Unif}([m])}\left[\rho^*_{\text{SVD}}(i)\right]\right) - \mathbb{E}_{i\sim\text{Unif}([m])}\left[\rho^*_{\text{SVD}}(i)\right]\right)^2.$$

Now, we give a concentration bound over the distribution of $W$ that limits the probability that

$\mathbb{E}_{i\sim\text{Unif}([m])}\left[\rho^*_{\text{SVD}}(i)\right]$ is much larger than its average over the distribution of $W$. We upper bound

$$\mathbb{P}_{u_i\sim\text{Unif}\left(\mathbb{S}^{d-1}\right)\forall i\in[m]}\left(\left|\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{r}\left(u_i^\top e_j\right)^2-\frac{r}{d}\right|>t\right)<2\exp\left(-m\cdot\frac{t^2/2}{1+t/3}\right).$$

This bound holds since $\sum_{j=1}^{r}\left(u_i^\top e_j\right)^2<1$ and is identically distributed for all $i\in[m]$ and all $j\in[d]$, and we can apply Bernstein's inequality (see Theorem 2.8.4 in [92]). Therefore, the desired upper bound holds with high probability.

□

Therefore, we see that in the case of uniform spherical columns for $W$, the sub-optimality gap increases as either the width increases or as the rank scale decreases. We also note that as the width increases, our lower bound more tightly concentrates.

*Remark* 3.3.12 (Non-Limiting Case). In the case where we do not take the limit as $m\to\infty$ and want to understand what happens for small $m$, we can apply the following approach to upper bound $\rho^*_{\text{SVD}}$: First,

$$\mathbb{E}_{W,i\sim\text{Unif}([m])}\left[\rho^*_{\text{SVD}}(i)\right]\leq\mathbb{E}_{W,i\sim\text{Unif}([m])}\left[\max_{\hat{E}_r}\left\|\hat{E}_r\frac{\Sigma V_i}{\|\Sigma V_i\|_2}\right\|_2\right].$$

The above inequality holds since the value induced by the selection of the top $r$ singular values given $W_i$ is less than or equal to the maximizing choice of $r$-sparse diagonal matrix $\hat{E}_r$. Conveniently, this removes $E_r$ as a random variable and we no longer have to worry about dependencies, since in the case of small $m$, $E_r$ and the uniform spherical random variable have dependencies. We also note, however, that this bound is not tight since the selection of the top $r$ singular values is not necessarily going to be the worst case every time – this approach just makes the object to bound easier to work with. We proceed similarly to Corollary 3.3.11 to see that we must upper bound

$$\sqrt{\mathbb{E}_{i\sim\text{Unif}([m])}\left[\mathbb{E}_{u_i\sim\text{Unif}(\mathbb{S}^{d-1})}\left[\max_{\hat{E}_r}u_i^\top\hat{E}_r u_i\right]\right]}.$$

We then can use the standard fact that $u_i^\top \hat{E}_r u_i$ is marginally distributed as $\text{Beta}(r/2, d/2)$, and apply the log moment generating function approach (see [93], Section 5.1 for an introduction) to upper bound this random variable over the set of possible $\hat{E}_r$ (e.g., the set of $r$-sparse $d$-dimensional binary vectors, of which the logarithm of the size is at most $\Theta(r \log(d/r))$). Note that this approach is approximately tight for independent random variables, and so is not necessarily tight for our situation where the set of $u_i^\top \hat{E}_r u_i$ are not independent. In particular we have that

$$\sqrt{\mathbb{E}_{i \sim \text{Unif}([m])} \left[ \mathbb{E}_{u_i \sim \text{Unif}(\mathbb{S}^{d-1})} \left[ \max_{\hat{E}_r} u_i^\top \hat{E}_r u_i \right] \right]} \leq \sqrt{\inf_{\lambda > 0} \frac{r \log(d/r) + \log\left({}_1F_1(r/2; d/2; \lambda)\right)}{\lambda}},$$

where ${}_1F_1(a; b; \lambda)$ is the confluent hypergeometric function of the first kind (or Kummer's function) [94]. After plugging in $a = r/2, b = d/2$, we get the moment generating function for $\text{Beta}(r/2, d/2)$. The optimum can be numerically solved via binary search in one-dimension since the gradient is initially negative and is also monotone non-decreasing, and there is a unique minimizer. This bound is always non-trivial since taking $\lambda \to \infty$ can be fairly easily seen to limit to 1 from below. We visualize how the upper bound on $\mathbb{E}[\rho_{\text{SVD}}]$ changes as a function of $m$ in Figure 3.3.

### 3.4 Experiments and Discussion

We present some empirical results for our low-rank initialization scheme for training EfficientNets [5] on the ImageNet dataset [6] with stochastic gradient descent and momentum, with tuned parameters and learning rate schedule. We study low-rank variants of these networks for various choices of *rank scale* – for parameter matrix $W \in \mathbb{R}^{d \times m}$, the rank scale is the fraction of $\min(d, m)$ that we require for our low-rank factorization $UV^\top$. We view the choice of rank scale as a trade-off parameter between computation and accuracy since in our settings, lower rank uniformly means worse generalization (though this may be false for other tasks and datasets). We compare the following initialization methods: 1) Baseline Low-Rank – apply the full-rank init distribution for $W$ to

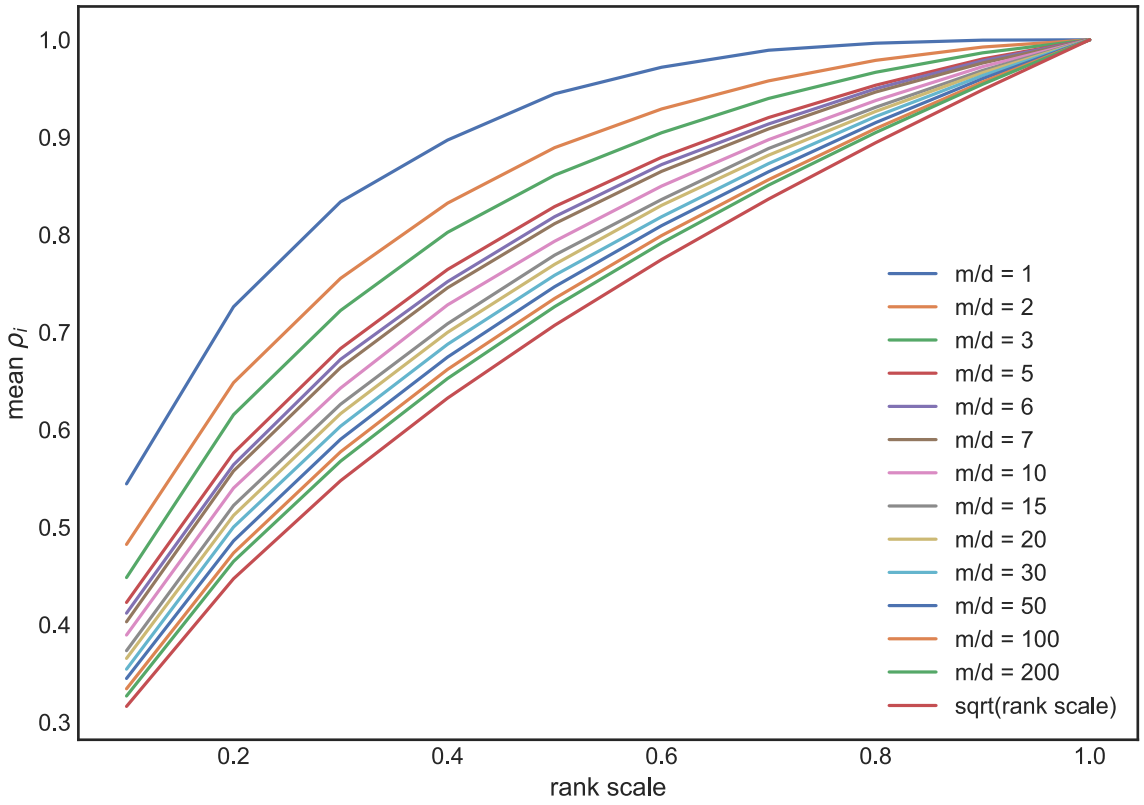



Figure 3.3: We visualize the effect of the choice of width, $m$, on the upper bound on $\mathbb{E}[\rho]$. For small $m$, the upper bound is close to the worst case where we upper bound with the maximum choice of $E_r$ as described in Remark 3.3.12, and for larger $m$, we see that the upper bound quickly approaches the asymptotic case corresponding to the upper bound of $\sqrt{r/d}$.

low-rank factors $U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{d \times r}$; 2) Spectral (Definition 3.3.3); 3) LFAI-Adam (Algorithm 2), implemented with Adam [95]; 4) LFAI-WS-Adam (Algorithm 3).

### 3.4.1 Details on Main Experimental Setup

We train ResNet50 [4] and EfficientNet [5] models with stochastic gradient descent and momentum on the 2012 ImageNet dataset [6], with tuned parameters and learning rate schedule. The low-rank version of this model simply replaces the convolution layers with low-rank convolutions, as described in [58]. The full-rank weight initialization is a truncated normal distribution $\mathcal{N}(0, 1/d)$, where $d$ is the number of input units for the layer, and where "truncation" refers to discarding and re-sampling any samples which are more than two standard deviations from the mean.

We also consider two kinds of regularization on the objective:

1. Weight decay: This method is the standard Frobenius norm regularization on the weights of the layers. In the low-rank setting, instead of penalizing $\|W\|_F^2$, we penalize $\|U\|_F^2 + \|V\|_F^2$.
2. Frobenius decay: This regularization approach is demonstrated by [58] to outperform weight decay in several settings they consider. Instead of separately regularizing the low rank factors, this approach penalizes $\|UV^T\|_F^2$.

We tune the regularization strength separately for both approaches and report the performance of the best regularization strength. Tuning the Frobenius decay regularization strength did not succeed for the EfficientNet models due to divergence during training. Thus we only report the weight decay results for the EfficientNet models. For ResNet-50, for Frobenius decay, we use a regularization strength of 0.3. For ResNet-50 weight decay, we use a regularization strength of $10^{-4}$. For the EfficientNet models with weight decay, we use a regularization strength of $10^{-5}$. We compare across multiple choices of rank scale $\frac{r}{d}$: $0.05, 0.1, 0.15, 0.2$.

We run each experiment three times for three different random seeds and average the results and include the standard error up to one standard deviation on the mean performance. The differences

across each experiment instance are due to the changes in random seeds used in both sampling at initialization and for sampling batches during optimization. We trained the models on TPU hardware.

For **NLRA**-based approaches, we choose the standard Gaussian as the input distribution. However, other choices are also feasible:

*Remark* 3.4.1 (On the Choice of Input Distribution). We minimize the function approximation loss over the Gaussian distribution. In practice, we may view this choice as a hyper-parameter to tune. For instance, another reasonable (but computationally expensive) approach is to minimize the function approximation loss over the real data distribution (after being transformed by the previous input layers). In our experiments, we did not notice much difference by switching to this initialization distribution, but it is plausible that in other problem settings other input distributions might perform better.

*Model Architecture*

The EfficientNet architecture for b0, b3, and b7 is standard and available in libraries such as Tensorflow [87]. For EfficientNet-b9, we define the width scale parameter to be 3.0, and the depth-scale parameter to be 3.2. For both architectures, we used the Swish non-linearity [96]. The Swish non-linearity can be expressed as $swish(x) = x \cdot \frac{1}{1+\exp(-x)}$, and is a smooth approximation to the ReLU nonlinearity that allows for small negative activation outputs for small magnitude inputs. For ResNet-50, we also simply use the standard available architecture in Tensorflow.

*Hyperparameters for Layerwise Initialization*

In our experiments, we implement both LFAI-Adam and LFAI-WS-Adam using the Adam optimizer [95]. Since our networks consist of low-rank convolutional layers (see [58] for guidance on how to efficiently define a low-rank convolutional layer), sample 1000 Gaussian vectors in the shape $(64, 64, \text{num. input filters})$ – we arbitrarily choose $64 \times 64$ patches since the specific dimension of the "image" does not matter too much, and we want to avoid making the dimension

of our training problem too large. The number of input filters is determined by the layer and the architecture – for the first layer, there are 3 input filters.

For Adam, we set the learning rate to $5 \times 10^{-3}$, the batch size to 512, and the number of steps per epoch to 128 after hyper-parameter searching over these parameters to determine the quickest convergence rate. With these parameters, Adam converges to the optimum for all low-rank layerwise optimization problems we tried within 6 epochs.

We reported results both for initializing Adam with spectral initialization (LFAI-WS-Adam) and with baseline initialization (LFAI-Adam) – almost universally LFAI-WS-Adam is better.

*Hyperparameters for Post-Initialization Optimization*

For ResNet-50 and each EfficientNet model, we trained the network for 62000 epochs using Stochastic Gradient Descent (SGD) with Momentum. We set the batch size to 4096. We set the momentum parameter to 0.9 and the learning rate schedule to follow a linear warm-up schedule for 1560 steps (choose values for the learning rate from 0 to an initial rate of 1.6), followed by a cosine curve with an initial rate of 1.6, decaying over the remaining 60440 steps (see the CosineDecay learning rate schedule in Tensorflow). We did not modify this training scheme across our experiments for simplicity (our goal was to compare the performance of different init schemes rather than to attain the optimal performance), explaining why our full-rank EfficientNet numbers do not match the numbers in the original EfficientNet paper [5].

### 3.4.2 Results

It turns out that either LFAI-WS-Adam or LFAI-Adam typically outperform the other initialization schemes across multiple rank scales and architecture widths. On average across all experiments (including on ResNet [4], our method gains on the order of 0.3% in accuracy, though the gain is larger for smaller rank scales and larger width networks. These results are also significant – we demonstrate a clear separation of our approach compared to others in 1-standard deviation confidence intervals, and our method tends to have lower variance in performance. Furthermore,

Table 3.1: The reported metrics are average top-1 accuracy as a percent on EfficientNet models of increasing width and depth, trained with Weight Decay regularization. We report standard error over three samples up to one standard deviation. The EfficientNet model variants we consider have width and depth scale parameters set to be (1, 1); (1.2, 1.4); (2.0, 3.1); (3.0, 3.2), corresponding to the b0, b3, b7, and b9 variants respectively. We see the empirical effect described in Section 3.3.4 – our methods (LFAI-Adam and LFAI-WS-Adam) have more significant improvements for smaller rank scales and larger width models.

| Rank Scale | Method | EfficientNet-b9 | EfficientNet-b7 | EfficientNet-b3 | EfficientNet-b0 |
|---|---|---|---|---|---|
| 0.05 | Baseline | 66.36 ± 0.23 | 58.61 ± 0.45 | 37.34 ± 0.29 | 31.22 ± 0.62 |
| | Spectral | 66.64 ± 0.23 | 58.84 ± 0.38 | **38.3 ± 0.09** | 30.96 ± 0.32 |
| | **LFAI-Adam** | 65.84 ± 0.28 | 59.00 ± 0.18 | **38.22 ± 0.28** | **32.39 ± 0.26** |
| | **LFAI-WS-Adam** | **66.90 ± 0.11** | **59.63 ± 0.37** | 38.04 ± 0.03 | 30.8 ± 0.19 |
| 0.10 | Baseline | 71.67 ± 0.19 | 68.89 ± 0.07 | **57.06 ± 0.24** | **48.24 ± 0.13** |
| | Spectral | 72.07 ± 0.15 | 68.83 ± 0.16 | 56.66 ± 0.35 | 47.79 ± 0.17 |
| | LFAI-Adam | 71.02 ± 0.32 | 68.52 ± 0.04 | **56.96 ± 0.19** | 47.42 ± 0.30 |
| | **LFAI-WS-Adam** | **72.28 ± 0.04** | **69.09 ± 0.16** | 56.76 ± 0.11 | 47.84 ± 0.24 |
| 0.15 | Baseline | 73.20 ± 0.05 | 71.67 ± 0.19 | 62.56 ± 0.07 | 54.99 ± 0.05 |
| | Spectral | **73.55 ± 0.15** | 71.55 ± 0.20 | 63.02 ± 0.07 | 54.97 ± 0.10 |
| | LFAI-Adam | 72.70 ± 0.18 | 71.25 ± 0.12 | 62.86 ± 0.12 | 54.85 ± 0.09 |
| | **LFAI-WS-Adam** | 73.31 ± 0.04 | **71.74 ± 0.02** | **63.55 ± 0.16** | **55.19 ± 0.10** |
| 0.20 | Baseline | 73.84 ± 0.04 | 72.16 ± 0.27 | 66.05 ± 0.23 | **59.78 ± 0.17** |
| | Spectral | **74.14 ± 0.19** | 72.45 ± 0.17 | 65.96 ± 0.29 | 59.42 ± 0.32 |
| | LFAI-Adam | 73.91 ± 0.09 | 72.32 ± 0.18 | 65.93 ± 0.06 | 59.20 ± 0.11 |
| | **LFAI-WS-Adam** | **74.01 ± 0.11** | **72.72 ± 0.03** | **66.32 ± 0.02** | 59.38 ± 0.18 |
| 1.0 | Full-Rank | 79.62 ± 0.05 | 78.70 ± 0.02 | 76.63 ± 0.19 | 74.50 ± 0.13 |

we also find empirical evidence of our theoretical claims in Section 3.3.4, despite the fact that we use the Swish activation while our theory was ReLU-based: we see the trend that as the rank-scale decreases, or as the widths of the networks increase (higher number EfficientNets correspond to larger widths), we see improvement in the top-1 accuracy gain, see Table 3.1.

While Frobenius Decay outperforms Weight Decay for smaller rank scales, the advantage erodes for larger rank scales. This effect was not observed by [58], possibly because we study larger models and datasets. We note that our method improves over other initialization methods regardless of the initialization scheme.

We also note that in our EfficientNet experiments (Table 3.1), the effect of width does not appear to be as strong as the effect of the rank scale. On the other hand, the effect of the smaller

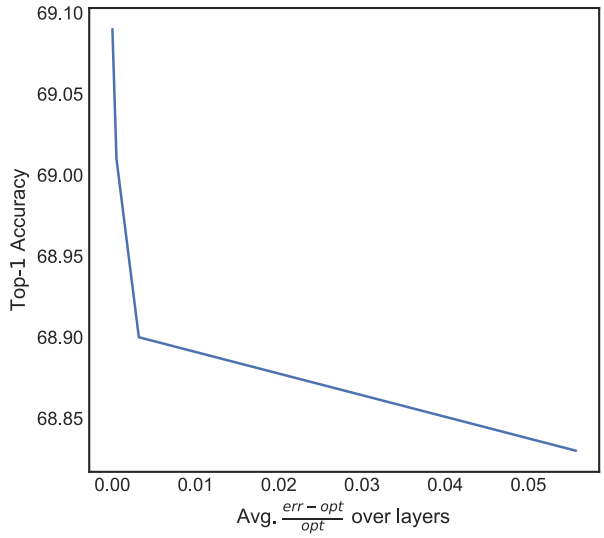



Figure 3.4: We vary the number of optimization steps used to implement WS-NLRA and plot the average normalized error across layers against the average downstream validation top-1 accuracy for training an EfficientNet-b7 model on ImageNet. Decreasing the function approximation error at init is beneficial for top-1 accuracy.

Table 3.2: Comparison of LFAI-WS-Adam Across Rank Scales for EfficientNet-b9. We show by how many percent points the best performing method, LFAI-WS-Adam, beats the other methods, for the Weight Decay regularization.

| Rank Scale | Top-1 Accuracy for LFAI-WS-Adam |
|---|---|
| 0.05 | 66.90 ± 0.11 (best by 0.36) |
| 0.1 | 72.28 ± 0.04 (best by 0.21) |
| 0.15 | 73.31 ± 0.04 (sub-optimal) |
| 0.2 | 74.01 ± 0.11 (overlaps w/best) |
| Full-Rank | 79.62 ± 0.05 |

rank scale is quite apparent in our experiments.

It would be interesting to develop a more thorough characterization of the impact of the ratios of the inputs and outputs of layers in a deep architecture on the effectiveness of the LFAI-Adam framework – it seems plausible that we would have to take into account some notion of average ratio between input and output sizes, and possibly the structure of the architecture itself as well.

In Table 3.6, we present our results for the ResNet-50 architecture [4]. In Tables 3.2, 3.3, 3.4, 3.5, 3.7, and 3.8, we summarize the performance of our best initialization method compared to the others.

Table 3.3: Comparison of LFAI-WS-Adam Across Rank Scales for EfficientNet-b7. We show by how many percent points the best performing method, LFAI-WS-Adam, beats the other methods, for the Weight Decay regularization.

| Rank Scale | Top-1 Accuracy for LFAI-WS-Adam |
|---|---|
| 0.05 | 59.63 ± 0.37 (best by 0.63) |
| 0.1 | 69.09 ± 0.16 (best by 0.20) |
| 0.15 | 71.74 ± 0.02 (best by 0.07) |
| 0.2 | 72.72 ± 0.03 (best by 0.27) |
| Full-Rank | 78.70 ± 0.02 |

Table 3.4: Comparison of LFAI-Adam Across Rank Scales for EfficientNet-b3. We show by how many percent points the best performing method, LFAI-Adam, beats the other methods, for the Weight Decay regularization.

| Rank Scale | Top-1 Accuracy for LFAI-Adam |
|---|---|
| 0.05 | 38.22 ± 0.28 (overlaps w/best) |
| 0.1 | 56.96 ± 0.19 (overlaps w/best) |
| 0.15 | 63.55 ± 0.16 (best by 0.53) |
| 0.2 | 66.32 ± 0.02 (best by 0.27) |
| Full-Rank | 76.63 ± 0.19 |

Table 3.5: Comparison of LFAI-Adam and LFAI-WS-Adam Across Rank Scales for EfficientNet-b0. We show by how many percent points LFAI-WS-Adam compares against the other methods, for the Weight Decay regularization.

| Rank Scale | Top-1 Accuracy for LFAI-Adam or LFAI-WS-Adam |
|---|---|
| 0.05 | 32.39 ± 0.26 (best by 1.17) |
| 0.1 | 47.84 ± 0.24 (suboptimal) |
| 0.15 | 55.19 ± 0.10 (best by 0.20) |
| 0.2 | 59.38 ± 0.18 (suboptimal) |
| Full-Rank | 74.50 ± 0.13 |

Table 3.6: Average Top-1 accuracy on ResNet-50 with different regularization methods. Note that our method tends to outperform the baselines regardless of whether Weight Decay or Frobenius Decay is used. For each rank scale, we display in bold all methods whose confidence intervals overlap.

| Rank Scale | Method | Weight Decay | Frobenius Decay |
|---|---|---|---|
| 0.05 | Baseline | 56.92 ± 0.14 | 57.73 ± 0.23 |
| | Spectral | 56.28 ± 0.39 | 57.42 ± 0.38 |
| | LFAI-Adam | 56.55 ± 0.04 | 57.61 ± 0.15 |
| | **LFAI-WS-Adam** | **57.37 ± 0.08** | **58.22 ± 0.31** |
| 0.10 | Baseline | 64.09 ± 0.12 | 65.61 ± 0.09 |
| | Spectral | 64.06 ± 0.19 | 65.69 ± 0.12 |
| | LFAI-Adam | 64.02 ± 0.17 | 65.55 ± 0.05 |
| | **LFAI-WS-Adam** | **64.74 ± 0.04** | **65.93 ± 0.05** |
| 0.15 | Baseline | 67.58 ± 0.15 | **68.15 ± 0.20** |
| | Spectral | 66.95 ± 0.16 | **68.21 ± 0.08** |
| | LFAI-Adam | 67.24 ± 0.26 | 67.13 ± 0.51 |
| | **LFAI-WS-Adam** | **67.74 ± 0.03** | **67.83 ± 0.42** |
| 0.20 | **Baseline** | **69.04 ± 0.33** | 69.22 ± 0.05 |
| | Spectral | 68.59 ± 0.19 | 69.04 ± 0.14 |
| | LFAI-Adam | 68.93 ± 0.20 | 68.87 ± 0.10 |
| | **LFAI-WS-Adam** | **69.37 ± 0.07** | **69.47 ± 0.09** |
| 1.0 | Full-Rank | 78.1 | 78.1 |

### 3.4.3 Downstream Generalization Error

While existing literature has studied the effects of various initializations on optimization, no results as far as we are aware connect the choice of initialization to downstream generalization performance, though this effect has been demonstrated empirically.

Thus, we empirically justify the benefits of function approximation at initialization for low-rank networks. We perform the following experiment: We fix rank scale 0.10, EfficientNet-b7 and initialization algorithm LFAI-WS-Adam, and investigate the effect of layerwise function approximation at initialization on downstream generalization. We implement layerwise function approximation with Adam, as mentioned above. We train the network on ImageNet training data starting from the initialization produced after $0 - 6$ epochs of running Adam starting from the spectral initialization, and measure the top-1 validation accuracy. For each number of epochs of training,

Table 3.7: Comparison of LFAI-WS-Adam Across Rank Scales for ResNet-50. We show by how many percent points the best performing method, LFAI-WS-Adam, beats the other methods, for the Weight Decay regularization.

| Rank Scale | Top-1 Accuracy for LFAI-WS-Adam |
|---|---|
| 0.05 | 57.37 ± 0.08 (best by 0.45) |
| 0.1 | 64.74 ± 0.04 (best by 0.65) |
| 0.15 | 67.74 ± 0.03 (best by 0.16) |
| 0.2 | 69.37 ± 0.07 (best by 0.33) |
| 0.25 | 70.37 ± 0.06 (best by 0.25) |
| Full-Rank | 78.1 |

Table 3.8: Comparison of LFAI-WS-Adam Across Rank Scales for ResNet-50. We show by how many percent points the best performing method, LFAI-WS-Adam, beats the other methods, for the Frobenius Decay regularization.

| Rank Scale | Top-1 Accuracy for LFAI-WS-Adam |
|---|---|
| 0.05 | 58.22 ± 0.31 (best by 0.49) |
| 0.1 | 65.93 ± 0.05 (best by 0.24) |
| 0.15 | 67.83 ± 0.42 (sub-optimal) |
| 0.2 | 69.47 ± 0.09 (best by 0.25) |
| 0.25 | 70.16 ± 0.09 (best by 0.19) |
| Full-Rank | 78.1 |

we run the experiment 3 times and take the average.

Then, on the $x$-axis, for each number of initialization training epochs, we plot the average multiplicative error *across* the 107 layers of EfficientNet-b7, paired with the corresponding downstream generalization error. In particular, if the optimum value of our objective (defined in Problem 3.3.6) at a given layer (at init) is opt, and we have reached $(1 + \epsilon_{\text{layer } i}) \cdot \text{opt}$, then we record the average $\epsilon$ across all layers for that number of epochs, and plot alongside it the corresponding average over downstream top-1 validation accuracy. We find that after 4 epochs, training converges and the optimization procedure has reached the optimum.

We plot our results in Figure 3.4, and observe that as LFAI-WS-Adam optimizes, accuracy improves, demonstrating that optimal function approximation at initialization improves downstream generalization.

# Chapter 4: Stronger Lower Bounds for Streaming SVM

In this chapter, we consider resource limitations from another perspective: in modern machine learning, large data streams are prevalent in practice (see [1]). For example, in the ads/recommender system setting, very large amounts of data arrive in a streaming fashion (as a concrete example: consider the Twitter Firehose [97]). It is not clear that the data arrive in an i.i.d. fashion, and it is also not the case that we can necessarily store all of the data without some cost. Thus, we are motivated to consider the streaming setting for machine learning: we want to know how much data we must store after one pass of the data stream in order to optimize a model. This work is currently in progress with my advisors Daniel Hsu and Alexandr Andoni.

## 4.1 Problem Statement

Concretely, we will consider the $\ell_2$ regularized SVM objective [98], since it is a common and simple objective function often applied in the practice of training linear models. Since it is simple to optimize, it is often a good baseline to use before resorting to fancier, more compute-heavy, methods.

The SVM objective is an interesting case study also because it is both strongly convex (Definition 1.3.10) and non-smooth (Definition 1.3.8). Existing works in the streaming literature (for instance, regarding the streaming optimization of generalized linear models) deal with smooth objectives [99]. As other prevalent methods (like ReLU deep networks) are also non-smooth (though non-convex), the SVM is also a good starting point for understanding the challenges arising in non-smooth optimization in the streaming setting.

**Definition 4.1.1** (Primal Soft-Margin SVM Objective)**.** For $N$ labelled data points $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$ with $\|x_i\|_2 \leq 1$ and $w \in \mathbb{R}^d$ the unknown model parameters, the *primal soft-margin SVM*

*objective* is defined as:

$$F_\lambda(w) = \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{N}\sum_{i=1}^{N}[1 - y_i(w^\top x_i)]_+$$

for regularization strength $\lambda > 0$ where $[\cdot]_+$ denotes $\max(0, \cdot)$. We also refer to $\lambda$ as the strong convexity parameter, since it determines the strong convexity constant for $F_\lambda$ (see Definition 1.3.10).

We consider the primal soft-margin SVM objective in the streaming setting. The goal is to minimize $F_\lambda(w)$ over $w \in \mathbb{R}^d$. The Streaming SVM problem is as follows:

**Definition 4.1.2** ($\epsilon$-Optimal One-Pass Streaming SVM Problem)**.** We receive data points $(x_i, y_i) \in \mathbb{R}^d \times \{\pm 1\}$ with $\|x_i\|_2 \leq 1$ from a stream of length $N$ in some pre-fixed but arbitrary order. We are also given a regularization parameter $\lambda \in \mathbb{R}^+$ which ensures the objective is strongly convex. At the end of the stream, we must output a weight vector $\hat{w} \in \mathbb{R}^d$ such that it is an $\epsilon$-approximate solution, i.e.

$$F_\lambda(\hat{w}) - F_\lambda(w) \leq \epsilon,$$

with probability at least $3/4$ over any randomness used to fix the stream order and any randomness used by the streaming algorithm.

Note that we consider the one-pass streaming setting. Our key interest will be in the space complexity of an algorithm that solves this problem. Particularly, we ask: *Does any one-pass (randomized) streaming algorithm that solves the* 1*-dimensional* $\epsilon$*-optimal streaming SVM require* $\Omega(1/\sqrt{\epsilon})$ *memory?*

## 4.2 Related Work

The most significant prior work that we build on is [100], which considers the same problem and achieves some preliminary results. In dimensions 1 and 2, [100] provide algorithms that outperform the naive method of using resevoir sampling to subsample the stream for $1/\lambda\epsilon$ points, and then running stochastic gradient descent over these points [101]. The algorithm proposed for the 1-dimensional problem takes space $O\left(\frac{1}{\sqrt{\epsilon}}\sqrt{\log(1/\epsilon)}\right)$. The algorithm proposed for the 2-

dimensional problem takes space $\tilde{O}(\epsilon^{-4/5})$, where the tilde hides logarithmic factors of $1/\epsilon$. [100] also provides a space complexity lower bound for one pass streaming of $\Omega(1/\sqrt{\epsilon})$ for dimensions 3 and higher; however, this lower bound requires $\lambda = \Theta(\sqrt{\epsilon})$ and does not apply to the case where $\lambda = \Theta(1)$. Thus far, there is no space lower bound for the 1-dimensional problem, and for dimension 2, the best known space lower bound is $\Omega(\epsilon^{-1/4})$ (and this lower bound requires $\lambda = \Theta(\epsilon)$).

When the data is separable with margin $\gamma$, it is possible to obtain space dependencies which depend on $\gamma$. Consider the following algorithm: Subsample $1/\epsilon\gamma^2$ elements from the stream using resevoir sampling, and implement the leave-one-out online-to-batch conversion using the Perceptron algorithm [102] on the subsampled stream. This algorithm has a space upper bound of $\Theta(d/\epsilon\gamma^2)$. If $\gamma$ grows large enough as a function of $1/\epsilon$ (say for instance $\gamma = 1/\sqrt{\epsilon}$), it is possible to attain space complexity sub-linear in $1/\epsilon$. Dimension reduction approaches in this setting can further remove the dependence on $d$ via random projections; however, this method introduces poly-logarithmic dependencies on $N$ (see e.g. [103], which attains a $\Theta(\log^2(N)/\epsilon \cdot \gamma^4)$ memory bound).

### 4.3 Main Result

We prove a nearly tight space lower bound for one-pass streaming SVM in 1 dimension, resolving the open problem. This lower bound also clearly applies in 2+ dimensions, and improves the 2-dimensional lower bound as well. Furthermore, our lower bound holds for $\lambda = \Theta(1)$ – previously, such lower bounds were only known for $d = \Theta\left(\log(1/\epsilon)\right)$. Our approach relies on the general reduction to communication lower bounds outlines in Section 1.3.5. In our case, we will consider the standard Augmented Index communication problem (see Section 1.3.5 for a definition), and come up with a novel reduction in 1 dimension. The theorem is as follows:

**Theorem 4.3.1** (Lower Bound: 1-Dim Streaming SVM)**.** *Consider any* 1*-dimensional* $\epsilon$*-optimal One-Pass Streaming SVM problem instance satisfying* $\epsilon < 2\lambda$ *and* $\lambda < \frac{3}{2} - \sqrt{2}$*, where* $\lambda$ *is the strong convexity parameter defined in Definition 4.1.2. Then, any successful (randomized) streaming algorithm for the Streaming SVM problem (Definition 4.1.2) requires memory at least* $\Omega\left(1/\sqrt{\epsilon}\right)$.

## 4.4 Proofs

### 4.4.1 Additional Notation and Terminology

**Definition 4.4.1** (Hinge Loss)**.** We define the *hinge loss* $\ell^{(z)}_{\text{hinge}} : \mathbb{R}^d \to \mathbb{R}$ for point $z \in \mathbb{R}^d$ to be $\ell^{(z)}_{\text{hinge}}(w) := \left[1 - z^\top w\right]_+$ where $[\cdot]_+ = \max(0, \cdot)$.

**Definition 4.4.2** (Hinge Points in 1-Dimension)**.** We will refer to the points of non-smoothness of the 1-dimensional SVM objective $F_\lambda(w)$ as *hinge points* – these are the points which have a sub-differential set of size $> 1$. Considering the terms in the objective defined by the data points (e.g. $f_i(w) = [1 - wz_i]_+$), the hinge points will correspond to the points $w = 1/z_i$.

### 4.4.2 Main Technique: Reduction from Augmented Index

Given an instance of Augmented Index (see Definition 1.3.14), we need to construct a 1-Dimensional Streaming SVM problem instance – this step consists of constructing a dataset $\mathbf{D}$ and a parameter $\lambda \in \mathbb{R}^+$ (Definition 4.1.2). However, there are several restrictions on how we may construct the dataset. In particular, we must construct the dataset with Alice and Bob and the restrictions implicit in the information they have available – that is, Alice must encode the information from her bit string into one part of the dataset, and Bob must encode his index into the other part of the dataset, only using information from the first $i - 1$ bits of the bitstring. This setup obeys the reduction from communication problems outlined in Section 1.3.5. Then we need to show that knowledge of an $\epsilon$-optimal point for the Streaming SVM problem instance solves the Augmented Index instance. First we give a useful definition:

**Definition 4.4.3** (Defining the $s_{\text{Bob}}$ Function)**.** Consider a set $X = [X_1, \ldots, X_m] \subset R$ of size $|X| = m$. Then define

$$s_{\text{Bob}}(X) := -\sum_{k=1}^{m} \frac{1}{X_k}.$$

If $X$ is chosen as a function of some integer index $i > 0$, then we adopt the shorthand

$$s_{\text{Bob}}(i) := s_{\text{Bob}}(X(i)).$$

We term this function $s_{\text{Bob}}$ to correspond to the phrase "Bob's slope", which will be relevant in the forthcoming reduction.

We now construct a Streaming SVM problem instance from an Augmented Index instance, which consists of defining a dataset $D$ and regularization parameter (strong convexity parameter) $\lambda \in \mathbb{R}^+$:

**Construction 4.4.4** (Streaming SVM Dataset $D$ from Augmented Index Instance)**.** Let $n > 0$ be a positive integer. We begin with an Augmented Index instance parameterized by $(\mathbf{a}, i) \in \{0,1\}^n \times [n]$. In this instance, Alice has $n$ bits $\mathbf{a} := \{a_j\}_{j=1}^n$, and Bob has an index $i \in [n]$. Let $c_{\text{Alice}} \in (0,1)$ – we will specify its value shortly. We now define a Streaming SVM instance over $N = \left\lfloor n/c_{\text{Alice}} \right\rfloor$ data points given the Augmented Index data, and we specify a choice of $\lambda$ as well as $c_{\text{Alice}}$ below. See Figures 4.1 and 4.2 for a visualization. We define the complete Streaming SVM problem instance $(\lambda, \mathbf{D})$:

1. First, select any real number $\lambda \in (0, \frac{3}{2} - \sqrt{2})$.
2. Select $c_{\text{Alice}} = \frac{1}{2} - \lambda - \sqrt{2\lambda}$.
3. Define an ordered list of points $A = \left[1, \ldots, 1 + \sqrt{\frac{1}{2\lambda}}\right] \subset \mathbb{R}^+$ to be $n$ evenly spaced points. Here, we use $A[j]$ to denote the $j^{th}$ element in this list.
4. Each point $A[j] \in A$ is associated with a hinge loss term $f_j(w) := \left[1 - (w/A[j])\right]_+$. Note $f_j(w) = 0$ for $w \geq A[j]$.
5. Define

$$K(i) := -\lambda N \cdot A[n-i+1] + \frac{1}{2A[n-i+1]}.$$

We define an ordered list of $N - n$ real-valued points $B \subset \mathbb{R}$ which collectively are a function

of $i$: We require that $B$ satisfies the relation

$$s_{\text{Bob}}(B(i)) = \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} + K(i).$$

In words, Bob will add several hinge losses so that the total slope due to Bob ensures that the optimum is at index $A[n-i+1]$ if $a_i = 1$, and far enough away otherwise (see Lemma 4.4.7). We also require that for all $k \in [N-n]$ and for all $i \in [n]$, $B_k(i)$ satisfies the condition that either $B_k(i) < 0$ or $B_k(i) \geq A[n]$.

6. We now define the dataset $\mathbf{D} := \{z_i\}_{i=1}^{N}$ where $z_i \in \mathbb{R}$ using $(\mathbf{a}, i)$: For each $j \in [n]$, if $\mathbf{a}[j] = 1$, include $1/A[n-j+1]$ in $\mathbf{D}$. Otherwise, include a copy of 0. For the remaining data points, we use index $i$ and add the set $\{1/B_k(i)\}_{k=1}^{N-n}$ to $\mathbf{D}$. Each of Bob's data points is associated with a hinge loss term $g_k(w,i) := \left[1 - (w/B_k(i))\right]_+$ where $g_k(w,i) = 0$ for $\frac{w}{B_k(i)} \geq 1$. Our constraints on $B_k(i)$ ensure that $g_k(w,i) = 1 - (w/B_k(i))$ for all $w \in (0, A[n]]$.

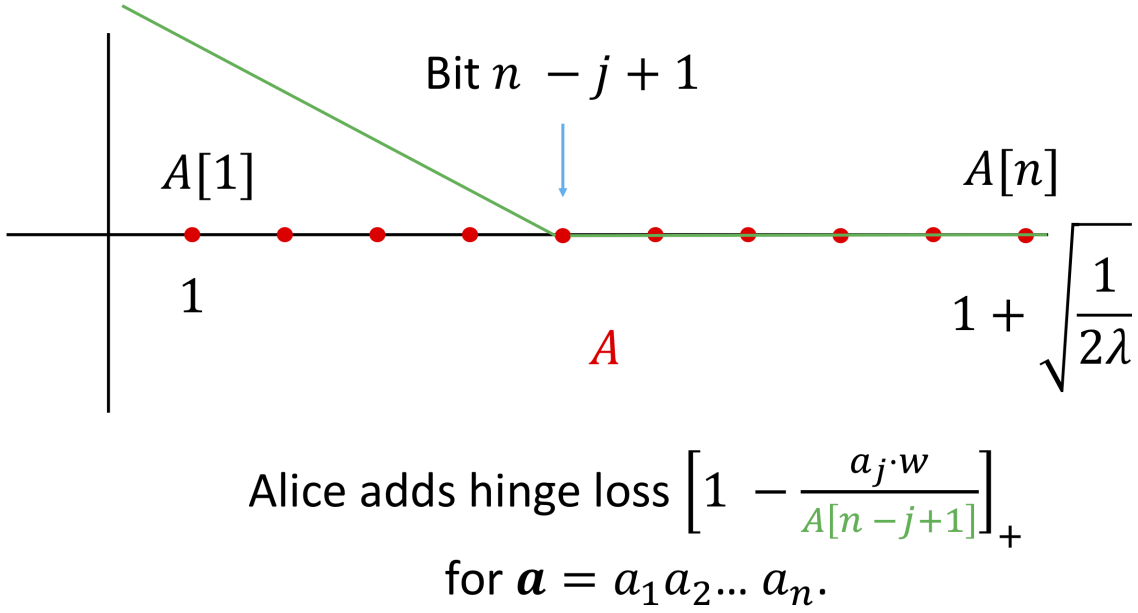
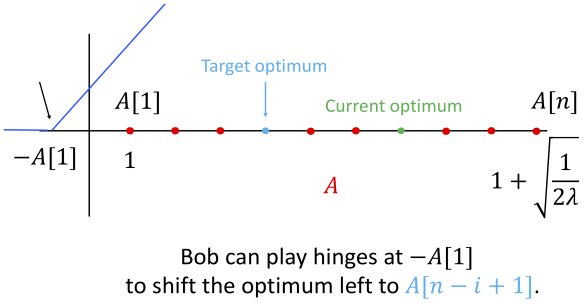
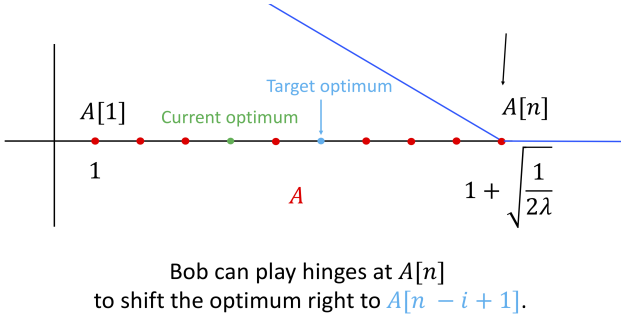



Figure 4.1: Alice's partial construction of dataset **D**.




(a) Bob shifts the optimum to the left.




(b) Bob shifts the optimum to the right.

Figure 4.2: Bob's partial construction of dataset **D**.

Now we present the lower bound proof. First recall the statement of the theorem:

**Theorem 4.3.1** (Lower Bound: 1-Dim Streaming SVM)**.** *Consider any* 1*-dimensional* $\epsilon$*-optimal One-Pass Streaming SVM problem instance satisfying* $\epsilon < 2\lambda$ *and* $\lambda < \frac{3}{2} - \sqrt{2}$*, where* $\lambda$ *is the strong convexity parameter defined in Definition 4.1.2. Then, any successful (randomized) streaming algorithm for the Streaming SVM problem (Definition 4.1.2) requires memory at least* $\Omega\left(1/\sqrt{\epsilon}\right)$.

We introduce and prove several lemmas before we give the full proof:

1. In Lemma 4.4.5, we establish conditions so that the optimum of the dataset corresponding to any Augmented Index instance (as specified in Construction 4.4.4) is located to the right of $A[1] > 0$.
2. In Lemmas 4.4.7 and 4.4.8, we establish that the optima for the constructed Streaming SVM instance corresponding to cases $a_i = 0$ or 1 in the corresponding Augmented Index instances are sufficiently separated so that a $\Theta(\epsilon)$-optimal algorithm for the Streaming SVM problem (Definition 4.1.2) can distinguish between the two cases.
3. Finally, in Lemma 4.4.9, we assert that the instance constructed in Construction 4.4.4 given an Augmented Index instance is feasible given the constraints implied by the knowledge that Alice and Bob have individually.

We now proceed to state and prove these pieces of the proof formally.

**Lemma 4.4.5** (Bounds on the Location of the Minimizer)**.** *Recall the constructed Streaming SVM instance from Construction 4.4.4. If* $s_{\text{Bob}}(i)$ *satisfies*

$$s_{\text{Bob}}(i) \leq \sum_{j=1}^{n} \left( \frac{a_j}{A[n-j+1]} \right) - \lambda \cdot N \cdot A[1],$$

*then* $w^* \geq A[1]$.

*Proof.* First, let us write down the SVM function given the construction in Construction 4.4.4: We

have

$$F_\lambda(w) = \frac{\lambda}{2}w^2 + \frac{1}{N}\sum_{j=1}^{n}\left[1 - a_j \cdot \frac{w}{A[n-j+1]}\right]_+ + \frac{1}{N}\sum_{k=1}^{N-n}\left[1 - \frac{w}{B_k(i)}\right]_+. \tag{4.1}$$

Now note that for any $\max_{k\in[N-n]:B_k(i)<0} B_k(i) \le w \le 0$, the sub-gradient (slope) of $F_\lambda(w)$ is unique and satisfies

$$\begin{aligned}
\frac{d}{dw}F_\lambda(w) &= \lambda w - \frac{1}{N}\sum_{j=1}^{n}\frac{a_j}{A[n-j+1]} - \frac{1}{N}\sum_{k=1}^{N-n}\frac{1}{B_k(i)} \\
&\le -\frac{1}{N}\sum_{j=1}^{n}\frac{a_j}{A[n-j+1]} + \frac{1}{N}s_{\text{Bob}}(i) \\
&\le -\frac{1}{N}\sum_{j=1}^{n}\frac{a_j}{A[n-j+1]} + \frac{1}{N}\sum_{j=1}^{n}\left(\frac{a_j}{A[n-j+1]} - \frac{\lambda}{c_{\text{Alice}}}A[1]\right) \\
&\le -\lambda \cdot A[1] \\
&< 0
\end{aligned} \tag{4.2}$$

applying the condition in the lemma statement. Now also note that $\frac{d}{dw}F_\lambda(w) < 0$ also applies for $w < \max_{k\in[N-n]:B_k(i)<0} B_k(i)$, since the value of the slope can only be smaller for any such $w$ (from positive slope to zero slope). Thus we can also rule out $w^* < \max_{k\in[N-n]:B_k(i)<0} B_k(i)$ as well. Therefore, we must have $w^* > 0$ since $F_\lambda(w)$ is strongly convex.

Next, we will show that $w^* \ge A[1]$ under our condition. Suppose for sake of contradiction that $w^* \in (0, A[1])$ while

$$s_{\text{Bob}}(i) \le \sum_{j=1}^{n}\left(\frac{a_j}{A[n-j+1]} - \frac{\lambda}{c_{\text{Alice}}}A[1]\right).$$

Then, $F_\lambda(w)$ is differentiable in this region since in this region, $\left[1 - a_j \cdot \frac{w}{A[n-j+1]}\right]_+ = 1 - a_j \cdot \frac{w}{A[n-j+1]}$ (as all $A[n-j+1] \ge A[1]$) and $\left[1 - \frac{w}{B_k(i)}\right]_+ = 1 - \frac{w}{B_k(i)}$, which holds when $B_k(i) \le -A[1] = 1$ and also when $B_k(i) = A[n]$ if $w \le A[n]$. Thus, under our assumption that $w^* \in (0, A[1])$, the optimality conditions of differentiable convex functions imply that the gradient of $F_\lambda(w)$ is zero at some $w^*$ in $(0, A[1])$. Since $F_\lambda(w)$ is also strongly convex, this point is unique.

Thus, the optimality conditions for a smooth convex function directly imply

$$\lambda w^* + \frac{1}{N}\left(s_{\text{Bob}}(i) - \sum_{j=1}^{n} \frac{a_j}{A[n-j+1]}\right) = 0, \tag{4.3}$$

and solving for $w^*$ yields

$$w^* = \frac{1}{\lambda \cdot N}\left(\sum_{j=1}^{n} \frac{a_j}{A[n-j+1]} - s_{\text{Bob}}(i)\right). \tag{4.4}$$

Recalling the hypothesis for the sake of contradiction, we must have $w^* < A[1]$. We thus have

$$\sum_{j=1}^{n}\left(\frac{a_j}{A[n-j+1]} - \frac{\lambda}{c_{\text{Alice}}}A[1]\right) < s_{\text{Bob}}(i),$$

which contradicts our assumption. Therefore, the condition

$$s_{\text{Bob}}(i) \leq \sum_{j=1}^{n}\left(\frac{a_j}{A[n-j+1]} - \frac{\lambda}{c_{\text{Alice}}}A[1]\right)$$

ensures that $w^* \notin (0, A[1])$. Since $w^* > 0$ and $w^* \notin (0, A[1])$, we have therefore established that $w^* \geq A[1]$. By the strong convexity of $F_\lambda(w)$, $w^*$ is at a finite point $\geq A[1]$ and is unique. □

*Remark* 4.4.6 (Edge Case for Applying Lemma 4.4.5). We note that given the values for $B$ and thus $s_{\text{Bob}}(i)$ from Construction 4.4.4, the restriction in the statement of Lemma 4.4.5 is satisfied in all cases except when Alice's bit string **a** is the string of all 0 bits. In this case, as we will see in Lemma 4.4.7, the optimum value can be slightly smaller than $A[1]$.

**Lemma 4.4.7** (Testing the Minimizer)**.** *Recall the Streaming SVM instance constructed in Construction 4.4.4. Let $w^*_{a_i}$ refer to the optimal point of the function depending on the value of $a_i$. If $a_i = 1$,*

$$w^*_1 = A[n-i+1].$$

*Otherwise, if $a_i = 0$, either*

$$w_0^* = A[n-i+1] - \frac{1}{2\lambda N \cdot A[n-i+1]}.$$

*or*

$$w_0^* \leq A[n-i].$$

*Proof.* Let $F_\lambda^{a_i}$ refer to the function as it depends on the value of $a_i = 0$ or 1. First we consider the case $a_i = 1$. We have $w_1^* = A[n-i+1]$ if and only if we have $0 \in \partial F_\lambda^1(A[n-i+1])$, by the strong convexity of $F_\lambda^1$. Since $F_\lambda^1$ is piece-wise convex quadratic and the hinge points are the points of non-smoothness, we will prove that Bob's dataset $B$ (chosen in Construction 4.4.4) induces a value of $s_{\text{Bob}}(i)$ such that for $w \in (A[n-i], A[n-i+1])$, $\frac{dF_\lambda^1(w)}{dw} < 0$ and for $w \in (A[n-i+1], A[n-i+2])$, $\frac{dF_\lambda^1(w)}{dw} > 0$. These conditions ensure by strong convexity that $0 \in \partial F_\lambda^1(A[n-i+1])$, and thus that $w_1^* = A[n-i+1]$. To begin, recall the value of $s_{\text{Bob}}(i)$ induced by the selection of Bob's dataset $B$ from Construction 4.4.4:

$$s_{\text{Bob}}(i) = \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} + K(i) = \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} - \lambda N \cdot A[n-i+1] + \frac{1}{2A[n-i+1]}.$$

Next, we calculate the most negative and most positive sub-gradients at $A[n-i+1]$ using the sub-gradient of $F_\lambda^1(w)$ along the intervals $w \in (A[n-i], A[n-i+1])$ and for $w \in (A[n-i+1], A[n-i+2])$ respectively, and demonstrate that they satisfy

$$\min_{g \in \partial F_\lambda(A[n-i+1])} g = \lambda A[n-i+1] + \frac{1}{N}\left(s_{\text{Bob}}(i) - \sum_{j=1}^{i} \frac{a_j}{A[n-j+1]}\right) < 0$$

and

$$\max_{g \in \partial F_\lambda(A[n-i+1])} g = \lambda A[n-i+1] + \frac{1}{N}\left(s_{\text{Bob}}(i) - \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]}\right) > 0.$$

In particular, these constraints hold (and prove that $w_1^* = A[n-i+1]$ since $0 \in \partial F_\lambda^1(A[n-i+1])$)

since after simplifying with respect to $s_{\text{Bob}}(i)$, we see that they are equivalent to the constraint

$$\sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} - \lambda N \cdot A[n-i+1] < s_{\text{Bob}}(i) < \sum_{j=1}^{i} \frac{a_j}{A[n-j+1]} - \lambda N \cdot A[n-i+1].$$

which clearly holds for the induced value of $s_{\text{Bob}}(i)$ when $a_i = 1$.

Now we handle the case $a_i = 0$: How much does the optimum of $F_\lambda^0$ differ from the optimum of $F_\lambda^0$? The sub-gradient set of $F_\lambda^0$ at $A[n-i+1]$ satisfies

$$\begin{aligned}
\min_{g \in \partial F_\lambda^0(A[n-i+1])} g &= \lambda A[n-i+1] + \frac{1}{N}\left(\sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} + K(i) - 0 - \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]}\right) \\
&= \lambda A[n-i+1] + \frac{1}{N}\left(\frac{1}{2A[n-i+1]} - \lambda N A[n-i+1]\right) \\
&= \frac{1}{2N \cdot A[n-i+1]} \\
&> 0
\end{aligned} \tag{4.5}$$

and thus the objective is differentiable and sub-optimal at $A[n-i+1]$. Instead, since the slope at $A[n-i+1]$ is positive, we must have $w^* < A[n-i+1]$. Now we have to identify the value of the optimum – how far away does it shift? There are two cases: $w^* \leq A[n-i]$, or $w^* \in (A[n-i], A[n-i+1])$. First we consider the second case. We have to set the value of the slope to 0 and solve, since there are no hinge points in $(A[n-i], A[n-i+1])$ by definition (and thus any optimizer in this regime has a sub-gradient set of cardinality 1 and $F_\lambda^0$ is differentiable everywhere in the interval). In particular:

$$\frac{dF_\lambda^0}{dw}(w^*) = \lambda w^* + \frac{1}{N}\left(\sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} + K(i) - \sum_{j=1}^{i} \frac{a_j}{A[n-j+1]}\right) = 0.$$

Using $a_i = 0$, we get

$$\lambda w^* + \frac{1}{N}\left(\sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} - \lambda N A[n-i+1] + \frac{1}{2A[n-i+1]} - \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]}\right) \tag{4.6}$$
$$= 0,$$

or simplifying,

$$\lambda w^* + \frac{1}{N}\left(-\lambda N A[n-i+1] + \frac{1}{2A[n-i+1]}\right) = 0,$$

and thus

$$w^* = A[n-i+1] - \frac{1}{2\lambda N A[n-i+1]}.$$

Then we need to check our assumption that $w^* \in (A[n-i], A[n-i+1])$. We require

$$A[n-i+1] - \frac{1}{2\lambda N A[n-i+1]} \geq A[n-i]$$

for this solution to not contradict the assumptions we used to derive it. If this holds, we are done. Otherwise, the first case must hold, and $w^* \leq A[n-i]$. □

Now we prove a lemma asserting that given an $\epsilon/16$-optimal solution $\hat{w}$ to the streaming SVM objective derived from Construction 4.4.4, $\hat{w}$ can distinguish between the cases $a_i = 0, 1$ in the Augmented Index instance.

**Lemma 4.4.8** (Sufficient Sub-optimality)**.** *Let $w_1^*$ and $w_0^*$ be the minimizers of the SVM objective defined by Construction 4.4.4 with strong convexity parameter $\lambda < 3/2 - \sqrt{2}$ and size $N = |\boldsymbol{D}| = 1/2\sqrt{\epsilon}$ in the two cases where $a_i = 0$ or 1. Let*

$$\Delta := \left|w_1^* - w_0^*\right|.$$

*Then, given an $\epsilon/16$-sub-optimal solution $\hat{w}$ to the streaming SVM problem, we have that*

$$\left|\hat{w} - w^*\right| < \frac{1}{2} \cdot \Delta$$

*Proof.* Recall that $A[n-i+1] \leq 1 + \sqrt{\frac{1}{2\lambda}}$. From Lemma 4.4.7, there are two cases. We have that one of the following two statements holds:

1. $w_1^* - w_0^* \geq \frac{1}{2\lambda N \cdot A[n]}$, since $A[n] \geq A[j]$ for all $j \in [n]$.
2. $w_1^* - w_0^* \geq A[n-i+1] - A[n-i] = \frac{1}{n\sqrt{2\lambda}}$, from the definition of $A$ in Construction 4.4.4.

In the first case, since $A[n] = 1 + \sqrt{\frac{1}{2\lambda}}$, we have

$$\begin{aligned} w_1^* - w_0^* &\geq \frac{1}{2\lambda N \cdot \left(1 + \sqrt{\frac{1}{2\lambda}}\right)} \\ &= \frac{2\sqrt{\epsilon}}{2\lambda \cdot \left(1 + \sqrt{\frac{1}{2\lambda}}\right)} \\ &= \frac{\sqrt{\epsilon}}{\lambda + \sqrt{\frac{\lambda}{2}}}. \end{aligned} \tag{4.7}$$

The same lower bound holds in the second case: we have

$$\begin{aligned} w_1^* - w_0^* &\geq \frac{1}{c_{\text{Alice}} \cdot N \cdot \sqrt{2\lambda}} \\ &> \frac{2\sqrt{\epsilon}}{\sqrt{\lambda}} \\ &> \frac{\sqrt{\epsilon}}{\lambda + \sqrt{\frac{\lambda}{2}}}. \end{aligned} \tag{4.8}$$

since $c_{\text{Alice}} < 1/2$ and $2\sqrt{\lambda} > \lambda + \sqrt{\lambda/2}$ as $\lambda < 1$. Therefore, to be able to distinguish between the cases $a_i = 0, 1$ with probability $> 3/4$, we must have that after conditioning on the success event

of an $\epsilon/16$-optimal one pass Streaming SVM algorithm, its output $\hat{w}$ satisfies

$$|w^*_{0 \text{ or } 1} - \hat{w}| < \frac{1}{2} \cdot \frac{\sqrt{\epsilon}}{\lambda + \sqrt{\frac{\lambda}{2}}}.$$

We demonstrate this is true. By $\lambda$-strong convexity of both $F^0_\lambda$ and $F^1_\lambda$, we have that for an $\epsilon/16$-optimal streaming algorithm

$$\begin{aligned} \frac{\lambda}{2} \cdot \left((w^*_{0 \text{ or } 1}) - \hat{w}\right)^2 &\leq \left|F^{0 \text{ or } 1}_\lambda(w^*_{0 \text{ or } 1}) - F^{0 \text{ or } 1}_\lambda(\hat{w})\right| \leq \frac{\epsilon}{16} \\ \left|(w^*_{0 \text{ or } 1}) - \hat{w}\right| &\leq \sqrt{\frac{\epsilon}{8\lambda}} = \frac{1}{2} \cdot \sqrt{\frac{\epsilon}{2\lambda}} \\ &< \frac{1}{2} \cdot \frac{\sqrt{\epsilon}}{\lambda + \sqrt{\frac{\lambda}{2}}} \end{aligned} \tag{4.9}$$

which holds when $\lambda$ satisfies

$$\sqrt{2\lambda} > \lambda + \sqrt{\frac{\lambda}{2}}$$

which holds when $\lambda < 1/2$, which is true since we assume $\lambda < \frac{3}{2} - \sqrt{2} < 1/2$. □

**Lemma 4.4.9** (Feasibility of Choosing $s_{\text{Bob}}(i)$)**.** *Fix the list of $N - n$ real-valued points $B$ and strong convexity parameter $\lambda < 3/2 - \sqrt{2}$ from Construction 4.4.4. If $N > \frac{1}{2\lambda + \sqrt{8\lambda}}$, it is possible for Bob to choose a list $B$ that satisfies the required relation from Construction 4.4.4 given knowledge of only $a_1, \ldots, a_{i-1}$ and index $i \in [n]$, namely that*

$$s_{\text{Bob}}(B(i)) = \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} - \lambda N \cdot A[n-i+1] + \frac{1}{2A[n-i+1]},$$

*Proof.* Bob must construct a list of $N - n$ real-valued points $B$ so that they satisfy

$$s_{\text{Bob}}(B(i)) = \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} - \lambda N \cdot A[n-i+1] + \frac{1}{2A[n-i+1]},$$

The right-hand side of the above expression only depends on the first $i-1$ terms of $\mathbf{a}$ and $i$ from the

Augmented Index instance, which is exactly the information that Bob receives. Now it remains to show that Bob can achieve any magnitude of $s_{\text{Bob}}(i)$ that might be necessary, given varying values of $\mathbf{a}$ and $i$. First we observe upper and lower bounds on

$$s_{\text{Bob}}(i) = M_i + \frac{1}{2A[n-i+1]},$$

where

$$M_i = \sum_{j=1}^{i-1} \frac{a_j}{A[n-j+1]} - \lambda \cdot N \cdot A[n-i+1].$$

We can upper bound $s_{\text{Bob}}(i)$ by taking $a_j = 1$ for all $j \in [n]$ and upper bounding $-\lambda \cdot N \cdot A[n-i+1] + \frac{1}{2A[n-i+1]} \leq -\lambda \cdot N \cdot A[1] + \frac{1}{2}$. We can lower bound $s_{\text{Bob}}(i)$ by taking $a_j = 0$ for all $j \in [n]$ and lower bounding $-\lambda \cdot N \cdot A[n-i+1] \geq -\lambda N \cdot A[n]$. Therefore:

$$-\lambda \cdot N \cdot A[n] \leq s_{\text{Bob}}(i) \leq \frac{1}{2} + \sum_{j=1}^{n} \frac{1}{A[j]} - \lambda \cdot N \cdot A[1].$$

Now we show that Bob only requires $N - n$ data points to achieve these magnitudes of $s_{\text{Bob}}(i)$, with hinge loss terms which will have non-zero slope at *any* point in $A$. First, we show that Bob can attain the lower bound: Suppose that Bob sets $B_k(i) = A[n]$ for all $N - n$ of his data points (note that these slopes are all negative and apply for all $w \leq A[n]$). Then we need

$$\lambda N A[n] = (N - n) \cdot \frac{1}{A[n]}$$

and thus

$$\lambda A[n]^2 = (1 - c_{\text{Alice}})$$

since $N - n = (1 - c_{\text{Alice}}) \cdot N$. Therefore, for $c_{\text{Alice}} \in (0, 1)$, we require

$$c_{\text{Alice}} = 1 - \lambda A[n]^2 \in (0, 1).$$

Thus, plugging in $A[n] = 1 + \sqrt{\frac{1}{2\lambda}}$, we have

$$1 - \lambda A[n]^2 = 1 - \left(\sqrt{\lambda} + \sqrt{1/2}\right)^2 = \frac{1}{2} - \lambda - \sqrt{2\lambda}$$

from which follows the requirement that

$$\frac{1}{2} - \lambda - \sqrt{2\lambda} \in (0, 1)$$

and thus

$$\lambda + \sqrt{2\lambda} < \frac{1}{2} \quad \text{and} \quad \lambda + \sqrt{2\lambda} > -1/2,$$

the second of which is trivially true since $\lambda > 0$. The first equation is satisfied for $\lambda < \frac{3}{2} - \sqrt{2} \approx 0.08$, which holds under our assumption on $\lambda$. Thus we can choose a valid $c_{\text{Alice}}$:

$$c_{\text{Alice}} = \frac{1}{2} - \lambda - \sqrt{2\lambda} < 1/2.$$

Since Bob sets $B_k(i) = A[n]$ for all $k \in [N - n]$, the corresponding slopes of the hinge functions are non-zero for all $A[i] \in A$.

Now we show that Bob can attain the upper bound. We consider two cases. First suppose $1/2 - \lambda N \cdot A[1] \leq 0$, and the upper bound we must attain is at most $n/A[1]$. In this case, it is easy to see that Bob can set $B_k(i) = -A[1]$ for $n < N - n$ of his $N - n$ points to attain the slope of $n/A[1]$, since $n < N/2$ because $c_{\text{Alice}} < 1 - c_{\text{Alice}}$ for our choice of $c_{\text{Alice}}$. Since the function $\left[1 + \frac{w}{A[1]}\right]_+ = 1 + \frac{w}{A[1]}$ for all $w \geq -A[1]$, and all $A[j] \in A$ satisfy $A[j] > -A[1]$, the slopes due to these terms apply at all $A[j]$ for $j \in [n]$. In the other case, $0 < 1/2 - \lambda N \cdot A[1] \leq 1/2$ instead. Bob only needs to add a data point $B_k(i) = -2$ to recover the extra term of $1/2$ (the slope will be $1/2$ for any point in $A$ since $-2 < A[1]$). Thus, as long as

$$c_{\text{Alice}} + \frac{1}{N} < 1 - c_{\text{Alice}},$$

and thus

$$N > \frac{1}{1 - 2c_{\text{Alice}}} = \frac{1}{1 - 2\left(\frac{1}{2} - \lambda - \sqrt{2\lambda}\right)} = \frac{1}{2\lambda + \sqrt{8\lambda}},$$

this modified argument works. Therefore, we have established that Bob can construct both $s_{\text{Bob}}(i)$ as small as $-\lambda A[n]$ and as large as $\sum_{j=1}^{n} \frac{1}{A[1]} \geq \sum_{j=1}^{n} \left(\frac{1}{A[j]} - \lambda\right)$.

Thus, all that remains is to show that Bob can reach any magnitude in-between these two extremes. This fact is also simple to see: Bob is allowed to set $B_k(i) < 0$ or $B_k(i) > A[n]$ to obtain exact values for $s_{\text{Bob}}(i)$. Since we can attain the maximum and the minimum, we can replace the points in the minimizing set as needed with real-valued $B_k(i) < 0$ to adjust the value of $s_{\text{Bob}}(i)$ all the way up to the maximal value attainable with $N - n$ points, while not changing the total number of points needed.

We remark that every choice of Bob is either setting $B_k(i) = A[n]$ for negative slope, or setting $B_k(i) < 0$ for a positive slope. Thus, there are no new points where the slope is disjoint in $[1, A[n]]$. Finally, we also note that all choices of $s_{\text{Bob}}(i)$ are valid in the sense that they preserve the property that the optimum satisfies $w^* \geq A[1]$ (see Lemma 4.4.5), since the required upper bound on $s_{\text{Bob}}(i)$ for this statement to hold obeys the constraint by definition (with an unimportant exception in the case where $\mathbf{a} = (0, \ldots, 0)$, see Remark 4.4.6). □

Now we give the proof of Theorem 4.3.1.

*Proof.* Given an instance of Augmented Index of size $\left(\frac{1}{2} - \lambda - \sqrt{2\lambda}\right) \cdot \frac{1}{2\sqrt{\epsilon}}$, we construct the Streaming SVM instance from Construction 4.4.4 of size $N = \frac{1}{2\sqrt{\epsilon}}$ (in the language of Construction 4.4.4, $c_{\text{Alice}} = \frac{1}{2} - \lambda - \sqrt{2\lambda}$). Lemmas 4.4.5 and 4.4.9 ensure the construction is valid: the optimum is guaranteed to lie near a point in $A$ and Bob can legitimately construct the desired magnitude of $s_{\text{Bob}}(i)$ given only $i$ and $a_1, \ldots, a_{i-1}$ (the definition of Bob's input in the Augmented Index instance) with $N - n$ points. Note that in Lemma 4.4.9, there is a requirement that

$$N > \frac{1}{2\lambda + \sqrt{8\lambda}}.$$

When $N = 1/2\sqrt{\epsilon}$ and $\epsilon < 2\lambda$ both hold, this requirement is satisfied. Lemma 4.4.9 is also the reason we must restrict the magnitude of the points in $A$ to satisfy $\leq \Theta(1/\sqrt{\lambda})$ (see Remark 4.4.10).

Then, we claim that the minimizers $w_0^*$ and $w_1^*$ in the two cases ($a_i = 0$ or $a_i = 1$) are spaced distance $\Delta$ apart with respect to the $\ell_2$ distance (Lemma 4.4.7). Then, in Lemma 4.4.8, we show that given the output $\hat{w}$ of an $\epsilon/16$-optimal streaming SVM algorithm (and conditioning on the success event),

$$d_{\ell_2}(\hat{w}, w^*) < \frac{\Delta}{2}.$$

Thus, a (randomized) one-pass streaming algorithm that solves the $\epsilon/16$-optimal one-pass Streaming SVM problem (Definition 4.1.2) determines whether $a_i = 0$ or $a_i = 1$ via a distance test which succeeds with probability at least $3/4$. Finally, applying the communication lower bound for Augmented Index of $\Omega(n)$ [13], we have proven a communication lower bound of $\Omega\left(\frac{\frac{1}{2}-\lambda-\sqrt{2\lambda}}{8\sqrt{\epsilon}}\right) = \Omega(1/\sqrt{\epsilon})$ as desired (since $\frac{1}{2} - \lambda - \sqrt{2\lambda} > 0$ and is treated like a constant function of $\epsilon$ for our restrictions on $\lambda$). □

*Remark* 4.4.10. The upper bound constraint on the size of $A[n]$ comes from the requirement that $\lambda A[n]^2 = 1 - c_{\text{Alice}} \in (0, 1)$ in Lemma 4.4.9. In particular, we must have

$$A[n] \leq \frac{1}{\sqrt{\lambda}}.$$

Since $A[1] < A[n]$, we must also require $\lambda < 1/A[1]^2$, but this inequality already holds under our assumptions ($A[1] = 1, \lambda < 3/2 - \sqrt{2} < 1/2$).

# Chapter 5: Conclusion and Future Work

In this thesis, we studied resource-limited algorithms for machine learning in the context of resource-efficient structured model classes and machine learning in the online streaming setting. We conclude by summarizing each chapter's results and pointing to directions for future work.

## 5.1 Sparse Monomials

In Chapter 2, we studied the problem of learning sparse monomials of highly-correlated features. Our work provides the first attribute-efficient analysis (handling arbitrarily high correlations) in a non-product distribution setting, which has been a major challenge in the prior work (e.g., [25]; [15]). By leveraging a folklore technique from applied statistics, namely applying the $\log(|\cdot|)$ transform to the features and responses, we reduced this problem to a sparse linear regression problem. By analyzing how the covariance matrix changes after the $\log(|\cdot|)$ transform, we show that our procedure works under the minimal conditions required for the model to be identifiable.

We summarize the conceptual contributions of the chapter as follows.

1. Learning degree-$k$ sparse polynomial functions with $\text{poly}(\log(p), k)$ samples in $p^{o(k)}$-time under non-product distributions is a challenging problem. Our work gives a new algorithmic line-of-attack for this problem, namely transforming both the response and the features such that each relevant variable participates in an $O(1)$-degree interaction in the transformed model, reducing the computational burden of searching for relevant variables from $p^{\Omega(k)}$ to $p^{O(1)}$. Although we study this general principle in a specialized setting, we believe our techniques (Lemma 2.4.1) can be useful for analyzing other instances of this algorithmic idea. The fact that no existing approach gives attribute-efficient algorithms with $p^{o(k)}$ run-time for the comparatively simple sparse monomial problem underscores the promise of this approach.

2. Our analysis uncovers a *blessing of non-linearity*. Specifically, the assumptions on the correlation structure needed to learn a class of sparse non-linear functions are less restrictive than those needed to learn sparse linear functions. We require only minimal assumptions on the dependence structure to ensure identifiability, a significant departure from previous results.
3. We demonstrate the minimum eigenvalue of the log-transformed data covariance matrix is strictly positive with high probability, regardless of the initial rank. Thus, nonlinear data transformations can destroy low-rank covariance structure, a principle which may be useful for other estimation problems.

We conclude with a few open problems. The most immediate is to find an efficient algorithm for learning sparse monomials in the presence of additive noise. [104] proved a Statistical Query (SQ) lower bound for the problem of learning sparse monomials in the agnostic setting, and thus no efficient SQ algorithm exists for this problem – it remains open whether a non-SQ algorithm exists. Another interesting question is to relax the Gaussian distribution assumption (e.g., to rotations of general product distributions), and to also try to handle larger families of sparse polynomials over highly-correlated features. Finally, it would be nice to obtain a better dependence on the maximum correlation parameter, since our sample complexity currently blows up as it approaches 1.

### 5.2 Low-Rank Initialization

In Chapter 3, we introduced a novel low-rank initialization framework for deep learning: Algorithms 2 and 3 empirically tend to outperform the existing method of choice, spectral initialization, while being essentially as efficient to implement. We view our approach as a simple drop-in substitute to spectral initialization that practitioners can try to increase performance. While we do not expect huge performance increases, since the method is simple and does not require significantly more compute or memory, Table 3.1 demonstrates a relatively reliable performance increase in generalization accuracy.

This chapter also contributes to the understanding of the mechanism behind successful low-rank initialization methods for deep learning: a critical component of methods like spectral ini-

tialization and our **NLRA** framework is the quality of the *function approximation* of the full-rank initialized parameters, rather than parameter approximation. We demonstrate this with empirical results showcasing the positive correlation between decreasing nonlinear low-rank approximation error and decreasing generalization error, and with theoretical guarantees and intuition for when our methods should outperform classic approaches (high input dimension together with larger network width and smaller target ranks) (Theorem 3.3.9 and Corollary 3.3.11).

We close with a few directions for future work: It would be interesting to 1) provably characterize the impact of initialization schemes on downstream generalization, low-rank or otherwise; 2) identify optimal *distributions* to sample low-rank weights from without optimization; 3) discover better low-rank training methods; 4) extend our theory beyond 1-hidden-layer ReLU networks to other activations, depths, and architectures; 5) extend our initialization framework to "efficiently-parameterized" networks beyond low-rank; 6) determine the complexity of the nonlinear low-rank approximation problem both in the general case as well as in an average case setting (perhaps when the weight matrix $W$ has columns drawn from the uniform spherical distribution), as well as understand the sample complexity and computational complexity of the problem when one is only provided sample access to the weights in both general and average case settings.

### 5.3 Memory-Bounded Streaming Optimization

The main task is to close the gap between upper and lower bounds in dimensions larger than 1 – in high dimension (say $d > \log(1/\epsilon)$), this gap is quadratic. We believe that stronger lower bounds are possible which can essentially close the gap using structural lemmas characterizing the properties of the SVM objective via local perturbation analysis near the optimum, and proving these lower bounds is currently a work in progress. Beyond streaming SVM, it would be interesting to consider the memory requirements for other non-smooth classes of streaming algorithms, though similar lower bounds likely hold for a broader class of instances beyond SVM (which is additionally strongly convex). It would also be fruitful to consider multi-pass settings of the problem, and to understand whether multiple rounds can non-trivially mitigate the memory cost.

# References


[1] C. Huyen, *Data distribution shifts and monitoring*, 2022.

[2] A. Andoni, R. Dudeja, D. Hsu, and K. Vodrahalli, "Attribute-efficient learning of monomials over highly-correlated variables," in *Algorithmic Learning Theory*, PMLR, 2019, pp. 127–161.

[3] K. Vodrahalli, R. Shivanna, M. Sathiamoorthy, S. Jain, and E. H. Chi, "Nonlinear initialization methods for low-rank neural networks," *CoRR*, vol. abs/2202.00834, 2022. arXiv: 2202.00834.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[5] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, PMLR, 2019, pp. 6105–6114.

[6] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[7] R. O'Donnell, "Analysis of boolean functions," in New York, NY, USA: Cambridge University Press, 2014, ch. 11 sec. 2, 8, pp. 334–338, 368–385, ISBN: 1107038324, 9781107038325.

[8] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015, ISBN: 1498712169, 9781498712163.

[9] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[10] R. J. Tibshirani, "The lasso problem and uniqueness," *Electronic Journal of Statistics*, vol. 7, no. none, pp. 1456 –1490, 2013.

[11] P. J. Bickel, Y. Ritov, and A. B. Tsybakov, "Simultaneous analysis of lasso and dantzig selector," *Annals of Statistics*, vol. 37, pp. 1705–1732, 4 2009.

[12] Y. E. Nesterov, *Introductory Lectures on Convex Optimization - A Basic Course*, ser. Applied Optimization. Springer, 2004, vol. 87, ISBN: 978-1-4613-4691-3.

[13] T. Roughgarden *et al.*, "Communication complexity (for algorithm designers)," *Foundations and Trends® in Theoretical Computer Science*, vol. 11, no. 3–4, pp. 217–404, 2016.

[14] A. Andoni, R. Dudeja, D. Hsu, and K. Vodrahalli, "Attribute-efficient learning of monomials over highly-correlated variables," in *Algorithmic Learning Theory, ALT 2019, 22-24 March 2019, Chicago, Illinois, USA*, A. Garivier and S. Kale, Eds., ser. Proceedings of Machine Learning Research, vol. 98, PMLR, 2019, pp. 127–161.

[15] A. Andoni, R. Panigrahy, G. Valiant, and L. Zhang, "Learning sparse polynomial functions," in *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '14, Portland, Oregon: Society for Industrial and Applied Mathematics, 2014, pp. 500–510, ISBN: 978-1-611973-38-9.

[16] S. Negahban and D. Shah, "Learning sparse boolean polynomials," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, IEEE, 2012, pp. 2032–2036.

[17] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 463–482, 2002.

[18] M. H. Quang, "Reproducing kernel hilbert spaces in learning theory," Ph.D. dissertation, Brown University, 2006.

[19] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[20] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine learning*, vol. 2, no. 4, pp. 285–318, 1988.

[21] D. Helmbold, R. Sloan, and M. K. Warmuth, "Learning integer lattices," *SIAM Journal on Computing*, vol. 21, no. 2, pp. 240–266, 1992.

[22] A. Blum, "On-line algorithms in machine learning," in *Online algorithms*, Springer, 1998, pp. 306–325.

[23] A. R. Klivans and R. A. Servedio, "Toward attribute efficient learning of decision lists and parities," *Journal of Machine Learning Research*, vol. 7, no. Apr, pp. 587–602, 2006.

[24] G. Valiant, "Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem," *Journal of the ACM (JACM)*, vol. 62, no. 2, p. 13, 2015.

[25] A. T. Kalai, A. Samorodnitsky, and S.-H. Teng, "Learning and smoothed analysis," in *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, IEEE, 2009, pp. 395–404.

[26] M. Kocaoglu, K. Shanmugam, A. G. Dimakis, and A. Klivans, "Sparse polynomial learning and graph sketching," in *Advances in Neural Information Processing Systems*, 2014, pp. 3122–3130.

[27] O. N. Keene, "The log transformation is special," *Statistics in medicine*, vol. 14, no. 8, pp. 811–819, 1995.

[28] S. Chen and D. Donoho, "Basis pursuit," in *28th Asilomar conf. Signals, Systems Computers*, 1994.

[29] S. F. Edwards and P. W. Anderson, "Theory of spin glasses," *Journal of Physics F: Metal Physics*, vol. 5, no. 5, p. 965, 1975.

[30] R. B. Bapat and V. S. Sunder, "On majorization and schur products," *Linear Algebra and its Applications*, vol. 72, pp. 107 –117, 1985.

[31] A. K. Kuchibhotla and A. Chakrabortty, "Moving beyond sub-gaussianity in high-dimensional statistics: Applications in covariance estimation and linear regression," *ArXiv e-prints*, 2018. arXiv: 1804.02605.

[32] J. W. Rae *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[33] K. Shin, H. Kwak, K.-M. Kim, S. Y. Kim, and M. N. Ramstrom, "Scaling law for recommendation models: Towards general-purpose user representations," *arXiv preprint arXiv:2111.11294*, 2021.

[34] T. Henighan *et al.*, "Scaling laws for autoregressive generative modeling," *arXiv preprint arXiv:2010.14701*, 2020.

[35] M. A. Gordon, K. Duh, and J. Kaplan, "Data and parameter scaling laws for neural machine translation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 5915–5922.

[36] U. Sharma and J. Kaplan, "A neural scaling law from the dimension of the data manifold," *arXiv preprint arXiv:2004.10802*, 2020.

[37] J. Kaplan *et al.*, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[38] S. Bubeck and M. Sellke, "A universal law of robustness via isoperimetry," *arXiv preprint arXiv:2105.12806*, 2021.

[39] R. Reed, "Pruning algorithms-a survey," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[40] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *arXiv preprint arXiv:2003.03033*, 2020.

[41] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 2, Morgan-Kaufmann, 1990.

[42] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems*, S. Hanson, J. Cowan, and C. Giles, Eds., vol. 5, Morgan-Kaufmann, 1993.

[43] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 1, Morgan-Kaufmann, 1989.

[44] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, 4860–4874, ISBN: 9781510860964.

[45] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[46] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer, "Base layers: Simplifying training of large, sparse models," *arXiv preprint arXiv:2103.16716*, 2021.

[47] H. Yang, W. Wen, and H. Li, "Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures," in *International Conference on Learning Representations*, 2019.

[48] J. Su *et al.*, "Sanity-checking pruning methods: Random tickets can win the jackpot," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 20 390–20 401.

[49] J. Frankle, G. K. Dziugaite, D. Roy, and M. Carbin, "Pruning neural networks at initialization: Why are we missing the mark?" In *International Conference on Learning Representations*, 2020.

[50] J. Fischer and R. Burkholz, "Plant'n'seek: Can you find the winning ticket?" *arXiv preprint arXiv:2111.11153*, 2021.

[51] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3779–3787.

[52] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[53] Anonymous, "Language model compression with weighted low-rank factorization," in *Submitted to The Tenth International Conference on Learning Representations*, under review, 2022.

[54] Y. Idelbayev and M. A. Carreira-Perpinán, "Low-rank compression of neural nets: Learning the rank of each layer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8049–8059.

[55] M. Tukan, A. Maalouf, M. Weksler, and D. Feldman, "No fine-tuning, no cry: Robust svd for compressing deep networks," *Sensors*, vol. 21, no. 16, p. 5599, 2021.

[56] B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré, "Scatterbrain: Unifying sparse and low-rank attention approximation," *arXiv preprint arXiv:2110.15343*, 2021.

[57] T. Dao *et al.*, "Kaleidoscope: An efficient, learnable representation for all structured linear maps," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[58] M. Khodak, N. A. Tenenholtz, L. Mackey, and N. Fusi, "Initialization and regularization of factorized neural layers," in *International Conference on Learning Representations*, 2021.

[59] K. A. Vahid, A. Prabhu, A. Farhadi, and M. Rastegari, "Butterfly transform: An efficient fft based neural architecture design," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2020, pp. 12 021–12 030.

[60] H. Wang, S. Agarwal, and D. Papailiopoulos, "Pufferfish: Communication-efficient models at no extra cost," *Proceedings of Machine Learning and Systems*, vol. 3, 2021.

[61] M. Dusenberry *et al.*, "Efficient and scalable bayesian neural nets with rank-1 factors," in *International conference on machine learning*, PMLR, 2020, pp. 2782–2792.

[62] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[63] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 1943–1955, 2015.

[64] K. Choromanski *et al.*, "Rethinking attention with performers," *arXiv preprint arXiv:2009.14794*, 2020.

[65] B. Chen *et al.*, "{mongoose}: A learnable {lsh} framework for efficient neural network training," in *International Conference on Learning Representations*, 2021.

[66] R. Panigrahy, X. Wang, and M. Zaheer, "Sketch based memory for neural networks," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, A. Banerjee and K. Fukumizu, Eds., ser. Proceedings of Machine Learning Research, vol. 130, PMLR, 2021, pp. 3169–3177.

[67] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," in *International Conference on Learning Representations*, 2019.

[68] H. Mostafa and X. Wang, "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization," in *International Conference on Machine Learning*, PMLR, 2019, pp. 4646–4655.

[69] Q. He, M. Dong, L. Schwiebert, and W. Shi, "Learning pruned structure and weights simultaneously from scratch: An attention based approach," *arXiv preprint arXiv:2111.02399*, 2021.

[70] B. Mussay, M. Osadchy, V. Braverman, S. Zhou, and D. Feldman, "Data-independent neural pruning via coresets," in *International Conference on Learning Representations*, 2020.

[71] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035.

[72] NVIDIA, "Nvidia ampere ga102 gpu architecture," White Paper, 2020.

[73] A. Mishra *et al.*, "Accelerating sparse deep neural networks," *arXiv preprint arXiv:2104.08378*, 2021.

[74] T. Bachlechner, B. P. Majumder, H. H. Mao, G. W. Cottrell, and J. McAuley, "Rezero is all you need: Fast convergence at large depth," *arXiv preprint arXiv:2003.04887*, 2020.

[75] K. Choromanski, C. Downey, and B. Boots, "Initialization matters: Orthogonal predictive state recurrent neural networks," in *International Conference on Learning Representations*, 2018.

[76] Y. N. Dauphin and S. Schoenholz, "Metainit: Initializing learning by learning to initialize," *Advances in Neural Information Processing Systems*, vol. 32, pp. 12 645–12 657, 2019.

[77] W. Hu, L. Xiao, and J. Pennington, "Provable benefit of orthogonal initialization in optimizing deep linear networks," *arXiv preprint arXiv:2001.05992*, 2020.

[78] X. S. Huang, F. Perez, J. Ba, and M. Volkovs, "Improving transformer optimization through better initialization," in *International Conference on Machine Learning*, PMLR, 2020, pp. 4475–4483.

[79] D. Mishkin and J. Matas, "All you need is a good init," *arXiv preprint arXiv:1511.06422*, 2015.

[80] J. Pennington, S. S. Schoenholz, and S. Ganguli, "Resurrecting the sigmoid in deep learning through dynamical isometry: Theory and practice," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4788–4798.

[81] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5393–5402.

[82] Y. Zhang, B. Li, Y. Liu, H. Wang, and C. Miao, "Initialization matters: Regularizing manifold-informed initialization for neural recommendation systems," *arXiv preprint arXiv:2106.04993*, 2021.

[83] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," *Found. Trends Theor. Comput. Sci.*, vol. 10, no. 1–2, 1–157, 2014.

[84] M. W. Mahoney, "Randomized algorithms for matrices and data," *Found. Trends Mach. Learn.*, vol. 3, no. 2, 123–224, 2011.

[85] C. Musco, C. Musco, and D. P. Woodruff, "Simple heuristics yield provable algorithms for masked low-rank approximation," *arXiv preprint arXiv:1904.09841*, 2019.

[86] I. Razenshteyn, Z. Song, and D. P. Woodruff, "Weighted low rank approximations with provable guarantees," in *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '16, Cambridge, MA, USA: Association for Computing Machinery, 2016, 250–263, ISBN: 9781450341325.

[87] M. Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[88] Y. Cho and L. Saul, "Kernel methods for deep learning," in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., vol. 22, Curran Associates, Inc., 2009.

[89] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[90] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 249–256.

[91] R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices," in *Compressed Sensing: Theory and Applications*, Y. C. Eldar and G. Kutyniok, Eds. Cambridge University Press, 2012, 210–268.

[92] ——, *High-dimensional probability: An introduction with applications in data science*. Cambridge university press, 2018, vol. 47.

[93] R. Van Handel, "Probability in high dimension," Princeton University, Tech. Rep., 2016.

[94] M. Abramowitz and I. A. Stegun, "Handbook of mathematical functions," in New York: Dover Publications, 1964, ch. 6, sec. 4, pp. 260–261, ISBN: 978-0-486-61272-0.

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

[96] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *CoRR*, vol. abs/1710.05941, 2017. arXiv: 1710.05941.

[97] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, "Is the sample good enough? comparing data from twitter's streaming API with twitter's firehose," *CoRR*, vol. abs/1306.5204, 2013. arXiv: 1306.5204.

[98] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, 273–297, 1995.

[99] J. Huggins, R. P. Adams, and T. Broderick, "Pass-glm: Polynomial approximate sufficient statistics for scalable bayesian glm inference," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3611–3621, 2017.

[100] A. Andoni, C. Burns, Y. Li, S. Mahabadi, and D. P. Woodruff, "Streaming complexity of svms," in *APPROX/RANDOM*, 2020.

[101] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for svm," *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

[102] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, ser. COLT' 98, Madison, Wisconsin, USA: Association for Computing Machinery, 1998, 209–217, ISBN: 1581130570.

[103] A. Blum, "Random projection, margins, kernels, and feature-selection," in *Proceedings of the 2005 International Conference on Subspace, Latent Structure and Feature Selection*, ser. SLSFS'05, Bohinj, Slovenia: Springer-Verlag, 2005, 52–68, ISBN: 3540341374.

[104] S. Goel, A. Gollakota, and A. Klivans, "Statistical-query lower bounds via functional gradients," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[105] A. Winkelbauer, "Moments and absolute moments of the normal distribution," *ArXiv e-prints*, 2014. arXiv: 1209.4340.

[106] E. M. Stein and R. Shakarchi, "Complex analysis," in Princeton, New Jersey: Princeton University Press, 2003, ch. 6, 7, pp. 159–204, ISBN: 9781400831159.

[107] J. Sondow, "An antisymmetric formula for euler's constant," *Mathematics Magazine*, vol. 71, pp. 219–220, 3 1998.

[108] H. Robbins, "A remark on stirling's formula," *The American Mathematical Monthly*, vol. 62, no. 1, pp. 26–29, 1955.

[109] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)* Baltimore, MD, USA: Johns Hopkins University Press, 1996, ISBN: 0-8018-5414-8.

[110] S. Du, J. Lee, Y. Tian, A. Singh, and B. Poczos, "Gradient descent learns one-hidden-layer CNN: Don't be afraid of spurious local minima," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 1339–1348.

[111] S. Goel, S. Karmalkar, and A. Klivans, "Time/accuracy tradeoffs for learning a relu with respect to gaussian marginals," *Advances in neural information processing systems*, 2019.

[112] J. Martens *et al.*, "Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping," *arXiv preprint arXiv:2110.01765*, 2021.

# Appendix A: Deferred Proofs from Chapter 2

## A.1 Proof of Proposition 2.3.1.

**Proposition A.1.1.** *A unique solution $\hat{\beta}$ to the transformed model in Eq.* (2.2)*) is the unique solution to the original model in Eq.* (2.1).

*Proof.* We proceed by reversing each step of the transformation and demonstrating that the solutions do not change. First, note that the logarithm is invertible over the positive reals, which allows us to undo the log transformation without any effect, since absolute value ensures the domain is non-negative. Thus only consider the modified problem using data $\hat{y}_i = |y_i|, \hat{x}_{i,j} = |x_{i,j}|$. Since absolute value distributes under multiplication, $\hat{y}_i = |y_i| = |\prod_{j\in S} x_{i,j}^{\beta_j}| = \prod_{j\in S} |x_{i,j}|^{\beta_j} = \prod_{j\in S} \hat{x}_{i,j}^{\beta_j}$ and the resulting data points $(\hat{x}_i, \hat{y}_i)$ still satisfy the monomial model. Thus, if there is a unique solution on the transformed data $(\log|x_i|, \log|y_i|)$, it must also be the unique solution on all of the data. □

## A.2 Supporting Lemmas and Proof for Lemma 2.4.1

### A.2.1 Calculating the First and Second Log-Moments

In order to use the ideas from Section 1.3.2, we first need to show that the function $\log|\cdot| \in L_2(\mathcal{N}(0,1))$, e.g., that $\mathbb{E}_{w\sim\mathcal{N}(0,1)}[\log^2|w|] = \alpha < \infty$. Along the way, it will also be useful to calculate and record $\mathbb{E}_{w\sim\mathcal{N}(0,1)}[\log|w|] = \tau < \infty$. In order to directly calculate these quantities, we use an idea from statistical physics called the replica trick ([29]). The idea is to note that $\frac{d}{d\nu}\mathbb{E}[a^\nu] = \mathbb{E}[\frac{d}{d\nu}a^\nu] = \mathbb{E}[(\log a)a^\nu]$. In general, if one takes $m$ derivatives, the result will be $\frac{d^m}{d\nu^m}\mathbb{E}[a^\nu] = \mathbb{E}[a^\nu \log^m(a)]$. Then, taking the limit as $\nu \to 0$ yields

**Lemma A.2.1** (Replica trick)**.** *Let $a$ be a non-negative random variable. Then,*

$$\mathbb{E}_a[\log^m a] = \lim_{\nu \to 0} \frac{d^m}{d\nu^m} \mathbb{E}_a[a^\nu]. \tag{A.1}$$

We refer to the LHS expression in Eq. (A.1) as the $m^{th}$ *log-moment* of $a$. Thus, as long as we can get an analytic expression for $\mathbb{E}_a[a^\nu]$ which is valid for $\nu \in \mathbb{R}^+$, we can take the continuous limit and derive expressions for the first and second log-moments of $a$, where $a = |w|, w \sim \mathcal{N}(0, 1)$. We also note that in the upcoming discussion, $\gamma$ refers to the *Euler-Mascheroni* constant.

We will apply the replica trick to calculate the first two log-moments. We first need to collect some lemmas from the literature.

**Lemma A.2.2** (Moments of Absolute Gaussian Distribution)**.** *We have for $\nu \in \mathbb{R}^+$ with $w \sim \mathcal{N}(0, 1)$*

$$\mathbb{E}_w[|w|^\nu] = \frac{1}{\sqrt{\pi}} 2^{\nu/2} \Gamma\left(\frac{\nu + 1}{2}\right)$$

*where $\Gamma$ is the gamma function.*

*Proof.* For derivation, see [105]. □

We will need some properties of the gamma function, several of which depend on the polygamma function $\psi$. We take these facts from [106], [94], and [107].

**Definition A.2.3** (Polygamma function)**.** The *polygamma function of order* 0 is defined by

$$\psi(x) := \frac{d}{dx} \log(\Gamma(x)) = \frac{\Gamma'(x)}{\Gamma(x)}.$$

The *polygamma function of order $i \geq 1$* is defined by

$$\psi^{(i)}(x) := \frac{d^i}{dx^i} \psi(x).$$

**Lemma A.2.4** (Properties of the Gamma and Polygamma functions)**.** *The derivative of* $\Gamma(x)$ *is given by*

$$\frac{d}{dx}\Gamma(x) = \Gamma(x)\psi(x).$$

*The Taylor series expansions for* $\psi^{(i)}(1+x)$ *are*

$$\psi(x+1) = -\gamma + \sum_{j=1}^{\infty}(-1)^{j+1}\zeta(j+1)x^j$$
$$\psi^{(i)}(x+1) = \sum_{j=0}^{\infty}(-1)^{i+j+1}\frac{(i+j)!}{j!}\zeta(i+j+1)x^j \text{ for } i \geq 1$$

*(convergence is for* $|x| < 1$*). Above,* $\gamma$ *is the Euler-Mascheroni constant and* $\zeta(s) = \sum_{n=1}^{\infty}\frac{1}{n^s}$ *is the zeta function.*

The first identity above follows from [94] and the antisymmetric formula for $\gamma$ given in [107].

Now, we can use these facts to calculate the first two log-moments of $|w|$.

**Lemma A.2.5** (First log-moment ($\tau$))**.** *Define*

$$\tau := \mathbb{E}_w[\log|w|] = -\frac{1}{2}\left(\log(2) + \gamma\right) \approx -0.635.$$

*We also record that* $\tau^2 \approx 0.403$.

*Proof.* We apply the replica trick to get

$$\begin{aligned}
\mathbb{E}_w[\log|w|] &= \lim_{\nu\to 0}\frac{d}{d\nu}\frac{1}{\sqrt{\pi}}2^{\nu/2}\Gamma(\frac{\nu+1}{2}) \\
&= \frac{1}{2}\log(2) + \frac{1}{2\sqrt{\pi}}\lim_{\nu\to 0}\Gamma'\left(\frac{1+\nu}{2}\right)\sqrt{2}^{\nu} \\
&= \frac{1}{2}\log(2) + \frac{1}{2\sqrt{\pi}}\left(1\cdot\sqrt{\pi}\cdot\lim_{\nu\to 0}\psi\left(\frac{1+\nu}{2}\right)\right) \\
&= \frac{1}{2}\left(\log(2) + \psi\left(\frac{1}{2}\right)\right)
\end{aligned}$$

where we used the derivative of $\Gamma$ and the fact that the limit existed individually for each term in the second product. Applying the Taylor expansion of $\psi(x+1)$ and plugging in $x = -1/2$, we get using properties of infinite geometric series that

$$\begin{aligned}
\psi(1/2) &= -\gamma + \sum_{j=1}^{\infty}(-1)^{j+1}\zeta(j+1)(-1/2)^j \\
&= -\left(\gamma + \sum_{j=1}^{\infty}\frac{1}{2^j}\zeta(j+1)\right) \\
&= -\left(\gamma + \sum_{j,n=1}^{\infty}\frac{1}{2^j}\frac{1}{n^{j+1}}\right) \\
&= -\left(\gamma + \sum_{n=1}^{\infty}\frac{1}{n}\sum_{j=1}^{\infty}\frac{1}{(2n)^j}\right) \\
&= -\left(\gamma + \sum_{n=1}^{\infty}\frac{1}{n}\left(\frac{1/2n}{1-1/2n}\right)\right) \\
&= -\left(\gamma + \sum_{n=1}^{\infty}\frac{1}{n(2n-1)}\right)
\end{aligned}$$

Then, consider the Taylor series for $\log(x)$ centered at $x = 1$, which has radius of convergence

$|x-1| \le 1$. We have, plugging in $x = 2$,

$$\begin{aligned}
\log(x) &= \sum_{n=1}^{\infty} (-1)^{n+1} \frac{(x-1)^n}{n} \\
\log(2) &= \sum_{n=1}^{\infty} (-1)^{n+1} \frac{1}{n} \\
&= (1 - \frac{1}{2}) + (\frac{1}{3} - \frac{1}{4}) + (\frac{1}{5} - \frac{1}{6}) + \dots \\
&= \frac{1}{2} \sum_{n=1}^{\infty} \frac{1}{n(2n-1)}.
\end{aligned}$$

Thus, we conclude that $\psi(1/2) = -(\gamma + 2\log(2))$, and overall that

$$\tau = \frac{1}{2}\left(\log(2) - \gamma - 2\log(2)\right) = -\frac{1}{2}\left(\log(2) + \gamma\right).$$

□

**Lemma A.2.6** (Second log-moment ($\alpha$))**.**

$$\alpha := \mathbb{E}_w[\log^2 |w|] = \frac{1}{4}\left(\gamma^2 + \frac{\pi^2}{2} + \log^2(2) + \gamma\log(4)\right) \approx 1.637.$$

*We also record that* $\alpha - \tau^2 \approx 1.234$.

*Proof.* We calculate the second derivative with respect to $\nu$ and evaluate it at $\nu = 0$, using the product rule and the derivatives of $\Gamma$ and $\psi$:

$$\begin{aligned}
\mathbb{E}_w[\log^2 |w|] &= \frac{d^2}{d\nu^2} \mathbb{E}_w[|w|^\nu]|_{\nu=0} \\
&= \frac{1}{2\sqrt{\pi}} \Big[ \log(2) \left( \log(\sqrt{2}) e^{\nu \log(\sqrt{2})} \Gamma\left(\frac{1+\nu}{2}\right) + \frac{1}{2} e^{\nu \log(\sqrt{2})} \Gamma\left(\frac{1+\nu}{2}\right) \psi\left(\frac{1+\nu}{2}\right) \right) \\
&+ \psi\left(\frac{1+\nu}{2}\right) \frac{d}{d\nu} \left( \sqrt{2}^\nu \Gamma\left(\frac{1+\nu}{2}\right) \right) + \frac{1}{2} \sqrt{2}^\nu \Gamma\left(\frac{1+\nu}{2}\right) \psi^{(1)}\left(\frac{1+\nu}{2}\right) \Big] \Big|_{\nu=0} \\
&= \frac{1}{4} \left[ \left( \log^2(2) - 2\log(2)(\gamma + \log(4)) + (\gamma + \log(4))^2 \right) + \left[ \psi^{(1)}\left(\frac{1+\nu}{2}\right) \right] \Big|_{\nu=0} \right]
\end{aligned}$$

where we used the results from Lemma A.2.1 to simplify, keeping in mind that we will shortly show that $\psi^{(1)}(1/2)$ exists. We use the Taylor series for $\psi^{(1)}(x+1)$ which converges for $|x| < 1$ and plug in $x = -1/2$:

$$\begin{aligned}
\psi^{(1)}(x+1) &= \sum_{j=0}^{\infty} (-1)^{j+2} \frac{(j+1)!}{j!} \zeta(j+2) x^j \\
\psi^{(1)}(1/2) &= \sum_{j=0}^{\infty} \frac{j+1}{2^j} \sum_{n=1}^{\infty} \frac{1}{n^{j+2}} \\
&= 2 \sum_{n=1}^{\infty} \frac{1}{n} \sum_{j=1}^{\infty} j \left(\frac{1}{2n}\right)^j \\
&= 4 \sum_{n=1}^{\infty} \frac{1}{(2n-1)^2} \\
&= 4 \left( \sum_{n=1}^{\infty} \frac{1}{n^2} - \sum_{n=1}^{\infty} \frac{1}{(2n)^2} \right) = 4 \left( \frac{\pi^2}{6} - \frac{\pi^2}{24} \right) = \frac{\pi^2}{2}
\end{aligned}$$

recalling that $\sum_{j=1}^{\infty} j c^j = \frac{c}{1-c} + \frac{c^2}{1-c} + \ldots = \frac{c}{(1-c)^2}$ and the fact that $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$. Plugging this value in to our previous formula, we conclude

$$\begin{aligned}
\alpha &= \frac{1}{4} \left[ \log^2(2) - 2\log(2)(\gamma + \log(4)) + (\gamma + \log(4))^2 + \frac{\pi^2}{2} \right] \\
&= \frac{1}{4} \left[ \log^2(2) + \frac{\pi^2}{2} - 2\gamma \log(2) - \log^2(4) + \gamma^2 + 2\gamma \log(4) + \log^2(4) \right] \\
&= \frac{1}{4} \left[ \gamma^2 + \log^2(2) + \frac{\pi^2}{2} + \gamma \log(4) \right].
\end{aligned}$$

□

The next lemma will be useful in the next section of the appendix, and uses similar ideas.

**Lemma A.2.7.**

$$\mathbb{E}_w[w^2 \log(|w|)] = 1 + \tau.$$

*Proof.* We have by integration by parts and the fact that $\varphi'(w) = -w\varphi(w)$

$$\begin{aligned}
\int w^2 \log(|w|)\varphi(w)dw\Big|_{\mathbb{R}} &= \int w \log(|w|) w\varphi(w)dw\Big|_{\mathbb{R}} \\
&= -\int w \log(|w|)\frac{d}{dw}\varphi(w)dw\Big|_{\mathbb{R}} \\
&= -\left(w \log(|w|)\varphi(w) - \int (1 + \log(|w|))\varphi(w)dw\right)\Big|_{\mathbb{R}} \\
&= 0 + 1 + \int \log(|w|)\varphi(w)dw\Big|_{\mathbb{R}} \\
&= 1 + \tau
\end{aligned}$$

since $\lim_{w\to\pm\infty} w \log(|w|)\varphi(w) = 0$. □

### A.2.2 Coefficients of the Hermite Expansion of $\log(|\cdot|)$

**Lemma A.2.8** (Coefficients of the Hermite Expansion for $\log(|\cdot|)$)**.** *The Hermite expansion of* $\log\left(|\cdot|\right)$

$$\log(a) = \sum_{\ell=0}^{\infty} c_\ell H_\ell(a)$$

*has* $c_0 = \tau$*, and for* $\ell \geq 1$*,*

$$c_{2\ell-1} = 0, \quad c_{2\ell} = \frac{(-1)^{\ell-1}2^{\ell-1}(\ell-1)!}{\sqrt{(2\ell)!}}.$$

*Moreover,*

$$\lim_{\ell\to\infty} c_{2\ell}^2 \cdot \ell^{3/2} = \frac{\sqrt{\pi}}{4},$$

*and for* $\ell \geq 2$*,*

$$c_{2\ell}^2 \geq \frac{1}{5} \cdot \frac{1}{\ell^{3/2}}.$$

*Proof.* Our goal is to calculate $\mathbb{E}_w[H_\ell(w)\log(|w|)]$. Recall that $\varphi(w)$ is the standard Gaussian density. We proceed by making use of several properties of Hermite polynomials from Section 1.3.2 and applying integration by parts. First define the indefinite integral and apply the property $H'_{i+1}(w) = \sqrt{i+1}H_i(w)$:

$$\begin{aligned}
A_i &= \int \log(|w|)H_i(w)\varphi(w)dw \\
&= \frac{1}{\sqrt{i+1}}\int H'_{i+1}(w)\log(|w|)\varphi(w)dw \\
&= \frac{1}{\sqrt{i+1}}\left(H_{i+1}(w)\log(|w|)\varphi(w) - \int H_{i+1}(w)\left(\frac{1}{w} - w\log(|w|)\right)\varphi(w)dw\right)
\end{aligned}$$

where we used the fact $\frac{d}{dw}\log(|w|)\varphi(w) = \left(\frac{1}{w} - w\log(|w|)\right)\varphi(w)$. Let $V_i(w) = H_i(w)\log(|w|)\varphi(w)$. Then, applying the relation $wH_i(w) = H'_i(w) + \sqrt{i+1}H_{i+1}(w)$, we get

$$\begin{aligned}
A_i &= \frac{1}{\sqrt{i+1}}\left(V_{i+1}(w) + \int \left(H'_{i+1}(w) + \sqrt{i+2}H_{i+2}(w)\right)\log(|w|)\varphi(w)dw\right. \\
&\qquad\qquad \left. - \int \frac{1}{\sqrt{i+1}}\frac{wH_i(w) - H'_i(w)}{w}\varphi(w)dw\right) \\
&= \frac{1}{\sqrt{i+1}}\left(V_{i+1}(w) + \sqrt{i+2}A_{i+2} + \sqrt{i+1}A_i - \frac{1}{\sqrt{i+1}}\int\left(H_i(w) - \frac{1}{w}H'_i(w)\right)\varphi(w)dw\right).
\end{aligned}$$

Assuming that $i > 0$, orthogonality implies $\int_{\mathbb{R}} H_i(w)\varphi(w)dw = 0$, and we cancel it out now (since eventually we will evaluate everything over $\mathbb{R}$). We simplify the equation to

$$V_{i+1}(w) + \sqrt{i+2}A_{i+2} + \sqrt{\frac{i}{i+1}}\int \frac{1}{w}H_{i-1}(w)\varphi(w)dw = 0.$$

Then we calculate $\frac{d}{dw}H_{i-1}(w)\varphi(w) = \varphi(w)\left(\sqrt{i-1}H_{i-2}(w) - wH_{i-1}(w)\right)$ and apply integration

by parts to the last integral to get

$$\begin{aligned}
&\int \frac{1}{w} H_{i-1}(w)\varphi(w)dw \\
&= V_i(w) - \int \left(\sqrt{i-1}\log(|w|)H_{i-2}(w) - \log(|w|)\left(wH_{i-1}(w)\right)\right)\varphi(w)dw \\
&= V_i(w) - \sqrt{i-1}A_{i-2} + \int \log(|w|)\left(H'_{i-1}(w) + \sqrt{i}H_i(w)\right)\varphi(w)dw \\
&= V_i(w) - \sqrt{i-1}A_{i-2} + \sqrt{i-1}A_{i-2} + \sqrt{i}A_i \\
&= V_i(w) + \sqrt{i}A_i.
\end{aligned}$$

Plugging this equality back in and then evaluating the integrals on $\mathbb{R}$ yields

$$\begin{aligned}
\left[V_{i+1}(w) + \sqrt{\frac{i}{i+1}}V_i(w)\right]\Bigg|_{\mathbb{R}} + \sqrt{i+2}A_{i+2}\Bigg|_{\mathbb{R}} + \frac{i}{\sqrt{i+1}}A_i\Bigg|_{\mathbb{R}} &= 0, \\
\sqrt{i+2}c_{i+2} &= -\frac{i}{\sqrt{i+1}}c_i, \\
\frac{-i}{\sqrt{(i+1)(i+2)}}c_i &= c_{i+2} \qquad \text{(A.2)}
\end{aligned}$$

since $\lim_{w\to\pm\infty} V_i(w) = 0$ for any $i$, as $\varphi(w)$ decays much faster than $\log(|w|)\cdot\text{poly}(w)$ grows. Note that this recurrence is only valid for $i > 0$, since we used that in the analysis. Now, recall that by definition, $c_0 = \tau$ since $H_0(w) = 1$. Furthermore, since $H_1(w) = w$, $c_1 = \mathbb{E}_w[w\log(|w|)] = 0$ since $w\log(|w|)$ is an odd function and the Gaussian distribution is symmetric. Then, we can calculate $H_2(w) = \frac{1}{\sqrt{2}}(x^2-1)$ and thus that

$$\begin{aligned}
c_2 &= \frac{1}{\sqrt{2}}\left(\mathbb{E}_w[w^2\log(|w|)] - \mathbb{E}_w[\log(|w|)]\right) \\
&= \frac{1}{\sqrt{2}}(1+\tau-\tau) = \frac{1}{\sqrt{2}}
\end{aligned}$$

using Lemma A.2.7. The rest of the coefficients are defined recursively by Eq. (A.2). In particular, we can find a closed form. First, note that since $c_1 = 0$, $c_{2n-1} = 0$ for all strictly positive integers

$n$. Iterating Eq. (A.2) gives

$$c_{2n} = \frac{(-1)^{n-1}2^{n-1}(n-1)!}{\sqrt{(2n)!}}.$$

Now, we can apply the well-known Stirling's approximation $\left(n! \asymp \sqrt{2\pi n}\left(n/e\right)^n\right)$ to get the asymptotic behavior of this quantity. We have

$$\begin{aligned} c_{2n} &\asymp (-1)^{n-1}2^{n-1}\sqrt{\frac{\sqrt{\pi}(n-1)}{\sqrt{n}}}\frac{e}{n-1}\left(\frac{n-1}{n}\right)^n 2^{-n} \\ &= (-1)^{n-1}\frac{\pi^{1/4}}{2}\left((n-1)\sqrt{n}\right)^{-1/2} e\left(1-\frac{1}{n}\right)^n \\ &\asymp (-1)^{n-1}\frac{\pi^{1/4}}{2}n^{-3/4} \end{aligned}$$

after noting that $\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n = e^{-1}$. Therefore, the behavior of $c_{2n}^2$ is given by

$$c_{2n}^2 \asymp \frac{\sqrt{\pi}}{4}\cdot\frac{1}{n^{3/2}}.$$

We note that this asymptotic behavior is quite tight, even up to constants, for sufficiently large $n$. We can also prove a fairly tight lower bound using [108], which gives the bound

$$\sqrt{2\pi n}n^n e^{-n}e^{1/(12n+1)} \le n! \le \sqrt{2\pi n}n^n e^{-n}e^{1/12n}$$

Plugging in the upper bound for $(n+1)!$ and the lower bound for $(2n)!$, we get that

$$\begin{aligned} 2^{n-1}\frac{(n-1)!}{\sqrt{(2n)!}} &\ge \frac{e\pi^{1/4}}{2}\sqrt{\frac{1}{n\sqrt{n}}}\left(\frac{n-1}{n}\right)^n e^{\frac{36n+11}{48n(12n-11)}} \\ &\ge \frac{e\pi^{1/4}}{8}\cdot 1\cdot n^{-3/4} \end{aligned}$$

for $n \geq 2$. Thus, for $n \geq 2$,

$$c_{2n}^2 \geq \frac{e^2\sqrt{\pi}}{64} n^{-3/2} > \frac{1}{5} n^{-3/2}.$$

□

**Lemma A.2.9** (Integrals of the Hermite Coefficients)**.** *Suppose* $|b| < 1$ *and* $a > 0$*. Then*

$$\int_a^\infty \ell^{-3/2} d\ell = \frac{2}{\sqrt{a}}$$

*and*

$$\int_a^\infty \ell^{-3/2} |b|^{2\ell} d\ell = \frac{2|b|^{2a}}{\sqrt{a}} + 2\sqrt{2\pi \log |b|^{-1}} \left( -1 + \mathrm{Erf}\left( \sqrt{a \log |b|^{-2}} \right) \right)$$

*where*

$$\mathrm{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

*We also have the following upper bound:*

$$\int_a^\infty \ell^{-3/2} |b|^{2\ell} d\ell \leq \frac{2|b|^{2a}}{\sqrt{a}} - 4\sqrt{2 \log |b|^{-1}} \frac{e^{2a \log |b|}}{\sqrt{\log |b|^{-2a}} + \sqrt{2 + \log |b|^{-2a}}}.$$

*Proof.* The first equality is by direct integration. Now we tackle the second equality. We apply

integration by parts to get

$$\begin{aligned}
\int \ell^{-3/2}|b|^{2\ell}d\ell &= |b|^{2\ell}(-2\ell^{-1/2}) + \int 2\ell^{-1/2}\cdot(2\log(|b|))|b|^{2\ell}d\ell \\
&= |b|^{2\ell}(-2\ell^{-1/2}) + 4\log(|b|)\int |b|^{2u^2}2du \\
&= |b|^{2\ell}(-2\ell^{-1/2}) + 8\log(|b|)\int e^{-u^2/\left(\frac{1}{2}\frac{1}{\log(|b|^{-1})}\right)}du \\
&= |b|^{2\ell}(-2\ell^{-1/2}) + \frac{-8\log(|b|^{-1})}{\sqrt{2\log(|b|^{-1})}}\frac{\sqrt{\pi}}{2}\frac{2}{\sqrt{\pi}}\int e^{-v^2}dv \\
&= -2\left(|b|^{2\ell}(\ell^{-1/2}) + \sqrt{2\pi\log(|b|^{-1})}\mathrm{Erf}\left(\sqrt{2\ell\log(|b|^{-1})}\right)\right).
\end{aligned}$$

Then since $\mathrm{Erf}(\infty) = 1$, we simply evaluate the integral and note that the Erf term (depending on $a$) is positive:

$$\int_a^\infty \ell^{-3/2}|b|^{2\ell}d\ell = -2\sqrt{2\pi\log(|b|^{-1})} + 2\left(|b|^{2a}a^{-1/2} + \sqrt{2\pi\log(|b|^{-1})}\mathrm{Erf}\left(\sqrt{2a\log(|b|^{-1})}\right)\right).$$

Section 7.1.13 of [94] gives

$$\begin{aligned}
\frac{1}{x+\sqrt{x^2+2}} < e^{x^2}\int_x^\infty e^{-t^2}dt \le \frac{1}{x+\sqrt{x^2+\frac{4}{\pi}}}, \\
1 - \frac{2}{\sqrt{\pi}}\frac{e^{-x^2}}{x+\sqrt{x^2+\frac{4}{\pi}}} \le \mathrm{Erf}(x) < 1 - \frac{2}{\sqrt{\pi}}\frac{e^{-x^2}}{x+\sqrt{x^2+2}}.
\end{aligned}$$

We can apply the upper bound on Erf to in order to get the final upper bound on the integral. □

## A.3 Supporting Lemmas and Proofs for Theorem 2.4.2

### A.3.1 Matrix Inequalities and Hadamard Powers of Matrices

In this section, we record some useful definitions and theorems about matrices and their Hadamard powers.

**Theorem A.3.1** (Gershgorin Circle Theorem)**.** *For matrix $A \in \mathbb{R}^{p\times p}$, every eigenvalue $\lambda(A)$ satisfies*

$$\lambda(A) \geq A_{ii} - \sum_{i\neq j} |A_{ij}|.$$

*In particular,*

$$\lambda_{\min}(A) \geq \min_i |A_{ii}| - (p-1) \max_{i\neq j} |A_{ij}|.$$

*Proof.* See [109]. □

**Definition A.3.2** (Hadamard Product and Power)**.** The *Hadamard product* of matrices $A, B$ is given by

$$[A \circ B]_{i,j} = A_{i,j} B_{i,j}.$$

The $m^{th}$ Hadamard power of $A$ is given by

$$A^{(m)} = \underbrace{A \circ A \circ \ldots \circ A}_{m \text{ times}}.$$

**Theorem A.3.3** (Schur Product Theorem (weak version))**.** *Suppose $A, B$ are both symmetric PSD square matrices. Then $A \circ B$ is also PSD.*

*Proof.* Write eigendecompositions of $A = \sum_i \mu_i a_i a_i^T$ and $B = \sum_i \nu_i b_i b_i^T$. Then

$$\begin{aligned} A \circ B &= \sum_{i,j} \mu_i \nu_j \left(a_i a_i^T\right) \circ \left(b_j b_j^T\right) \\ &= \sum_{i,j} \mu_i \nu_j \left(a_i \circ b_j\right) \left(a_i \circ b_j\right)^T \end{aligned} \tag{A.3}$$

Then we have that $\mu_i, \nu_i \geq 0$ and $\left(a_i \circ b_j\right) \left(a_i \circ b_j\right)^T$ is PSD. Thus $A \circ B$ is PSD. □

**Theorem A.3.4** (Eigenvalues of Hadamard Powers)**.** *Suppose $A, B \in \mathbb{R}^{p\times p}$ both PSD. Let $b$ denote $B$'s diagonal. Then*

$$\prod_{i=j}^{p} \lambda_i(A \circ B) \geq \prod_{i=j}^{p} \lambda_i(A) b_i \tag{A.4}$$

*for all $j \in [p]$, where $\lambda_i$ is the $i^{th}$ smallest eigenvalue.*

*Proof.* See Theorem 3 from [30]. □

### A.3.2 Proof of Theorem 2.4.2

*Lower Bounding the Population Minimal Eigenvalue*

As a warm-up, we first prove prove a lower bound on $\lambda_{\min}(\Sigma)$.

**Theorem A.3.5** (Minimum Eigenvalue of Population Correlation Matrix)**.** *The following lower bounds on $\lambda_{\min}(\Sigma)$ hold:*

1. $\lambda_{\min}(\Sigma) \geq \frac{\pi^2}{8}\lambda_{\min}(\Phi)$.
2. $\lambda_{\min}(\Sigma) \geq \sum_{\ell=1}^{\frac{1}{2}\left\lfloor \frac{\log(p-1)}{\log(\frac{1}{1-\epsilon})} \right\rfloor} \frac{\lambda_{\min}\left(\Phi^{(2\ell)}\right)}{5\ell^{3/2}} + \frac{2}{5}\sqrt{\frac{2\log((1-\epsilon)^{-1})}{\log\left(\frac{p-1}{1-\epsilon}\right) + \max\{2, \log((1-\epsilon)^{-1})\}}}$.

*Note that the first lower bound is positive whenever $\Phi$ is full-rank, and the second bound is always strictly positive, even if $\Phi$ is low-rank.*

*Remark* A.3.6 (Intuition for Theorem 2.4.2). In the case that $\Phi$ is not low-rank, we automatically have a constant multiplicative factor improvement on the minimum eigenvalue when applying the log transformation. However, the true magic happens in the second bound – even if $\lambda_{\min}(\Phi) = 0$, we can *still achieve a positive lower bound* on $\lambda_{\min}\left(\mathbb{E}_z[zz^T]\right)$. The intuitive reason this phenomenon occurs is because the Hadamard power destroys the potential co-linear structure in $\Phi$ – this is precisely how the nonlinearity of the logarithm comes into play.

*Proof.* For ease of notation, through out this proof, we define $|\rho_{\max}| = 1 - \epsilon$. We recall that $\Sigma = \mathbb{E}[zz^T]$ where $z = \log|x|, x \sim \mathcal{N}(0, \Phi)$. By Lemma 1.3.4, we have

$$\Sigma_{i,j} = \sum_{\ell=0}^{\infty} c_\ell^2 \Phi_{i,j}^\ell$$

where $c_\ell = \mathbb{E}_w[H_\ell(w)\log(|w|)]$. This means,

$$\Sigma = \sum_{\ell=0}^{\infty} c_\ell^2 \Phi^{(\ell)}.$$

Continuing with the proof, we have

$$\begin{aligned}
\lambda_{\min}\left(\mathbb{E}_z[zz^T]\right) &= \min_{\|v\|_2=1} \sum_{i,j=1}^{p} v_i v_j \sum_{\ell=0}^{\infty} c_\ell^2 \Phi_{i,j}^\ell \\
&= \min_{\|v\|_2=1} \sum_{\ell=0}^{\infty} c_\ell^2 \sum_{i,j=1}^{p} v_i v_j \Phi_{ij}^\ell \\
&\geq \sum_{\ell=0}^{\infty} c_\ell^2 \left( \min_{\|v\|_2=1} \sum_{i,j=1}^{p} v_i v_j \Phi_{i,j}^\ell \right) \\
&= \sum_{\ell=0}^{\infty} c_\ell^2 \lambda_{\min}\left(\Phi^{(\ell)}\right) \\
&= c_0^2 \overbrace{\lambda_{\min}\left(\mathbf{1}_{p\times p}\right)}^{0} + \sum_{\ell=1}^{\infty} c_\ell^2 \lambda_{\min}\left(\Phi^{(\ell)}\right) \\
&= \sum_{\ell=1}^{\infty} c_\ell^2 \lambda_{\min}\left(\Phi^{(\ell)}\right)
\end{aligned} \tag{A.5}$$

where $\Phi^{(\ell)}$ denotes the $\ell^{th}$ element-wise (Hadamard) power of $\Phi$ (see Definition A.3.2). Then, using Theorem A.3.4, we have that $\lambda_{\min}(A \circ B) \geq \lambda_{\min}(A) \cdot B_{p,p}$, where $\circ$ denotes Hadamard product. Therefore, since the diagonal entries are all 1 and $1^\ell = 1$, we have for all $\ell \geq 1$ that

$$\lambda_{\min}\left(\Phi^{(\ell)}\right) \geq \lambda_{\min}(\Phi) \cdot 1$$

and we immediately get the bound

$$\lambda_{\min}\left(\mathbb{E}_z[zz^T]\right) \geq \sum_{\ell=1}^{\infty} c_\ell^2 \lambda_{\min}(\Phi) = \lambda_{\min}(\Phi) \sum_{\ell=1}^{\infty} c_\ell^2 = \left(\alpha - \tau^2\right) \lambda_{\min}(\Phi) = \frac{\pi^2}{8} \lambda_{\min}(\Phi).$$

However, this bound can be greatly improved by judiciously applying the well-known Gershgorin circle theorem (Theorem A.3.1). In order to apply this bound, we need to ensure that the Gershgorin bound will be strictly positive. Therefore, we truncate the summation carefully. Define

$$\ell_{\text{threshold}} = 1 + \left\lceil \frac{\log(p-1)}{\log(1/|\rho_{\max}|)} \right\rceil.$$

Note that for $\ell \geq \ell_{\text{threshold}}$, we have

$$\begin{aligned}(p-1)|\rho_{\max}|^\ell &\leq (p-1)|\rho_{\max}|^{1+\left\lceil \frac{\log(p-1)}{\log(1/|\rho_{\max}|)} \right\rceil} \\ &\leq \frac{|\rho_{\max}|(p-1)}{p-1} < 1\end{aligned}$$

Applying Gershgorin to the truncated tail of the sum, we bound

$$\begin{aligned}\lambda_{\min}\left(\mathbb{E}_z[zz^T]\right) &\geq \sum_{\ell=1}^{\infty} c_\ell^2 \lambda_{\min}\left(\Phi^{(\ell)}\right) \\ &= \sum_{\ell=1}^{\ell_{\text{threshold}}-1} c_\ell^2 \lambda_{\min}\left(\Phi^{(\ell)}\right) + \sum_{\ell=\ell_{\text{threshold}}}^{\infty} c_\ell^2 \left(1 - (p-1)|\rho_{\max}|^\ell\right).\end{aligned}$$

We know from Theorem A.3.3 that taking the Hadamard power of a PSD matrix yields a PSD matrix, thus the first summation term is non-negative.

We can further control this bound by plugging in estimates for $c_\ell^2$ from Lemma A.2.8: Recall that we have $c_{2\ell}^2 \geq \frac{1}{5}\frac{1}{\ell^{3/2}}$ and $c_{2\ell-1}^2 = 0$. Then, supposing $\ell_{\text{threshold}}$ is even for simplicity, we can

re-write our bound as

$$\begin{aligned}
&\lambda_{\min}\left(\mathbb{E}_z[zz^T]\right) \\
&\geq \sum_{\ell=1}^{(\ell_{\text{threshold}}-2)/2} c_{2\ell}^2 \lambda_{\min}\left(\Phi^{(2\ell)}\right) + \frac{1}{5}\sum_{\ell=\ell_{\text{threshold}}/2}^{\infty} \ell^{-3/2}\left(1-(p-1)|\rho_{\max}|^{2\ell}\right) \\
&= \sum_{\ell=1}^{(\ell_{\text{threshold}}-2)/2} \frac{\lambda_{\min}\left(\Phi^{(2\ell)}\right)}{5\ell^{3/2}} + \frac{1}{5}\left(\sum_{\ell=\ell_{\text{threshold}}/2}^{\infty} \ell^{-3/2} - (p-1)\sum_{\ell=\ell_{\text{threshold}}/2}^{\infty} \ell^{-3/2}|\rho_{\max}|^{2\ell}\right).
\end{aligned}$$

We now focus on lower bounding the second term further, letting

$$L = \sum_{\ell=1}^{(\ell_{\text{threshold}}-2)/2} \frac{\lambda_{\min}\left(\Phi^{(2\ell)}\right)}{5\ell^{3/2}}.$$

Recall that for a non-negative function $f$, we can upper and lower bound its summation as follows:

$$\int_a^{\infty} f(\ell)d\ell \leq \sum_{\ell=a}^{\infty} f(\ell) \leq f(\ell) + \sum_{\ell=a+1}^{\infty} f(\ell) \leq f(\ell) + \int_a^{\infty} f(\ell)d\ell.$$

Then, applying Lemma A.2.9, and plugging in the integral bounds, we get

$$\begin{aligned}
&\lambda_{\min}\left(\mathbb{E}_z[zz^T]\right) \\
&\geq L + \frac{1}{5}\left(\frac{2-2(p-1)|\rho_{\max}|^{\ell_{\text{threshold}}}}{\sqrt{\ell_{\text{threshold}}/2}} + \frac{4(p-1)\sqrt{2\log(|\rho_{\max}|^{-1})}e^{-\ell_{\text{threshold}}\log(|\rho_{\max}|^{-1})}}{\sqrt{\ell_{\text{threshold}}\log(|\rho_{\max}|^{-1})}+\sqrt{2+\ell_{\text{threshold}}\log(|\rho_{\max}|^{-1})}}\right) \\
&= L + \frac{2}{5}\left(\frac{1}{\sqrt{\ell_{\text{threshold}}/2}}(1-|\rho_{\max}|) + \frac{2(p-1)|\rho_{\max}|^{\ell_{\text{threshold}}}\sqrt{2\log(|\rho_{\max}|^{-1})}}{\sqrt{\log(|\rho_{\max}|^{-\ell_{\text{threshold}}})}+\sqrt{2+\log(|\rho_{\max}|^{-\ell_{\text{threshold}}})}}\right) \\
&> L + \frac{2}{5}\left((1-|\rho_{\max}|)\sqrt{\frac{2}{1+\left\lceil\frac{\log(p-1)}{\log(1/|\rho_{\max}|)}\right\rceil}} + |\rho_{\max}|\sqrt{\frac{2\log(|\rho_{\max}|^{-1})}{\log\left(\frac{p-1}{|\rho_{\max}|}\right)+2}}\right) \\
&> L + \frac{2}{5}\left((1-|\rho_{\max}|)\sqrt{\frac{2\log(|\rho_{\max}|^{-1})}{\log\left(\frac{p-1}{|\rho_{\max}|}\right)+\log(|\rho_{\max}|^{-1})}} + |\rho_{\max}|\sqrt{\frac{2\log(|\rho_{\max}|^{-1})}{\log\left(\frac{p-1}{|\rho_{\max}|}\right)+2}}\right)
\end{aligned}$$

where we upper bounded $\lceil x \rceil \leq x + 1$. Then, we can simplify the expression to

$$\lambda_{\min}\left(\mathbb{E}_z[zz^T]\right) > L + \frac{2}{5}\sqrt{\frac{2\log(|\rho_{\max}|^{-1})}{\log\left(\frac{p-1}{|\rho_{\max}|}\right) + \max(2, \log(|\rho_{\max}|^{-1}))}}$$

where if $|\rho_{\max}| \geq e^{-2}$, we have that $\log(|\rho_{\max}|^{-1}) \leq 2$, which is the desired result. □

*Bounds that use Sparsity*

In this section, we demonstrate bounds on the minimum eigenvalue which are independent of dimension $p$: instead, the sparsity $k$ plays a role.

In order to fully take advantage of the sparsity assumption, we prove a restricted analogue to the Gershgorin circle theorem ([109]) we used previously.

**Lemma A.3.7** (Restricted Gershgorin Circle Theorem)**.** *Let $A \in \mathbb{R}^{p\times p}$ be a symmetric matrix. Let $\alpha \geq 1$ and $T \subset [p]$. Then,*

$$\tilde{\lambda}(\alpha, T, A^{1/2}) \geq \min_i A_{ii} - |T| \cdot (1+\alpha)^2 \cdot \max_{i\neq j} |A_{ij}|.$$

*Proof.* Given in Appendix A.5. □

We can use these results directly to replace dimension $p$ with sparsity $k$ in the statements in Theorem 2.4.2. The proof is by direct application of Lemma A.3.7.

**Corollary A.3.8.** *The following lower bound holds:*

$$\tilde{\lambda}\left(k, \Sigma^{1/2}\right) \geq \sum_{\ell=1}^{\frac{1}{2}\left\lfloor \frac{\log(16k))}{\log(\frac{1}{1-\epsilon})}\right\rfloor} \frac{\tilde{\lambda}\left(k, [\Phi^{(2\ell)}]^{1/2}\right)}{5\ell^{3/2}} + \frac{2}{5}\sqrt{\frac{2\log((1-\epsilon)^{-1})}{\log\left(\frac{16k}{1-\epsilon}\right) + \max\{2, \log((1-\epsilon)^{-1})\}}} \tag{A.6}$$

This improvement is quite notable in that it completely removes dependence on dimension $p$. Potentially, the bound could be a lot better as typically $k \ll p$ in high-dimensional settings.

This improvement is also valuable because it now shifts dependence on $\lambda_{\min}(\Phi)$ to dependence on $\tilde{\lambda}(k, \Phi^{1/2})$, which is potentially much larger and positive even in the case where $\lambda_{\min}(\Phi) = 0$.

## A.4 Proof of Lemma 2.4.5 and Theorem 2.3.2.

### A.4.1 Bounding the Empirical Restricted Eigenvalue

We denote the sample and population covariance matrices of the log-transformed covariates by $\Sigma$ and $\hat{\Sigma}$:

$$\Sigma := \mathbb{E}_z[zz^T]$$
$$\hat{\Sigma} := \frac{1}{n}\sum_{i=1}^{n} z^{(i)} z^{(i)T}.$$

Theorem 2.4.2 gives us a bound on $\tilde{\lambda}(k, \Sigma^{1/2})$. In this section we apply the results of [31] to convert this into a bound on $\tilde{\lambda}(k, \hat{\Sigma}^{1/2})$ by analyzing $|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})|$. The following lemma shows that it is sufficient to analyze $\|\Sigma - \hat{\Sigma}\|_\infty$.

**Lemma A.4.1.** *We have,*

$$|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})| \leq 16k\|\Sigma - \hat{\Sigma}\|_\infty.$$

*Where, $\|\cdot\|_\infty$ denotes the entry-wise $\infty-$norm.*

*Proof.* We note that,

$$|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})| \leq \max_{v:\|v\|_2=1, \|v_S^c\|_1 \leq 3\|v_S\|_1} v^T(\Sigma - \hat{\Sigma})v.$$

Furthermore for any $v$ which satisfies $\|v\|_2 = 1, \|v_{S^c}\|_1 \leq 3\|v_S\|_1$, we have,

$$\begin{aligned} v^T(\Sigma - \hat{\Sigma})v &\overset{(1)}{\leq} \|v\|_1 \|(\Sigma - \hat{\Sigma})v\|_\infty \\ &= \|v\|_1 \max_i |\langle \Sigma_{i,\cdot} - \hat{\Sigma}_{i,\cdot}, v\rangle| \\ &\overset{(2)}{\leq} \|v\|_1^2 \|\Sigma - \hat{\Sigma}\|_\infty. \end{aligned}$$

In the above display, the inequalities marked (1) and (2) both follow from Holder's Inequality. Furthermore,

$$\begin{aligned} \|v\|_1 &= \|v_S\|_1 + \|v_{S^c}\|_1 \\ &\overset{(3)}{\leq} (1+3)\|v_S\|_1 \\ &\leq 4\sqrt{k}\|v_S\|_2 \\ &\overset{(4)}{\leq} 4\sqrt{k}. \end{aligned}$$

In the above display, the inequality marked (3) follows from $\|v_{S^c}\|_1 \leq 3\|v_S\|_1$, the inequality marked (4) follows from $\|v_S\|_2 \leq \|v\|_2 \leq 1$. Consequently, we have,

$$|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})| \leq 16k\|\Sigma - \hat{\Sigma}\|_\infty.$$

□

To analyze $\|\Sigma - \hat{\Sigma}\|_\infty$ we appeal to the concentration results from [31]. To do so we need to verify two conditions on our covariates:

1. The log-transformed covariates $z_i$ are entry-wise (marginally) subexponential. This is done in Lemma A.4.2.

2. An upper bound on the quantity $\Gamma$ defined as:

$$\Gamma^2 := \max_{j,k\in[p]} \frac{1}{n}\sum_{i=1}^{n} \mathrm{var}\left(z_j^{(i)} z_k^{(i)}\right).$$

This is done in Lemma A.4.3.

**Lemma A.4.2.** *Let $w \sim \mathcal{N}(0,1)$. Then, $z = \log(|w|)$ is 1-subexponential.*

*Proof.* It is sufficient to show that, $\forall t > 0$,

$$\mathbb{P}\left[|\log(|w|)| > t\right] \leq 2\exp(-t).$$

To show this, we bound the upper tail and the lower tail separately. First let us consider the upper tail,

$$\begin{aligned}
\mathbb{P}[\log(|w|) > t] &= \mathbb{P}[|w| > e^t] \\
&= 2\mathbb{P}[w > e^t] \\
&\overset{(1)}{\leq} \sqrt{\frac{2}{\pi}}\frac{e^{-e^{2t}/2}}{e^t} \\
&\leq \sqrt{\frac{2}{\pi}}e^{-t}.
\end{aligned}$$

In the inequality marked (1) we used the standard estimate for Gaussian tails: $\mathbb{P}[w > \delta] \leq \sqrt{\frac{2}{\pi}}\frac{\exp(-\delta^2/2)}{\delta}$. To bound the lower tail, we use standard estimates on Gaussian anti-concentration,

$$\begin{aligned}
\mathbb{P}[\log(|w|) < -t] &= \mathbb{P}[|w| < e^{-t}] \\
&= \frac{1}{\sqrt{2\pi}}\int_{-e^{-t}}^{e^{-t}} \exp(-a^2/2)da \\
&\leq \sqrt{\frac{2}{\pi}}e^{-t}.
\end{aligned}$$

Combining the estimates of the lower and upper tail, we get,

$$\mathbb{P}\left[|\log(|w|)| > t\right] \leq 2\sqrt{\frac{2}{\pi}}\exp(-t) < 2\exp(-t)$$

as desired. □

**Lemma A.4.3.** *We have the following upper bound on* $\Gamma$*:*

$$\Gamma^2 \leq 48.$$

*Proof.* Since $z_i$ are identically distributed,

$$\Gamma^2 = \max_{j,k\in[p]} \operatorname{var}\left(z_j z_k\right).$$

We have,

$$\begin{aligned}
\operatorname{var}(z_j z_k) &\leq \mathbb{E}[(z_j z_k)^2] \\
&\overset{(1)}{\leq} \sqrt{\mathbb{E}[z_i^4]\mathbb{E}[z_j^4]} \\
&\overset{(2)}{=} \mathbb{E}[z_i^4].
\end{aligned}$$

In the above display we used the Cauchy-Schwarz Inequality to obtain the inequality marked (1) and the fact that $z_i$ and $z_j$ have the same marginal distribution in the equality marked (2). To bound

$\mathbb{E}[z_i^4]$ we use the concentration result proved in Lemma A.4.2.

$$\begin{aligned}\mathbb{E}[z_i^4] &= \int_0^\infty \mathbb{P}(z_i^4 > t)\, dt \\ &= \int_0^\infty \mathbb{P}(|z_i| > t^{1/4})\, dt \\ &\overset{(3)}{\leq} \int_0^\infty 2\exp(-t^{1/4})\, dt \\ &= 48.\end{aligned}$$

In the above display, we used Lemma A.4.2 for the inequality marked (3). □

We can now apply Theorem 4.1 of [31] to control $\|\Sigma - \hat{\Sigma}\|_\infty$ and hence control $|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})|$.

**Lemma A.4.4.** *Let $\delta \in (0,1)$ be an arbitrary confidence parameter. With probability $1-\delta$,*

$$|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})| \leq Ck\left(\sqrt{\frac{(\log(3/\delta) + 2\log(p))}{n}} + \frac{\log^2(n)(\log(3/\delta) + 2\log(p))^2}{n}\right).$$

*In the above display $C$ is a universal constant.*

*Proof.* From Lemma A.4.2, we know that the log-transformed covariates are marginally subexponential. Applying Theorem 4.1 of [31] for marginally subexponential random variables, we have with probability atleast $1 - 3e^{-t}$,

$$\|\Sigma - \hat{\Sigma}\|_\infty \leq C\left(\sqrt{\frac{\Gamma(t + 2\log(p))}{n}} + \frac{\log^2(n)(t + 2\log(p))^2}{n}\right),$$

where $C$ is a universal constant. Substituting the bound on $\Gamma$ from Lemma A.4.3 and then applying Lemma A.4.1 we get,

$$|\tilde{\lambda}(k, \Sigma^{1/2}) - \tilde{\lambda}(k, \hat{\Sigma}^{1/2})| \leq Ck\left(\sqrt{\frac{(t + 2\log(p))}{n}} + \frac{\log^2(n)(t + 2\log(p))^2}{n}\right).$$

Substituting $t = \log(3/\delta)$ gives us the required bound. □

We are now ready to present the proof of Theorem 2.3.2 which is restated and proved below.

**Theorem A.4.5.** *Let $\delta \in (0,1)$ be an arbitrary confidence parameter. Suppose the covariance matrix $\Phi$ satisfies $\Phi_{i,i} = 1$, $\forall i \in [p]$ and $\max_{i \neq j} |\Phi_{i,j}| < 1 - \epsilon$. Then, we have that log-transformed design matrix satisfies the restricted eigenvalue bound:*

$$\tilde{\lambda}(k, \hat{\Sigma}^{1/2}) \geq \frac{1}{5}\sqrt{\frac{\epsilon}{\log(16k) + 2}},$$

*with probability $1 - \delta$, provided,*

$$n \geq \frac{Ck^2}{\epsilon} \log^2\left(\frac{2pk}{\delta}\right) \log^2\left(\frac{k}{\epsilon} \log\left(\frac{2pk}{\delta}\right)\right).$$

*In the above display, $C$ is a universal constant.*

*Proof.* For the ease of notation, we define $|\rho_{\max}| = 1 - \epsilon$. From Theorem A.3.2, we know that,

$$\begin{aligned}
\tilde{\lambda}(k, \Sigma^{1/2}) &\geq \frac{2}{5}\left(\frac{2\log\left(\frac{1}{1-\epsilon}\right)}{\log\left(\frac{16k}{1-\epsilon}\right) + \max\{2, \log\left(\frac{1}{1-\epsilon}\right)\}}\right)^{1/2} \\
&\overset{(1)}{\geq} \frac{2}{5}\sqrt{\frac{2\log\left(\frac{1}{1-\epsilon}\right)}{\log(16k) + 2 + 2\log\left(\frac{1}{1-\epsilon}\right)}} \\
&\overset{(2)}{\geq} \frac{\sqrt{2}}{5} \min\left(1, \sqrt{\frac{\log\left(\frac{1}{1-\epsilon}\right)}{2 + \log(16k)}}\right)
\end{aligned}$$

In the display marked above, we used the fact that $\max(a, b) \leq a + b$ in the inequality marked (1). In the inequality marked (2) we used the fact for any $x, c \geq 0$, we have $\frac{x}{x+c} \geq \frac{1}{2}\min\left(\frac{x}{c}, 1\right)$. By

Lemma A.4.4, we know that with probability $1 - \delta$,

$$\tilde{\lambda}(k, \hat{\Sigma}^{1/2}) \geq \frac{\sqrt{2}}{5} \min\left(1, \sqrt{\frac{\log\left(\frac{1}{1-\epsilon}\right)}{2 + \log(16k)}}\right) - Ck\left(\sqrt{\frac{(\log(3/\delta) + 2\log(p))}{n}} + \frac{\log^2(n)(\log(3/\delta) + 2\log(p))^2}{n}\right).$$

Hence there exists a constant $C$ such that if,

$$\frac{n}{\log^2(n)} \geq Ck^2 \left(\log(3/\delta) + 2\log(p)\right)^2 \max\left(1, \frac{\log(16k) + 2}{\log\left(\frac{1}{1-\epsilon}\right)}\right),$$

we have that $\hat{\Sigma}$ satisfies the restricted eigenvalue bound:

$$\tilde{\lambda}(k, \hat{\Sigma}^{1/2}) \geq \frac{1}{5} \min\left(1, \sqrt{\frac{\log\left(\frac{1}{1-\epsilon}\right)}{2 + \log(16k)}}\right)$$

Finally, we clean up this bound. First we note that $\log\left(\frac{1}{1-\epsilon}\right) \geq \epsilon$. Hence, if $n$ is large enough so that,

$$\frac{n}{\log^2(n)} \geq \frac{Ck^2 \log(2k)}{\epsilon} \log^2\left(\frac{2p}{\delta}\right),$$

we have, with probability $1 - \delta$,

$$\tilde{\lambda}(k, \hat{\Sigma}^{1/2}) \geq \frac{1}{5}\sqrt{\frac{\epsilon}{2 + \log(16k)}}$$

Finally, we note to satisfy the requirement on the sample size, it is sufficient that,

$$n \geq \frac{Ck^2 \log(2k)}{\epsilon} \log^2\left(\frac{2p}{\delta}\right) \log^2\left(\frac{k \log(2k)}{\epsilon} \log\left(\frac{2p}{\delta}\right)\right).$$

□

## A.5 Proof of the Gershgorin's Circle Theorem for Restricted Eigenvalue

In this section, we prove the Gershgorin's theorem for the restricted eigenvalue. Let $A$ be a $p \times p$ symmetric matrix and $S$ be an arbitrary subset of $[p]$. Let $\tilde{\lambda}(\alpha, S, A^{1/2})$ denote the Restricted eigenvalue defined as:

$$\tilde{\lambda}(\alpha, S, A^{1/2}) = \min v^T A v \text{ subject to: } \|v\|_2 = 1, \|v_{S^c}\|_1 \leq \alpha \|v_S\|_1.$$

The goal is to prove the following theorem.

**Theorem A.5.1.** *For $\alpha \geq 1$, we have,*

$$\tilde{\lambda}(\alpha, S, A^{1/2}) \geq \min_{i \in [p]} A_{i,i} - (1+\alpha)^2 \cdot |S| \cdot \max_{i \neq j} |A_{ij}|.$$

Let $v^\star$ be the optimizer of the Restricted Eigenvalue problem. To simplify notation, we will short hand the optimal objective $\tilde{\lambda}(\alpha, S, A^{1/2})$ as $\lambda^\star$. Without loss of generality we can assume $|v_i^\star| > 0 \; \forall i \in [p]$. This is because of the following reason: Let $T$ denote the support of the optimal $v^\star$. It is straightforward to see that $\lambda^\star$ and $v^\star(T)$ are the optimal objective value and the optimizer of the following problem:

$$\min v^T A(T,T) v \text{ subject to: } \|v\|_2 = 1, \|v_{T \cap S^c}\|_1 \leq \alpha \|v_{T \cap S}\|_1.$$

If $T \neq [p]$, then we can make the arguments that follow for the optimization problem defined in the display above.

The proof of the usual Gershgorin Theorem begins with the optimality condition for the unconstrained eigenvalue problem. Taking cue from the original proof, we first derive an optimality condition for the restricted eigenvalue problem. We then utilize this to prove a lower bound on $\lambda^\star$.

*Proof.* We first write the local optimality condition at $v^\star$. For $\lambda \in \mathbb{R}$ and $q \geq 0$, we form the Lagrangian:

$$L(v, \lambda, q) = v^T A v - \lambda \|v\|_2^2 + 2q \left( \|v_{S^c}\|_1 - \alpha \|v_S\|_1 \right).$$

Since $|v_i^\star| > 0 \; \forall i$, by the method of Lagrange multipliers, the local optimality condition at $v^\star$ is:

$$\exists \lambda \in \mathbb{R}, q \geq 0 \text{ such that } \nabla_v L(v^\star, \lambda, q) = 0.$$

This means,

$$Av^\star - \lambda v^\star - qu = 0. \tag{A.7}$$

Where, the vector $u \in \mathbb{R}^p$ is defined as:

$$u_i = \begin{cases} \alpha \text{sign}(v_i^\star) & i \in S \\ -\text{sign}(v_i^\star) & i \notin S. \end{cases}$$

Taking dot-product with $v^\star$ on both sides of equation A.7, we get,

$$\begin{aligned} \lambda^\star &= \lambda + q(\alpha \|v_S\|_1 - \|v_{S^c}\|_1) \\ &\geq \lambda. \end{aligned}$$

Hence to lower bound $\lambda^\star$, it is sufficient to lower bound $\lambda$. Let $i$ be the coordinate that maximizes $|v_i^\star|$. Then, we have,

$$\sum_{j \neq i} A_{ij} v_j^\star + A_{ii} v_i^\star - \lambda v_i^\star = q u_i. \tag{A.8}$$

However, since $q$ is unknown, to eliminate it we consider another coordinate $k$. This coordinate $k$

is chosen so that: If $i \in S$, $k \notin S$ and if $i \notin S$, then $k \in S$. We have,

$$\sum_{j \neq k} A_{kj} v_j^\star + A_{kk} v_k^\star - \lambda v_k^\star = q u_k. \tag{A.9}$$

Hence, we can eliminate $q$ between equations A.8 and A.9,

$$\frac{v_i^\star}{u_i} A_{ii} - \frac{v_k^\star}{u_k} A_{kk} - \lambda \left( \frac{v_i^\star}{u_i} - \frac{v_k^\star}{u_k} \right) = \frac{1}{u_k} \left( \sum_{j \neq k} A_{kj} v_j^\star \right) - \frac{1}{u_i} \left( \sum_{j \neq i} A_{ij} v_j^\star \right).$$

Taking absolute values,

$$\left| \frac{v_i^\star}{u_i} A_{ii} - \frac{v_k^\star}{u_k} A_{kk} - \lambda \left( \frac{v_i^\star}{u_i} - \frac{v_k^\star}{u_k} \right) \right| = \left| \frac{1}{u_k} \left( \sum_{j \neq k} A_{kj} v_j^\star \right) - \frac{1}{u_i} \left( \sum_{j \neq i} A_{ij} v_j^\star \right) \right|.$$

Dividing throughout by $|\frac{v_i^\star}{u_i}|$:

$$\begin{aligned} \left| A_{ii} - \frac{v_k^\star u_i}{u_k v_i^\star} A_{kk} - \lambda \left( 1 - \frac{v_k^\star u_i}{v_i^\star u_k} \right) \right| &= \left| \frac{u_i}{u_k} \left( \sum_{j \neq k} A_{kj} \frac{v_j^\star}{v_i^\star} \right) - \left( \sum_{j \neq i} A_{ij} \frac{v_j^\star}{v_i^\star} \right) \right| \\ &\leq |S| \cdot (1 + \alpha) \cdot \left( \max_{l \neq m} |A_{lm}| \right) \cdot \left( \frac{|u_i|}{|u_k|} + 1 \right) \\ &\leq |S| \cdot (1 + \alpha)^2 \cdot \left( \max_{l \neq m} |A_{lm}| \right). \end{aligned}$$

Next we note because of the choice of $k$ (if $i \in S$, $k \notin S$, if $i \notin S$, $k \in S$) and the definition of $u$,

$$\rho := -\frac{v_k^\star u_i}{u_k v_i^\star} \geq 0.$$

Dividing through out by $1 + \rho$ gives,

$$\left| \frac{A_{ii} + \rho A_{kk}}{1 + \rho} - \lambda \right| \leq \frac{|S| \cdot (1 + \alpha)^2 \cdot \left( \max_{l \neq m} |A_{lm}| \right)}{1 + \rho} \leq |S| \cdot (1 + \alpha)^2 \cdot \left( \max_{l \neq m} |A_{lm}| \right).$$

Next noting that,

$$\frac{A_{ii} + \rho A_{kk}}{1 + \rho} \geq \min_i A_{ii},$$

we have the following lower bound,

$$\lambda^\star \geq \lambda \geq (\min_i A_{ii}) - |S| \cdot (1 + \alpha)^2 \cdot \left( \max_{l \neq m} |A_{lm}| \right).$$

□

# Appendix B: Deferred Proofs from Chapter 3

## B.1 Problem Statement Remarks

We also have the following two remarks regarding our framing of the problem.

*Remark* B.1.1 (Recovering Low-Rank Factors). Note that we have framed the learning problem in terms of recovering a low-rank matrix $Y^*$. To fulfill the promise of improved computational efficiency, the original motivation for considering this learning problem, we would apply a final post-processing step after learning $Y^*$ by exactly decomposing it into the product of two rank $r$ matrices using SVD.

*Remark* B.1.2 (Convolutional Architecture). In this problem we only consider fully-connected architecture. However, since a convolution is a linear operator, we can make use of results which efficiently learn one-layer convolutional ReLU networks [110] to first recover the convolutional filters before reducing to our low-rank recovery procedure.

## B.2 Hermite Decomposition of ReLU

[111] derives properties of the Hermite expansion for the univariate ReLU:

**Lemma B.2.1** (Hermite Expansion Properties for Univariate ReLU [111])**.** *Let* $\{c_i\}_{i=0}^{\infty}$ *be the Hermite coefficients for ReLU. Then,*

$$c_k = \begin{cases} 1/\sqrt{2\pi} & \text{if } k = 0, \\ \frac{1}{2} & \text{if } k = 1, \\ \frac{1}{\sqrt{2\pi k!}} \left(H_k(0) + kH_{k-2}(0)\right) & \text{if } k \geq 2. \end{cases}$$

Using the above properties, we can derive the explicit form of the Hermite coefficients for the ReLU activation:

**Lemma B.2.2** (Hermite Coefficients for ReLU)**.** *Let $\{c_i\}_{i=0}^{\infty}$ be the Hermite coefficients for ReLU. Then,*

$$c_k = \begin{cases} 1/\sqrt{2\pi} & \text{if } k = 0, \\ \frac{1}{2} & \text{if } k = 1, \\ 0 & \text{if } k = 2m+1, m \geq 1, \\ \sqrt{\frac{1}{2\pi}\frac{1}{4^m} \cdot \binom{2m}{m} \cdot \frac{1}{(2m-1)^2}} & \text{if } k = 2m, m \geq 1. \end{cases}$$

*Proof.* First we show that $c_k = 0$ for odd $k > 1$. This is easy to check since if $k$ is odd, so is $k-2$ and checking that there is no constant term for odd Hermite polynomials $H_k$ yields that the whole expression is 0.

For the rest of the even terms, plug in the standard formula

$$H_{2m}(0) = (-1)^m \frac{(2m)!}{m! \cdot 2^m}$$

to the recurrence given in the expansion properties and simplify using $\binom{2m}{m} = \frac{(2m)!}{m!m!}$. □

**Lemma B.2.3** (Hermite Analysis of ReLU Correlation)**.** *Suppose we have univariate standard Gaussians $g_1, g_2$ which are $\rho$-correlated ($\rho \in [0,1]$). Let $\sigma(x) = \max(0,x)$ be the ReLU activation. Then,*

$$\begin{aligned} &2\mathbb{E}_{g_1,g_2}[\sigma(g_1)\sigma(g_2)] \\ &= \frac{1}{\pi}\left(1 + \frac{\pi}{2}\rho + \sum_{\ell=1}^{\infty} \frac{1}{4^\ell} \cdot \binom{2\ell}{\ell} \cdot \frac{1}{(2\ell-1)^2}\rho^{2\ell}\right) \\ &=: \sqrt{h}(\rho). \end{aligned} \tag{B.1}$$

*Proof.* Apply Lemma 1.3.4 and Lemma B.2.2 and simplify the algebra. □

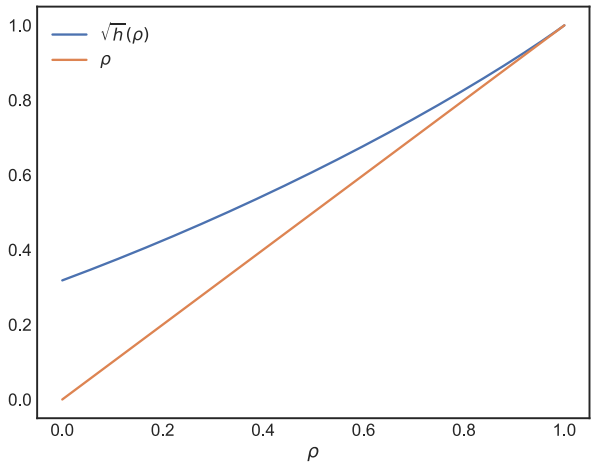



Figure B.1: $\sqrt{h(\rho)}$ (see Definition 3.3.7) plotted against the linear function $\rho$. Here, $\rho$ is a correlation between the inputs to the arc-cosine kernel.

**Lemma B.2.4** (Convexity and Monotonicity of $\sqrt{h}$ and $h$)**.** *The function $\sqrt{h}$ defined by Definition B.1 is both convex and monotone non-decreasing on $[-1, 1]$, and is also bounded between $[0, 1]$ for inputs in $[-1, 1]$.*

*Proof.* First we compute the derivative:

$$\frac{d\sqrt{h}(\rho)}{d\rho} = \frac{1}{2} + \frac{1}{\pi}\sum_{\ell=1}^{\infty} \frac{1}{4^\ell}\binom{2\ell}{\ell}\frac{2\ell}{(2\ell-1)^2}\rho^{2\ell-1}.$$

Now we observe the derivative is non-negative for $\rho \in [-1, 1]$. To prove this fact, first note that since the coefficients of the derivative are all positive and all powers of $\rho$ are odd, the derivative is minimized at $\rho = -1$. Thus, evaluating

$$\frac{1}{2} - \frac{1}{\pi}\sum_{\ell=1}^{\infty} \frac{1}{4^\ell}\binom{2\ell}{\ell}\frac{2\ell}{(2\ell-1)^2} = 0,$$

we see that the derivative is lower bounded by 0 and thus $\sqrt{h}(\rho)$ is monotone non-decreasing on $[-1, 1]$. Since $\sqrt{h}(\rho)$ is a positive combination of convex functions (linear functions and even powers are convex), it is also convex. To prove it is bounded, using monotonicity, we only need consider the extremes at $\rho = 0, 1$. We have $h(0) = \frac{1}{\pi}$, and we have (using the closed form given by

Definition 3.3.7)

$$h(1) = \frac{\left(\sqrt{1-1^2} + (\pi - 0) \cdot 1\right)}{\pi} = 1$$

which proves the statement. □

*Remark* B.2.5 (Convexity and Monotonicity of C-Maps). It is worth noting that the analysis in [112] proves that a wide variety of related functions to $\sqrt{h}$ are convex and monotone.

### B.3 Proofs for Section 3.3.4

We recall the following useful definition:

**Definition B.3.1** (ReLU Kernel: $1^{st}$-Order Arc-Cosine Kernel)**.** We define a function known in the literature as the *first-order arc-cosine kernel* [88], and is defined by

$$k(x, y) := \|x\|_2 \|y\|_2 \cdot \sqrt{h}(\rho_{xy})$$

where

$$\rho_{xy} := \frac{x^\top y}{\|x\|_2 \|y\|_2}$$

and

$$\sqrt{h}(\rho_{xy}) = \frac{\left(\sqrt{1-\rho_{xy}^2} + \left(\pi - \cos^{-1}\left(\rho_{xy}\right)\right)\rho_{xy}\right)}{\pi}.$$

This function will be very relevant to our analysis of ReLU SVD.

We provide an accompanying Hermite expansion of $\sqrt{h}$ in Lemma B.2.3.

*Characterizing the ReLU SVD*

**Lemma B.3.2** (ReLU Decomposition)**.** *Let ReLU be defined by* $\sigma(x) = \max(0, x)$*. Then,*

$$\sigma(x) = \frac{x + |x|}{2}. \tag{B.2}$$

*Proof.* Observe that if $x \leq 0$, $\sigma(x) = 0$. Otherwise, it equals $x$. □

**Lemma B.3.3** (ReLU Norm)**.**

$$\mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left\|\sigma(x^\top Y)\right\|_2^2\right] = \frac{1}{2}\|Y\|_F^2 . \tag{B.3}$$

*Proof.* We have

$$\begin{aligned}
\left\|\sigma(Y^\top x)\right\|_2^2 &= \sum_{i=1}^m \sigma(Y_i^\top x)^2 \\
&= \frac{1}{4}\sum_{i=1}^m \left(Y_i^\top x + |Y_i^\top x|\right)^2 \\
&= \frac{1}{4}\sum_{i=1}^m 2\left(Y_i^\top x\right)^2 + 2\left(Y_i^\top x\right)|Y_i^\top x| \\
&= \frac{1}{2}\left\|Y^\top x\right\|_2^2 + \frac{1}{2}\sum_{i=1}^m \left(Y_i^\top x\right)^2 \operatorname{sgn}(Y_i^\top x).
\end{aligned} \tag{B.4}$$

Now we take expectation over $x \sim \mathcal{N}(0, I)$.

$$\begin{aligned}
\mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left\|Y^\top x\right\|_2^2\right] &= \mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\operatorname{Tr}\left(x^\top Y Y^\top x\right)\right] \\
&= \operatorname{Tr}\left(Y^\top \mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[xx^\top\right] Y\right) \\
&= \operatorname{Tr}\left(Y^\top Y\right) \\
&= \|Y\|_F^2 .
\end{aligned} \tag{B.5}$$

For the second term, apply linearity of expectation and condition on the events that $Y_i^\top x > 0$ and $Y_i^\top x < 0$. Since $Y_i$ is fixed and $x$ is an isotropic Gaussian (which is spherically symmetric), the

probability that $Y_i^\top x > 0$ is equal to the probability that $Y_i^\top x < 0$. Thus, letting $q_i = \mathbb{P}\left(Y_i^\top x > 0\right)$,

$$\begin{aligned} & q_i \cdot \sum_{i=1}^{m} \mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left(Y_i^\top x\right)^2 \operatorname{sgn}(Y_i^\top x) | Y_i^\top x > 0\right] \\ +(1-q_i) \cdot & \sum_{i=1}^{m} \mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\left(Y_i^\top x\right)^2 \operatorname{sgn}(Y_i^\top x) | Y_i^\top x < 0\right] \\ & = 0 \end{aligned} \tag{B.6}$$

and we get the desired result. □

**Lemma B.3.4** (Reduction to 1-Dimensional Correlation)**.** *Consider $W, Y \in \mathbb{R}^{d\times m}$, and ReLU nonlinearity $\sigma$, which is applied elementwise. Then,*

$$\mathbb{E}_{x\sim\mathcal{N}(0,I)}\left[\langle\sigma(x^\top Y), \sigma(x^\top W)\rangle\right] \tag{B.7}$$

$$= \sum_{i=1}^{m} \|W_i\|_2 \|Y_i\|_2 \cdot \mathbb{E}_{g_i^1, g_i^2}[\sigma(g_i^1)\sigma(g_i^2)] \tag{B.8}$$

*where $g_i^1, g_i^2$ are univariate standard Gaussians with correlation $\mathbb{E}_{g_i^1,g_i^2}[{g_i^1}^\top g_i^2] = \rho_i$ and $\rho_i = \frac{W_i^\top Y_i}{\|W_i\|_2\|Y_i\|_2} \in [0,1]$. $W_i, Y_i$ are the columns of $W, Y$ respectively.*

*Proof.* First apply linearity of expectation to get a sum over $m$ correlations corresponding to column vectors $Y_i, W_i$. Then, using the positive homogeneous property of ReLU ($\sigma(c\cdot x) = c\cdot\sigma(x)$ for $c \geq 0$), we can normalize $Y_i, W_i$ and pull out their $\ell_2$ norms. Then using joint Gaussinity, we can replace $Y_i^\top x$ and $W_i^\top x$ with $g_i^1, g_i^2$. Finally, observe

$$\begin{aligned} \mathbb{E}_{x\sim\mathcal{N}(0,I)}[\operatorname{Tr}\left(Y_i^\top x x^\top W_i\right)] \cdot \frac{1}{\|Y_i\|_2\|W_i\|_2} &= \frac{\operatorname{Tr}\left(Y_i^\top I W_i\right)}{\|Y_i\|_2\|W_i\|_2} \\ &= \frac{Y_i^\top W_i}{\|Y_i\|_2\|W_i\|_2}. \end{aligned} \tag{B.9}$$

□

*Lower Bounding the Gap*

First we begin with a characterization of the optimal vector $\rho$ from Theorem 3.3.8.

**Lemma B.3.5** (Re-writing $\rho$)**.** *Consider the optimal choice of $\rho \in \mathbb{R}^m$ from Theorem 3.3.8. Then, for all $i \in [m]$, we can write*

$$\rho_i^* = \left\| Z^* \hat{W}_i \right\|_2,$$

*for $Z^* \in \mathbb{R}^{r \times d}$ a matrix with orthogonal rows (of potentially non-unit norm) and where we define $\hat{W}_i = W_i / \|W_i\|_2$.*

*Proof.* First recall we have

$$\rho_i = \frac{Y_i^\top W_i}{\|Y_i\|_2 \|W_i\|_2}$$

from Theorem 3.3.8. We utilize a special property of the function $h$: $h$ is convex and monotone-increasing for all $\rho \in [-1, 1]$ (Lemma B.2.4). This property ensures that any choice of $\rho_i < 0$ is always dominated by a choice of $\rho_i = 0$.

Therefore, we always want to maximize the value of $\rho_i$ subject to the low-rank constraint whenever it is the case that the parameter choices made for $\rho_i$ do not affect the other $\rho_i$ (since overall we want to maximize $\sum_{i=1}^m \|W_i\|_2^2 \cdot h(\rho_i)$) – in other words, we first optimize over the separable parameters. Now applying SVD, write $Y_i = D \Lambda B_i$ for column $i \in [m]$ of $Y$. Here we have $D \in \mathbb{R}^{d \times r}$, $\Lambda \in \mathbb{R}^{r \times r}$, and $B_i \in \mathbb{R}^r$. Thus

$$Y_i^\top W_i = B_i^\top Z W_i,$$

where $Z = \Lambda D^\top \in \mathbb{R}^{r \times d}$ ensures that $Y$ is rank $r$. We know that we must maximize $\rho_i$ for each $i \in [m]$. We first fix an arbitrary choice of $\Lambda^*, D^*$ and thus $Z^*$ and then optimize over $B_i$ for each $i \in [m]$. By Cauchy-Schwarz, $B_i^* = Z^* W_i$. Plugging this back in yields

$$\rho_i^* = \frac{\|Z^* W_i\|_2^2}{\|Z^* W_i\|_2 \|W_i\|_2} = \frac{\|Z^* W_i\|_2}{\|W_i\|_2} = \left\| Z^* \hat{W}_i \right\|_2,$$

where we define $\hat{W}_i = W_i / \|W_i\|_2$ and note that $Z^* \in \mathbb{R}^{r \times d}$ has the property that $Z^* Z^{*\top} = \Lambda^{*2} \in \mathbb{R}^{r \times r}$ is a diagonal matrix.

□

**Theorem B.3.6** (Lower Bound on Suboptimality of SVD)**.** *Recall the objective $\mathcal{R}(Y)$ from Problem 3.3.6, where we require that $Y \in \mathbb{R}^{d \times m}$ is a rank-r matrix. Let $W = U\Sigma V^\top \in \mathbb{R}^{d \times m}$ be the SVD of $W$. Define $\rho^*_\sigma \in \mathbb{R}^m$ as the correlations $\|\Lambda^* D^{*\top} \hat{W}_i\|_2$ for column $i$ of $W$, where $\Lambda^* \in \mathbb{R}^{r \times r}$ and $D^* \in \mathbb{R}^{d \times r}$ are as in Lemma B.3.5 and $\hat{W}_i = W_i / \|W_i\|_2$. As shorthand, denote $Y(\rho)$ as the associated low-rank matrix for correlation vector $\rho \in \mathbb{R}^m$ (computed as described in Theorem 3.3.8). Denote $\rho^*_{\text{SVD}}$ to be the optimal correlations in the case where we pick $\Lambda^*$ to correspond to the top r singular values, and $D^* = U$ as in SVD. Then, we have the following lower bound for the suboptimality of the SVD solution $Y_{\text{SVD}}$:*

$$\left(\mathcal{R}(Y_{\text{SVD}})) - \mathcal{R}(Y(\rho^*_\sigma))\right) \geq \frac{1}{2} \|w \odot \sqrt{h(\rho^*_{\text{SVD}})} - \rho^*_{\text{SVD}}\|_2^2 \tag{B.10}$$

*where $h$ is defined in Definition B.1, where $w = \left[\|W_1\|_1, \quad \ldots, \|W_m\|_2\right]$ is a vector of column norms of $W$, and where $\odot$ is the element-wise product.*

*Proof.* First consider that the value of the ReLU SVD objective as a function of arbitrary $\rho$ and $\beta_i = \|Y_i\|_2$ for column $i$ of $Y$ is

$$\frac{\|W\|_F^2}{2} - \sum_{i=1}^{m} \left[\|W_i\|_2 \sqrt{h}(\rho_i)\beta_i - \frac{1}{2}\beta_i^2\right]$$

as computed in Theorem 3.3.8. We know that choosing $\Lambda^*$ to correspond to the top $r$ singular values and $D^* = U$ is not necessarily optimal, and so if we bound the difference between the SVD solution and the choice of $\rho^*_{\text{SVD}}$ with correct scaling, we are also lower bounding the difference between the SVD solution and the optimal choice of $\rho^*_\sigma$ (and correct scaling). If we choose $\Lambda^*$ to correspond to the top $r$ singular values and $D^* = U$ (sub-optimally) and then set $\beta_i = \|W_i\|_2 \cdot$

$\sqrt{h}(\rho^*_{\text{SVD}}(i))$ optimally, we get that the value of the objective is

$$\frac{\|W\|_F^2}{2} - \frac{1}{2}\sum_{i=1}^{m} \|W_i\|_2^2 \cdot h(\rho^*_{\text{SVD}}(i)).$$

On the other hand, if we use the SVD solution, we can note that since we pick the same $\Lambda^*$ and $D^*$, the SVD solution only differs in that we plug in $\beta_i = \rho^*_{\text{SVD}}(i)$ to the objective instead (Lemma 3.3.10): Thus, the resulting value of that objective is

$$\frac{\|W\|_F^2}{2} - \sum_{i=1}^{m} \left[ \|W_i\|_2 \cdot \sqrt{h}(\rho^*_{\text{SVD}}(i))\rho^*_{\text{SVD}}(i) - \frac{1}{2}\rho^*_{\text{SVD}}(i)^2 \right].$$

Therefore, computing the absolute difference to get the suboptimality gap, we get

$$\begin{aligned} &\sum_{i=1}^{m} \|W_i\|_2 \cdot \sqrt{h}(\rho^*_{\text{SVD}}(i))\rho^*_{\text{SVD}}(i) - \frac{1}{2}\rho^*_{\text{SVD}}(i)^2 \\ &+ \sum_{i=1}^{m} \frac{1}{2}\|W_i\|_2^2 \cdot h(\rho^*_{\text{SVD}}(i)) \\ &= \frac{1}{2}\sum_{i=1}^{m} \left( \|W_i\|_2 \cdot \sqrt{h}(\rho^*_{\text{SVD}}(i)) - \rho^*_{\text{SVD}}(i) \right)^2. \end{aligned} \tag{B.11}$$

□

Using the lower bound in Theorem 3.3.9, we can now characterize the conditions on the full-rank matrix $W \in \mathbb{R}^{d\times m}$ where the optimum of $\mathcal{R}(Y)$ from Problem 3.3.6 and the SVD solution are significantly different.

**Corollary B.3.7.** *Consider $W \in \mathbb{R}^{d\times m}$ and the problem of finding the best nonlinear approximation low-rank $Y \in \mathbb{R}^{d\times m}$, as described in Theorem 3.3.8. Then, cases where more correlation terms $\rho^*_{\text{SVD}}(i)$ are smaller result in the SVD solution having larger sub-optimality gaps (measured with respect to $\mathcal{R}(Y)$ defined in Problem 3.3.6).*

*Proof.* From the proofs of Theorem 3.3.8 and Theorem 3.3.9, we only need to consider the difference between $\sqrt{h}(\rho^*_{\text{SVD}}(i))$ and $\rho^*_{\text{SVD}}(i)$. Note that the gap between $\sqrt{h}(\rho^*_{\text{SVD}}(i))$ and $\rho^*_{\text{SVD}}(i)$

increases as $\rho^*_{\text{SVD}}(i)$ decreases (since $\sqrt{h}(\rho^*_{\text{SVD}}(i))$ is a convex and monotone increasing upper bound to $\rho^*_{\text{SVD}}(i)$ which converges at $\rho^*_{\text{SVD}}(i) = 1$, see Lemma B.2.4 and Corollary B.1). □

*Remark* B.3.8 (Orthogonality Intuition for Corollary B.3.7). Corollary B.3.7 roughly translates to a statement about the approximate orthogonality of the columns of $W$ – if the columns of $W$ are all mostly orthogonal, it is hard to achieve large correlations $\rho_i = \langle \frac{W_i}{\|W_i\|_2}, \frac{Y_i}{\|Y_i\|_2} \rangle$ with a low-rank $Y \in \mathbb{R}^{d\times m}$ since we can only select $Y_i$ which span a certain subspace – if the columns are approximately orthogonal, they do not live in a degenerate subspace and if for instance they are chosen uniformly randomly on the sphere, they will typically be near orthogonal to the vectors of the low-rank subspace spanned by the columns of $Y$. Therefore, since the $\rho_i$ will be typically small in this setting, by Corollary B.3.7, there will be a larger sub-optimality gap for the SVD.

When $\max_i \rho^*_{\text{SVD}}(i) < 1$ (larger $\rho^*_{\text{SVD}}(i)$ correspond to smaller gaps), we can prove a stronger result:

**Corollary B.3.9.** *Suppose the columns of $W \in \mathbb{R}^{d\times m}$ satisfy $\|W_i\|_2 = 1$ for all $i \in [m]$. If we have that $\max_{i\in[m]} \rho^*_{\text{SVD}}(i) < \rho_{\max} < 1$, then the sub-optimality gap grows as $m$ increases. Furthermore, the sub-optimality gap is monotone non-decreasing as $r$ decreases.*

*Proof.* Using Theorem 3.3.9 and the fact that $\|W_i\|_2 = 1$ for all $i \in [m]$, we have

$$\begin{aligned}
&\frac{1}{2}\left\|\sqrt{h}(\rho^*_{\text{SVD}}) - \rho^*_{\text{SVD}}\right\|_2^2 \\
&= \frac{1}{2}\sum_{i=1}^{m}\left(\sqrt{h}(\rho^*_{\text{SVD}}(i)) - \rho^*_{\text{SVD}}(i)\right)^2 \\
&\geq \frac{m}{2}\cdot\left(\sqrt{h}(\rho_{\max}) - \rho_{\max}\right)^2,
\end{aligned} \tag{B.12}$$

where we used the fact that $\sqrt{h}(\rho^*_{\text{SVD}}(i)) - \rho^*_{\text{SVD}}(i)$ decreases as $\rho^*_{\text{SVD}}(i)$ increases. Then note that the maximum correlation attainable by the low-rank SVD approximation, $\rho_{\max}$, is a monotone non-decreasing function of the rank $r$. This fact follows because by reducing the rank, we only further restrict the choice of subspace which the columns $Y_i \in \mathbb{R}^d$ can live in. Applying this fact, we have that as $r$ decreases, by Corollary B.3.7, $\left(\sqrt{h}(\rho^*_{\max}) - \rho^*_{\max}\right)^2$ does not decrease. □

**Lemma B.3.10.** *For $\rho \in \mathbb{R}$, let*

$$f(\rho) := \left(\sqrt{h}(\rho) - \rho\right)^2.$$

*Then, $f : \mathbb{R} \to \mathbb{R}$ is convex and non-increasing in $\rho$.*

*Proof.* We compute $f'(\rho)$ to get

$$f'(\rho) = \frac{-2}{\pi^2} \cdot \arccos(\rho) \cdot \left(\sqrt{1-\rho^2} - \rho \cdot \arccos(\rho)\right)$$

Note that $\sqrt{h}(\rho) - \rho > 0$ for $\rho \in [0, 1]$ (Lemma B.2.3) and also $\arccos(\rho) \geq 0$ for $\rho \in [0, 1]$. Thus, $f'(\rho) \leq 0$. Now also note from our previous characterization (Lemma B.2.3) that $\sqrt{h}(\rho) - \rho > 0$ is monotone non-increasing in $\rho$ for $\rho \in [0, 1]$, and that this is likewise true for $\arccos(\rho)$ – thus the function is also monotone non-increasing. Putting these two facts together gives the result. □

**Lemma B.3.11.** *For $u \in \mathbb{R}^d$ uniformly distributed on the $\ell_2$ sphere and $E_r$ a diagonal binary matrix consisting of $r$ ones, we have that*

$$\mathbb{E}_u \left[u^\top E_r u\right] = \frac{r}{d}.$$

*Proof.* Note that $u$ is radially symmetric, and thus for standard basis vectors $e_1, \ldots, e_d$, we have $u^\top e_i$ has the same distribution for all $i \in [d]$. Since $\|u\|_2 = 1$, we have

$$1 = \mathbb{E}\left[u^\top u\right] = \sum_{j=1}^{d} \mathbb{E}\left[\left(u^\top e_j\right)^2\right] = d \cdot \mathbb{E}\left[\left(u^\top e_1\right)^2\right],$$

and therefore

$$\mathbb{E}\left[\left(u^\top e_1\right)^2\right] = 1/d.$$

Thus

$$\mathbb{E}_u \left[u^\top E_r u\right] = \sum_{j=1}^{r} \left(u^\top e_j\right)^2 = r/d.$$

□