Generalization Bounds for Data-Driven Numerical Linear Algebra

Peter Bartlett* UC Berkeley
Piotr Indyk† MIT
Tal Wagner‡ Microsoft Research

June 17, 2022

Abstract

Data-driven algorithms can adapt their internal structure or parameters to inputs from unknown application-specific distributions, by learning from a training sample of inputs. Several recent works have applied this approach to problems in numerical linear algebra, obtaining significant empirical gains in performance. However, no theoretical explanation for their success was known.

In this work we prove generalization bounds for those algorithms, within the PAC-learning framework for data-driven algorithm selection proposed by Gupta and Roughgarden (SICOMP 2017). Our main results are closely matching upper and lower bounds on the fat shattering dimension of the learning-based low rank approximation algorithm of Indyk et al. (NeurIPS 2019). Our techniques are general, and provide generalization bounds for many other recently proposed data-driven algorithms in numerical linear algebra, covering both sketching-based and multigrid-based methods. This considerably broadens the class of data-driven algorithms for which a PAC-learning analysis is available.

1 Introduction

Traditionally, algorithms are formally designed to handle a single unknown input. In reality, however, computational problems are often solved on multiple different yet related inputs over time. It is therefore natural to tailor the algorithm to the inputs it encounters, and leverage — or *learn* from — past inputs, in order to improve performance on future ones.

This paradigm addresses common scenarios in which we need to make design choices about the algorithm, such as setting parameters or selecting algorithmic components. It advocates making those design choices in an automated way, based on past inputs viewed as a training set, premised to share underlying similarities with future inputs. Rather than trying to model or gauge these similarities explicitly, the idea is to let a learning mechanism detect and exploit them implicitly. This is often referred to as *self-improving*, *data-driven* or *learning-based* algorithm design, and by now has been applied successfully to a host of computational problems in various domains [ACC+11, GR17, Bal20, MV20].

In particular, this approach has recently become popular in efficient algorithms for computational problems in linear algebra. Matrix computations are very widely used and are often hard to scale, leading to an enormous body of work on developing fast approximate algorithms for them. These algorithms often rely on internal auxiliary matrices, like the sketching matrix in sketch-based methods [Woo14] or the prolongation matrix in algebraic multigrid methods [BHM00], and the choice of auxiliary matrix is crucial for good performance. Classically, these auxiliary matrices are chosen either at random (from carefully designed distributions which are oblivious to the input) or handcrafted via elaborate heuristic methods. However, a recent

*peter@berkeley.edu
†indyk@mit.edu
‡tal.wagner@gmail.com

surge of work on learning-based linear algebra shows they can be successfully learned in an automated way from past inputs [IVY19, ALN20, LLV+20, LGM+20, IWW21].

The empirical success of learning-based algorithms has naturally led to seeking theoretical frameworks for reasoning about them. [GR17] suggested modeling the problem in terms of statistical learning, and initiated a PAC-learning theory for algorithm selection. In their formulation, the inputs are drawn independently from an unknown distribution $\mathcal{D}$, and the goal is to choose an algorithm from a given class $\mathcal{L}$, so as to approximately optimize the expected performance on $\mathcal{D}$. One then proves upper bounds on the *pseudo-dimension* or the *fat shattering dimension* of $\mathcal{L}$, which are analogs of the VC-dimension for real-valued function classes. By classical VC theory, such bounds imply *generalization*, i.e., that an approximately optimal algorithm for $\mathcal{D}$ can be chosen from a bounded number of samples (proportional to either notion of dimension). Building on this framework, subsequent works have developed sets of tools for proving pseudo-dimension bounds for learning-based algorithms for various problems, focusing on combinatorial optimization and mechanism design [BNVW17, BDSV18, BSV18, BSV20, BDD+21, BPSV21].

However, existing techniques do not yield generalization bounds for the learning-based linear algebraic algorithms mentioned above, and so far, no generalization bounds were known for them.

Our contribution. We develop a new approach for proving PAC-learning generalization bounds for learning-based algorithms, which is applicable to linear algebraic problems. For concreteness, we mostly focus on the learning-based low rank approximation algorithm of [IVY19], called IVY. Operating on input matrices of order $n \times n$, IVY learns an auxiliary sketching matrix specified by n real parameters. Our main results is, loosely speaking, that the fat shattering dimension of IVY is $\widetilde{\Theta}(n)$, which yields a similar bound on its sample complexity.[1] The precise bound we prove also depends on the fatness parameter ϵ, the target low rank k, and the row-dimension m and sparsity s of the learned sketching matrix; for now, it is instructive to think of all of them as constants. See Section 2.3 for the full result.

We proceed to show that the tools we develop also lead to pseudo-dimension or fat shattering dimension bounds for various other learning-based linear algebraic algorithms known in the literature, which in addition to IVY include the learned low rank approximation algorithms of [ALN20], [LLV+20] and [IWW21], and the learned regression algorithm of [LGM+20]. These algorithms represent both the sketch-based and the multigrid-based approaches in numerical linear algebra.

Our work significantly advances the line of research on the statistical foundations of learning-based algorithms, by extending the PAC-learning framework to a prominent and well-studied area of algorithms not previously covered by it, through developing a novel set of techniques.

2 Background and Main Results

2.1 Generalization Framework for Learning-Based Algorithms

The following is the PAC-learning framework for algorithm selection, due to [GR17]. Let $\mathcal{X}$ be a domain of inputs for a given computational problem, and let $\mathcal{D}$ be a distribution over $\mathcal{X}$. Let $\mathcal{L}$ be a class of algorithms that operate on inputs from $\mathcal{X}$. We identify each algorithm $L \in \mathcal{L}$ with a function $L : \mathcal{X} \to [0,1]$ that maps the result of running L on x to a loss $L(x)$, where we shift and normalize the loss to be in $[0,1]$ for convenience. Let $L^* \in \mathcal{L}$ be an optimal algorithm in expectation on $\mathcal{D}$, i.e., $L^* = \operatorname{argmin}_{L \in \mathcal{L}} \mathbb{E}_{x \sim \mathcal{D}}[L(x)]$.

We wish to find an algorithm in $\mathcal{L}$ that approximately matches the performance of L^*. To this end, we draw ℓ samples $\tilde{X} = \{x_1, \ldots, x_\ell\}$ from $\mathcal{D}$, and apply a *learning procedure* that maps $\tilde{X}$ to an algorithm $L \in \mathcal{L}$. We say that $\mathcal{L}$ is (ϵ, δ)-*learnable* with ℓ samples if there is a learning procedure that maps $\tilde{X}$ to

[1] Throughout, we use $\widetilde{O}(f)$ for $O(f \cdot \operatorname{polylog}(f))$.

$L \in \mathcal{L}$ such that

$$\Pr_{\tilde{X}}[\mathbb{E}_{x\sim D}[L(x)] \leq \mathbb{E}_{x\sim D}[L^*(x)] + \epsilon] \geq 1 - \delta.$$

A canonical learning procedure is *Empirical Risk Minimization* (ERM), which maps $\tilde{X}$ to an algorithm that minimizes the average loss over the samples, i.e., to $\operatorname{argmin}_{L\in\mathcal{L}} \frac{1}{\ell}\sum_{i=1}^{\ell} L(x_i)$. We say that $\mathcal{L}$ admits (ϵ, δ)-*uniform convergence* with ℓ samples if

$$\Pr_{\tilde{X}}\left[\forall L \in \mathcal{L} \quad \left|\frac{1}{\ell}\sum_{i=1}^{\ell} L(x_i) - \mathbb{E}_{x\sim\mathcal{D}}[L(x)]\right| \leq \epsilon\right] \geq 1 - \delta.$$

It is straightforward that (ϵ, δ)-uniform convergence implies $(2\epsilon, \delta)$-learnability with ERM with the same number of samples. To bound the number of samples, we have the following notions.

Definition 2.1 (pseudo-dimension and fat shattering dimension)**.** *Let $X = \{x_1, \ldots, x_N\} \subset \mathcal{X}$. We say that X is* pseudo-shattered *by $\mathcal{L}$ if there are thresholds $r_1, \ldots, r_N \in \mathbb{R}$ such that,*

$$\forall I \subset \{1, \ldots, N\}\ \exists L \subset \mathcal{L} \quad \textit{s.t.} \quad L(x_i) > r_i \Leftrightarrow i \in I.$$

For $\gamma > 0$, we say that X is γ-fat shattered *by $\mathcal{L}$ if there are thresholds $r_1, \ldots, r_N \in \mathbb{R}$ such that,*

$$\forall I \subset \{1, \ldots, N\}\ \exists L \subset \mathcal{L} \quad \textit{s.t.} \quad i \in I \Rightarrow L(x_i) > r_i + \gamma \quad \textit{and} \quad i \notin I \Rightarrow L(x_i) < r_i - \gamma.$$

The pseudo-dimension *of $\mathcal{L}$, denoted $\operatorname{pdim}(\mathcal{L})$, is the maximum size of a pseudo-shattered set. The* γ-fat shattering dimension *of $\mathcal{L}$, denoted $\operatorname{fatdim}_\gamma(\mathcal{L})$, is the maximum size of a γ-fat shattered set.*

Note that the pseudo-dimension is an upper bound on the γ-fat shattering dimension for every $\gamma > 0$. Classical results in PAC-learning theory show that these quantities govern the number of samples needed for (ϵ, δ)-uniform convergence, and thus for (ϵ, δ)-learning with ERM.[2] Therefore, a typical goal is to prove upper and lower bounds on them.

2.2 Learning-Based Low Rank Approximation (LRA)

Let $A \in \mathbb{R}^{n\times d}$ be an input matrix, with $n \geq d$. By normalization, we assume throughout the paper that $\|A\|_F^2 = 1$. Let $[A]_k$ denote the optimal rank-k approximation of A in the Frobenius norm, i.e.,

$$[A]_k = \operatorname{argmin}_{A' \in \mathbb{R}^{n\times d} \text{ of rank } k} \|A - A'\|_F^2.$$

It is well-known that $[A]_k$ can be computed using the singular value decomposition (SVD), in time $O(nd^2)$. Since this running time does not scale well for large matrices, a very large body of work has been dedicated to developing fast approximate LRA algorithms, which output an matrix A' of rank k that attains error close to $[A]_k$, see surveys by [Mah11, Woo14, MT20].

SCW. The SCW algorithm for LRA is due to [CW13], building on [Sar06, CW09]. It uses an auxiliary *sketching matrix* $S \in \mathbb{R}^{m\times n}$, where its row-dimension m is called the *sketching dimension*, and is chosen to be slightly larger than k and much smaller than n. The algorithm is specified in Algorithm 1.

In [CW13], S is chosen at random from an data-oblivious distribution as follows. Let $s \in \{1, \ldots, m\}$ be a *sparsity* parameter. One chooses s uniformly random entries in each column of S, and chooses the value of each of them uniformly at random from $\{1, -1\}$. The rest of the entries in S are zero. [CW13] show that given $\epsilon > 0$, by setting the sketching dimension to $m = \widetilde{O}(k^2/\epsilon^2)$, even with sparsity $s = 1$, SCW returns A' of rank k that satisfies $\|A - A'\|_F^2 \leq (1 + \epsilon)\|A - [A]_k\|_F^2$ with high probability (over the random choice of S), while running in time nearly linear in the size of A.

[2] For example, a standard result is that $O(\epsilon^{-2} \cdot (\operatorname{pdim}(\mathcal{L}) + \log(\delta^{-1})))$ samples suffice for (ϵ, δ)-uniform convergence, and thus also for (ϵ, δ)-learning with ERM. The pseudo-dimension yields somewhat tighter upper bounds for learning than the fat shattering dimension, while the latter also yields lower bounds. See, for example, Theorems 19.1, 19.2, 19.5 in [AB09].

Input: $A \in \mathbb{R}^{n\times d}$, target rank k, auxiliary matrix $S \in \mathbb{R}^{m\times n}$. **Output:** $A' \in \mathbb{R}^{n\times d}$ of rank k.

1: Compute the product SA.
2: If SA is a zero matrix, return the zero matrix of order $n \times d$.
3: Compute the SVD U, Σ, V^T of SA.
4: Compute the product AV.
5: Compute and return $[AV]_k V^T$.

Algorithm 1: The SCW low rank approximation algorithm

IVY. The IVY algorithm due to [IVY19] is a learning-based variant of SCW. The sparsity pattern of S is chosen similarly to SCW (s non-zero entries per column, chosen uniformly at random) and remains fixed. However, the values of the non-zero entries in S are now trainable parameters, learned from a training set of input matrices. In particular, S is chosen by minimizing the empirical loss $\sum_{A\in\mathcal{A}_{train}} \|A - \mathrm{SCW}_k(S,A)\|_F^2$, where $\mathrm{SCW}_k(S,A)$ denotes the output matrix of Algorithm 1, using stochastic gradient descent (SGD) on a training set $\mathcal{A}_{train} \subset \mathbb{R}^{n\times d}$.

To cast IVY in the statistical learning framework from Section 2.1, let $\mathcal{A}$ denote the set of possible input matrices (i.e., all $A \in \mathbb{R}^{n\times d}$ with $\|A\|_F^2 = 1$), and let $\mathcal{S}$ denote the set of possible sketching matrices (i.e., all $S \in \mathbb{R}^{m\times n}$ with the fixed sparsity pattern specified above). Every $S \in \mathcal{S}$ gives rise to an LRA algorithm, whose output rank-k approximation of A is $A' = \mathrm{SCW}_k(S,A)$, and whose associated loss function $L_k^{\mathrm{SCW}}(S,\cdot) : \mathcal{A} \to [0,1]$ is given by[3]

$$L_k^{\mathrm{SCW}}(S,A) = \|A - \mathrm{SCW}_k(S,A)\|_F^2.$$

Given samples from an unknown distribution $\mathcal{D}$ over $\mathcal{A}$, the learning problem is to choose $S \in \mathcal{S}$ which has approximately optimal loss in expectation over $\mathcal{D}$. Our objective is to prove generalization bounds for learning a sketching matrix $S \in \mathcal{S}$, or equivalently, for learning the class of loss functions $\mathcal{L}_{\mathrm{IVY}} = \{L_k^{\mathrm{SCW}}(S,\cdot)\}_{S\in\mathcal{S}}$.

2.3 Our Main Results

We prove closely matching upper and lower bounds on the fat shattering dimension of IVY.

Theorem 2.2. *For every $\epsilon > 0$ smaller than a sufficiently small constant, the ϵ-fat shattering dimension of IVY, $\mathrm{fatdim}_\epsilon(\mathcal{L}_{\mathrm{IVY}})$, satisfies*

$$\mathrm{fatdim}_\epsilon(\mathcal{L}_{\mathrm{IVY}}) \le O(ns \cdot (m + k\log(d/k) + \log(1/\epsilon))).$$

Furthermore, if $\epsilon < 1/(2\sqrt{k})$, then $\mathrm{fatdim}_\epsilon(\mathcal{L}_{\mathrm{IVY}}) \ge \Omega(ns)$.

These results translate to sample complexity bounds on its uniform convergence and learnability:

Theorem 2.3. *Let $\epsilon, \delta > 0$ be smaller than sufficiently small constants. The number of samples needed for (ϵ,δ)-uniform convergence for IVY, and thus for (ϵ,δ)-learning the sketching matrix of IVY with ERM, is*

$$O(\epsilon^{-2} \cdot (ns \cdot (m + k\log(d/k) + \log(1/\epsilon)) + \log(1/\delta))).$$

Furthermore, if $\epsilon \le 1/(256\sqrt{k})$, then (ϵ,ϵ)-uniform convergence for IVY requires $\Omega(\epsilon^{-2} \cdot ns/k)$ samples, and if $\epsilon < 1/(2\sqrt{k})$, then (ϵ,δ)-learning IVY with any learning procedure requires $\Omega(\epsilon^{-1} + ns)$ samples.

[3] It can be shown that the loss never exceeds $\|A\|_F^2$ for any A and S, and recalling that we assume that all input matrices are normalized so that $\|A\|_F^2 = 1$, the loss is contained in $[0,1]$.

For the sake of intuition, let us comment on typical settings for the various sizes in these results. The input matrix is of order $n \times d$, where we think of n, d as being arbitrarily large.[4] The target rank k can be thought of as a small integer constant, and the sketching dimension m as only slightly larger (the larger m is, the slower but more accurate the LRA algorithm would be). Both are generally independent of n, d. Concretely, the empirical evaluations in [IVY19, IVWW19, IWW21, ALN20, LLV+20] use k up to 40, and m up to $4k$, for matrices with thousands of rows and columns. The sparsity s is often chosen to be the smallest possible, $s = 1$, while some have found it beneficial to make it slightly larger, up to 8 [TYUC19, MT20]. The upshot is that the upper and lower bounds in Theorem 2.2 are essentially matching up to a factor of $\log(d/\epsilon)$. Furthermore, both are essentially proportional to ns, which is the number of non-zero entries in the sketching matrix, i.e., the number of trainable parameters that IVY learns.

Other learning-based algorithms. While we focus on IVY for concreteness, our techniques also yield bounds for many other learning-based linear algebraic algorithms. See Section 6.

Remark 2.4 (on computational efficiency)**.** *Like most prior work on PAC-learning, our results focus on the sample complexity of the learning problem, rather than the computational efficiency of learning (see, e.g., Remark 3.5 in [GR17]). It is currently unknown whether there exist efficient ERM learners for the data-driven algorithms considered in this work. In particular, while in practice IVY uses SGD to minimize the empirial loss, it is not known to provably converge to a global minimum. The computational efficiency of ERM for backpropagation-based algorithms (like IVY) remains a challenging open problem.*

Remark 2.5 (on the precision of real number representation)**.** *We prove Theorem 2.2 without any restriction on the numerical precision of real numbers in the computational model, that is, even when they may have unbounded precision. If one considers only bounded precision models, say where real numbers are represented by b-bit machine words, then the total number of possible sketching matrices is 2^{nsb}, leading trivially to a bound of $O(nsb)$ on the pseudo-dimension of IVY. On the other hand, the lower bound in Theorem 2.2 holds even with 1-bit machine words. In summary, both our upper and lower bounds are proven in the most general computational model.*

2.4 Technical Overview

Our starting point is a general framework of [GJ95] for bounding the VC-dimension or the pseudo-dimension. They showed that if the functions in a class (which in our case are the losses of candidate algorithms) can be computed by a certain type of algorithm, which we call a *GJ algorithm*, then the pseudo-dimension of the function class can be bounded in terms of the running time of that algorithm. We employ a simple yet crucial refinement of this framework, relying on more refined complexity measures of GJ algorithms than the running time.

Under this framework, ideally we would have liked to compute the SCW loss with a GJ algorithm which is efficient in these refined complexity measures, thus obtaining a bound on the pseudo-dimension of IVY. Unfortunately, it is not clear how to compute the SCW loss at all with a GJ algorithm, since the GJ framework only allows a narrow set of operations. Nonetheless, we show it can be *approximated* by a GJ algorithm, by relying on recent advances in numerical linear algebra, and specifically on "gap-independent" analyses of power method iterations for LRA, that do not depend on numerical properties of the input matrix (like eigenvalue gaps) [RST10, HMT11, BDMI14, Woo14, WC15, MM15]. We thus obtain a pseudo-dimension bound for the approximate loss, which translates to a fat shattering dimension bound for the true SCW loss, yielding generalization results for IVY, as well as for other learning-based numerical linear algebra algorithms.

[4] Recall we assume that $d \leq n$ by convention. For all conceptual purposes it suffices to consider the square case $n = d$.

Proof roadmap. Section 3 presents the [GJ95] framework. In Section 4, as a warm-up, we prove tight bounds for IVY in the simple case $m = k = 1$. Section 5 contains the main proof of this paper, establishing the upper bound from Theorem 2.2. The lower bound is proven in Appendix C, together completing the proof of Theorem 2.2. Theorem 2.3 follows from Theorem 2.2 using mostly standard arguments, given in Appendix D.

2.5 Related Work

[IVY19] gave a "safeguarding" technique for IVY, showing it can be easily modified to guarantee that the learned algorithm never performs worse than the oblivious SCW algorithm on future matrices. The modification is simply to concatenate an oblivious sketching matrix vertically to the learned sketching matrix. While this result guarantees that learning does not hurt, it does not show that learning can help, and provides no generalization guarantees.

[IWW21] prove "consistency" results for their learning-based algorithms, showing that the learned sketching matrix performs well on the training input matrices from which it was learned. These results have no bearing on future matrices, and again provide no generalization guarantees.

[BDD+21] recently gave a general technique for proving generalization bounds for learning-based algorithm in the statistical learning framework of Section 2.1, based on piecewise decompositions of dual function classes. While there are some formal connections between their techniques and ours, their approach does not yield useful bounds for the linear algebraic algorithms that we study, since these algorithms do not exhibit a sufficiently simple piecewise dual structure as the technique of [BDD+21] requires. We discuss this in more detail in Appendix E.

3 The Goldberg-Jerrum Framework

Our upper bounds are based on a general framework due to [GJ95] for bounding the VC-dimension. We instantiate a refined version of it, which still follows immediately from their proofs. For completeness and self-containedness, Appendix A reproduces the proof of the variant we state below.

Definition 3.1. *A **GJ algorithm** Γ operates on real-valued inputs, and can perform two types of operations:*

- *Arithmetic operations of the form $v'' = v \odot v'$, where $\odot \in \{+, -, \times, \div\}$.*
- *Conditional statements of the form "if $v \geq 0$ ... else ...".*

In both cases, v, v' are either inputs or values previously computed by the algorithm.

Every intermediate value computed by Γ is a multivariate rational function (i.e., the ratio of two polynomials) of its inputs. The degree of a rational function is the maximum of the degrees of the two polynomials in its numerator and denominator, when written as a reduced fraction. We now define two complexity measures of GJ algorithms.

Definition 3.2. *The **degree** of a GJ algorithm is the maximum degree of any rational function it computes of the inputs. The **predicate complexity** of a GJ algorithm is the number of distinct rational functions that appear in its conditional statements.*

Theorem 3.3. *Using the notation of Section 2.1, suppose that each algorithm $L \in \mathcal{L}$ is specified by n real parameters. Suppose that for every $x \in \mathcal{X}$ and $r \in \mathbb{R}$ there is a GJ algorithm $\Gamma_{x,r}$ that given $L \in \mathcal{L}$, returns "true" if $L(x) > r$ and "false" otherwise. Suppose $\Gamma_{x,r}$ has degree Δ and predicate complexity p. Then, the pseudo-dimension of $\mathcal{L}$ is at most $O(n \log(\Delta p))$.*

Let us comment on the relation between this statement and the original theorem from [GJ95]. Their statement is that if $\Gamma_{x,r}$ has running time t then the pseudo-dimension of $\mathcal{L}$ is $O(nt)$. They prove it by implcitly proving Theorem 3.3, as it is not hard to see that if the running time is t then both the degree and the predicate complexity are at most 2^t.

To illustrate why the refinement stated in Theorem 3.3 could be useful, let us first consider the degree. Consider computing q power method iterations for a matrix $M \in \mathbb{R}^{n\times n}$, i.e., computing $M^q\pi$ with some initial vector $\pi \in \mathbb{R}^n$. The running time is $t = O(n^2 q)$, so the pseudo-dimension upper bound we get based on the running time alone is $O(nt) = O(n^3 q)$. However, the degree of every entry in $M^q\pi$ (when considered as a rational function of the entries of M and π) is just $q+1$, and hence Theorem 3.3 yields the better upper bound $O(n \log q)$.

As for the predicate complexity, consider for example a GJ algorithm for choosing the minimum of r numbers (this operation will be useful for us in derandomizing the power method). The running time is $t = O(r)$, and this is tight since GJ algorithms adhere to the comparison model, so the pseudo-dimension upper bound we get based on the running time alone is $O(nt) = O(nr)$. However, the predicate complexity is $\binom{r}{2}$, and the degree is just 1, since choosing the minimum among $v_1, \ldots, v_r \in \mathbb{R}$ involves only the polynomials $v_i - v_j$ for every $i < j$. Hence, Theorem 3.3 yields the better upper bound $O(n \log r)$. The same reasoning applies to sorting r numbers (see Appendix E.1 for an example that uses this operation), and to other useful subroutines.

Remark 3.4 (oracle access to optimal solution)**.** *It is also worth observing that $\Gamma_{x,r}$ has free access to all information about x — and in particular, to the optimal solution of the computational problem addressed by $\mathcal{L}$ with x as the input— without any computational cost. For example, in LRA, $\Gamma_{x,r}$ has free access to the exact SVD factorization of the input matrix A. (Note that the difficulty for $\Gamma_{x,r}$ lies in computing the losses of other than optimal solutions, namely the SCW loss induced by the given sketching matrix S). Similarly, in a regression problem $\min_y \|Ay - b\|$, the input x is the pair A, b, and $\Gamma_{x,r}$ has free access to the optimal solution $y^* = \operatorname{argmin}_y \|Ay - b\|$. These observations are useful in proving bounds for some of the algorithms we consider in Section 6.*

4 Warm-up: The Case $m = k = 1$

As a warm-up, we consider the simple case where both the target rank k and the sketching dimension is m are 1. We show an upper bound of $O(n)$ on the pseudo-dimension and a lower bound of $\Omega(n)$ on the ϵ-fat shattering dimension for every $\epsilon \in (0, 0.5)$ (this immediately implies that both are $\Theta(n)$). This case is particularly simple because the SCW loss admits a closed-form expression, given next (the proof appears in Appendix B.1). Note that the sketching matrix in this case is just a vector, denoted $w \in \mathbb{R}^n$ for the rest of this section, and it is specified by n real parameters.

Fact 4.1. *The loss of SCW with input matrix $A \in \mathbb{R}^{n\times d}$, sketching vector $w \in \mathbb{R}^n$ and target rank $k = 1$ is $\|A - \frac{1}{\|A^T w\|^2} AA^T ww^T A\|_F^2$.*

Theorem 4.2. *The pseudo-dimension of IVY in the $m = k = 1$ case is $\Theta(n)$, and the ϵ-fat shattering dimension is $\Theta(n)$ for every $\epsilon \in (0, 0.5)$*

Proof. Since the pseudo-dimension is an upper bound on the fat shattering dimension, it suffices to prove the upper bound for the former and the lower bound for the latter. For a fixed matrix A and threshold $r \in \mathbb{R}$, a GJ algorithm $\Gamma_{A,r}$ can evaluate the loss formula from Fact 4.1 on a given w with degree $O(1)$, and compare the loss to r with predicate complexity 1. Therefore, by Theorem 3.3, the pseudo-dimension is $O(n)$.

For the lower bound, we first argue that if A is a rank-1 matrix, then SCW with sketching vector w attains zero loss if and only if $w^T A \neq 0$. Indeed, by Fact 4.1, if $w^T A = 0$ then the loss is $\|A\|_F^2$, which

is non-zero for a rank-1 matrix. If $w^T A \neq 0$, we write A in SVD form $A = \sigma u v^T$ with $\sigma \in \mathbb{R}$, $u \in \mathbb{R}^n$, $v \in \mathbb{R}^d$, and $\|u\| = \|v\| = 1$, and by plugging this into Fact 4.1, the loss is 0.

Now, consider the set of matrices $A_1, \ldots, A_n \in \mathbb{R}^{n \times d}$, where A_i is all zeros except for a single 1-entry in row i, column 1. For every $I \subset \{1, \ldots, n\}$, let $w_I \in \mathbb{R}^n$ be its indicator vector. By the above, SCW with sketching vector w_I attains zero loss on the matrices $\{A_i : i \in I\}$, and loss $\|A\|_F^2 = 1$ on the rest. Therefore this set of n matrices is ϵ-fat shattered for every $\epsilon \in (0, 0.5)$. □

5 Proof of the Upper Bound

In this section we prove the upper bound in Theorem 2.2 on the fat shattering dimension of IVY. We start by stating a formula for the output rank-k matrix of SCW (Algorithm 1). For a matrix M, we use $M^\dagger$ to denote its Moore-Penrose pseudo-inverse, and recall that we use $[M]_k$ to denote its best rank-k approximation. All of the proofs omitted from this section are collected in Appendix B.

Lemma 5.1. *The output matrix of the SCW algorithm equals* $[A(SA)^\dagger(SA)]_k$.

5.1 Computing Projection Matrices

Next, we give a GJ algorithm for computing orthogonal projection matrices. Recall that the orthogonal projection matrix on the row-space of a matrix Z is given by $Z^\dagger Z$.

Lemma 5.2. *There is a GJ algorithm that given an input matrix Z with k rows, computes $Z^\dagger Z$. The algorithm has degree $O(k)$ and predicate complexity at most 2^k. Furthermore, if Z is promised to have full rank k, then the predicate complexity is zero.*

Proof. We start by proving the "furthermore" part, supposing Z has full rank k. Since Z has full rank k, by a known identity for the pseudo-inverse we can write $Z^\dagger Z = Z^T(ZZ^T)^{-1}Z$. We use the matrix inversion algorithm from [Csa76] (similar/identical algorithms have appeared in other places as well), to invert an invertible $k \times k$ matrix M. Define the following matrices,

$$B_1 = I, \quad B_i = MB_{i-1} - \frac{\mathrm{tr}(MB_{i-1})}{k-1} \cdot I.$$

The inverse is then given by,

$$M^{-1} = \frac{k}{\mathrm{tr}(MB_k)} \cdot B_k.$$

Note that each entry of B_i is a polynomial of degree $i-1$ in the entries of M, and therefore each entry of M^{-1} is a rational function of degree $k-1$ in the entries of M. In our case $M = ZZ^T$, and the desired output is $Z^T(ZZ^T)^{-1}Z$, and therefore it has degree $O(k)$. This finishes the proof of the "furthermore" part.

We proceed to showing the lemma without the full rank assumption. Suppose M has rank $r \leq k$ which could be strictly smaller than k (and r need not be known to the algorithm). If we find a matrix $Y \in \mathbb{R}^{r \times d}$ whose rows form a basis for the row-space of M, then we could write the desired projection matrix as $Z^\dagger Z = Y^\dagger Y = Y^T(YY^T)^{-1}Y$, and compute it with a GJ-algorithm of degree $O(r) \leq O(k)$ in the entries of Y, as above.

So, it remains to compute such a matrix Y. To this end we use another result from [Csa76]: if M is a $k \times k$ matrix with characteristic polynomial $f_M(\lambda) = \det(\lambda I - M) = \sum_{i=0}^k c_i \lambda^{k-i}$, then $c_i = -\frac{1}{i}\mathrm{tr}(MB_i)$, with B_i defined as above. In particular, the free coefficient c_k can be computed by a GJ algorithm of degree $O(k)$. This yields a GJ algorithm for determining whether a $k \times k$ matrix has full rank, since this holds if and only if $c_k \neq 0$.

Now, to compute Y from Z, we can simply go over the rows of Z one by one, tentatively add each row to our (partial) Y, and keep or remove it depending on whether Y still has full row rank. To check this, we compute YY^T (which is a square matrix of order at most $r \leq k$) and check whether it has full rank as above. Overall, Y and therefore the output $Y^T(YY^T)^{-1}Y$ are computed using a GJ algorithm with degree $O(k)$. Choosing Y involves conditional statements with up to 2^k rational functions, which are the free coefficients of the characteristic polynomials of every subset of the k rows of Z, so the predicate complexity is at most 2^k. □

We remark that this lemma already yields an upper bound on the pseudo-dimension of IVY in the case $m = k$. The full regime $m \geq k$ is more involved and will occupy the rest of this section.

Corollary 5.3. *The pseudo-dimension of IVY with $m = k$ is $O(nsk)$.*

Proof. S has $m = k$ rows, hence SA has rank at most k, hence $[A(SA)^\dagger(SA)]_k = A(SA)^\dagger(SA)$. By Lemma 5.1, the SCW loss is $\|A - A(SA)^\dagger(SA)\|_F^2$. By Lemma 5.2, it can be computed with a GJ algorithm of degree $O(k)$ and predicate complexity 2^k. Recalling that the learned sketching matrix in IVY is specified by ns real parameters, the corollary follows from Theorem 3.3. □

5.2 Derandomized Power Method Iterations

For the rest of this section we denote $B = A(SA)^\dagger(SA)$ for brevity. Note that B is of order $n \times d$. By Lemma 5.1 the SCW loss equals $\|A - [B]_k\|_F^2$, while Lemma 5.2 shows how to compute B with a GJ algorithm. Our next goal is to approximately compute $[B]_k$ with a GJ algorithm.

To this end we use a result on iterative methods for LRA, due to [MM15], building on ideas from [RST10, HMT11, WC15, BDMI14, Woo14]. It states that given B, an initial crude approximation for $[B]_k$ can be refined into a good approximation with a small number of powering iterations. The next theorem is somewhat implicit in [MM15]; see Appendix B.3 for details.

Theorem 5.4 ([MM15])**.** *Suppose we have a matrix $P \in \mathbb{R}^{d \times k}$ that satisfies*

$$\|B - (BP)(BP)^\dagger B\|_F^2 \leq O(kd) \cdot \|B - [B]_k\|_F^2. \tag{1}$$

Then, $Z = (BB^T)^q BP$ with $q = O(\epsilon^{-1} \log(d/\epsilon))$ satisfies $\|B - ZZ^\dagger B\|_F^2 \leq (1+\epsilon) \cdot \|B - [B]_k\|_F^2$.

Normally, P is chosen at random as a matrix of independent gaussians, which satisfies Equation (1) with high probability. Since GJ algorithms are deterministic, we derandomize this approach using subsets of the standard basis.

Lemma 5.5. *Suppose $k < d$. For every $B \in \mathbb{R}^{n \times d}$, there is a subset of size k of the standard basis in $\mathbb{R}^d$, such that if we organize its elements into a matrix $P \in \mathbb{R}^{d \times k}$, it satisfies Equation (1).*

5.3 The Proxy Loss

Let $L_k^{\mathrm{SCW}}(S, A)$ denote the loss of SCW with input matrix A, sketching matrix S and target rank k. By Lemma 5.1, $L_k^{\mathrm{SCW}}(S, A) = \|A - A(SA)^\dagger(SA)\|_F^2$. We now approximate this quantity with a GJ algorithm. Formally, given $\epsilon > 0$, we define a proxy loss $\hat{L}_{k,\epsilon}(S, A)$ as the output of the following GJ algorithm, which operates on an input S with a fixed A. Recall that d is the column dimension of A.

1. Compute $B = A(SA)^\dagger(SA)$ using the GJ algorithm from Lemma 5.2.
2. Let $\{P_i : i = 1 \dots \binom{d}{k}\}$ be the set of matrices in $\mathbb{R}^{d \times k}$ whose columns form all of the possible k-subsets of the standard basis in $\mathbb{R}^d$.

3. For every P_i, compute $Z_i = (BB^T)^q BP_i$, where $q = O(\epsilon^{-1}\log(d/\epsilon))$.

4. Choose Z as the Z_i that minimizes $\|B - Z_i Z_i^\dagger B\|_F^2$, using Lemma 5.2 to compute $Z_i Z_i^\dagger$.

5. Compute and return the proxy loss, $\hat{L}_{k,\epsilon}(S, A) = \|A - ZZ^\dagger B\|_F^2$.

Given $r \in \mathbb{R}$, we can return "true" if $\hat{L}_{k,\epsilon}(S, A) > r$ and "false" otherwise, obtaining a GJ algorithm that fits the assumptions of Theorem 3.3.

Lemma 5.6. *This GJ algorithm has degree* $\Delta = O(mk\epsilon^{-1}\log(d/\epsilon))$ *and predicate complexity* $p \le 2^m \cdot 2^{O(k)} \cdot (d/k)^{3k}$.

Proof. Since SA has m rows, computing $(SA)^\dagger(SA)$ with Lemma 5.2 in step 1 has degree $O(m)$ and predicate complexity 2^m. For each P_i, the q powering iterations in step 3 blow up the degree by q. Since Z_i has k columns, computing $Z_i Z_i^\dagger$ with (the transposed version of) Lemma 5.2 blows up the degree by $O(k)$ and the predicate complexity by 2^k. Choosing Z in step 4 entails pairwise comparisons between $\binom{d}{k}$ values, blowing up the predicate complexity by $\binom{\binom{d}{k}}{2} \le e^2(ed/k)^{2k}$. Step 5 blows up the degree by only $O(1)$ and does not change the predicate complexity (note that $ZZ^\dagger$ has already been computed). The final check whether $\hat{L}_{k,\epsilon}(S, A) > r$ is one of $\binom{d}{k} \le (ed/k)^k$ polynomials (one per possible value of Z). Overall, the algorithm has degree $O(mkq) = O(mk\epsilon^{-1}\log(d/\epsilon))$, and predicate complexity at most $2^m \cdot 2^{O(k)} \cdot (d/k)^{3k}$. □

Next we show that the proxy loss approximates the true SCW loss.

Lemma 5.7. *For every* S, A, *it holds that* $0 \le \hat{L}_{k,\epsilon}(S, A) - L_k^{\mathrm{SCW}}(S, A) \le \epsilon$.

Proof. Given S, A, let $B = A(SA)^\dagger(SA)$, and let $U \in \mathbb{R}^{n\times k}$ a matrix whose columns are the top k left-singular vectors of B. This means $[B]_k = UU^T B$. Therefore, by Lemma 5.1, we have $L_k^{\mathrm{SCW}}(S, A) = \|A - UU^T B\|_F^2$. Let Z be the matrix computed in step 4 of the GJ algorithm for $\hat{L}_{k,\epsilon}(S, A)$, and recall we have $\hat{L}_{k,\epsilon}(S, A) = \|A - ZZ^\dagger B\|_F^2$. On one hand, since Z has k columns, $ZZ^\dagger B$ has rank at most k. Therefore, the optimality of $[B]_k = UU^T B$ as a rank-k approximation of B implies,

$$\|B - UU^T B\|_F^2 \le \|B - ZZ^\dagger B\|_F^2. \tag{2}$$

On the other hand, by Lemma 5.5, some P_i considered in step 2 of the GJ algorithm satisfies Equation (1). By Theorem 5.4, this implies that its corresponding Z_i (computed in step 3) satisfies $\|B - Z_i Z_i^\dagger B\|_F^2 \le (1+\epsilon)\cdot\|B - UU^T B\|_F^2$, and consequently, Z satisfies

$$\|B - ZZ^\dagger B\|_F^2 \le (1+\epsilon)\cdot\|B - UU^T B\|_F^2. \tag{3}$$

Since $(SA)^\dagger(SA)$ is a projection matrix, we have by the Pythagorean identity,

$$\begin{aligned}
\hat{L}_{k,\epsilon}(S, A) &= \|A - ZZ^\dagger B\|_F^2 \\
&= \|A - ZZ^\dagger A(SA)^\dagger(SA)\|_F^2 \\
&= \|A(SA)^\dagger(SA) - ZZ^\dagger A(SA)^\dagger(SA)\|_F^2 + \|A(I - (SA)^\dagger(SA))\|_F^2 \\
&= \|B - ZZ^\dagger B\|_F^2 + \|A - B\|_F^2,
\end{aligned}$$

and similarly, $L_k^{\mathrm{SCW}}(S, A) = \|A - UU^T B\|_F^2 = \|B - UU^T B\|_F^2 + \|A - B\|_F^2$. Putting these together, $\hat{L}_{k,\epsilon}(S, A) - L_k^{\mathrm{SCW}}(S, A) = \|B - ZZ^\dagger B\|_F^2 - \|B - UU^T B\|_F^2$. From Equations (2) and (3) we now

get, $0 \leq \hat{L}(S,A) - L_k^{\mathrm{SCW}}(S,A) \leq \epsilon \cdot \|B - UU^T B\|_F^2$. The lemma follows since both $(I - UU^T)$ and $(SA)^\dagger(SA)$ are projection matrices, implying that

$$\|(I - UU^T)B\|_F^2 \leq \|B\|_F^2 = \|A(SA)^\dagger(SA)\|_F^2 \leq \|A\|_F^2,$$

and we recall that we assume throughout that $\|A\|_F^2 = 1$. □

We can now complete the proof of the upper bound in Theorem 2.2. Let us recall notation: Let $\mathcal{A}$ be set possible input matrices to the LRA problem (i.e., all matrices $A \in \mathbb{R}^{n \times d}$ with $\|A\|_F^2 = 1$), and let $\mathcal{S}$ be the set of all possible sketching matrices that IVY can learn (which are all matrices of order $m \times n$ with the fixed sparsity pattern used by SCW and IVY). The class of IVY losses (whose fat shattering dimension we aim to bound) is $\mathcal{L}_{\mathrm{IVY}} = \{L_k^{\mathrm{SCW}}(S,\cdot) : \mathcal{A} \to [0,1]\}_{S \in \mathcal{S}}$, and the class of proxy losses is $\hat{\mathcal{L}}_\epsilon = \{\hat{L}_{k,\epsilon}(S,\cdot) : \mathcal{A} \to [0,1]\}_{S \in \mathcal{S}}$.

By Lemma 5.6, given $A \in \mathcal{A}$ and $S \in \mathcal{S}$, the proxy loss $\hat{L}_{k,\epsilon}(S,A)$ can be computed by a GJ algorithm with degree $\Delta = O(mk\epsilon^{-1}\log(d/\epsilon))$ and predicate complexity $p \leq 2^m \cdot 2^{O(k)} \cdot (d/k)^{3k}$. Since each $S \in \mathcal{S}$ is defined by ns real parameters, Theorem 3.3 yields that

$$\mathrm{pdim}(\hat{\mathcal{L}}_\epsilon) = O(ns\log(\Delta p)) = O(ns \cdot (m + k\log(d/k) + \log(1/\epsilon))). \tag{4}$$

Let $A_1, \ldots, A_N \in \mathcal{A}$ be a subset of matrices of size N which is ϵ-fat shattered by $\mathcal{L}_{\mathrm{IVY}}$. Recall that by Definition 2.1, this means that there are thresholds $r_1, \ldots, r_N \in \mathbb{R}$, such that for every $I \subset \{1, \ldots, N\}$ there exists $S \in \mathcal{S}$ such that $L_k^{\mathrm{SCW}}(S, A_i) > r_i + \epsilon$ if $i \in I$ and $L_k^{\mathrm{SCW}}(S, A_i) < r_i - \epsilon$ if $i \notin I$. By Lemma 5.7, this implies that $\hat{L}_{k,\epsilon}(S, A_i) > r_i$ if and only if $i \in I$. Thus, $A_1, \ldots, A_N$ is pseudo-shattered by $\hat{\mathcal{L}}_\epsilon$, implying that $N \leq \mathrm{pdim}(\hat{\mathcal{L}}_\epsilon)$. Since this holds for any subset of size N which is ϵ-fat shattered by $\mathcal{L}_{\mathrm{IVY}}$, we have $\mathrm{fatdim}_\epsilon(\mathcal{L}_{\mathrm{IVY}}) \leq \mathrm{pdim}(\hat{\mathcal{L}}_\epsilon)$. The upper bound in Theorem 2.2 now follows from Equation (4).

6 Other Learning-Based Algorithms in Numerical Linear Algebra

Generally, our approach is applicable whenever the loss incurred by a given algorithm (or equivalently, by a given setting of parameters) on a fixed input can be computed by an efficient GJ algorithm (in the efficiency measures of Definition 3.2), when given "oracle access" to the optimal solution for that fixed input.[5] In this section, we show that in addition to IVY, our method also yields generalization bounds for various other data-driven and learning-based algorithms for problems in numerical linear algebra (specifically, LRA and regression) that have appeared in the literature. The first two algorithms discussed below (Butterfly LRA and Multi-sketch LRA) are variants of IVY, and the bounds for them essentially follow from Theorem 2.2. For the other two algorithms (Few-shot LRA and Learned Multigrid) we describe the requisite GJ algorithms below.

Butterfly learned LRA. [ALN20] suggested a learned LRA algorithm similar to IVY, except that the learned sparse sketching matrix is replaced by a dense but implicitly-sparse *butterfly* gadget matrix (a known and useful gadget that arises in fast Fourier transforms). This gadget induces a sketching matrix $S \in \mathbb{R}^{m \times n}$ specified by $O(m \log n)$ learned parameters, where each entry is a product of $\log n$ of those parameters. Since they use the IVY loss to learn the matrix, our results give the same upper bound as Theorem 2.2 on the fat shattering dimension of their algorithm, times $\log n$ to account for the initial degree (in the GJ sense of Definition 3.1) of the entries in S.

[5] See Remark 3.4 regarding oracle access to the optimal solution. This oracle access was not needed for proving our bounds for IVY, but t will be needed for two of the algorithms discussed in this section: Few-shot LRA and Learned Multigrid.

Multi-sketch learned LRA. [LLV+20] propose a more involved learned LRA algorithm, which uses two learned sketching matrices. Still, they train each of them using the IVY loss, and therefore the upper bound from Theorem 2.2 holds for their algorithm as well, up to constants.

Few-shot learned LRA. [IWW21] use a different loss than IVY for learned LRA: $\text{loss}(S, A) = \|U_k^T S^T S U - I_0\|_F^2$, where U is the left-factor in the SVD of A, U_k is its restriction to the top-k columns, and I_0 is the identity of order k concatenated on the right with zero columns to match the number of columns in U. Recall that in the GJ framework, the GJ algorithm has free "oracle access" to U (see Remark 3.4). Therefore, this loss can be computed by GJ algorithm of degree 4 and predicate complexity 1, yielding an upper bounded of $O(ns)$ on its pseudo-dimension by Theorem 3.3.

Learned multigrid regression. [LGM+20] presented a learning-based algorithm for linear regression, which is based on the well-studied *multigrid* paradigm for solving numerical problems. The specific algorithm they build on is known as 2-level algebraic multigrid (AMG). It approximates the solution to a regression problem $\min_x \|Ax - b\|_2^2$, where A is a square matrix of order $n \times n$, by iterative improvements that use a sparse auxiliary matrix $P \in \mathbb{R}^{m \times n}$ called a *prolongation matrix*. While there is a large body of literature on heuristic methods for choosing P, [LGM+20] suggest to instead learn it using a graph neural network. Similarly to IVY, they keep the sparsity pattern of P fixed, and learn the values of its non-zero entries as trainable parameters.

Let $x^{(i)}$ denote the approximate solution produced by 2-level AMG in iteration i, starting from some fixed initial guess $x^{(0)}$. For details of how $x^{(i)}$ is computed in practice, we refer to [LGM+20]. For our purposes, it suffices to note that there is a closed-form formula for obtaining $x^{(i)}$ from the true optimal solution x^*, described next. Let $s_1, s_2 \geq 1$ be two fixed (non-learned) integer parameters of the algorithm. Let L be the lower-triangular part of the input matrix A (including the diagonal). [LGM+20] restrict their algorithm for input matrices A and prolongation matrices P such that both L and $P^T AP$ are invertible. We then have the identity,

$$x^{(i)} = x^* + (I - L^{-1}A)^{s_2} \cdot (I - P(P^T AP)^{-1} P^T A) \cdot (I - L^{-1}A)^{s_1} \cdot (x^{(i-1)} - x^*). \tag{5}$$

Let $\|P\|_0$ denote the number of non-zeros in the fixed sparsity pattern of P, and recall that m is the row-dimension of P. We show that the pseudo-dimension of 2-level AMG with q iterations is $O(\|P\|_0 \cdot q \log m)$, by describing a GJ algorithm Γ for computing its loss $\|Ax^{(q)} - b\|_2^2$. To this end, note that Γ has "oracle access" to any information about A (see Remark 3.4). In particular, $(I - L^{-1}A)^{s_1}$, $(I - L^{-1}A)^{s_2}$ and x^* are all available to it. Furthermore, as shown in the proof of Lemma 5.2, Γ can compute $(P^T AP)^{-1}$ using degree $O(m)$ and predicate complexity zero. Using Equation (5), it can compute $x^{(i)}$ from $x^{(i-1)}$ while increasing the degree by only 2 (from multiplying $(P^T AP)^{-1}$ by P on the left and by P^T on the right). Iterating this, Γ can compute $x^{(q)}$ and $\|Ax^{(q)} - b\|_2^2$ from the initial guess $x^{(0)}$ using degree $O(m^q)$ and predicate complexity zero. Comparing the loss $\|Ax^{(q)} - b\|_2^2$ to a given threshold r increases the predicate complexity to 1. The upper bound $O(\|P\|_0 \cdot q \log m)$ now follows from Theorem 3.3.

Acknowledgments

We thank Sebastien Bubeck for helpful discussions on statistical learning, and the anonymous reviewers for useful comments. This work was supported in part by NSF TRIPODS program (award DMS-2022448); Simons Investigator Award; GIST-MIT Research Collaboration grant; MIT-IBM Watson collaboration.

References

[AB09] Martin Anthony and Peter L Bartlett, *Neural network learning: Theoretical foundations*, cambridge university press, 2009.

[ACC+11] Nir Ailon, Bernard Chazelle, Kenneth L Clarkson, Ding Liu, Wolfgang Mulzer, and C Seshadhri, *Self-improving algorithms*, SIAM Journal on Computing **40** (2011), no. 2, 350–375.

[ALN20] Nir Ailon, Omer Leibovich, and Vineet Nair, *Sparse linear networks with a fixed butterfly structure: Theory and practice*, arXiv preprint arXiv:2007.08864 (2020).

[Bal20] Maria-Florina Balcan, *Data-driven algorithm design*, Beyond Worst Case Analysis of Algorithms (Tim Roughgarden, ed.), Cambridge University Press, 2020.

[BDD+21] Maria-Florina Balcan, Dan DeBlasio, Travis Dick, Carl Kingsford, Tuomas Sandholm, and Ellen Vitercik, *How much data is sufficient to learn high-performing algorithms? generalization guarantees for data-driven algorithm design*, STOC, 2021.

[BDMI14] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail, *Near-optimal column-based matrix reconstruction*, SIAM Journal on Computing **43** (2014), no. 2, 687–717.

[BDSV18] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik, *Learning to branch*, International conference on machine learning, PMLR, 2018, pp. 344–353.

[BHM00] William L Briggs, Van Emden Henson, and Steve F McCormick, *A multigrid tutorial*, SIAM, 2000.

[BNVW17] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White, *Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems*, Conference on Learning Theory, PMLR, 2017, pp. 213–274.

[BPSV21] Maria-Florina Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik, *Sample complexity of tree search configuration: Cutting planes and beyond*, NeurIPS, 2021.

[BSV18] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik, *A general theory of sample complexity for multi-item profit maximization*, Proceedings of the 2018 ACM Conference on Economics and Computation, 2018, pp. 173–174.

[BSV20] _____, *Refined bounds for algorithm configuration: The knife-edge of dual class approximability*, International Conference on Machine Learning, PMLR, 2020, pp. 580–590.

[Csa76] L Csanky, *Fast parallel matrix inversion algorithms*, SIAM Journal on Computing **5** (1976), no. 4, 618–623.

[CW09] Kenneth L Clarkson and David P Woodruff, *Numerical linear algebra in the streaming model*, Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 205–214.

[CW13] _____, *Low rank approximation and regression in input sparsity time*, Proceedings of the forty-fifth annual ACM symposium on Theory of Computing, 2013, pp. 81–90.

[DRVW06] Amit Deshpande, Luis Rademacher, Santosh S Vempala, and Grant Wang, *Matrix approximation and projective clustering via volume sampling*, Theory of Computing **2** (2006), no. 1, 225–247.

[GJ95] Paul W Goldberg and Mark R Jerrum, *Bounding the vapnik-chervonenkis dimension of concept classes parameterized by real numbers*, Machine Learning **18** (1995), no. 2-3, 131–148.

[GR17] Rishi Gupta and Tim Roughgarden, *A pac approach to application-specific algorithm selection*, SIAM Journal on Computing **46** (2017), no. 3, 992–1017.

[HMT11] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review **53** (2011), no. 2, 217–288.

[IVWW19] Pitor Indyk, Ali Vakilian, Tal Wagner, and David P Woodruff, *Sample-optimal low-rank approximation of distance matrices*, Conference on Learning Theory, PMLR, 2019, pp. 1723–1751.

[IVY19] Piotr Indyk, Ali Vakilian, and Yang Yuan, *Learning-based low-rank approximations*, NeurIPS, 2019.

[IWW21] Piotr Indyk, Tal Wagner, and David Woodruff, *Few-shot data-driven algorithms for low rank approximation*, NeurIPS, 2021.

[Kol06] Vladimir Koltchinskii, *Local rademacher complexities and oracle inequalities in risk minimization*, The Annals of Statistics **34** (2006), no. 6, 2593–2656.

[LGM+20] Ilay Luz, Meirav Galun, Haggai Maron, Ronen Basri, and Irad Yavneh, *Learning algebraic multigrid using graph neural networks*, ICML, 2020.

[LLV+20] Simin Liu, Tianrui Liu, Ali Vakilian, Yulin Wan, and David P Woodruff, *Learning the positions in countsketch*, arXiv preprint arXiv:2007.09890 (2020).

[LT13] Michel Ledoux and Michel Talagrand, *Probability in banach spaces: Isoperimetry and processes*, Springer Science & Business Media, 2013.

[Mah11] Michael W Mahoney, *Randomized algorithms for matrices and data*, arXiv preprint arXiv:1104.5557 (2011).

[MM15] Cameron Musco and Christopher Musco, *Randomized block krylov methods for stronger and faster approximate singular value decomposition*, NeurIPS (2015).

[MT20] Per-Gunnar Martinsson and Joel A Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica **29** (2020), 403–572.

[MV20] Michael Mitzenmacher and Sergei Vassilvitskii, *Algorithms with predictions*, Beyond Worst Case Analysis of Algorithms (Tim Roughgarden, ed.), Cambridge University Press, 2020.

[RST10] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert, *A randomized algorithm for principal component analysis*, SIAM Journal on Matrix Analysis and Applications **31** (2010), no. 3, 1100–1124.

[Sar06] Tamas Sarlos, *Improved approximation algorithms for large matrices via random projections*, 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), IEEE, 2006, pp. 143–152.

[Ste15] Ingo Steinwart, *Measuring the capacity of sets of functions in the analysis of erm*, Measures of Complexity, Springer, 2015, pp. 217–233.

[TYUC19] Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher, *Streaming low-rank matrix approximation with an application to scientific simulation*, SIAM Journal on Scientific Computing **41** (2019), no. 4, A2430–A2463.

[War68] Hugh E Warren, *Lower bounds for approximation by nonlinear manifolds*, Transactions of the American Mathematical Society **133** (1968), no. 1, 167–178.

[WC15] Rafi Witten and Emmanuel Candes, *Randomized algorithms for low-rank matrix factorizations: sharp performance bounds*, Algorithmica **72** (2015), no. 1, 264–281.

[Woo14] David P Woodruff, *Sketching as a tool for numerical linear algebra*, Theoretical Computer Science **10** (2014), no. 1-2, 1–157.

A Proof of the Goldberg-Jerrum Framework (Theorem 3.3)

In this section we prove Theorem 3.3, the main theorem of the GJ framework presented in Section 3. The proof is taken from [GJ95] with slight modifications, and we include it here for completeness.

We start with an intermediate result about polynomial formulas.

Definition A.1. *A* polynomial formula $f : \mathbb{R}^n \to \{True, False\}$ *is a DNF formula over boolean predicates of the form* $P(x_1, \ldots, x_n) \geq 0$*, where* P *is a polynomial in the real-valued inputs to* f*.*

Theorem A.2. *Using the notation of Section 2.1, suppose that each algorithm* $L \in \mathcal{L}$ *is specified by* n *real parameters. Suppose that for every* $x \in \mathcal{X}$ *and* $r \in \mathbb{R}$ *there is a polynomial formula* $f_{x,r}$ *over* n *variables, that given* $L \in \mathcal{L}$ *checks whether* $L(x) > r$*. Suppose further that* $f_{x,r}$ *has* p *distinct polynomials in its predicates, and each them has degree at most* Δ*. Then, the pseudo-dimension of* $\mathcal{L}$ *is* $O(n \log(\Delta p))$*.*

Proof. Let $x_1, \ldots, x_N$ be a shattered set of instances, and $r_1, \ldots, r_N$ the corresponding loss thresholds. We need to show $N = O(n \log(\Delta p))$.

For every $i = 1, \ldots, N$, let us denote $f_i = f_{x_i, r_i}$ for brevity. Then, for a given $L \in \mathcal{L}$, f_i checks whether $L(x_i) > r_i$. Thus, $x_1, \ldots, x_N$ being a shattered set means that the family of N-dimensional boolean strings $\{f_1(L), \ldots, f_N(L) : L \in \mathcal{L}\}$ has size 2^N.

The truth value of each f_i is determined by the signs of p different polynomials in n variables, each of degree at most Δ. Therefore, a boolean string $f_1(L), \ldots, f_N(L)$ is determined by the signs of at most pN different polynomials in n variables, each of degree at most Δ. We now use the following classical theorem,

Theorem A.3 ([War68])**.** *Suppose* $N \geq n$*. Then* N *polynomials in* n *variables, each of degree at most* Δ*, take at most* $O(N\Delta/n)^n$ *different sign patterns.*

In our case, if $N \leq n$ then the conclusion $N = O(n \log(\Delta p))$ is trivial, thus we may assume $N > n$ and hence $pN \geq n$. Now by Warren's theorem, $2^N = \{f_1(L), \ldots, f_N(L) : L \in \mathcal{L}\} = O(pN\Delta/n)^n$, and by solving for N we get $N = O(n \log(\Delta p))$ as desired. □

Now we can prove Theorem 3.3. We can describe $\Gamma_{x,r}$ by a computation tree, where arithmetic operations correspond to nodes with one child, conditional statements correspond to nodes with two children, and leaves correspond to output values. We can transform the tree into a polynomial DNF formula (as defined in the previous section) that checks whether $L(x) > r$, by ORing the ANDs of the conditional statements along each root-to-leaf computation path that ends with a "true" leaf.

Each branching node in the tree (i.e., a node with two children) corresponds to a conditional statement in $\Gamma_{x,r}$, which has the form of a rational predicate, meaning it determines whether $R \geq 0$ for some rational function R of the inputs of $\Gamma_{x,r}$. It can be easily checked that we can replace each such rational predicate

with $O(1)$ polynomials predicates (that determine whether $P \geq 0$ for some polynomial in the inputs of $\Gamma_{x,r}$) of degree no larger than that of R. Since $\Gamma_{x,r}$ has predicate complexity p, the obtained formula has at most $O(p)$ distinct polynomials in its predicates. Since $\Gamma_{x,r}$ has degree Δ, each of these polynomials has degree at most Δ. Theorem 3.3 now follows from Theorem A.2.

B Omitted Proofs from Sections 4 and 5

B.1 Proof of Fact 4.1

If $w^T A = 0$ then SCW returns a zero matrix, whose loss is $\|A\|_F^2$, and the statement holds. Now assume $w^T A \neq 0$. We go over the steps of SCW (Algorithm 1):

- Compute the row vector $w^T A$.
- Compute the SVD of $w^T A$. Note that for any row vector z^T, its SVD $U\Sigma V^T$ is given by U being a 1×1 matrix whose only entry is 1, Σ being a 1×1 matrix whose only entry is $\|z\|$, and V^T being the row vector $\frac{1}{\|z\|} z^T$. So for $z^T = w^T A$ we have $V^T = \frac{1}{\|A^T w\|} w^T A$.
- Compute $[AV]_1$, the best rank-1 approximation of AV. But in our case AV equals $\frac{1}{\|A^T w\|} AA^T w$, which is already rank 1, so its best rank-1 approximation is itself, $[AV]_1 = \frac{1}{\|A^T w\|} AA^T w$.
- Return $[AV]_1 V^T$, which in our case equals $\frac{1}{\|A^T w\|^2} AA^T w w^T A$.

So, the output rank-1 matrix of SCW is $\frac{1}{\|A^T w\|^2} AA^T w w^T A$, and its loss is as stated.

B.2 Proof of Lemma 5.1

Write the SVD of $A(SA)^\dagger(SA)$ as $U\Sigma V^T$. Note that $(SA)^\dagger(SA)$ is the orthogonal projection on the row-space of SA, in which every vector is a linear combination of the rows of A. Therefore, projecting the rows of A onto the row-space of SA spans all of the row-space of SA, or in other words, the row-spans of $A(SA)^\dagger(SA)$ and of SA are the same. Consequently, the rows of V^T form an orthonormal basis for the row-space of SA. This means in particular that VV^T is the orthogonal projection on the row-space of SA, thus $(SA)^\dagger(SA) = VV^T$, so we can write the matrix from the lemma statement as $[A(SA)^\dagger(SA)]_k = [AVV^T]_k$.

We recall that SCW returns the best rank-k approximation of A in the row-space of SA (see [Woo14]), meaning it returns ZV^T where Z is a rank-k matrix that minimizes $\|A - ZV^T\|_F^2$. So, we need to show that

$$\forall\ Z \ \text{ of rank } k, \quad \|A - [AVV^T]_k\|_F^2 \leq \|A - ZV^T\|_F^2. \tag{6}$$

We use the following observation.

Claim B.1. $[AVV^T]_k VV^T = [AVV^T]_k$.

Proof. Recall that $A(SA)^\dagger(SA) = AVV^T$, thus the SVD of AVV^T is $U\Sigma V^T$, thus $[AVV^T]_k = U_k \Sigma_k V_k^T$. Since $V_k^T VV^T = V_k^T$ (projecting a subset of k rows of V^T onto the row-space of V^T does not change anything), we have $[AVV^T]_k VV^T = U_k \Sigma_k V_k^T VV^T = U_k \Sigma_k V_k^T = [AVV^T]_k$. □

Proceeding with the proof of Lemma 5.1, we now have for every Z of rank k,

$$\begin{aligned}\|A - [AVV^T]_k\|_F^2 &= \|AVV^T - [AVV^T]_k VV^T\|_F^2 + \|A(I - VV^T) - [AVV^T]_k(I - VV^T)\|_F^2 \\ &= \|AVV^T - [AVV^T]_k\|_F^2 + \|A - AVV^T\|_F^2 \\ &\leq \|AVV^T - ZV^T\|_F^2 + \|A - AVV^T\|_F^2 \\ &= \|A - ZV^T\|_F^2,\end{aligned}$$

where the first and last equalities are the Pythagorean identity (orthogonally projecting onto and against VV^T), the second equality is by Claim B.1, and the inequality is by the optimality of $[AVV^T]_k$ as a rank-k approximation of AVV^T (since Z has rank k). This proves Equation (6), proving the lemma.

B.3 Proof of Theorem 5.4

In this section we explain how to read the statement of Theorem 5.4 from [MM15]. All section, theorem and page numbers refer to the arXiv version of their paper.[6]

The relevant result in [MM15] is the Frobenius-norm analysis of their Algorithm 1 (which they call "Simultaneous Iteration"), stated in their Theorem 11. As they state in their Section 5.1, for the purpose of low-rank approximation in the Frobenius norm, it suffices to return $\mathbf{Q}$ instead of $\mathbf{Z}$ (in the notation of their Algorithm 1). Thus, steps 4–6 of Algorithm 1 can be skipped. Since (again in the notation of their Algorithm 1) $\mathbf{Q}$ is the result of orthonormalizing the columns of $\mathbf{K}$, we clearly have $\mathbf{K}\mathbf{K}^\dagger = \mathbf{Q}\mathbf{Q}^\dagger$ (this is because $\mathbf{M}\mathbf{M}^\dagger$ is the projection matrix on the column space of any matrix $\mathbf{M}$, and $\mathbf{K}$ and $\mathbf{Q}$ have the same column spaces). Since for our purpose we only need the projection matrix $\mathbf{Q}\mathbf{Q}^\dagger$ (rather than the orthonormal basis $\mathbf{Q}$), we can also skip step 3, and simply use the matrix $\mathbf{K}$ as the output of their Algorithm 1, while maintaining the guarantee of their Theorem 11 (with $\mathbf{Z}\mathbf{Z}^T$ replaced by $\mathbf{K}\mathbf{K}^\dagger$).

Next, while their Algorithm 1 chooses the initial matrix $\mathbf{\Pi}$ as a random gaussian matrix, they state in their Section 5.1 that their analysis (and in particular, their Theorem 11) holds for every initial matrix $\mathbf{\Pi}$ that satisfies the guarantee of their Lemma 4, which is equivalent to satisfying our Equation (1).

Finally, note that they set the number of iterations in step 1 of their Algorithm 1 to $q = O(\epsilon^{-1}\log(d))$, while we set it in Theorem 5.4 to $q = O(\epsilon^{-1}\log(d/\epsilon))$. This is because in their setting they may assume w.l.o.g. that $\epsilon^{-1} = \mathrm{poly}(d)$ (see bottom of their page 10), while in our setting this is not the case.

Altogether, the statement of Theorem 5.4 follows.

B.4 Proof of Lemma 5.5

For completeness, we discuss two ways to prove the lemma.

Proof 1. We start by noting that a significantly stronger version of Lemma 5.5 follows from Theorem 1.3 of [DRVW06]. Slightly rephrasing (and transposing) their theorem, it states that for every matrix $B \in \mathbb{R}^{n\times d}$, there is a distribution (that depends on B) over matrices $P \in \mathbb{R}^{d\times k}$ whose columns are standard basis vectors, such that if we let $\tilde{B}_k$ be the projection of the columns of B onto the columns-space of BP, namely $\tilde{B}_k = (BP)(BP)^\dagger B$, then we have

$$\mathbb{E}_P\|B - \tilde{B}_k\|_F^2 \leq (k+1)\cdot\|B - [B]_k\|_F^2.$$

In particular, there exists a supported P that satisfies this equation without the expectation, yielding Lemma 5.5.

[6] https://arxiv.org/abs/1504.05477v4.

Remark. Note that this is in fact a quantitatively stronger version of Lemma 5.5, since the $O(kd)$ term in Equation (1) is replaced here by $(k+1)$, which [DRVW06] furthermore show is the best possible. However, this improvement does not strengthen the final bounds we obtain in our theorems. This is because the analysis of the power method (Theorem 5.4) requires the $\log d$ term in the number of iterations q even if the initial approximation (Equation (1)) is up to a factor of $(k+1)$ instead of $O(kd)$. Technically, this stems from the analysis in [MM15]: In the equation immediately after their eq. (4) on page 11, the $\log d$ term is needed not just to eliminate d from the numerator, but also to gain a $d^{O(1)}$ term in the denominator.

Proof 2. Since the aforementioned result of [DRVW06] is difficult and stronger than we require, we now also give a more basic proof of Lemma 5.5, for completeness.

Let $B \in \mathbb{R}^{n\times d}$ be written in SVD form as $B = U\Sigma V^T$. Let V_k^T be the matrix with the top k rows of V^T, and V_{-k}^T the matrix with the remaining rows. We use the following fact from [Woo14], which originates in [BDMI14] and is also used in [MM15] (see their Lemma 14).

Lemma B.2. *Let $P \in \mathbb{R}^{d\times k}$ be any matrix such that the $k \times k$ matrix $V_k^T P$ has rank k. Then,*

$$\|B - (BP)(BP)^\dagger B\|_F^2 \le \|B - [B]_k\|_F^2 \cdot \left(1 + \|V_{-k}^T P\|_2^2 \cdot \|(V_k^T P)^\dagger\|_2^2\right).$$

We will construct a matrix $P \in \mathbb{R}^{d\times k}$ whose columns are distinct standard basis vectors of $\mathbb{R}^d$. Since it would thus have orthonomal columns, as does V_{-k}, we would have $\|V_{-k}^T P\|_2^2 \le \|V_{-k}^T\|_2^2 \cdot \|P\|_2^2 = 1$. Therefore, using Lemma B.2, for P to satisfy Equation (1) and thus prove Lemma 5.5, it suffices to construct it such that $V_k^T P$ has rank k and satisfies $\|(V_k^T P)^\dagger\|_2^2 \le d$. This is equivalent to showing that the smallest of the k singular values of $V_k^T P$ is at least $1/\sqrt{d}$, which is what we do in the rest of the current proof.

We construct P by the following process. Initialize $Z_1 \in \mathbb{R}^{k\times d}$ as $Z_1 \leftarrow V_k^T$, and a zero matrix $P \in \mathbb{R}^{d\times k}$. For $i = 1, \ldots, k$:

1. Let $z_1^i, \ldots, z_d^i$ denote the columns of Z_i.
2. Let $j_i \leftarrow \operatorname{argmax}_{j\in[d]} \|z_j^i\|_2^2$.
3. Set column i of P to be e_{j_i}.
4. Let Π_i be the orthogonal projection matrix against $\operatorname{span}(v_{j_1}, \ldots, v_{j_i})$.
5. $Z_{i+1} \leftarrow \Pi_i V_k^T$.

Let $v_1, \ldots, v_d$ denote the columns of V_k^T. Observe that the columns of $V_k^T P$ are $v_{j_1}, \ldots, v_{j_k}$.

Claim B.3. *For every $i = 1, \ldots, k$ we have $\|z_{j_i}^i\|_2^2 \ge \frac{k-i+1}{d}$.*

Proof. Let $i \in [k]$. Since V_k^T has orthonormal rows, each of its k singular values equals 1, and any best rank-$(i-1)$ approximation of it, $[V_k^T]_{i-1}$, satisfies $\|V_k^T - [V_k^T]_{i-1}\|_F^2 = k - (i-1)$. Since Π_{i-1} is a projection against $i-1$ directions, $(I - \Pi_{i-1})V_k^T$ can be viewed as a rank-$(i-1)$ approximation of V_k^T, and therefore

$$\|Z_i\|_F^2 = \|\Pi_{i-1} V_k^T\|_F^2 = \|V_k^T - (I - \Pi_{i-1})V_k^T\|_F^2 \ge \|V_k^T - [V_k^T]_{i-1}\|_F^2 = k - i + 1.$$

Hence the average squared-ℓ_2 mass of columns in Z_i is at least $\frac{k-i+1}{d}$, and hence the column with maximal ℓ_2-norm — which we recall is defined to be $z_{j_i}^i$ — has squared ℓ_2-norm at least $\frac{k-i+1}{d}$. □

Claim B.4. *For every $i = 1, \ldots, k$, $z_{j_i}^i$ is spanned by $v_{j_1}, \ldots, v_{j_i}$.*

Proof. Observe that $z^i_{j_i} = \Pi_{i-1} v_{j_i}$ (with the convention that Π_0 is the identity). Therefore, $z^i_{j_i} = v_{j_i} - (I - \Pi_{i-1})v_{j_i}$, and the claim follows by noting that $I - \Pi_{i-1}$ is the orthogonal projection onto the subspace spanned by $v_{j_1}, \ldots, v_{j_{i-1}}$. □

Claim B.5. *$z^1_{j_1}, \ldots, z^k_{j_k}$ are pairwise orthogonal.*

Proof. Let $i < i'$. By Claim B.4 $z^i_{j_i}$ is spanned by $v_{j_1}, \ldots, v_{j_i}$. At the same time, $z^{i'}_{j_{i'}}$ is a column of the matrix $Z_{i'}$, whose columns have been orthogonally projected against $v_{j_1}, \ldots, v_{j_{i'-1}}$, a set that contains $v_{j_1}, \ldots, v_{j_i}$. Thus $z^{i'}_{j_{i'}}$ is orthogonal to $z^i_{j_i}$. □

Claim B.6. *Let $i \in [k]$. Then v_{j_i} can be written uniquely as a linear combination of $z^1_{j_1}, \ldots, z^i_{j_i}$, such that the coefficient of $z^i_{j_i}$ is* 1.

Proof. We have shown in Claim B.5 that $z^1_{j_1}, \ldots, z^k_{j_k}$ are orthogonal and in Claim B.3 that each is non-zero, so they form a basis of $\mathbb{R}^k$. Thus v_{j_i} is written uniquely as a linear combination of them. As noted in the proof of Claim B.4, we have $z^i_{j_i} = \Pi_{i-1} v_{j_i} = v_{j_i} - (I - \Pi_{i-1})v_{j_i}$, or equivalently $v_{j_i} = z^i_{j_i} + (I - \Pi_{i-1})v_{j_i}$.

We recall that $(I - \Pi_{i-1})$ is the orthogonal projection onto the subspace $W = \mathrm{span}(v_{j_1}, \ldots, v_{j_{i-1}})$, whose dimension is at most $i - 1$. By Claim B.4, W contains $z_{j_1}, \ldots, z_{j_{i-1}}$. Since $z_{j_1}, \ldots, z_{j_k}$ form a basis of $\mathbb{R}^k$, we now get that W is spanned by $z_{j_1}, \ldots, z_{j_{i-1}}$, and cannot contain any $z_{j_{i'}}$ with $i' \geq i$. In particular, $(I - \Pi_{i-1})v_{j_i}$ is written uniquely as a linear combination of $z_{j_1}, \ldots, z_{j_{i-1}}$. Recalling that $v_{j_i} = z^i_{j_i} + (I - \Pi_{i-1})v_{j_i}$, the claim follows. □

We can finally complete the proof of Lemma 5.5. For every $i \in [k]$, let q_i denote the unit-length vector in the direction of $z^i_{j_i}$, i.e., $q_i = \frac{1}{\|z^i_{j_i}\|} z^i_{j_i}$. Let $Q \in \mathbb{R}^{k \times k}$ be the matrix whose columns are $q_1, \ldots, q_k$. Claim B.6 means that we can write $V_k^T P$ (since its columns, we recall, are $v_{j_1}, \ldots, v_{j_k}$) as $V_k^T P = QR$, where R is an upper triangular matrix, whose diagonal entries are $\|z^i_{j_i}\|$ for $i = 1, \ldots, k$. Since Q has orthonormal columns (from Claim B.5 and the fact that we normalized its columns), this is a QR-decomposition of $V_k^T P$. Hence, $V_k^T P$ and R have the same singular values. The diagonal entries of R are its eigenvalues, which are also its singular values as all are non-negative. By Claim B.3, each diagonal entry of R is at least $1/\sqrt{d}$. This implies the same lower bound on the smallest singular value of R and thus of $V_k^T P$, which as explained earlier, implies Lemma 5.5.

C Proof of the Lower Bound

In this section we prove the lower bound in Theorem 2.2, restated next.

Theorem C.1. *For every $s \leq k \leq m$ and $\epsilon \in (0, \frac{1}{2\sqrt{k}})$, the ϵ-fat shattering dimension of IVY with target low rank k, sketching dimension m and sparsity s is $\Omega(ns)$.*

Proof. We may assume w.l.o.g. that $m = k$, since we can always augment a sketching matrix with k rows with $m - k$ additional zero rows, without changing the result of SCW.

We start with the special case $s = k$, where the sketching matrix S is allowed to be fully dense, and the desired lower bound is $\Omega(nk)$. Let A_0 be the $n \times k$ matrix whose i-th column is e_i (the standard basis vector) for all $i = 1 \ldots k$. For every $i \in \{1 \ldots k\}$ and $t \in \{k+1 \ldots n\}$, Let $A_{(i,t)}$ be given by replacing the i-th column of A_0 with e_t. Observe that each $A_{(i,t)}$ has rank k, and each of its singular values equals 1. We will show that the set of matrices $Z = \{A_{(i,t)} : i = 1 \ldots k,\ t = k+1 \ldots n\}$ is shattered, which would imply the lower bound since its size is $k(n-k) = \Omega(nk)$.

Let $Z' \subset Z$. It suffices to exhibit a sketching matrix $S \in \mathbb{R}^{k \times n}$ (with unbounded sparsity, since $s = k$) such that for every $A_{(i,t)}$, the SCW loss of using S for a rank-k approximation of A is 0 if $A_{(i,t)} \in Z'$,

and at least 1 otherwise. We set the first k columns of S to be the order-k identity matrix. Then, for every $t = k+1, \ldots, n$, let $J_t = \{i \in 1 \ldots k : A_{(i,t)} \in Z'\}$. In the t-th column of S, we put 1's in rows J_t and 0's in the remaining rows.

Fix $A_{(i,t)}$. Recall that SCW returns the best rank-k approximation of A in the row-space of $SA_{(i,t)}$, which is a $k \times k$ matrix.

- Suppose $A_{(i,t)} \notin Z'$. In this case, we argue that the i-th row of $SA_{(i,t)}$ is zero. This implies that the row-space of $SA_{(i,t)}$ has dimension at most $k-1$, so SCW incurs loss at least 1.

  Indeed, row i of $SA_{(i,t)}$ equals the sum of rows j of $A_{(i,t)}$ for which $S(i,j) = 1$. The only non-zero rows in $A_{(i,t)}$, by construction, are rows $\{1...k\} \setminus \{i\}$ and row t. Since the first k columns of S are the identity, $S(i,j) = 0$ for every $j \in \{1...k\} \setminus \{i\}$. Since $i \notin J_t$, then by construction of S we have $S(i,t) = 0$. Overall, the i-th row of $SA_{(i,t)}$ is zero.

- Suppose $A_{(i,t)} \in Z'$. In this case, we argue that the row-space of $SA_{(i,t)}$ has dimension k. Since the ambient row-dimension is also k, this means SCW returns a perfect rank-k approximation, i.e., zero loss.

  Indeed, again, the only non-zero rows of $A_{(i,t)}$ are rows $\{1...k\} \setminus \{i\}$ and row t. But in this case $i \in J_t$, thus $S(i,t) = 1$ and thus row i of $SA_{(i,t)}$ equals e_i. For every $j \in \{1...k\} \setminus \{i\}$, row j of $SA_{(i,t)}$ equals either $e_j + e_i$ (if $j \in J_t$) or just e_j (if $j \notin J_t$). It is thus clear that the rows of $SA_{(i,t)}$ span all of $\mathbb{R}^k$.

In conclusion, the SCW loss is 0 on matrices in Z' at least 1 on matrices in $Z \setminus Z'$.

Next we extend it to an $\Omega(ns)$ lower bound for any $s = 1, \ldots, k$. We assume for simplicity that s is an integer divisor of k and that k is an integer divisor of ns.

We partition an input matrix $A \in \mathbb{R}^{n \times k}$ into k/s diagonal blocks of order $(ns/k) \times s$ each (everything outside the diagonal blocks is zero). To construct the shattered set, we first set each diagonal block in A to $A_0 \in \mathbb{R}^{(ns/k) \times s}$ (as defined above), then choose one "critical" block $b \in \{1, \ldots, k\}$ and replace its A_0 with $A_{(i,t)}$ as defined above, with $i \in \{1, \ldots, s\}$ and $t \in \{s+1, \ldots, ns/k\}$. Denote the resulting matrix by $A_{(b,i,t)}$. The total size of the shattered set $\{A_{(b,i,t)}\}$ is $\frac{k}{s} \cdot s \cdot (\frac{ns}{k} - s) = (n-k)s = \Omega(ns)$. The corresponding sketching matrix, arising from the dense construction above, has block-diagonal structure with blocks of order $s \times (ns/k)$ each, and in particular has at most s nonzeros per column, as needed.

The correctness of the construction follow immediately from the dense case. In more detail, let Z'' be a subset of the shattered set $\{A_{(b,i,t)}\}$. Fix $A_{(b,i,t)}$ (not necessarily in Z''). The $k \times k$ matrix $SA_{(b,i,t)}$ has block-diagonal structure with blocks of order $s \times s$. Each non-critical block equals the order-s identity, while the critical block, by the proof of the dense case above, equals the order-s identity if $A_{(b,i,t)} \in Z''$ and has rank at most $s-1$ otherwise. Therefore, if $A_{(b,i,t)} \in Z''$ then SCW with sketching matrix S finds a zero-loss rank-k approximation of every $A_{(b,i,t)} \in Z''$, and incurs loss at least 1 for every $A_{(b,i,t)} \notin Z''$.

Finally, recall that we need to normalize our matrices to have squared Frobenius norm 1. In the above construction it is instead k, so we need to divide each entry by $1/\sqrt{k}$. This means that the loss is zero for matrices in the shattered set, and is at least $1/\sqrt{k}$ for matrices outside the shattered set. Thus, the set is ϵ-fat shattered for every $\epsilon \in (0, \frac{1}{2\sqrt{k}})$. □

D Proof of Theorem 2.3

D.1 Uniform Convergence and ERM Upper Bound

By classical results in learning theory, the fat shattering dimension can be used to obtain an upper bound on the number of samples needed for uniform convergence and for ERM learning (see for example Theorem

19.1 in [AB09]), and we could simply plug Theorem 2.2 into these results. However, we can get a sharper bound by exploiting the proxy loss from Section 5.3, since for the latter we have an upper bound on the pseudo-dimension, which yields sharper bounds for ERM learning than the fat shattering dimension.

Let $\ell_{\mathrm{IVY}} = \ell_{\mathrm{IVY}}(\epsilon, \delta)$ be the number of samples required for (ϵ, δ)-uniform convergence for the family IVY losses, $\mathcal{L}_{\mathrm{IVY}} = \{L_k^{\mathrm{SCW}}(S, \cdot) : \mathcal{A} \to [0,1]\}_{S \in \mathcal{S}}$. Let $\hat{\ell} = \hat{\ell}(\epsilon, \delta)$ be the number of samples required for (ϵ, δ)-uniform convergence for the family of proxy losses, $\hat{\mathcal{L}}_\epsilon = \{\hat{L}_{k,\epsilon}(S, \cdot) : \mathcal{A} \to [0,1]\}_{S \in \mathcal{S}}$. Fix $\hat{\ell}$ matrices $A_1, \ldots, A_{\hat{\ell}} \in \mathcal{A}$. Lemma 5.7 implies

$$\forall\, S \in \mathcal{S}, \quad \left| \frac{1}{\hat{\ell}} \sum_{i=1}^{\hat{\ell}} \hat{L}_{k,\epsilon}(S, A_i) - \frac{1}{\hat{\ell}} \sum_{i=1}^{\hat{\ell}} L_k^{\mathrm{SCW}}(S, A_i) \right| \leq \epsilon$$

and

$$\forall\, S \in \mathcal{S}, \quad \left| \mathbb{E}_{A \sim \mathcal{D}}[\hat{L}_{k,\epsilon}(S, A)] - \mathbb{E}_{A \sim \mathcal{D}}[L_k^{\mathrm{SCW}}(S, A)] \right| \leq \epsilon.$$

Let $\mathcal{D}$ be a distribution over $\mathcal{A}$. By definition of $\hat{\ell}$, with probability at least $1 - \delta$ over sampling $A_1, \ldots, A_{\hat{\ell}}$ independently from $\mathcal{D}$, we have

$$\forall\, S \in \mathcal{S}, \quad \left| \frac{1}{\hat{\ell}} \sum_{i=1}^{\hat{\ell}} \hat{L}_{k,\epsilon}(S, A_i) - \mathbb{E}_{A \sim \mathcal{D}}[\hat{L}_{k,\epsilon}(S, A)] \right| \leq \epsilon,$$

which combined with the above, implies

$$\forall\, S \in \mathcal{S}, \quad \left| \frac{1}{\hat{\ell}} \sum_{i=1}^{\hat{\ell}} L_k^{\mathrm{SCW}}(S, A_i) - \mathbb{E}_{A \sim \mathcal{D}}[L_k^{\mathrm{SCW}}(S, A)] \right| \leq 3\epsilon.$$

Consequently, $\ell_{\mathrm{IVY}}(3\epsilon, \delta) \leq \hat{\ell}(\epsilon, \delta)$. By classical results in learning theory (see, e.g., Theorem 3.2 in [GR17]), the sample complexity can be upper bounded using the pseudo-dimension, and in particular, $\hat{\ell}(\epsilon, \delta) = O(\epsilon^{-2} \cdot (\mathrm{pdim}(\hat{\mathcal{L}}_\epsilon) + \log(1/\delta)))$. By Equation (4) in Section 5.3, $\mathrm{pdim}(\hat{\mathcal{L}}_\epsilon) = O(ns \cdot (m + k \log(d/k) + \log(1/\epsilon)))$. Putting everything together,

$$\ell_{\mathrm{IVY}}(3\epsilon, \delta) = O(\epsilon^{-2} \cdot (ns \cdot (m + k \log(d/k) + \log(1/\epsilon)) + \log(1/\delta))).$$

As a consequence, that many samples suffice for $(6\epsilon, \delta)$-learning IVY with ERM. The upper bound in Theorem 2.3 follows by scaling ϵ down by a constant.

D.2 Uniform Convergence Lower Bound

We proceed to the lower bound on the number of samples needed for IVY to admit (ϵ, ϵ)-uniform convergence. It relies on some known results about the connection between uniform convergence and the fat shattering dimension, which we now detail.

We introduce some notation. Let $\mathcal{L}$ be a class of functions $L : \mathcal{X} \to [0,1]$, and let $\mathcal{D}$ be a distribution over $\mathcal{X}$. For every $L \in \mathcal{L}$, denote its expected loss over $\mathcal{D}$ by

$$z(L) := \mathbb{E}_{x \sim \mathcal{D}}[L(x)].$$

Given ℓ i.i.d. samples $(x_1, \ldots, x_\ell) \sim \mathcal{D}^\ell$, denote the empirical loss over the samples by

$$\hat{z}_\ell(L) := \frac{1}{\ell} \sum_{i=1}^{\ell} L(x_i).$$

Suppose $\mathcal{L}$ admits (ϵ, ϵ)-uniform convergence with ℓ samples. This can now be written as

$$\Pr_{\mathcal{D}^\ell}\left[\sup_{L\in\mathcal{L}} |\hat{z}_\ell(L) - z(L)| \le \epsilon\right] \ge 1 - \epsilon. \tag{7}$$

Our goal for this section is to prove a lower bound on ℓ. We begin the proof. Equation (7) implies

$$\mathbb{E}_{\mathcal{D}^\ell}\left[\sup_{L\in\mathcal{L}} |\hat{z}_\ell(L) - z(L)|\right] \le 2\epsilon. \tag{8}$$

This expectation can be bounded using Rademacher averages. Define the centered class $\mathcal{L}_c$ of $\mathcal{L}$ as

$$\mathcal{L}_c := \{L - z(L) : L \in \mathcal{L}\}.$$

For the given sample $(x_1, \ldots, x_\ell)$, define the Rademacher average of $\mathcal{L}_c$ as

$$\mathrm{Rad}_\ell(\mathcal{L}_c) := \mathbb{E}_{\sigma_1,\ldots,\sigma_\ell}\left[\sup_{L'\in\mathcal{L}_c}\left|\frac{1}{\ell}\sum_{i=1}^{\ell}\sigma_i L'(x_i)\right|\right] = \mathbb{E}_{\sigma_1,\ldots,\sigma_\ell}\left[\sup_{L\in\mathcal{L}}\left|\frac{1}{\ell}\sum_{i=1}^{\ell}\sigma_i\left(L(x_i) - z(L)\right)\right|\right],$$

where each of $\sigma_1, \ldots, \sigma_\ell$ is independently uniform in $\{1, -1\}$. The desymmetrization inequality for Rademacher processes (see [Kol06]) states that

$$\mathbb{E}_{\mathcal{D}^\ell}\left[\sup_{L\in\mathcal{L}} |\hat{z}_\ell(L) - z(L)|\right] \ge \frac{1}{2}\mathbb{E}_{\mathcal{D}^\ell}\left[\mathrm{Rad}_\ell(\mathcal{L}_c)\right]. \tag{9}$$

A version of Sudakov's minorization inequality for Rademacher processes (Equation (26) in [Ste15], based on Corollary 4.14 in [LT13]) states that

$$\mathrm{Rad}_\ell(\mathcal{L}_c) \ge \frac{\alpha_1}{\sqrt{\ell}} \cdot \left(\ln\left(2 + \frac{\alpha_2}{\Psi_\ell(\mathcal{L}_c)}\right)\right)^{-1/2} \cdot \sup_{\gamma>0} \gamma\sqrt{\ln \mathcal{N}_2(\gamma, \mathcal{L}, \ell)}, \tag{10}$$

where $\alpha_1, \alpha_2 > 0$ are absolute constants, $\mathcal{N}_2(\epsilon, \mathcal{L}, \ell)$ is the *covering number*, and

$$\Psi_\ell(\mathcal{L}_c) := \sup_{L'\in\mathcal{L}_c}\sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}(L'(x_i))^2}.$$

We forgo the definition of the covering number, since we only need the standard fact that it can be lower-bounded in terms of the fat shattering dimension. It is encompassed in the following claim.

Claim D.1. *Let $\gamma \in [4\epsilon, 1)$ and $\delta \in (0, \frac{1}{100})$. Suppose $\mathcal{L}$ admits (ϵ, δ)-uniform convergence with ℓ samples, and $\mathrm{fatdim}_{16\gamma}(\mathcal{L}) > 1$. Then $\ln \mathcal{N}_2(\gamma, \mathcal{L}, \ell) \ge \frac{1}{8} \cdot \mathrm{fatdim}_{16\gamma}(\mathcal{L})$.*

Proof. Let ℓ' be the number of samples required for $(\frac{1}{2}\gamma, \delta)$-learning $\mathcal{L}$ (see Section 2.1 for the definition of (ϵ, δ)-learning). Theorem 19.5 in [AB09] gives the lower bound $\ell' \ge \frac{1}{16\alpha}(\mathrm{fatdim}_{\gamma/(2\alpha)}(\mathcal{L}) - 1)$ for any $0 < \alpha < 1/4$. Setting $\alpha = 1/32$ and using $\mathrm{fatdim}_{16\gamma}(\mathcal{L}) > 1$ yields $\ell' \ge 2 \cdot \mathrm{fatdim}_{16\gamma}(\mathcal{L}) - 2 \ge \mathrm{fatdim}_{16\gamma}(\mathcal{L})$. Since (ϵ, δ)-uniform convergence implies $(2\epsilon, \delta)$-learning and thus $(\frac{1}{2}\gamma, \delta)$-learning (as $\gamma \ge 4\epsilon$), we have $\ell \ge \ell' \ge \mathrm{fatdim}_{16\gamma}(\mathcal{L})$. Now we can apply Lemma 10.5 and Theorem 12.10 from [AB09], which yield $\ln \mathcal{N}_2(\gamma, \mathcal{L}, \ell) \ge \ln \mathcal{N}_1(\gamma, \mathcal{L}, \ell) \ge \frac{1}{8} \cdot \mathrm{fatdim}_{16\gamma}(\mathcal{L})$. □

Putting together everything so far (i.e., combining Equations (8) to (10) and claim D.1), we get

$$2\epsilon \geq \frac{1}{2}\mathbb{E}_{\mathcal{D}^\ell}\left[\frac{\alpha_1}{8\sqrt{\ell}} \cdot \left(\ln\left(2+\frac{\alpha_2}{\Psi_\ell(\mathcal{L}_c)}\right)\right)^{-1/2} \cdot \sup_{\gamma\in[4\epsilon,1)} \gamma\sqrt{\mathrm{fatdim}_{16\gamma}(\mathcal{L})}\right],$$

provided that $\mathrm{fatdim}_{16\gamma}(\mathcal{L}) > 1$ (as needed for Claim D.1).

The derivations so far have been for any $\mathcal{L}$ that admits (ϵ,ϵ)-uniform convergence with ℓ samples. Now we begin specializing the arguments for IVY, that is, for $\mathcal{L} = \mathcal{L}_{\mathrm{IVY}}$. We choose $\gamma = 1/(64\sqrt{k})$. Note that the assumption in Theorem 2.3 is that $\epsilon \leq 1/(256\sqrt{k})$, ensuring that $\gamma \geq 4\epsilon$. By the lower bound in Theorem 2.2, $\mathrm{fatdim}_{16\gamma}(\mathcal{L}) = \Omega(ns)$. Plugging these above, we get

$$\epsilon \geq \Omega(1)\cdot\mathbb{E}_{\mathcal{D}^\ell}\left[\sqrt{\frac{ns}{\ell k}\cdot\left(\ln\left(2+\frac{\alpha_2}{\Psi_\ell(\mathcal{L}_c)}\right)\right)^{-1}}\right]. \tag{11}$$

It remains to bound $\Psi_\ell(\mathcal{L}_c)$ from below. We show it is lower-bounded by a constant even for very simple input distributions for IVY, supported on only two matrices (and indeed, for any distribution such that there is a loss function $L \in \mathcal{L}$ with loss 0 on half the distribution mass and loss 1 on the other half). Recall that

$$\Psi_\ell(\mathcal{L}_c) := \sup_{L'\in\mathcal{L}_c}\sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}(L'(x_i))^2} = \sup_{L\in\mathcal{L}}\sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}(L(x_i)-z(L))^2}.$$

Let $e_1,\ldots,e_n \in \mathbb{R}^n$ be the standard basis in $\mathbb{R}^n$. Let $A_0, A_0', A_1, A_1' \in \mathbb{R}^{n\times d}$ be defined as follows: the first k columns of A_0' are $e_1,\ldots,e_k$, and the rest are zero; the first k columns of A_1' are $e_{k+1},\ldots,e_{2k}$, and the rest are zero; $A_0 = \frac{1}{\sqrt{k}}A_0'$; and $A_1 = \frac{1}{\sqrt{k}}A_1'$. Note that $\|A_0\|_F^2 = \|A_1\|_F^2 = 1$. Let $\mathcal{D}$ be the uniform distribution over $\{A_0, A_1\}$. Let $L_0 \in \mathcal{L}$ be the IVY loss function induced by the sketching matrix $S_0 \in \mathbb{R}^{k\times n}$ whose rows are $e_1^T,\ldots,e_k^T$. (In the notation of the previous sections, $L_0 = L_k^{\mathrm{SCW}}(S_0,\cdot) \in \mathcal{L}_{\mathrm{IVY}}$.) It is not hard to see that $L_0(A_0) = 0$ and $L_0(A_1) = 1$. In particular, $z(L_0) = \frac{1}{2}$. Therefore, for any sample $x_1,\ldots,x_\ell$ from $\mathcal{D}$,

$$\Psi_\ell(\mathcal{L}_c) \geq \sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}(L_0(x_i)-z(L_0))^2} = \sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}\left(\frac{1}{2}\right)^2} = \frac{1}{2}.$$

The proof is now easily completed. Plugging the above bound on $\Psi_\ell(\mathcal{L}_c)$ into Equation (11) yields $\epsilon \geq \Omega(1)\cdot\mathbb{E}_{\mathcal{D}^\ell}\left[\sqrt{ns/(\ell k)}\right]$. Since n, s, ℓ, k are constants w.r.t. the sample from D^ℓ, we can dispense with the expecation and write $\epsilon \geq \Omega(\sqrt{ns/(\ell k)})$. Rearranging yields $\ell \geq \Omega(\epsilon^{-2}ns/k)$.

D.3 General Learning Lower Bound

The lower bound $\Omega(\epsilon^{-1}+ns)$ on the number of samples needed for (ϵ,δ)-learning IVY with any learning procedure follows by plugging the lower bound on the fat shattering dimension from Theorem 2.2 into Theorem 19.5 in [AB09].

E Connection to Prior Work

In this section we discuss the connection of our techniques to prior techniques for proving statistical generalization bounds for data-driven algorithms, and specifically to [GR17] and [BDD+21]. The goal is to place our work in the context of related work, and also to explain why previous techniques do not give useful generalization bounds for the linear algebraic algorithms considered in this paper.

E.1 Illustrative Example 1: Learning Greedy Heuristics for Knapsack

[GR17] discuss learned heuristics for the Knapsack problem as an illustrative example for their statistical learning framework. Recall that in the Knapsack problem, the input is n items with values $v_1, \ldots, v_n > 0$, respective costs $c_1, \ldots, c_n > 0$, and a cost limit $C > 0$. The goal is to return $I \subset \{1, \ldots, n\}$ that maximizes the total value $\sum_{i \in I} v_i$ under the cost constraint $\sum_{i \in I} c_i < C$. This problem is well-known to be NP-hard.

[GR17] consider the following family of greedy heurstics for Knapcask. Given a parameter $\rho \in R$, define the rank of item i as v_i / c_i^ρ, and let L_ρ be the greedy heuristic that adds the highest ranked items to I as long as the cost limit is not exceeded. They then let ρ be a learnable parameter, and prove that the pseudo-dimension of the class of heuristics $\mathcal{L} = \{L_\rho\}_{\rho \in \mathbb{R}}$ is $O(\log n)$.

The crux of the proof is the observation that for a given instance, the "loss" (or in this case the utility, since this is a combinatorial maximization problem) of a solution I is fully determined by the result of the $\binom{n}{2}$ comparison $v_i / c_i^\rho \geq^? v_j / c_j^\rho$, or equivalently $\rho \geq^? \log(v_j / v_i) / \log(c_j / c_i)$. This means that the parameter space $\mathbb{R}$ is partitioned into $\binom{n}{2}$ intervals such that the "loss" (utility) of the given instance is constant on each interval. The pseudo-dimension can be then bounded by the log of the number of intervals. The duality-based framework developed by [BDD+21] is a vast generalization of this argument, and recovers the same bound for Knapsack.

The GJ framework presented in Section 3 recovers it as well, for the same underlying reason. Since the "loss" (utility) of a given instance is determined by the $\binom{n}{2}$ comparisons $\rho \geq^? \log(v_j / v_i) / \log(c_j / c_i)$, which are polynomial predicates of degree 1 in the variable ρ, it can be computed by a GJ algorithm with degree 1 and predicate complexity $\binom{n}{2}$. The upper bound $O(\log n)$ on the pseudo-dimension thus follows from Theorem 3.3.[7]

E.2 Illustrative Example 2: IVY in the Case $m = k = 1$

In order to compare our techniques with the duality-based framework of [BDD+21], it is instructive to consider the simple case $m = k = 1$ of IVY. In Section 4, we proved a tight bound of $O(n)$ on the pseudo-dimension of IVY in this case.

Our proof showed that for a given input matrix $A \in \mathbb{R}^{n \times d}$, the loss equals $\|A\|_F^2$ if the sketching vector $w \in \mathbb{R}^n$ satisfies $w^T A = 0$, and equals 0 otherwise. By the main definition of [BDD+21], this means that the dual class of losses is $(\mathcal{F}, \mathcal{G}, d)$-piecewise decomposable, where $\mathcal{F}$ is the class of constant-valued functions, $\mathcal{G}$ is the class of n-dimensional linear threshold functions, and d is the column-dimension of A (note that d is the number of functions from $\mathcal{G}$ involved in the condition $w^T A = 0$). Denoting the dual classes of $\mathcal{F}$ and $\mathcal{G}$ by $\mathcal{F}^*$ and $\mathcal{G}^*$ respectively (see [BDD+21] for the definition of dual classes), we have $\operatorname{pdim}(\mathcal{F}^*) = 0$ and $\operatorname{VCdim}(\mathcal{G}^*) = n + 1$. Therefore, the main theorem of [BDD+21] gives a bound of $O(n \log n)$ on the pseudo-dimension of IVY with $m = k = 1$, which is looser than the tight bound by $\log n$.

We remark that since the condition $w^T A = 0$ is equivalent to $\|w^T A\|^2 = 0$, the dual class is also $(\mathcal{F}, \mathcal{G}_2, 1)$-piecewise decomposable where $\mathcal{G}_2$ is a class of quadratic threshold functions in n variables. It can be checked that $\operatorname{VCdim}(\mathcal{G}_2^*) = \frac{1}{2}(n+1)(n+2)$, leading to an even looser bound of $O(n^2 \log n)$ on the pseudo-dimension.

E.3 The General Case

Finally, let us point out a formal connection between the GJ framework and the duality framework of [BDD+21]. The proof of Theorem 3.3 in fact shows that if a class of algorithms admits a GJ algorithm with degree Δ and predicate complexity p for computing the loss, then the dual class as defined by [BDD+21] is $(\mathcal{F}, \mathcal{G}, p)$-piecewise decomposable, where $\mathcal{F}$ is the class of constant-valued functions, and $\mathcal{G}$ is the class of

[7] Note that each heuristic L_ρ is specified by the single real parameter ρ, so n in the notation of Theorem 3.3 is 1.

polynomial threshold functions of degree $\Delta' = O(\Delta)$ in n variables (where n is the number of parameters that specify an algorithm, as in Theorem 3.3). The VC-dimension of the dual class $\mathcal{G}^*$ can be upper-bounded by the number of monomials in n variables of degree at most Δ', which is $\binom{n+\Delta'}{n}$.[8] Therefore, the main theorem of [BDD+21] implies an upper bound of $O\left(\binom{n+\Delta}{n} \log\left(\binom{n+\Delta}{n} \cdot p\right)\right)$ on the pseudo-dimension. Unfortunately, this is typically much looser than the bound $O(n \log(\Delta p))$ in Theorem 3.3. On the other hand, the result of [BDD+21] is much more general, and can handle decomposability beyond constant functions $\mathcal{F}$ and polynomial thresholds $\mathcal{G}$.

Specifically for IVY, the main component in our proof of Theorem 2.2 was a GJ algorithm of degree $\Delta = O(mk(d/\epsilon)^{O(1/\epsilon)})$ and predicate complexity $p \leq 2^m \cdot 2^k \cdot (ed/k)^{3k}$ for computing the proxy loss (Lemma 5.6). While the main result of [BDD+21] can technically be applied here as described above, the pseudo-dimension upper bound it gives is super-polynomial in n (as opposed to the linear dependence on n in Theorem 2.2), so it does not seem to be useful in our setting.

[8] The VC-dimension of the dual class could in principle be even smaller, if the primal class has a simpler structure than the dual class. However, in typical scenarios the primal class is less nicely structured than the dual class, which is the motivation for the work of [BDD+21]. Furthermore, if the primal class is indeed simpler, then there is no reason to go through duality at all.