Proceedings of Machine Learning Research vol 145:1–34, 2022 3rd Annual Conference on Mathematical and Scientific Machine Learning

SHRIMP: Sparser Random Feature Models via Iterative Magnitude Pruning

Yuege Xie[†] YUEGE@ODEN.UTEXAS.EDU
University of Texas at Austin

Bobby Shi[†] BHSHI@UTEXAS.EDU
University of Texas at Austin

Hayden Schaeffer HSCHAEFF@ANDREW.CMU.EDU
Carnegie Mellon University

Rachel Ward RWARD@MATH.UTEXAS.EDU
University of Texas at Austin

Editors: Bin Dong, Qianxiao Li, Lei Wang, Zhi-Qin John Xu

Abstract

Sparse shrunk additive models and sparse random feature models have been developed separately as methods to learn low-order functions, where there are few interactions between variables, but neither offers computational efficiency. On the other hand, ℓ_2-based shrunk additive models are efficient but do not offer feature selection as the resulting coefficient vectors are dense. Inspired by the success of the iterative magnitude pruning technique in finding lottery tickets of neural networks, we propose a new method—**S**parser **R**andom Feature Models via **IMP** (ShRIMP)[1]—to efficiently fit high-dimensional data with inherent low-dimensional structure in the form of sparse variable dependencies. Our method can be viewed as a combined process to construct and find sparse lottery tickets for two-layer dense networks. We explain the observed benefit of SHRIMP through a refined analysis of the generalization error for thresholded Basis Pursuit and resulting bounds on eigenvalues.

From function approximation experiments on both synthetic data and real-world benchmark datasets, we show that SHRIMP obtains better than or competitive test accuracy compared to state-of-the-art sparse feature and additive methods such as SRFE-S, SSAM, and SALSA. Meanwhile, SHRIMP performs feature selection with low computational complexity and is robust to the pruning rate, indicating a robustness in the structure of the obtained subnetworks. We gain insight into the lottery ticket hypothesis through SHRIMP by noting a correspondence between our model and weight/neuron subnetworks.

Keywords: Random Feature Model, Pruning, Sparse Feature

1. Introduction

Kernel regression is an established choice for learning a target function from data with solid theoretical foundation (Hearst et al., 1998; Zhang, 2005). Kernel methods are general-purpose methods for estimating a function by fitting measurements to a representative function in a Reproducing Kernel Hilbert Space (Campbell, 2002). The power of the method is derived from the Representer Theorem which connects finite measurements and continuous function space; however, kernel ridge regression does not take into account additional structure in the underlying target function, which can be

[†] Equal contribution. Correspondence to: Yuege Xie.

1. Code is available at https://github.com/rhshi/sparse-rf.

limiting when additional structure is known to be present. In many physical settings (Harris, 2019), functions may arise naturally as sums of functions, each with a limited variable interaction. Such low-order structure (Kuo et al., 2010) may also be used to reduce the complexity of the system being modeled (Potts and Schmischke, 2021a,b). The Multiple Kernel Learning line of literature (Gönen and Alpaydın, 2011; Bach, 2008; Xu et al., 2010) and more recent methods such as shrunk additive models (Kandasamy and Yu, 2016; Liu et al., 2020) have been developed to exploit such low-order structure. However, these methods are computationally inefficient due to the cost of kernel ridge regression and minimal ℓ_1-norm optimization (Liu et al., 2020), as well as repeated computation of the kernel at test and prediction periods.

In an independent line of work, the well-studied random features model as introduced in Rahimi and Recht (2008c) allows for an approximation to the kernel function of interest without the computational cost of constructing a full kernel matrix, with bounds given in Rahimi and Recht (2008b,a); Avron et al. (2017); Cortes et al. (2010). However, neither the generic random features model nor the shrunk additive model offers the possibility of simple model compression or feature selection. Kernel methods have been investigated in the context of feature selection (Kumar et al., 2009) and in sparse additive models (Huang et al., 2010; Yin et al., 2012; Ravikumar et al., 2009), and recent work by Hashemi et al. (2021) introduces the sparse random features model with coefficient vector recovered using basis pursuit. However, it is a priori unclear what coefficient sparsity means, especially in the ℓ_1 sense, for random feature models.

Inspired by the success of the iterative magnitude pruning (IMP) technique for finding sparse subnetworks of neural networks with comparable performance (Frankle and Carbin, 2019; Zhou et al., 2019), we propose a new ℓ_2-based method—Sparser Random Feature Models via Iterative Magnitude Pruning (SHRIMP)—to efficiently fit high-dimensional data with inherent low-order structure. This method can be viewed as a combined process to construct and find lottery tickets of two-layer dense networks: it randomly initializes a fixed first layer with only low-order interactions by using sparse random weights instead of dense weights (Stage I) and applies neuron pruning to find a sparser subnetwork by IMP (Stage II). From experiments on both synthetic data and real-world benchmark datasets, we show that SHRIMP is better or competitive against both random sparse feature models and shrunk additive models. We offer a refined analysis of the thresholded ℓ_1-based sparse random feature model based on Hashemi et al. (2021), and through our experiments and further discussion offer insight into the success of our method.

We connect our work to the larger active literature on random features learning and the theory of neural networks. Neural networks in certain regimes have been found to be kernel machines (Jacot et al., 2018; Chizat et al., 2019), and thus linear regression has been revisited as a model for neural network behavior (Liang and Rakhlin, 2020; Hastie et al., 2019). In particular, there has been a concentrated interest in the generalization and double descent behaviors of random features regression (Montanari et al., 2019; Jacot et al., 2020; d'Ascoli et al., 2020); these works study ReLU features, while the closest model to our setting is that of Liao et al. (2020). *As random features regression has become essential for studying neural networks in the kernel regime, we hope that random features pruning can be used to study neural network pruning*; much of the current theory on neural network pruning has been existential in nature (Orseau et al., 2020; Malach et al., 2020; Pensia et al., 2020). Moreover, the lottery ticket hypothesis has one of the greatest implications for model compression, helping alleviate problems related to massive neural networks, such as unequal access to computing resources among researchers, and the environmental impact of deep learning.

Our main contributions are as follows:

1. We propose a two-stage algorithm, SHRIMP, to learn sparse random feature models with low-order interactions efficiently via an iterative magnitude pruning technique. It is surprising that ℓ_2-based SHRIMP finds sparser models than basis pursuit, which is a staple among algorithms used for sparse recovery and feature selection.

2. Experiments on both synthetic and real-world datasets verify that with proper choice of the order q in the low-order function model, SHRIMP obtains better or matching test performance compared to existing sparse feature or shrunk additive models while at the same time being scalable to high-dimensional settings. Beyond the effectiveness and efficiency, we show that SHRIMP is robust to parameters such as pruning rate and exhibits surprising support recovery ability with odd/even separation.

3. We offer a refined analysis of the main theorem from Hashemi et al. (2021) that allows us to connect our experimental findings with initial theoretical results from the compressed sensing literature; one perspective of our method is that it situates itself between ℓ_2-based and ℓ_0-based methods. Our analysis on the evolution of the spectrum of the Gram matrix during the SHRIMP algorithm iterations offers additional insight into the benefit of SHRIMP over random pruning.

4. By connecting our SHRIMP model to the process of finding winning lottery tickets for a two-layer fully connected neural network initialized with a random sparse subnetwork and with neurons pruned by IMP, we shed light on the successful performance of IMP as a mechanism for finding lottery tickets.

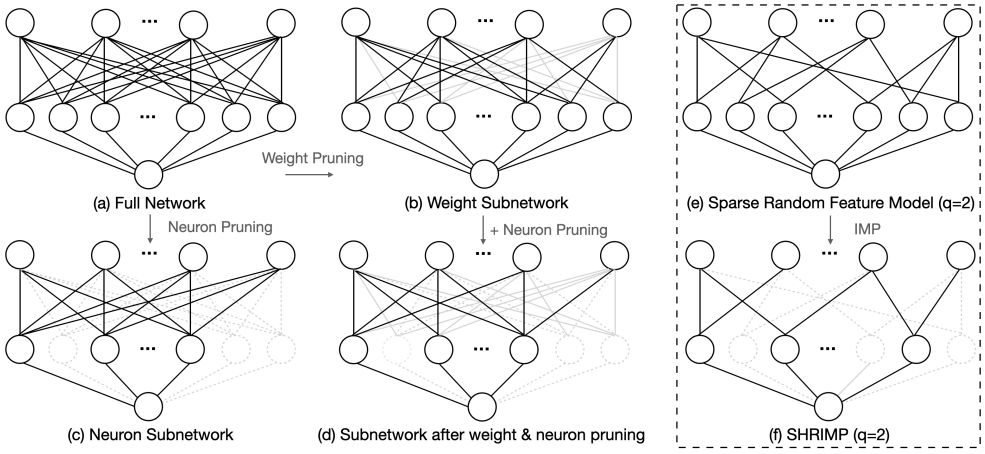

Figure 1: Illustration of the relationship between SHRIMP and different types of pruning.

1.1. Related Work

Sparse Random Feature Models and Shrunk Additive Models. In high-dimensional settings arising from modeling physical systems, the underlying governing function is well-approximated as being a sum of *low-order* functions; that is, the function can be written as a sum of component functions, such that only q variables out of a total of d variables are active in each component,

with $q \ll d$ (DeVore et al., 2011; Kuo et al., 2010). Recent methods such as SALSA (Kandasamy and Yu, 2016) and SSAM (Liu et al., 2020) are kernel-based methods that directly exploit such low-order structure. Separately, the random Fourier features model of Rahimi and Recht (2008c) is a popular choice in approximating a kernel function when there are many data samples and the construction of the kernel matrix is computationally expensive. Additive kernels were considered in Vedaldi and Zisserman (2012), while coefficient sparsity was investigated in Yen et al. (2014); Özcelikkale (2020). The first work (to our knowledge) to combine these approaches—exploiting low-order structure and random features together—was the work of Hashemi et al. (2021), which uses an ℓ_1-based approach with *sparse* random features in order to learn a low-order function. A detailed comparison of SHRIMP with sparse random feature models (Hashemi et al., 2021; Elesedy et al., 2020) and shrunk additive models (Kandasamy and Yu, 2016; Liu et al., 2020) is listed in Table 1.

Table 1: Comparison of sparse random feature models: SRFE-S (Hashemi et al., 2021), shrunk additive models such as SALSA (Kandasamy and Yu, 2016), SSAM (Liu et al., 2020), and IMP in Linear Regression (Elesedy et al., 2020). Here, d denotes the data dimension, q denotes the interaction order of features, and T_p indexes the pruning iteration. The feature number with N_s corresponds to the number of non-zero components in the feature vector by ℓ_1-regularization; N_{best} denotes the number of features remaining after IMP.

Property	SHRIMP	SALSA	SSAM	SRFE-S	IMP in LR
Sample sparsity	$\surd$	$\times$	$\surd$	$\surd$	$\times$
Low-order Interaction	$\surd$	$\surd$	$\surd$	$\surd$	$\times$
Feature sparsity	$\surd$	$\times$	$\surd$	$\surd$	$\surd$
Computational Efficiency	$\surd$	$\surd$	$\times$	$\times$	$\surd$
Regularization	implicit ℓ_2	ℓ_2-norm	ℓ_1-norm	ℓ_1-norm	implicit ℓ_2
Feature model	random feature	kernel	kernel	random feature	linear
# Features*	$\binom{d}{q} \cdot n \to N_{best}$	$\binom{d}{q}$	$\binom{d}{q} \to N_s$	$\binom{d}{q} \cdot n \to N_s$	$d \to d - T_p$

Lottery Ticket Hypothesis and Iterative Magnitude Pruning. Frankle and Carbin (2019) proposed the lottery ticket hypothesis and a corresponding iterative magnitude pruning (IMP) procedure for compressing overparameterized neural networks, which is a primary inspiration for our algorithm. The IMP procedure prunes weights based on their magnitude and then retrains the pruned subnetwork from the same initial weights as the original network at each pruning iteration. IMP is demonstrated empirically to find a sparse subnetwork (i.e., winning ticket) with comparable test accuracy to the original dense network. Follow-up work (Zhou et al., 2019; Ramanujan et al., 2020; Malach et al., 2020) theoretically proves that for an overparameterized neural network with randomly-initialized weights under some conditions, there exists a subnetwork of it that can obtain competitive performance. Furthermore, Elesedy et al. (2020) initiated a theoretical analysis of a simplified form of IMP, where the algorithm prunes a single weight per iteration, and the solution is obtained by gradient flow (i.e., min ℓ_2-norm estimator) in linear models. However, this is not a practical sparse estimation method in linear models due to the inefficiency of one-weight-per-iteration pruning and computation of gradient descent with $t \to \infty$. Instead, our work applies proportional IMP to neuron pruning and utilizes the implicit regularization of the pseudo-inverse to reduce com-

putational cost significantly. The work Zhang et al. (2021a) analyzes the geometric structure of the model throughout the pruning process; we corroborate these ideas by providing results on the eigenvalues of the Gram matrix throughout the pruning. However, our work does not focus solely on the lottery ticket hypothesis but should serve as a stand-alone method for low-order function approximation.

1.2. Notations

Throughout the paper, $\boldsymbol{A}^\dagger$ denotes the Moore-Penrose inverse of a matrix $\boldsymbol{A}$, and hence $\boldsymbol{c} = \boldsymbol{A}^\dagger \boldsymbol{y}$ is the minimal ℓ_2-norm estimator in the overparameterized regime, and the least-square estimator in the underparameterized regime. We denote the number of data points by m, the dimension of data by d, and the number of features by N. Measurement noise e_k is defined as $y_k = f(\boldsymbol{x}_k) + e_k$ with either $|e_k| \leqslant E = 2\nu$ or e_k i.i.d. drawn from $\mathcal{N}(0, \nu^2)$, $\forall k \in [m]$.

2. Preliminaries

Low order functions arise naturally in the physical world and are used as a form of the reduced-complexity model for such systems (Potts and Schmischke, 2021a,b). Let us first recall the definition for an order-q function, as well as definitions for bounded ρ-norm functions and q-sparse feature weights from Hashemi et al. (2021).

Definition 1 (Order-q Function, (Hashemi et al., 2021)) *For any $d, q, K \in \mathbb{N}_+$ with $q \leqslant d$, a function $f : \mathbb{C}^d \to \mathbb{C}$ is an order-q function of at most K terms if there exist functions $g_1, \ldots, g_K : \mathbb{C}^q \to \mathbb{C}$ such that*

$$f(x_1, \ldots, x_d) = \frac{1}{K} \sum_{j=1}^{K} g_j(x_{j_1}, \ldots, x_{j_q}) = \frac{1}{K} \sum_{j=1}^{K} g_j(\boldsymbol{x}|_{\mathcal{S}_j}), \tag{1}$$

where $\mathcal{S}_j \subseteq [d]$ is an index subset of $[d]$ and $\boldsymbol{x}|_{\mathcal{S}_j}$ is the restriction of $\boldsymbol{x}$ onto the indices.

In general, such a decomposition is not unique. However, the set of order-q functions forms a vector space, as the sum of two order-q functions is itself an order-q function, and the space is closed under scalar multiplication. Additionally, if we let $\|\cdot\|$ be a function norm, then we can define

$$\|f\| = \inf \sqrt{\frac{1}{K}(\|g_1\|^2 + \cdots + \|g_K\|^2)},$$

where the infimum is taken over all possible order-q decompositions of f. If each g_j lies in a Reproducing Kernel Hilbert Space (RKHS), then f lies in the direct sum of the component Reproducing Kernel Hilbert Spaces with RKHS norm defined as above (Aronszajn, 1950).

Definition 2 (Bounded ρ-norm Function, (Hashemi et al., 2021)) *Fix a probability density function $\rho : \mathbb{R}^d \to \mathbb{R}$ and a function $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$. A function $f : \mathbb{R}^d \to \mathbb{C}$ has finite ρ-norm with respect to $\phi(\boldsymbol{x}; \boldsymbol{w})$ if it belongs to the class*

$$\mathcal{F}(\phi, \rho) := \left\{ f(\boldsymbol{x}) = \int_{\boldsymbol{w} \in \mathbb{R}^d} \alpha(\boldsymbol{w}) \phi(\boldsymbol{x}; \boldsymbol{w}) d\boldsymbol{w} : \|f\|_\rho := \sup_{\boldsymbol{w}} \left| \frac{\alpha(\boldsymbol{w})}{\rho(\boldsymbol{w})} \right| < \infty \right\}. \tag{2}$$

Definition 3 (q**-sparse Feature Weights, (Hashemi et al., 2021))** *Let $d, q, n \in \mathbb{N}_+$ with $q \leqslant d$, and let $\rho : \mathbb{R}^q \to \mathbb{R}$ be a probability distribution. A collection of $N = n\binom{d}{q}$ weight vectors $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_N$ is called a set of q-sparse feature weights if it is generated as follows: for each index subset $\mathcal{S}_j \subseteq [d], |\mathcal{S}_j| = q$ draw n i.i.d. random vectors $\mathbf{z}_1, \ldots, \mathbf{z}_n \sim \rho$, and construct q-sparse features $\{\boldsymbol{\omega}_{j_k}\}_{k=1}^n$ by setting $\mathrm{supp}(\boldsymbol{\omega}_{j_k}) = \mathcal{S}_j$ and $\boldsymbol{\omega}_{j_k}|_{\mathcal{S}_j} = \mathbf{z}_k, k \in [n]$.*

Let $\boldsymbol{X} \in \mathbb{R}^{m\times d}$ be a data matrix consisting of m d-dimensional samples, and let $\boldsymbol{W} \in \mathbb{R}^{N\times d}$ be the matrix of N d-dimensional feature weights constructed according to Def. 3. Construct the random feature matrix $\boldsymbol{A} = \phi(\boldsymbol{X}\boldsymbol{W}^*)$ so that $\boldsymbol{A}\boldsymbol{A}^*$ approximates a kernel matrix of interest. For example, let $\phi(\cdot) := [\cos(\cdot)\ \sin(\cdot)]$ [2] (with $\boldsymbol{A} = [\sin(\boldsymbol{X}\boldsymbol{W}^*)\ \cos(\boldsymbol{X}\boldsymbol{W}^*)]$), $q = d$, and ρ be the normal distribution, $\mathbf{E}_W[\boldsymbol{A}\boldsymbol{A}^*]$ is the kernel matrix of the Gaussian kernel (Rahimi and Recht, 2008c). In the more general case of $q \leqslant d$, with ρ as a normal distribution, $\mathbf{E}_W[\boldsymbol{A}\boldsymbol{A}^*]$ is the kernel matrix corresponding to a *direct sum* of Gaussian kernels, each defined over $\mathbb{R}^q \times \mathbb{R}^q$. See the appendix for more precise approximation bounds.

3. Sparser Random Feature Models via Iterative Magnitude Pruning

Algorithm 1 SHRIMP: Sparser Random Feature Models via Iterative Magnitude Pruning

Input: Training dataset $(\mathcal{X}, \mathcal{Y}) \in \mathbb{C}^{m\times d} \times \mathbb{C}^m$, parametric basis function $\phi(\cdot\,; \boldsymbol{w})$, feature sparsity level q, pruning rate $p \in (0, 1)$, iterations of pruning $T \in \mathbb{N}_+$.

Stage I. Constructing: Draw $N = n\binom{d}{q}$ q-sparse features $\{\boldsymbol{w}_j\}_{j=1}^N$ according to Def. 3, form the matrix $\boldsymbol{W} \in \mathbb{R}^{N\times d}$, and construct a random feature matrix $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ by $\boldsymbol{A} = \phi(\boldsymbol{X}\boldsymbol{W}^*)$.

Stage II. Pruning: Compute the min ℓ_2-norm solution $\boldsymbol{c}_0 = \boldsymbol{A}^\dagger \boldsymbol{y}$ and set $\mathcal{P}_0 = \{1, \ldots, N\}$.

for $t = 1, \ldots, T$ **do**

Get the feature index set $\mathcal{P}_t$ by pruning $p \times |\mathcal{P}_{t-1}|$ features from $\boldsymbol{c}_{t-1}$ with $\boldsymbol{c}_t^s$ denoting the ascending sorted (by absolute value) array of $\boldsymbol{c}_t$ by

$$\mathcal{P}_t = \{i \in \mathcal{P}_{t-1} : |\boldsymbol{c}_{t-1,i}| \geqslant \boldsymbol{c}^s_{t-1,p\times|\mathcal{P}_{t-1}|}\}.$$

Update the min ℓ_2-norm solution: $\boldsymbol{c}_t = \boldsymbol{A}^\dagger_{\mathcal{P}_t}\boldsymbol{y}$, where $\boldsymbol{A}_{\mathcal{P}_t}$ is the column submatrix of $\boldsymbol{A}$ with index set $\mathcal{P}_t$.

end

Output: the pruned minimal ℓ_2 norm estimator and feature index pair set $\{(\boldsymbol{c}_t, \mathcal{P}_t)\}_{t=1}^T$.

In this section, we first present the SHRIMP algorithm and then illustrate its connection to the lottery ticket hypothesis and network pruning. To address the challenge of targeting low-order additive structure efficiently with feature selection, we propose a two-step SHRIMP method (Algorithm 1) to find a sparse low-order random feature subnetwork of a fully connected neural network: first, we initialize a sparse random feature model by constructing low-order random feature weights as a subnetwork of the dense network, according to Def. 3; second, SHRIMP finds a sparse winning lottery ticket $\boldsymbol{c}_{t^\star}$ by forming a set of sparse min ℓ_2-norm estimators $\{\boldsymbol{c}_t\}_{t=1}^T$ via IMP and selecting the best model via a validation dataset. At test time, with the best model $\{\boldsymbol{c}_{t^\star}, \mathcal{P}_{t^\star}\}$ chosen, we

2. Here sin and cos are element-wise operations on the matrix $\boldsymbol{X}\boldsymbol{W}^*$. We concatenate two resulting matrices $\sin(\boldsymbol{X}\boldsymbol{W}^*)$ and $\cos(\boldsymbol{X}\boldsymbol{W}^*)]$ together.

transform the test data via $\boldsymbol{A}^{test} = \phi(\boldsymbol{X}^{test}_{\mathcal{P}_{t^\star}} \boldsymbol{W}^*_{\mathcal{P}^\star_t})$ and predict via $\boldsymbol{y}^{test} = \boldsymbol{A}^{test}\boldsymbol{c}_{t^\star}$. For synthetic data, the validation and test data are randomly drawn from the same distribution as the training data; for real-world data, we randomly split the training data into training and validation sets.

Discussion on Computational Complexity. At each step t, SHRIMP gets the min ℓ_2-norm solution by solving a linear system $\boldsymbol{c}_t = \boldsymbol{A}^\dagger_{\mathcal{P}_t}\boldsymbol{y}$ rather than computing pseudo-inverse directly, so each step has computational cost *at most* $\mathcal{O}(N_t m^2 + m^3)$ in the worst case (it is possible to leverage random sketching to solve $\boldsymbol{A}_{\mathcal{P}_{t+1}}$ based on the previous computation of $\boldsymbol{A}_{\mathcal{P}_t}$). Additionally, $|\mathcal{P}_t| = N_t = N(1-p)^t$ and $t \in [0, T]$ with $p \in (0,1)$ and $(1-p)^T N = 1$. Hence, N_t is at most $\sum_{t=0}^{T} N(1-p)^t = N\frac{(1-1/N)}{p} = \frac{N-1}{p} \ll NT$. Meanwhile, the complexity of ℓ_1 minimization is known to be at least polynomial in N; for example, the complexity of interior-point methods to obtain the min ℓ_1-norm estimator is $\mathcal{O}(N^6)$. SRFE is solved using the spgl1 package in Python/MATLAB (van den Berg and Friedlander, 2019, 2008). A detailed comparison of computational time required by SHRIMP and SRFE with different scales of data is in Figure 2.

Connection to Neural Network Pruning. Consider a two-layer fully connected neural network $f(\boldsymbol{x})$ with activation function $\phi(\cdot)$ (see also Figure 1(a)),

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} a_i \phi(\boldsymbol{w}_i^* \boldsymbol{x}). \tag{3}$$

Winning tickets are defined as sparse subnetworks that reach test accuracy comparable to the original network (Frankle and Carbin, 2019). Finding winning lottery tickets is known to be computationally hard in the worst case (Frankle and Carbin, 2019; Malach et al., 2020; Zhang et al., 2021b), and this poses a challenge to understanding why certain pruning methods tend to work well in practice. Pruning methods for neural networks generally fall into two categories: weight pruning and neuron pruning (Malach et al., 2020). Weight pruning (Figure 1 (a) to (b)) involves a set of binary mask vectors $\boldsymbol{u}_i \in \{0,1\}^d$, equivalent to pruning the first layer weights, resulting in the network

$$\tilde{f_w}(\boldsymbol{x}) = \sum_{i=1}^{N} a_i \phi((\boldsymbol{w}_i \odot \boldsymbol{u}_i)^* \boldsymbol{x}), \tag{4}$$

while neuron pruning (Figure 1 (a) to (c)) involves a set of binary scalars $b_i \in \{0,1\}$, equivalent to pruning entire neurons, resulting in the network

$$\tilde{f_n}(\boldsymbol{x}) = \sum_{i=1}^{N} (b_i a_i) \phi(\boldsymbol{w}_i^* \boldsymbol{x}). \tag{5}$$

A sparse subnetwork is the result of applying both types of pruning, as shown in Figure 1(d). The SHRIMP method first fixes the weight subnetwork at a specified sparsity level determined by the choice of low-order parameter q (Figure 1(e)); it then adaptively "prunes" the neurons by finding a sparse coefficient vector (Figure 1(f)). In other words, for each subset $\mathcal{S}_j \subseteq [d], |\mathcal{S}_j| = q$, we define $\boldsymbol{u}_j \in \{0,1\}^d, \mathrm{supp}(\boldsymbol{u}_j) = \mathcal{S}_j$, and then adaptively prune $\boldsymbol{c}$ so that the result is

$$f^\star(\boldsymbol{x}) = \frac{1}{K} \sum_{j=1}^{K} \sum_{\ell=1}^{n} (b_{j,\ell} c_{j,\ell}) \phi((\boldsymbol{w}_{j,\ell} \odot \boldsymbol{u}_j)^* \boldsymbol{x}), \tag{6}$$

where $K = \binom{d}{q}$. Using the notation of Algorithm 1, if s is the sparsity level of $\boldsymbol{c}_{t^\star}$, then we can compress the number of nonzero entries of $\boldsymbol{W}_{\mathcal{P}_t^\star}, \boldsymbol{c}_{t^\star}$ to $(q+1)s$. This is in contrast to the standard random feature model with dense $\boldsymbol{W}, \boldsymbol{c}$, where there are $(d+1)N$ non-zeros. Note that SHRIMP performs best when the target function is a sum of low-order components, so that the selected model is actually a pruned sparse model. Experiments shown in Table 2 corroborate this finding, with "Avg size" (#features) much smaller than N. Explicitly capturing the low-order structure is an advantage of SHRIMP.

4. Experiments

In this section, show generalization error results on synthetic and real-world datasets. We then illustrate several benefits of SHRIMP compared to other approaches including computational efficiency, robustness to pruning rate, and sparse support recovery, as well as other benefits of iterative magnitude pruning.

4.1. Function Approximation

To demonstrate the performance of SHRIMP on low-order functions, we first test different models on synthetic functions with $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]^T$. See the appendix for additional experiments.

- Simple additive functions: $f_1(\boldsymbol{x}) = \sum_{i=1}^{d-1} x_i + \exp(-x_d)$ and $f_2(\boldsymbol{x}) = \cos(x_1) + \sin(x_2)$;
- Functions with pairwise behavior (from Liu et al. (2020)): $f_3(\boldsymbol{x}) = (2x_1 - 1)(2x_2 - 1)$ and $f_4(\boldsymbol{x}) = (2x_1 - 1)(2x_2 - 1) + (2x_1 - 1)(2x_3 - 1) + (2x_2 - 1)(2x_3 - 1)$;
- Low-order non-smooth functions Hashemi et al. (2021): $f_5(\boldsymbol{x}) = \mathrm{sinc}(x_1)\mathrm{sinc}(x_3)^3 + \mathrm{sinc}(x_2)$;
- Ishigami example used for uncertainty and sensitivity analysis Ishigami and Homma (1990); Hashemi et al. (2021): $f_6(\boldsymbol{x}) = \sin(x_1) + 7\sin^2(x_2) + 0.1x_3^4 \sin(x_1)$;
- An order-2 function with many order-1 components: $f_7(\boldsymbol{x}) = \cos(x_1)x_3 + x_2^2 x_4 + \sum_{i=3}^{d} x_i$.

Experimental Results. Table 2 reports the test mean-squared error (MSE) for the function approximations $\{f_i(\boldsymbol{x})\}_{i=1}^7$ and corresponding optimal q[3]. As shown in Table 2, SHRIMP consistently outperforms other function approximations, often by an order of magnitude. This is consistent across low- (s_l^{q*}) and high-dimensional (s_h^{q*}) settings. In particular, SHRIMP performs notably better than SALSA Kandasamy and Yu (2016), which explicitly constructs the additive kernel matrix and performs kernel regression. One notable example is f_6, the Ishigami function; SHRIMP is the only method that attains a test MSE less than 1 in the low-dimensional setting. In addition, SHRIMP succeeds in finding sparse models. The best-performing SHRIMP model often has orders of magnitude fewer features, performing significant model compression (from initial 20000 features to "Avg size" in Table 2), and often moving from the overparameterized to the underparameterized setting via this compression. When more variables are involved, e.g., f_1, which is order-1 but involves all the variables, #features retained also increases. On the other hand, for f_2, which is a simple sum involving two trigonometric terms, SHRIMP retains only 19 features yet attains the best test error by a significant margin.

3. See the appendix for more results on $q = d$.

Table 2: Comparison of the test errors of SHRIMP, Min ℓ_2, SALSA, and SRFE-S with $q = q_*$. Avg size denotes the average pruned model size of SHRIMP taken over three runs. $^\dagger s$ denotes the setting with different (m, d, q) pairs: $s_l^{q*} = (140, 10, q_*), s_h^{q*} = (1400, 100, q_*)$, where l denotes low dimension and h denotes high dimension, and all methods use $q = q_*$ (the ground-truth order). The best MSE for each $f_i(\boldsymbol{x})$ over all models is in purple. From the comparison between s_l^{q*} and s_h^{q*}, SHRIMP is the best overall, and is significantly more scalable to the high-dimensional setting.

Setting†	Model	$f_1(\boldsymbol{x})$	$f_2(\boldsymbol{x})$	$f_3(\boldsymbol{x})$	$f_4(\boldsymbol{x})$	$f_5(\boldsymbol{x})$	$f_6(\boldsymbol{x})$	$f_7(\boldsymbol{x})$
s_l^{q*}	SRFE-S	7.85e-04	1.98e-05	1.15e-01	1.27e-01	7.52e-03	1.11	7.49e-02
	Min ℓ_2	4.37e-20	5.45e-24	8.20e-02	5.94e-02	7.36e-03	7.18	2.98e-02
	SALSA	1.59e-12	1.26e-15	8.80e-02	6.14e-02	7.32e-03	6.99	2.72e-02
	SHRIMP	1.37e-22	7.90e-32	4.98e-12	2.54e-12	6.39e-04	2.58e-02	2.83e-05
	Avg size*	3100.33	29	147	171.67	39	80.33	187
s_h^{q*}	SRFE-S	1.52e-03	8.71e-06	1.99	4.54	1.16e-01	4.59	4.40e-01
	Min ℓ_2	1.68e-20	3.51e-24	2.01	4.75	1.16e-01	8.34	1.49e-01
	SALSA	1.99e-11	2.54e-13	1.60	3.16	8.49e-02	7.35	1.34e-01
	SHRIMP	1.61e-22	1.11e-30	1.26e-02	5.11e-01	1.50e-02	2.68	5.82e-02
	Avg size	3355	19	61.33	64	42.67	13	229
	Order q_*	1	1	2	2	3	2	2

4.2. Real-world Datasets

We test SHRIMP on eight real-world datasets from the UCI repository (http://archive.ics.uci.edu/ml) and follow the experimental setup (https://github.com/kirthevasank/salsa) in Kandasamy and Yu (2016). We compare the test errors with shrunk additive models, SALSA (Kandasamy and Yu, 2016) and SSAM Liu et al. (2020), and the sparse model by Lasso. For the experiments with SHRIMP, we use 90% of the original training dataset as training data and 10% as validation data to select the best model from models trained with $q \in \{1, 2, 3, \ldots, d_{order}\}$ and the pruning rate $p(\%) \in \{15, 25, 35\}$, which is a similar selection strategy to SALSA Kandasamy and Yu (2016). For practical consideration, we usually only use $d_{order} = \max\{10, d\}$, and then $q \in \{1, 2, \ldots, \max\{10, d\}\}$. From Table 3, SHRIMP attains the best test errors on the Propulsion and Galaxy datasets and has comparable results on all other datasets while still being significantly more efficient to implement.

4.3. Application to State Estimation for Robotics

We apply SHRIMP to a real-world application—state estimation for robotics[4]—and compare the performance and execution time with SALSA Kandasamy and Yu (2016). According to the data scource (Wei et al., 2022), the truck data is obtained by tele-operating a scale 1/5, four-wheel drive, Ackermann steering vehicle. We apply SHRIMP and SALSA to infer next-timestamp positions,

4. This problem has a low-order nature since it is a dynamic system, for example, next time step positions $pos_{x,t+1} = pos_{x,t} + vel_{x,t} * \delta_t$, where $vel_{x,t}$ depends on other features except pos_x and pos_y, so the order is less than the number of total features.

Table 3: Test MSE on eight real-world benchmark datasets. The results of SALSA, SSAM, and Lasso are taken from Liu et al. (2020) and Kandasamy and Yu (2016). The best MSE for each dataset is in purple.

Dataset (m, d)	SHRIMP (q, n_b)	SALSA (q)	SSAM	Lasso
Propulsion $(200, 15)$	1.02×10^{-6} $(2, 84)$	8.81×10^{-3} (8)	NA	2.48×10^{-2}
Galaxy $(2000, 20)$	5.41×10^{-6} $(3, 575)$	1.35×10^{-4} (4)	NA	2.39×10^{-2}
Airfoil $(750, 41)$	2.65×10^{-1} $(2, 30)$	5.18×10^{-1} (5)	4.87×10^{-1}	5.20×10^{-1}
CCPP $(2000, 59)$	6.55×10^{-2} $(2, 49)$	6.78×10^{-2} (2)	6.94×10^{-2}	7.40×10^{-2}
Insulin $(256, 50)$	1.24×10^{0} $(1, 60)$	1.02×10^{0} (3)	1.01×10^{0}	1.11×10^{0}
Telemonit $(1000, 19)$	6.00×10^{-2} $(4, 86)$	3.47×10^{-2} (9)	6.89×10^{-2}	8.63×10^{-2}
Housing $(256, 12)$	3.94×10^{-1} $(7, 15)$	2.62×10^{-1} (1)	3.79×10^{-1}	4.4×10^{-1}
Skillcraft $(1700, 18)$	5.81×10^{-1} $(8, 21)$	5.47×10^{-1} (1)	5.43×10^{-1}	6.65×10^{-1}

angles, velocities, and angular velocity (ω) given current positions, angles, velocities, angular velocity, and controls (with twist linear and twist angular features) by training and validating with $\boldsymbol{X} \in \mathbb{R}^{1699\times 9}$ and test on different trajectories with test data $\boldsymbol{X}_{test} \in \mathbb{R}^{1399\times 9}$. We use the same early stopping criterion for SALSA and SHRIMP (with 25% as the pruning rate and $N = 2000$). The mean and standard deviation of training and inference time is based on the of function approximations seven states in Table 4. As shown in Table 4, the test MSE of SHRIMP on state estimation (pos_x, pos_y, angle_x, and angle_y) is significantly better than that estimated by SALSA; the test MSE on velocities of SHRIMP are slightly higher but very close to that of SALSA. SHRIMP shows the benefit of computational efficiency in both training and inference time over SALSA. SALSA needs to compute a kernel for inference, but SHRIMP only inferences with a low-dimensional weight vector.

Model	pos_x	pos_y	angle_x	angle_y	vel_x	vel_y	ω	training_time	inference_time
SHRIMP	1.64e-07	1.73e-07	2.93e-08	3.12e-08	7.59e-03	8.49e-03	1.39e-02	9.28 ± 1.99	5.94e-02 $\pm$ 5.05e-03
SALSA	3.50e-04	2.82e-04	1.44e-05	2.29e-05	6.26e-03	7.15e-03	9.24e-03	31.49 ± 4.90	1.05e+00 $\pm$ 1.90e-01

Table 4: Test MSE and training/inference time comparison: SHRIMP vs. SALSA.

4.4. Properties of SHRIMP

Computational Efficiency. To compare the time efficiency of SHRIMP to SRFE-S (Hashemi et al. (2021) with ℓ_1-minimization), we approximate the function $f_s(\boldsymbol{x}) = 3\cos(x_3) + 4\sin(x_4) + 2\sin(x_2)$ using varying values for the parameters m, d, and N. Figure 2 Left shows that at equal parameter choices, m, d, N, SHRIMP is significantly faster than SRFE-S. In particular, as m increases, the cost of SHRIMP over varying N increases linearly, while the cost of basis pursuit increases exponentially. In addition to offering better generalization error, SHRIMP is significantly less computationally intensive.

To compare training and inference time efficiency of SHRIMP to SALSA Kandasamy and Yu (2016), we implement SHRIMP in MATLAB and compare it with the official code of SALSA with

function and test them on the same function $f(\boldsymbol{x})$ with only $q = 1, \ldots, 5$ for both methods without any early stopping. Figure 2 Right shows that as the number of data increases[5], the training and inference time of SALSA increase fast since both stages need to compute kernels. In contrast, SHRIMP (with a proper choice of N) [6] is more computationally efficient and scalable with large datasets, especially for inference time; even with ten times more initial features ($N = 2000$), the inference time grows slowly since the resulting model is sparse ($N_{best} \ll N$). This shows the benefits of random sparse features over kernel methods in the applications of online model severing and mobile device inference, which requires less model size.

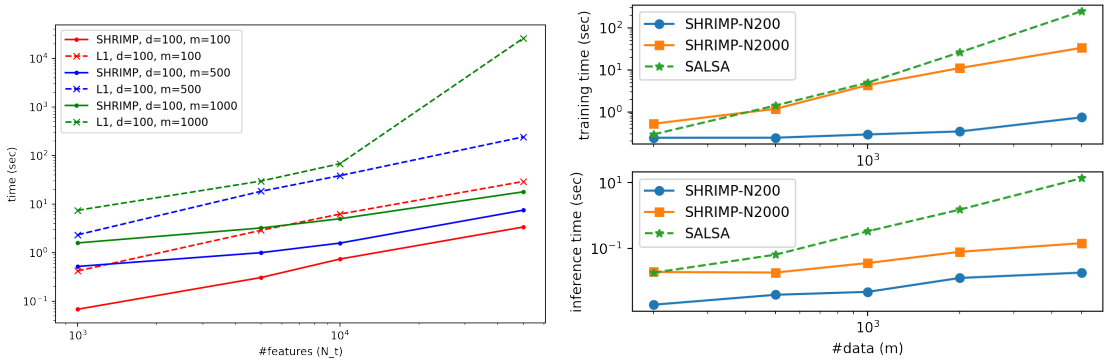

Figure 2: Time comparison. Left: training and validation time of SHRIMP vs. SRFE-S (L_1) as a function of #features in Python; Right: training (with validation) and inference time of SHRIMP vs. SALSA as a function of #data in MATLAB.

Robustness to Pruning Rate. We study the robustness of SHRIMP to the pruning rate on real-world datasets by using a range of pruning rates $p(\%) \in \{15, 20, 25, 30, 35, 40, 45, 50\}$ and comparing the best q chosen by validation dataset and corresponding test MSE. Figure 3 shows that the best q is almost invariant over all pruning rates, and the corresponding test MSEs remain within a small range. Hence, SHRIMP exhibits robustness to the pruning rate, indicating a corresponding robustness in the structure of good subnetworks.

Sparse Support Recovery. We illustrate the power of SHRIMP as a method for sparse support recovery on a simple order-1 additive function $f_s(\boldsymbol{x}) = 3\cos(x_3) + 4\sin(x_4) + 2\sin(x_2)$ with separate component functions on each coordinate. We sample $m = 1000$ points uniformly from $[-1, 1]^5$ and apply SHRIMP with $N = 10000$ ($N_0 = 20000$ features), $q = q_* = 1$, and pruning rate $p = 20\%$. Figure 4 shows that both the sparse support set and the even/odd property are recovered by SHRIMP. At first, when $N_t = 20000$, the min ℓ_2-norm solution has many small weights distributed across false coordinates $\{x_3, x_5\}$. At the first key point $N_t = 8192$, most unnecessary weights on sin have been pruned; after $N_t = 879$, remaining weights are only on $\{x_2, x_3, x_4\}$ with correct even/odd partition. The best test MSE is at $N_t = 38$, where the resulting vector is extremely sparse

5. Note that since the training and inference time of SALSA and SSAM do not depend on the number of features N and SSAM is generally slower than SALSA due to the computation of ℓ_1 minimization, we provide two Ns for SHRIMP to compare their computational costs with SALSA.
6. Note that if we start SHRIMP with an extremely large N and the number of data is small ($m \ll N$), the elapsed time for first few iterations will be long and the training time will be longer than SALSA.

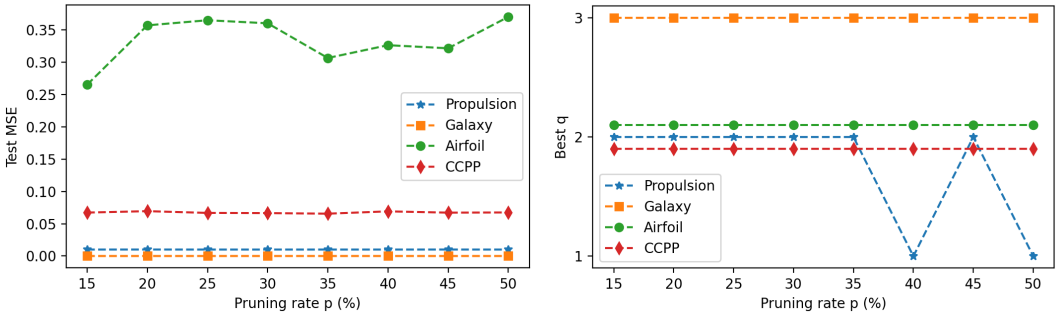

Figure 3: Robustness of pruning rate of SHRIMP shown by test MSE and the corresponding best q over four datasets. Note the lines of best q of Airfoil and CCPP are at 2 and shifted on purpose for illustration. Test MSE of propulsion is also shifted 0.01 up for illustration.

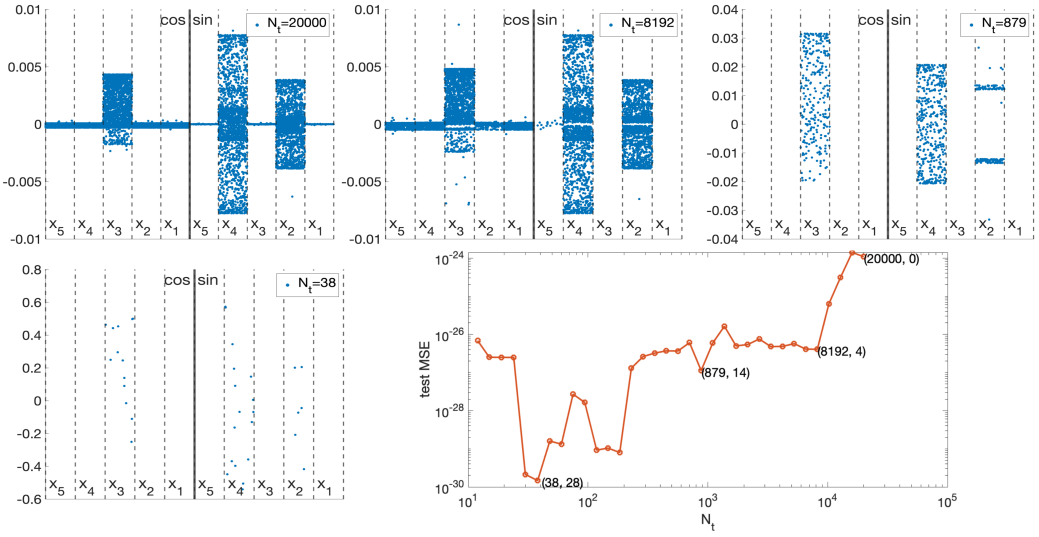

Figure 4: Illustration of support recovery of SHRIMP with $f_s(\boldsymbol{x}) = 3\cos(x_3) + 4\sin(x_4) + 2\sin(x_2)$. Top and bottom left: Support recovery plots with $N_t = 20000$ $(t = 0), 8192$ $(t = 4), 879$ $(t = 14), 38$ $(t = 28)$ with y-axis corresponding to the weight magnitude; Bottom right: Test MSE tracking of the pruning from $N_t = 20000$ to $N_t = 12$ with pruning iteration $t \in [0, 33]$.

and matches the support set exactly. Along the pruning process, the subnetworks found by SHRIMP maintain a comparative or better test error with only a small fraction of weights. In other words, winning tickets found by SHRIMP exhibit the ability to recover sparse low-order interactions in random feature models.

Benefits of Iterative Magnitude Pruning Compared to Random Pruning. We explore the role of IMP in sparse random feature models by showing the test MSE curves for approximating functions $\{f_2(\boldsymbol{x}), f_5(\boldsymbol{x}), f_7(\boldsymbol{x})\}$ as defined above, using a different number of features N_t (see Figure 5). We train and evaluate models with SHRIMP, minimal ℓ_2- and ℓ_1-norm (SRFE-S Hashemi et al.

(2021)) estimators with the same $\{N_t\}$ set. Figure 5 illustrates the role and benefit of IMP in finding sparse winning subnetworks. SHRIMP (in blue) has a similar computational cost compared to plain min-ℓ_2 (in orange), the only difference being in sorting and comparing the absolute weights; at the same time, SHRIMP achieves better test error with a sparser resulting model (i.e., lower N_t). The resulting sparse subnetwork behaves better than even the overparameterized solution of plain min ℓ_2 in the middle plot of $f_5(\boldsymbol{x})$, which also shows the double descent curve. SRFE-S is inefficient due to the computation of ℓ_1 minimization and is comparably flatter than the other two models, which indicates that it does not benefit from a smaller N_t solution. Hence, the pruning by IMP is efficient and obtains sparser and better subnetworks than the models obtained by ℓ_1 and ℓ_2 regularization.

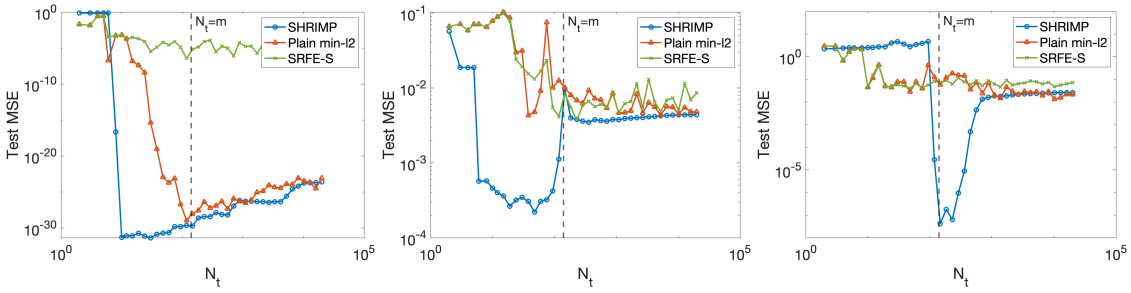

Figure 5: Test MSE of sparse random feature models obtained by SHRIMP, min ℓ_2-norm estimation, min ℓ_1-norm estimation (SRFE-S). From left to right: $f_2(\boldsymbol{x})$, $f_5(\boldsymbol{x})$, $f_7(\boldsymbol{x})$.

Spectrum of SHRIMP pruning compared to Random Pruning. Figure 6 shows the maximal and minimal eigenvalues of $A_S A_S^\top / N_t$ throughout pruning for the function $f_7(\boldsymbol{x}) = \cos(x_1)x_3 + x_2^2 x_4 + \sum_{j=3}^{d} x_j$ (Note that we observe similar spectrum patterns for other kinds of functions as well). We notice that for all methods excluding SHRIMP use variance $1/q$—which results in the best generalization error over all cases—these values are essentially constant (up to numerical instability for small N_t). The case for small variance is predicted by the random features approximation of Rahimi and Recht (2008c), as the kernel approximation is good, while the case for high variance is predicted by Hashemi et al. (2021), where mutual coherence is low. However, SHRIMP with low variance has a decreasing maximum eigenvalue throughout the pruning process, providing some explanation for the good performance of SHRIMP (with Theorem 5); it is important to both perform magnitude pruning and choose a proper variance, as SHRIMP is a two-stage procedure.

5. Theoretical Analysis

In this section, we first provide Theorem 5, improving the analysis of the generalization error for thresholded Basis Pursuit from Hashemi et al. (2021). Thresholded Basis Pursuit performs basis pursuit followed by a pruning step, keeping only the top s entries of the resulting coefficient vector. Our analysis refines the result of Hashemi et al. (2021) by exposing the role of the maximum singular value of $\boldsymbol{A}$ in the resulting generalization bound. Moreover, we remove the explicit dependence on the number of features N, demonstrating that a smaller maximum singular value indicates better generalization. For the proofs of all statements in this section, we refer the reader to the appendix.

For sake of comparing with SRFE-S in Hashemi et al. (2021), we restate SRFE-S according to our two-stage paradigm in Definition 4.

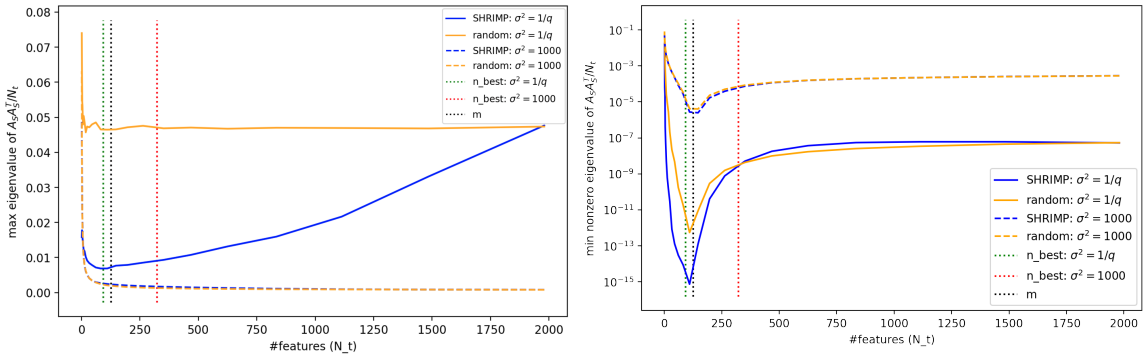

Figure 6: Maximal and minimal eigenvalue of $A_S A_S^\top/N_t$ ($A_S^\top A_S/N_t$ in the underparameterized setting) with weight vectors $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ for $f_{a_1}(\boldsymbol{x})$ (i.e., $f_7(\boldsymbol{x})$). Blue: SHRIMP; Orange: Random pruning. Test MSE for low variance $\sigma^2 = 1/q$ (solid): SHRIMP(1.19e-05); Random pruning (0.023). Test MSE for high variance $\sigma^2 = 1000$ (dashed): SHRIMP (4.52); Random pruning (4.07).

Definition 4 (Sparse Random Feature Expansion with Sparse Feature Weights (SRFE-S)) *With the same input as Algorithm 1 and a stability parameter η, SRFE-S constructs a random feature matrix $\boldsymbol{A}$ following Stage I in Algorithm 1 and solves*

$$\boldsymbol{c}^\sharp = \arg\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_1 \quad s.t. \quad \|\boldsymbol{A}\boldsymbol{c} - \boldsymbol{y}\| \leqslant \eta\sqrt{m} \tag{7}$$

in Stage II. The resulting pruned estimator $\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}$ by SRFE-S keeps the s largest (in magnitude) coefficients on the support set $\mathcal{S}^\sharp$ and sets $\boldsymbol{c}_j^\sharp = 0, \forall j \in [N]\backslash\mathcal{S}^\sharp$.

Theorem 5 (Generalization Bounds for Thresholded Basis Pursuit) *For a bounded ρ-norm function f as defined in Def. 2, construct the dictionary matrix $\boldsymbol{A}$ from Stage I in Algorithm 1 with m samples $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^m$, where $\boldsymbol{x}_k \sim \mathcal{N}(\mathbf{0}, \gamma^2 \boldsymbol{I}_d)$, $y_k = f(\boldsymbol{x}_k) + e_k$ with $|e_k| \leqslant 2\nu$ or $e_k \sim \mathcal{N}(0, \nu^2)$, and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_d)$, and $\phi(\boldsymbol{x}; \boldsymbol{\omega}) = \exp(i\langle \boldsymbol{x}, \boldsymbol{\omega}\rangle)$. Assume the conditions the following conditions: (1) $\gamma^2\sigma^2 \geqslant \frac{1}{2}\left(\left(\frac{\sqrt{41}(2s-1)}{2}\right)^{\frac{2}{d}} - 1\right)$; (2) number of features satisfies $N = \frac{4}{\epsilon^2}\left(1 + 4\gamma\sigma d\sqrt{1 + \sqrt{\frac{12}{d}\log\frac{m}{\delta}}} + \sqrt{\frac{1}{2}\log\frac{1}{\delta}}\right)^2$; (3) number of measurements $m \geqslant 4(2\gamma^2\sigma^2 + 1)^d \log\frac{N^2}{\delta}$. Suppose $f_{\mathcal{S}^\sharp}^\sharp$ is estimated by BP (i.e., min ℓ_1-norm estimator) with $\eta = \min\{\eta', \tilde{\eta}\}$, where $\eta' = \sqrt{2(\epsilon^2 \|f\|_\rho^2 + 4\nu^2)} + \sqrt{\frac{\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})}{m}}\kappa_{s,2}(\boldsymbol{c}^\star)$ and $\tilde{\eta} = 2\sqrt{\epsilon^2\|f\|_\rho^2 + 2\nu^2 + \kappa_{s,1}^2(\boldsymbol{c}^\star)}$ with approximation error ϵ and $\kappa_{s,p}(\boldsymbol{c}) := \min\{\|\boldsymbol{c} - \boldsymbol{z}\|_{\ell_p} : \boldsymbol{z} \text{ is } s\text{-sparse}\}$. Apply an additional pruned step with sparsity s, then with probability at least $1 - 5\delta$, the generalization error is bounded by*

$$\sqrt{\int_{\mathbb{R}^d} \left|f_{\mathcal{S}^\sharp}^\sharp(\boldsymbol{x}) - f^\star(\boldsymbol{x})\right|^2 d\mu} \leqslant \left(\frac{8}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}} \left(2s\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 + (\kappa_{s,1}(\boldsymbol{c}^\star))^2\right)^{\frac{1}{2}} + 2\left\|\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}_s^\star\right\|_2 + \kappa_{s,1}(\boldsymbol{c}^\star), \tag{8}$$

where

$$\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2 \leqslant C\min\left\{2\sqrt{\epsilon^2\|f\|_\rho^2 + 2\nu^2 + \kappa_{s,1}^2(\boldsymbol{c}^\star)}, \sqrt{2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)} + \sqrt{\frac{\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})}{m}}\epsilon\|f\|_\rho\right\},$$

and $\kappa_{s,1}(\boldsymbol{c}^\star)$ *is bounded by* $\frac{N-s}{N}\|f\|_\rho$.

Theorem 5 connects the numerical results on the decaying maximum singular value of $\boldsymbol{A}$, the empirical success of SHRIMP, and refined theory from a similar setting (finding a sparse coefficient vector in a low-order random features model that has good generalization error). Moreover, in our new analysis, the role of the norm of the smallest entries of the coefficient vector is explicitly revealed: the smaller the norm of the vector of small entries, the better the implied generalization. This result is directly connected to Belkin et al. (2020). Thus, we connect the ℓ_1-based methods of previous work with ℓ_2-based methods through this new analysis.

Corollary 6 (Generalization Bounds for Order-q Functions) *Fix* $\epsilon > 0$. *For an order-*q *function as in Def. 1 with at most* K *terms, and fix the sparsity* $s = nK$ *with* $N = n\binom{d}{q}$ *and* $K \ll \binom{d}{q}$. *Assume the following conditions:* $\gamma^2\sigma^2 \geqslant \frac{1}{2}\left(\left(\frac{\sqrt{41}(2s-1)}{2}\right)^{\frac{2}{q}} - 1\right)$, *number of features* $N = \frac{4}{\epsilon^2}\left(1 + 4\gamma\sigma d\sqrt{1+\sqrt{\frac{12}{d}\log\frac{m}{\delta}}} + \sqrt{\frac{q}{2}\log\frac{d}{\delta}}\right)^2$, *and number of measurements* $m \geqslant 4(2\gamma^2\sigma^2+1)^{\max\{2q-d,0\}}(\gamma^2\sigma^2+1)^{\min\{2q,2d-2q\}}\log\frac{N^2}{\delta}$. *Then the generalization error corresponding to the thresholded* ℓ_1 *estimator with the* s *largest elements (in magnitude) is bounded by* $\mathcal{O}\left(\left(1 + C's^{\frac{1}{2}}m^{-\frac{1}{4}}\log^{\frac{1}{4}}(\frac{1}{\delta})\right)\sqrt{\epsilon^2\binom{d}{q}|||f|||^2 + E^2}\right)$ *with probability at least* $1-5\delta$, *where* $|||f||| := \frac{1}{K}\sum_{j=1}^K \|g_j\|_\rho$.

Note that Corollary 6 improves the generalization bound of SRFE-S (Def. 4) in Hashemi et al. (2021) from depending on the number of features N to the sparsity level s in the $1 + C's^{\frac{1}{2}}m^{-\frac{1}{4}}\log^{\frac{1}{4}}(\frac{1}{\delta})$ term.

Then we shed light on the maximal and minimal eigenvalues of the Gram matrices of SHRIMP observed in Figures 6. The bounds in Proposition 7 below are obtained using techniques inspired by Chen and Schaeffer (2021).

Proposition 7 (Bounds on Eigenvalues of Gram Matrix) *Consider data* $\{\boldsymbol{x}_1,\dots,\boldsymbol{x}_m\}$ *i.i.d. drawn from* $\mathcal{N}(0,\gamma^2\boldsymbol{I}_d)$, *weights* $\{\boldsymbol{\omega}_1,\dots,\boldsymbol{\omega}_N\}$ *i.i.d. drawn from* $\mathcal{N}(0,\sigma^2\boldsymbol{I}_q)$, *and the Fourier feature matrix* $a_{j,k} = \phi(\boldsymbol{x}_j,\boldsymbol{\omega}_k)$, *where* $\phi(\boldsymbol{x},\boldsymbol{\omega}) = \exp(i\langle\boldsymbol{x},\boldsymbol{\omega}\rangle)$. *Fix the feature sparsity* $q \leqslant d$ *as in Def. 1 and consider the regime* $m \leqslant N$. *Let* $\lambda_k(\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*)$ *be the* k*th eigenvalue of the scaled Gram matrix. Then the expectation of the maximum eigenvalue* λ_1 *and the minimum eigenvalue* λ_m *of the matrix* $\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*$ *satisfy*

$$\mathbb{E}\lambda_1 \geqslant 2 - \frac{(N-1)m}{N^2} + \frac{(N-1)(m^2-m)}{N^2}\left(4\gamma^2\sigma^2+1\right)^{-\frac{q}{4}}, \tag{9}$$

$$\mathbb{E}\lambda_m \leqslant \frac{c-1}{c} + \frac{1}{m} + \left(\frac{c-1}{c}m+1\right)\left(4\gamma^2\sigma^2+1\right)^{-\frac{q}{4}}, \tag{10}$$

where $c = N/m$.

Remark 8 *Using Markov's inequality, for* $c \to 1^+$, *i.e.* $N = m$, *we have*

$$\lambda_m\left(\frac{1}{N}AA^*\right) \leqslant N^p\left(4\gamma^2\sigma^2+1\right)^{-\frac{q}{4}} + N^{p-1} \tag{11}$$

with probability $1 - N^{-p}$ *for* $0 < p < 1$. *If* $\gamma^2\sigma^2 = \mathcal{O}(1)$, *then we observe the benefit of small* q *(i.e., low-order interactions); if* $q \to d$ *(in the high dimensional setting), then the minimum eigenvalue becomes arbitrarily small while the maximum remains above 2, and thus the system is ill-conditioned. In particular, the conditioning is directly related to the size of* $\left(4\gamma^2\sigma^2 + 1\right)^{-\frac{q}{4}}$.

Further Discussion. We connect our results to ℓ_0-based methods. First we note the explicit connection to SINDy (Zhang and Schaeffer, 2019), which algorithmically is similar to SHRIMP, except instead of pruning the smallest magnitude coefficients, it prunes the all entries smaller than some threshold. However, their results are about coefficient recovery, and generalization bounds for ℓ_0-based methods are sparse in the literature. From Remark 2 in Nikolova (2013), the iterates of SHRIMP are *each* local minimizers of an ℓ_0-regularized problem, which gives insight to the behavior of SHRIMP; However, SHRIMP arrives in an adaptive and greedy nature, depending on the solution of the previous one. Further discussion on these topics is given in the appendix.

6. Discussion

We propose a new method, **S**parser **R**andom Feature Models with **IMP**, to exploit low-order additive structure in a learning problem, which often occurs in many domains of interest. In this method, we explicitly construct a sufficiently overparameterized sparse feature matrix in order to approximate a given underlying low-order function, and then prune coefficients by adaptively solving a min ℓ_2-norm problem and applying iterative magnitude pruning. This can be seen as an instance of feature selection or neuron pruning in the neural network pruning literature. We test our method on both synthetic and real datasets: SHRIMP vastly exceeds other methods on synthetic data; it is often better or at least competitive on real datasets. We illustrate the relationship between low-order structure and pruning, corresponding to weight and neuron pruning, respectively, and show the IMP has the greatest effect when combined with sparse feature models. Our analysis provides generalization bounds for thresholded BP and bounds on eigenvalues of Gram Matrix, which explains the benefits of our method. We hope to shed some light on the lottery ticket hypothesis in a simple model, similar to how regression is once again being studied in the context of deep learning theory; our method corresponds to certain pruning methods in two-layer neural networks.

More robust generalization bounds can be given to our SHRIMP model—for example, in the context of random features regression, studying the eigenspectrum of a pruned sub-Gram matrix throughout our algorithm is a possible extension of our work. Another possible future direction is to adaptively discover the low-order structure as we go, instead of fixing the parameter q in advance. This results in a setting more closely tied to practical pruning and allows for greater flexibility (e.g., if the underlying function is a sum of functions of various orders), and may shed light on what a pruned network is learning.

Acknowledgments

B. Shi, R. Ward, and Y. Xie were supported in part by AFOSR MURI FA9550-19-1-0005, NSF DMS 1952735, NSF HDR-1934932, and NSF 2019844. H. Schaeffer was supported in part by AFOSR MURI FA9550-21-1-0084 and NSF DMS-1752116. We thank Jiayi Wei for insightful discussion and providing the robotics data.

References

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 1950.

Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *International Conference on Machine Learning*, pages 253–262. PMLR, 2017.

Jimmy Ba, Murat Erdogdu, Taiji Suzuki, Denny Wu, and Tianzong Zhang. Generalization of two-layer neural networks: An asymptotic viewpoint. In *International conference on learning representations*, 2019.

Francis R Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9(6), 2008.

Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, 2(4):1167–1180, 2020.

Colin Campbell. Kernel methods: a survey of current techniques. *Neurocomputing*, 48(1-4):63–84, 2002.

Zhijun Chen and Hayden Schaeffer. Conditioning of random feature matrices: Double descent and generalization error. *arXiv preprint arXiv:2110.11477*, 2021.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

Corinna Cortes, Mehryar Mohri, and Ameet Talwalkar. On the impact of kernel approximation on learning accuracy. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, pages 113–120. PMLR, 2010.

Ronald DeVore, G. Petrova, and Przemyslaw Wojtaszczyk. Approximation of functions of few variables in high dimensions. *Constructive Approximation*, 33:125–143, 02 2011.

Stéphane d'Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double trouble in double descent: Bias and variance (s) in the lazy regime. In *International Conference on Machine Learning*, pages 2280–2290. PMLR, 2020.

Bryn Elesedy, Varun Kanade, and Yee Whye Teh. Lottery tickets in linear models: An analysis of iterative magnitude pruning. *arXiv preprint arXiv:2007.08243*, 2020.

Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013. ISBN 0817649476.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

Kameron Decker Harris. Additive function approximation in the brain. *arXiv preprint arXiv:1909.02603*, 2019.

Abolfazl Hashemi, Hayden Schaeffer, Robert Shi, Ufuk Topcu, Giang Tran, and Rachel Ward. Generalization bounds for sparse random feature expansions. *arXiv preprint arXiv:2103.03191*, 2021.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

Jian Huang, Joel L. Horowitz, and Fengrong Wei. Variable selection in nonparametric additive models. *The Annals of Statistics*, 38(4):2282 – 2313, 2010.

T. Ishigami and T. Homma. An importance quantification technique in uncertainty analysis for computer models. In *[1990] Proceedings. First International Symposium on Uncertainty Modeling and Analysis*, 1990.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

Arthur Jacot, Berfin Simsek, Francesco Spadaro, Clément Hongler, and Franck Gabriel. Implicit regularization of random feature models. In *International Conference on Machine Learning*. PMLR, 2020.

Kirthevasan Kandasamy and Yaoliang Yu. Additive approximations in high dimensional nonparametric regression via the salsa. In *International conference on machine learning*, pages 69–78. PMLR, 2016.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling techniques for the nystrom method. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 304–311. PMLR, 16–18 Apr 2009.

Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

Frances Kuo, Ian Sloan, Grzegorz Wasilkowski, and Henryk Wozniakowski. On decompositions of multivariate functions. *Math. Comput.*, 79:953–966, 04 2010.

Tengyuan Liang and Alexander Rakhlin. Just interpolate: Kernel "ridgeless" regression can generalize. *The Annals of Statistics*, 48(3), Jun 2020. ISSN 0090-5364.

Zhenyu Liao, Romain Couillet, and Michael Mahoney. A random matrix analysis of random fourier features: beyond the gaussian kernel, a precise phase transition, and the corresponding double descent. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Guodong Liu, Hong Chen, and Heng Huang. Sparse shrunk additive models. In *International Conference on Machine Learning*, pages 6194–6204. PMLR, 2020.

Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

Andrea Montanari, Feng Ruan, Youngtak Sohn, and Jun Yan. The generalization error of max-margin linear classifiers: High-dimensional asymptotics in the overparametrized regime. *arXiv preprint arXiv:1911.01544*, 2019.

Mila Nikolova. Description of the minimizers of least squares regularized with ℓ_0-norm. uniqueness of the global minimizer. *SIAM Journal on Imaging Sciences*, 6(2):904–937, 2013.

Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33, 2020.

Ayca Özcelikkale. Sparse recovery with non-linear fourier features. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5715–5719. IEEE, 2020.

Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2599–2610. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1b742ae215adf18b75449c6e272fd92d-Paper.pdf.

Daniel Potts and Michael Schmischke. Approximation of high-dimensional periodic functions with fourier-based methods. *SIAM Journal on Numerical Analysis*, 59(5):2393–2429, 2021a.

Daniel Potts and Michael Schmischke. Interpretable approximation of high-dimensional data. *arXiv preprint arXiv:2103.13787*, 2021b.

Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, 2008a.

Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: replacing minimization with randomization in learning. In *NIPS*. Citeseer, 2008b.

Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008c.

Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's Hidden in a Randomly Weighted Neural Network? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11890–11899, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5.

Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman. Sparse additive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(5):1009–1030, 2009.

E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008. doi: 10.1137/080714488. URL http://link.aip.org/link/?SCE/31/890.

E. van den Berg and M. P. Friedlander. SPGL1: A solver for large-scale sparse reconstruction, December 2019. https://friedlander.io/spgl1.

Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):480–492, 2012. doi: 10.1109/TPAMI.2011.153.

Jiayi Wei, Jarrett Holtz, Isil Dillig, and Joydeep Biswas. Steady: Simultaneous state estimation and dynamics learning from indirect observations. *arXiv preprint arXiv:2203.01299*, 2022.

Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, 2010.

Ian En-Hsu Yen, Ting-Wei Lin, Shou-De Lin, Pradeep K Ravikumar, and Inderjit S Dhillon. Sparse random feature algorithm as coordinate descent in hilbert space. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Junming Yin, Xi Chen, and Eric P Xing. Group sparse additive models. In *Proceedings of International Conference on Machine Learning*. NIH Public Access, 2012.

Linan Zhang and Hayden Schaeffer. On the convergence of the sindy algorithm. *Multiscale Modeling & Simulation*, 17(3):948–972, 2019.

Shuai Zhang, Meng Wang, Sijia Liu, Pin-Yu Chen, and Jinjun Xiong. Why lottery ticket wins? a theoretical perspective of sample complexity on pruned neural networks. *arXiv preprint arXiv:2110.05667*, 2021a.

Tong Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17(9):2077–2098, 2005.

Zhenyu Zhang, Xuxi Chen, Tianlong Chen, and Zhangyang Wang. Efficient lottery ticket finding: Less data is more. In *International Conference on Machine Learning*, pages 12380–12390. PMLR, 2021b.

Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in Neural Information Processing Systems*, 32:3597–3607, 2019.

Appendix

The appendix is organized as follows:

- Appendix A: Experimental Details of Function Approximation
- Appendix B: Additional Experiments
- Appendix C: Proofs of Theorems in Section 5
- Appendix D: Further Discussion

Appendix A. Experimental Details of Function Approximation

Data Generation. We generate the data $\boldsymbol{X} \times \boldsymbol{y} \in \mathbb{R}^{m\times d} \times \mathbb{R}^m$ in the following way: (1) sample m d-dimensional points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ with $\boldsymbol{x}_j \sim \mathrm{Unif}[-1,1]^d$, except for the Ishigami function [7] from $\mathrm{Unif}[-\pi,\pi]^d$; (2) For each, function f_i, $y_j = f_i(\boldsymbol{x}_j)$. Note that we include no additive noise in our experiments, although the observed behavior is robust to the presence of noise.

Experimental Set-up. In the function approximation experiments, models are evaluated in both low-dimensional (s_l^{q*} and s_l^d with $m = 140, d = 10$) and high-dimensional (s_h^{q*} and s_h^d with $m = 1400, d = 100$) settings. The results for $q = q_*$ are in Table 2, and the full results with both sparse ($q = q_*$) and dense features ($q = d$) are in Table 5 in the appendix. For SRFE-S, Min ℓ_2, and SHRIMP in s_l^{q*} and s_h^{q*}, we sample $\mathbf{w}$ according to Def. 3 and $\rho = \mathcal{N}\left(\mathbf{0}, q^{-1}I_q\right)$ with $q = q_*$ as the actual order of those low-order functions.[8], We set $n = N/\binom{d}{q}$ with $N = 10000$ in our experiments and form the random feature matrix $\boldsymbol{W} \in \mathbb{R}^{N\times d}$. The dictionary $\boldsymbol{A} = [\cos(\boldsymbol{X}\boldsymbol{W}^\top), \sin(\boldsymbol{X}\boldsymbol{W}^\top)] \in \mathbb{R}^{m\times 2N}$. For SHRIMP, we set 0.2 as the pruning rate and validate on 10% of the training set to choose the best pruned model. For SALSA, we form the kernel matrix $\boldsymbol{K}$ by $K_{ij} = \sum_{k=1}^{\binom{d}{q}} \exp\left(-\frac{\left\|\boldsymbol{x}_i|_{\mathcal{S}_k} - \boldsymbol{x}_j|_{\mathcal{S}_k}\right\|^2}{2q}\right)$; note that $\mathbb{E}_W[\boldsymbol{A}\boldsymbol{A}^\top] = \boldsymbol{K}$.

Each model is evaluated by the average of test mean squared errors over three runs. For the sake of completeness, we also experiment with the same functions with $q = d$ for all functions (s_l^d and s_h^d in Table 2—which corresponds to standard kernel regression and random feature regression with standard Gaussian kernel. Here, random weights are drawn $\mathbf{W} \sim \mathcal{N}\left(\mathbf{0}, d^{-1}I_d\right)$ and fully dense.

Appendix B. Additional Experiments

B.1. Comparing Function Approximations with Sparse and Dense Features

Table 5 shows the full results of function approximation with sparse ($q = q_*$) and dense features ($q = d$). As shown in Table 5, for both the low-dimensional (s_l^{q*}, s_l^d) and high-dimensional (s_h^{q*}, s_h^d) settings, models with $q = q_*$ (i.e., s_l^{q*} and s_h^{q*}), where the low order q matches the actual order of functions, have significantly better performance for all methods over corresponding models with $q = d$ (i.e., s_l^d and s_h^d). This shows the benefit of our use of low-order structure compared to previous random features work with dense features. However, with dense features $q = d$, the advantage of

7. It is the traditional sampling way for Ishigami function.
8. In the low-order case, since the actual orders are known and are small enough such that $\binom{d}{q} < N$, we can use the actual order q_*.

pruning over other methods fades: pruning over all functions performs comparably to standard ℓ_1 and ℓ_2 based methods in both low and high dimensions since all features add to the representative capacity, and when SHRIMP does perform worse, it is very slight. This is also exhibited as the average size of the model is much larger when using dense features as opposed to sparse features.

Table 5: Full comparison of the test errors of SHRIMP, Min ℓ_2, SALSA, and SRFE-S with $q = q_*$ and $q = d$. *Avg size denotes the average pruned model size of SHRIMP over three runs. $^\dagger s$ denotes the setting with different (m, d, q) pairs: $s_l^{q*} = (140, 10, q_*), s_l^d = (140, 10, d), s_h^{q*} = (1400, 100, q_*), s_h^d = (1400, 100, d)$, where l denotes low dimension and h denotes high dimension. The best MSE for each $f_i(\boldsymbol{x})$ over all models is in purple. In s_l^{q*} and s_h^{q*}, all the methods use $q = q_*$ (the ground-truth low order) to interpolate the functions and achieve better test performance than s_l^d and s_h^d (which use dense features), respectively. From the comparison of s_l^{q*} and s_h^{q*}, our SHRIMP method is scalable to the high-dimensional setting.

Setting†	Model	$f_1(\boldsymbol{x})$	$f_2(\boldsymbol{x})$	$f_3(\boldsymbol{x})$	$f_4(\boldsymbol{x})$	$f_5(\boldsymbol{x})$	$f_6(\boldsymbol{x})$	$f_7(\boldsymbol{x})$
s_l^{q*}	SRFE-S	7.85e-04	1.98e-05	1.15e-01	1.27e-01	7.52e-03	1.11	7.49e-02
	Min ℓ_2	4.37e-20	5.45e-24	8.20e-02	5.94e-02	7.36e-03	7.18	2.98e-02
	SALSA	1.59e-12	1.26e-15	8.80e-02	6.14e-02	7.32e-03	6.99	2.72e-02
	SHRIMP	1.37e-22	7.90e-32	4.98e-12	2.54e-12	6.39e-04	2.58e-02	2.83e-05
	Avg size*	3100.33	29	147	171.67	39	80.33	187
s_l^d	SRFE-S	5.35e-02	1.47e-03	5.56e-02	1.64e-01	4.11e-03	1.27e+01	9.17e-02
	Min ℓ_2	1.71e-02	1.93e-03	3.50e-02	1.01e-01	4.62e-03	1.45e+01	6.15e-02
	SALSA	1.68e-02	1.91e-03	3.38e-02	9.67e-02	4.62e-03	1.44e+01	6.05e-02
	SHRIMP	1.70e-02	1.63e-03	3.51e-02	9.81e-02	6.57e-03	1.60e+01	5.91e-02
	Avg size	6726	77.67	1872	12020	450.33	5342.33	39.33
s_h^{q*}	SRFE-S	1.52e-03	8.71e-06	1.99	4.54	1.16e-01	4.59	4.40e-01
	Min ℓ_2	1.68e-20	3.51e-24	2.01	4.75	1.16e-01	8.34	1.49e-01
	SALSA	1.99e-11	2.54e-13	1.60	3.16	8.49e-02	7.35	1.34e-01
	SHRIMP	1.61e-22	1.11e-30	1.26e-02	5.11e-01	1.50e-02	2.68	5.82e-02
	Avg size	3355	19	61.33	64	42.67	13	229
s_h^d	SRFE-S	1.43e-01	2.35e-02	1.55e+00	3.05e+00	8.20e-02	1.69e+01	2.06e-01
	Min ℓ_2	6.01e-02	2.33e-02	1.55e+00	3.05e+00	8.20e-02	1.40e+01	8.33e-02
	SALSA	2.62e+04	1.23e+03	4.63e+03	3.34e+04	3.20e+02	2.44e+04	4.66e+04
	SHRIMP	6.07e-02	2.35e-02	1.55e+00	3.07e+00	9.33e-02	1.41e+01	8.32e-02
	Avg size	6663	14933.33	8199	6908.67	717	14730.67	8328.67
	Order q_*	1	1	2	2	3	2	2

B.2. Additional Iterative Magnitude Pruning Curves

We show comprehensive curves in Figure 7 and 8 to illustrate the role of IMP in sparse random feature models using the functions defined in Section 4 with different number of features N_t. For sake of completeness, Figure 7 shows more types of test MSE curves with approximating functions $f_1(\boldsymbol{x}), f_3(\boldsymbol{x}), f_4(\boldsymbol{x}), f_6(\boldsymbol{x})$, which are not included in Section 4 due to page limit. Except for $f_1(\boldsymbol{x})$,

where the pruned curve has a little better test performance over the best solution of SRFE-S but with more number of features since the coefficient vector of function $f_1(\boldsymbol{x}) = \sum_{i=1}^{d-1} x_i + \exp(-x_d)$ is comparably dense with random sparse features, SHRIMP find sparser estimators with better performance than SRFE-S and plain min ℓ_2-norm estimator on other functions.

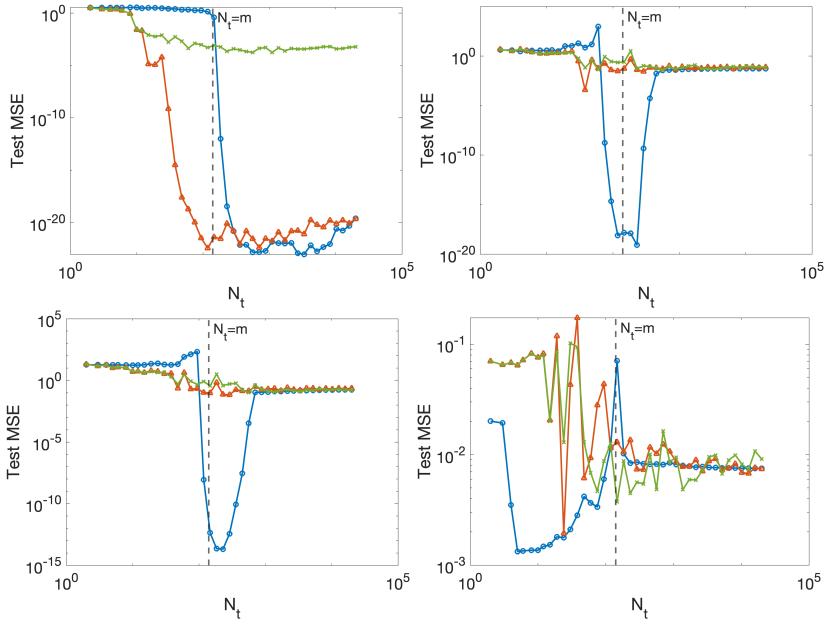

Figure 7: Test MSE of sparse random feature models obtained by SHRIMP, min ℓ_2-norm estimation, min ℓ_1-norm estimation (SRFE-S). From left to right and from top to bottom: $f_1(\boldsymbol{x}), f_3(\boldsymbol{x}), f_4(\boldsymbol{x}), f_6(\boldsymbol{x})$ as defined in Section 4. Blue: SHRIMP; Orange: Plain min ℓ_2; Green: SRFE-S.

Furthermore, we include additional curves comparing the aforementioned methods to the naive pruning method (where the weights are kept fixed without retraining after each pruning step) in Figure 8. As shown in those figures, naive pruning usually has worse performance than the original minimal ℓ_2-norm solution, let alone SHRIMP, which verifies the benefit of retraining from the same initialization of iterative magnitude pruning as Frankle and Carbin (2019) suggests.

B.3. Results on High-Order Functions

In addition to low-order functions, we present test MSE curves of high-order functions with different methods in Figure 9 for completeness. All experiments are with $m = 140, d = 10, q = d$ since the ground-truth order $q^* = d$, and the same experimental settings as the synthetic experiments in Section 4. For $f_{h_1}(\boldsymbol{x})$ and $f_{h_2}(\boldsymbol{x})$, SHRIMP results in better performance with sparser models. However, for $f_{h_3}(\boldsymbol{x})$, which has underlying dense weights, SHRIMP can only have comparable performance to SRFE-S but with a sparser coefficient vector.

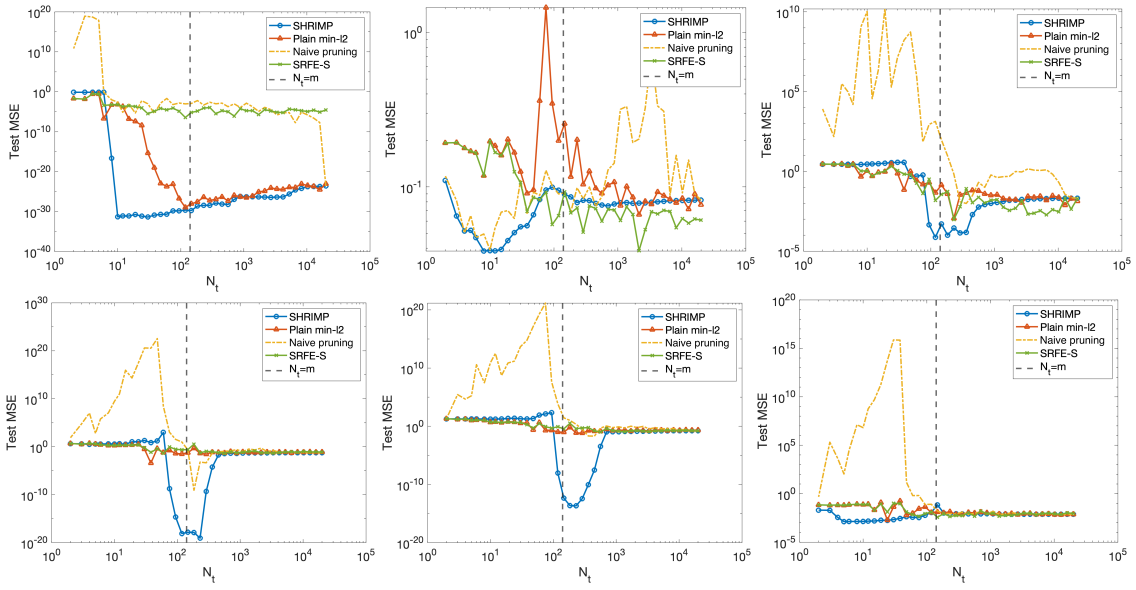

Figure 8: Test MSE of sparse random feature models obtained by SHRIMP, native pruning method, min ℓ_2-norm estimation, min ℓ_1-norm estimation (SRFE-S). From top to bottom and left to right: $f_2(\boldsymbol{x}), f_5(\boldsymbol{x}), f_7(\boldsymbol{x}), f_3(\boldsymbol{x}), f_4(\boldsymbol{x}), f_6(\boldsymbol{x})$ as defined in Section 4, respectively.

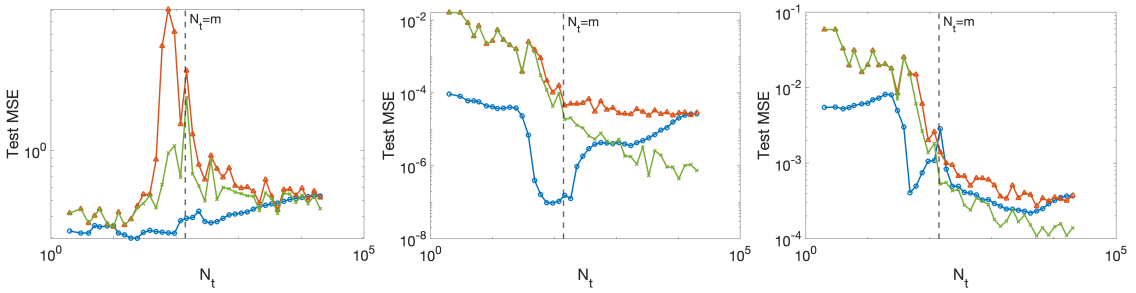

Figure 9: Test MSE of high-order functions approximated by sparse random feature models obtained by SHRIMP, min ℓ_2-norm estimation, min ℓ_1-norm estimation (SRFE-S). Left: $f_{h_1}(\boldsymbol{x}) = \sin(\sum_{i=1}^{d} x_i)$; Middle: $f_{h_2}(\boldsymbol{x}) = \cos(\prod_{i=1}^{d} x_i)$; Right: $f_{h_3}(\boldsymbol{x}) = (1 + \|\boldsymbol{x}\|_2)^{-1/2}$. Blue: SHRIMP; Orange: Plain min ℓ_2; Green: SRFE-S.

B.4. Experiments on Different Variances of Random Features

We show the performance of random feature models with different variances of $\boldsymbol{w}$ on functions with varying smoothness. We compare SHRIMP and min ℓ_2 solutions on the following functions:

- Sum of low-frequency functions only (i.e., smooth function): $f_{a_1(\boldsymbol{x})} = \cos(x_1)x_3 + x_2^2 x_4 + \sum_{j=3}^{d} x_j$
- Low-freq + High-freq: $f_{a_2(\boldsymbol{x})} = \cos(x_1 + x_2) + 5\cos(2x_3 + 10x_4)$

- High-freq + High-freq: $f_{a_3(\boldsymbol{x})} = \sin(9x_1) + 10\cos(10x_2)$

For all functions, we set $d = 10, q = q_*, m = 200, N = 1500$, and average over 10 trials. We give the test errors of SHRIMP vs the minimum ℓ_2-norm solution at three different variances: $1/q, 1, 100$ in Table 6. For the smooth function f_{a_1} with only low-frequency component functions, SHRIMP outperforms the min ℓ_2 norm at all corresponding variances, and as σ^2 increases, performance degrades. This implies that sampling at lower frequencies may have an implicit bias toward smooth functions. When there are higher-frequency component functions, such as f_{a_2} and f_{a_3}, performance improves as σ^2 increases; in this case, we need to sample at higher frequencies. Moreover, for f_{a_3}, increasing σ^2 allows for more gains: for this function a $\sigma^2 \approx 170$ seems to do the best from a course sweep over σ^2 ranging up to 200.

We can visualize the behavior of different types of functions with high variance $\boldsymbol{w}$ through the spectrum in Figure 10. We plot the maximum eigenvalue of the Gram matrix throughout SHRIMP and random pruning for f_{a_1} and f_{a_3}, where the weights are drawn from $\mathcal{N}(\mathbf{0}, 100\boldsymbol{I})$, a relatively high variance. We notice that for f_{a_1}, the maximum eigenvalue of SHRIMP qualitatively matches that of random pruning throughout the pruning process. However, for f_{a_3}, the behavior of the maximum eigenvalue of SHRIMP seems to more closely match that of SHRIMP with low variance on smooth functions (such as those given in the main paper), where the maximum eigenvalue of SHRIMP is smaller than that of random pruning for essentially the entire pruning process. We discuss some explanations for this behavior in Section D, as well as some limitations of our current theory in explaining this phenomenon.

Model	min ℓ_2			SHRIMP			Optimal q_*
σ^2	$1/q$	1	100	$1/q$	1	100	
f_{a_1}	0.041	0.034	2.452	1.19e-05	2.99e-04	0.137	2
f_{a_2}	55.205	40.023	9.734	14.411	12.778	4.935	2
f_{a_2}	186.283	184.581	47.864	73.381	50.650	10.334	1

Table 6: Test MSE of $\{f_{a_i}\}_{i=1}^3$ with random feature models with different variances.

B.5. Experiments on Kernel Approximation

We show an example of kernel approximation with $f(\boldsymbol{x}) = x_4^2 + x_2x_3 + x_1x_2 + x_4$ and $d = 5$ in this section to illustrate the benefit of sparse random features beyond kernel approximation capacity. As Figure 11 shows, the minimal ℓ_2 regression on a random features model is equivalent to the minimum RKHS kernel regression on the kernel matrix corresponding to the random feature matrix. In Figure 11, the blue lines represent the test MSE of the minimal ℓ_2 estimator for sparse random feature models (i.e., $\hat{\boldsymbol{c}} = \boldsymbol{A}^\dagger \boldsymbol{y}$) with increasing number of features (N), while the orange lines represent the test MSE of estimators from kernel regression with the kernel $\boldsymbol{K}$ defined in Section 4. Notice that the top-middle plot, $q = 2$ equals the actual order q_* of the function, and exhibits very interesting behavior as N grows. Instead of asymptotically approaching the orange line as the plots with $q = 3, 4, 5$ do, the test error curve of the sparse random feature model with min ℓ_2-norm estimator is significantly better than what can be obtained with kernel regression, which is

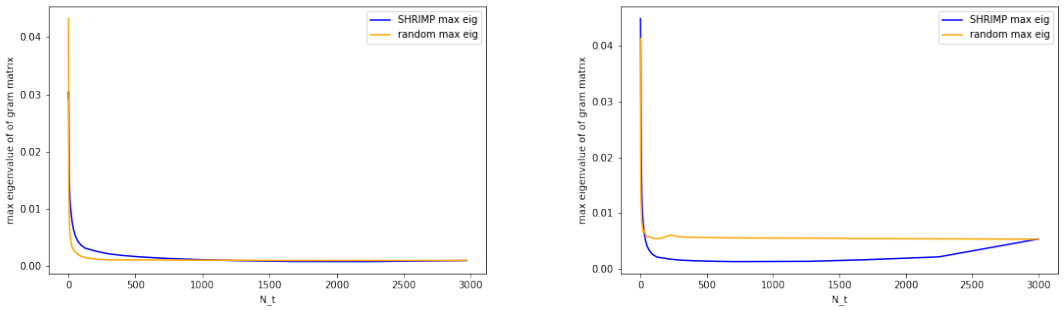

Figure 10: Maximum eigenvalue of $A_S A_S^\top / N_t$ with weight vectors drawn from $\mathcal{N}(\mathbf{0}, 100\boldsymbol{I})$. Left: f_{a_1}; Right: f_{a_3}.

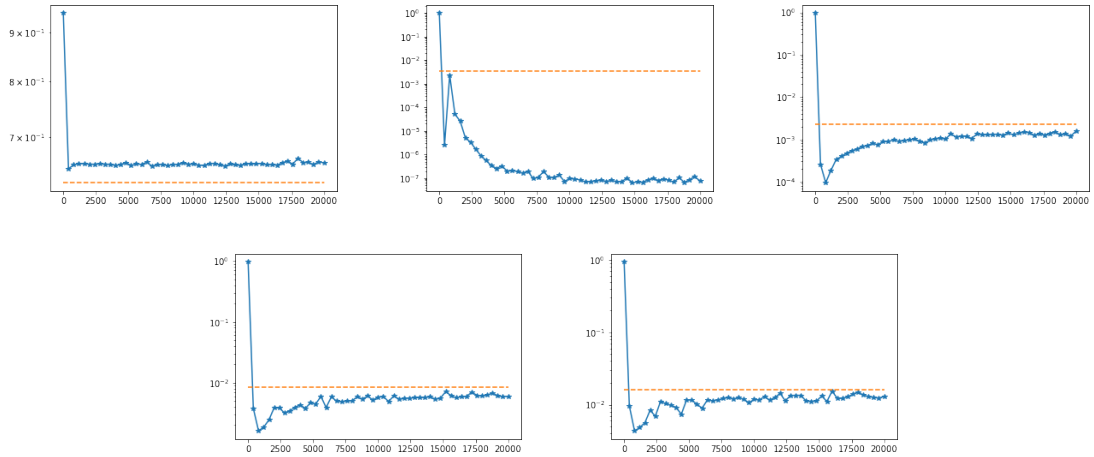

Figure 11: Illustration of benefit of sparse random feature models beyond a RKHS understanding by comparing with kernel approximation method on $f(\boldsymbol{x}) = x_4^2 + x_2x_3 + x_1x_2 + x_4$. From top to down and from left to right: $q = 1, 2, 3, 4, 5$.

surprising. This implies a *benefit of sparse random feature models beyond a RKHS understanding*, which would necessitate a more robust statistical study beyond approximation capacities.

Appendix C. Proofs of Theorems in Section 5

C.1. Proof of Theorem 5

Proof [Proof of Theorem 5] Denote $\boldsymbol{c}^\sharp$ as the minimal ℓ_1 norm solution obtained by basis pursuit, $\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}$ as the pruned solution with zeros on $i \notin \mathcal{S}^\sharp$, where $\mathcal{S}^\sharp$ is the support set of the s largest coefficients of $\boldsymbol{c}^\sharp$, and $\boldsymbol{c}_s^\star$ as $\boldsymbol{c}^\star$ supported on $\mathcal{S}^\star$, which is the support set of the s largest coefficients

of $\boldsymbol{c}^\star$. Since both $\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}$ and $\boldsymbol{c}_s^\star$ are s-sparse, we have for any $\boldsymbol{z}$,

$$\begin{aligned}
\left|f_{\mathcal{S}^\sharp}^\sharp(\boldsymbol{z}) - f^\star(\boldsymbol{z})\right|^2 &= \left|[\phi(\boldsymbol{z};\boldsymbol{\omega}_1),\ldots,\phi(\boldsymbol{z};\boldsymbol{\omega}_N)](\boldsymbol{c}^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp})\right|^2 \\
&= \left|[\phi(\boldsymbol{z};\boldsymbol{\omega}_1),\ldots,\phi(\boldsymbol{z};\boldsymbol{\omega}_N)](\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}_s^\star) + (\boldsymbol{c}_s^\star - \boldsymbol{c}^\star)\right|^2 \\
&\leqslant 2\left|[\phi(\boldsymbol{z};\boldsymbol{\omega}_1),\ldots,\phi(\boldsymbol{z};\boldsymbol{\omega}_N)](\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}_s^\star)\right|^2 \\
&\quad + 2|[\phi(\boldsymbol{z};\boldsymbol{\omega}_1),\ldots,\phi(\boldsymbol{z};\boldsymbol{\omega}_N)](\boldsymbol{c}_s^\star - \boldsymbol{c}^\star)|^2 \\
&\leqslant 4s\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 + 2\left|\sum_{j\notin\mathcal{S}^\star}\phi(\boldsymbol{z},\boldsymbol{\omega}_j)c_j^\star\right|^2 \\
&\leqslant 4s\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 + 2\left|\sum_{j\notin\mathcal{S}^\star}|\phi(\boldsymbol{z},\boldsymbol{\omega}_j)||c_j^\star|\right|^2 \\
&\leqslant 4s\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 + 2(\kappa_{s,1}(\boldsymbol{c}^\star))^2
\end{aligned} \tag{12}$$

We provide two ways to bound $\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2$ by using an alternative η' or $\tilde{\eta}$ instead of η.

1. **With the max singular value of $\boldsymbol{A}$ (or $\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$).** From (86) in Hashemi et al. (2021), we have $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{c}^\star\|^2 \leqslant 2m(\epsilon^2\|f\|_\rho^2 + 4\nu^2)$, then

$$\begin{aligned}
\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{c}_s^\star\|_2 &\leqslant \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{c}^\star\|_2 + \|\boldsymbol{A}(\boldsymbol{c}^\star - \boldsymbol{c}_s^\star)\|_2 \\
&\leqslant \sqrt{2m(\epsilon^2\|f\|_\rho^2 + 4\nu^2)} + \sqrt{\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})}\kappa_{s,2}(\boldsymbol{c}^\star) \\
&= \eta'\sqrt{m}
\end{aligned} \tag{13}$$

   Then $\eta' = \sqrt{2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)} + \sqrt{\frac{\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})}{m}}\kappa_{s,2}(\boldsymbol{c}^\star)$.

2. **With $\kappa_{s,1}$.** Follow the proof idea of (86), we have

$$\begin{aligned}
\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{c}_s^\star\|^2 &\leqslant 2\left(\sum_{k=1}^m (f(\boldsymbol{x}_k) - f^\star(\boldsymbol{x}_k)^2 + 4\nu^2 m\right) \\
&\leqslant 2\left(\sum_{i=1}^m 2(f(\boldsymbol{x}_k) - f_s^\star(\boldsymbol{x}_k))^2 + 2(f^\star(\boldsymbol{x}_k) - f_s^\star(\boldsymbol{x}_k))^2 + 4\nu^2 m\right) \\
&\leqslant 4m\big(\epsilon^2\|f\|_\rho^2 + \kappa_{s,1}^2(\boldsymbol{c}^\star) + 2\nu^2\big) := \tilde{\eta}^2 m
\end{aligned} \tag{14}$$

   where $(f(\boldsymbol{x}_k) - f_s^\star(\boldsymbol{x}_k))^2$ is bounded from Lemma 2 in Hashemi et al. (2021). Then $\tilde{\eta} = 2\sqrt{\epsilon^2\|f\|_\rho^2 + 2\nu^2 + \kappa_{s,1}^2(\boldsymbol{c}^\star)}$.

From Stability of BP-based Sparse Reconstruction (Foucart and Rauhut, 2013) (which is also Lemma 6 in Hashemi et al. (2021)), we have

$$\|\boldsymbol{c}^\sharp - \boldsymbol{c}_s^\star\|_2 \leqslant C'\frac{\kappa_{s,1}(\boldsymbol{c}_s^\star)}{\sqrt{s}} + C\min\{\eta', \tilde{\eta}\} = C\min\{\eta', \tilde{\eta}\} \tag{15}$$

where $\kappa_{s,1}(\boldsymbol{c}_s^\star) = 0$ since $\boldsymbol{c}_s^\star$ is s-sparse. Then, if the coherence of $\boldsymbol{A}$ satisfies $\mu_{\boldsymbol{A}} \leqslant \frac{4}{\sqrt{41}(2s-1)}$, we have

$$\|\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}_s^\star\|_2 \leqslant 3\|\boldsymbol{c}^\sharp - \boldsymbol{c}_s^\star\|_2 \leqslant 3C\min\{\eta', \tilde{\eta}\} := C\min\{\eta', \tilde{\eta}\} \tag{16}$$

by redefining C.

Then for McDiarmid's inequality, we have

$$\begin{aligned} |\nu(\boldsymbol{z}_k) - \nu(\tilde{\boldsymbol{z}_k})| &\leqslant \frac{2\big(4s\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 + 2(\kappa_{s,1}(\boldsymbol{c}^\star))^2\big)}{m} := \Delta_v \\ \Rightarrow t = \Delta_v\sqrt{\frac{m}{2}\log\left(\frac{1}{\delta}\right)} &= 2\sqrt{\frac{2}{m}\log\left(\frac{1}{\delta}\right)}\big(2s\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 + (\kappa_{s,1}(\boldsymbol{c}^\star))^2\big) \end{aligned} \tag{17}$$

Putting everything together, we have

$$\begin{aligned} \sqrt{\int_{\mathbb{R}^d}\left|f_{\mathcal{S}^\sharp}^\sharp(\boldsymbol{x}) - f^\star(\boldsymbol{x})\right|^2 d\mu} \leqslant\ & m^{-\frac{1}{2}}\sqrt{\sum_{k=1}^m\left|f_{\mathcal{S}^\sharp}^\sharp(\boldsymbol{z}_k) - f^\star(\boldsymbol{z}_k)\right|^2} \\ & + \left(\frac{8}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}}\big(2s\|\boldsymbol{c}^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 + (\kappa_{s,1}(\boldsymbol{c}^\star))^2\big)^{\frac{1}{2}} \end{aligned} \tag{18}$$

For the first term, following (96) in Hashemi et al. (2021), we bound as follows:

$$m^{-\frac{1}{2}}\sqrt{\sum_{k=1}^m\left|f_{\mathcal{S}^\sharp}^\sharp(\boldsymbol{z}_k) - f^\star(\boldsymbol{z}_k)\right|^2} \leqslant 2\left\|\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}_s^\star\right\|_2 + \kappa_{s,1}(\boldsymbol{c}^\star) \tag{19}$$

where $\boldsymbol{A}$ satisfies $2s$-RIP condition and $\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}_s^\star$ is $2s$-sparse.

Therefore, with probability at least $1-\delta$, we have the refined bound as follows:

$$\sqrt{\int_{\mathbb{R}^d}\left|f_{\mathcal{S}^\sharp}^\sharp(\boldsymbol{x}) - f^\star(\boldsymbol{x})\right|^2 d\mu} \leqslant \left(\frac{8}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}}\left(2s\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 + (\kappa_{s,1}(\boldsymbol{c}^\star))^2\right)^{\frac{1}{2}} + 2\left\|\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}_s^\star\right\|_2 + \kappa_{s,1}(\boldsymbol{c}^\star) \tag{20}$$

where

$$\|\boldsymbol{c}_s^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2 \leqslant C\min\left\{\sqrt{2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)} + \sqrt{\frac{\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})}{m}}\kappa_{s,2}(\boldsymbol{c}^\star), 2\sqrt{\epsilon^2\|f\|_\rho^2 + 2\nu^2 + \kappa_{s,1}^2(\boldsymbol{c}^\star)}\right\}.$$

Furthermore, if we plug in $\kappa_{s,2}(\boldsymbol{c}^\star) \leqslant \epsilon\|f\|_\rho$ from (89) in Hashemi et al. (2021), we have

$$\begin{aligned} &\sqrt{\int_{\mathbb{R}^d}\left|f_{\mathcal{S}^\sharp}^\sharp(\boldsymbol{x}) - f^\star(\boldsymbol{x})\right|^2 d\mu} \leqslant \left(\frac{8}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}}\cdot \\ &\left(2s\left(C\min\left\{\sqrt{2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)} + \sqrt{\frac{\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})}{m}}\epsilon\|f\|_\rho, 2\sqrt{\epsilon^2\|f\|_\rho^2 + 2\nu^2 + \kappa_{s,1}^2(\boldsymbol{c}^\star)}\right\}\right)^2 + \kappa_{s,1}^2(\boldsymbol{c}^\star)\right)^{\frac{1}{2}} \\ &+ 2C\min\left\{\sqrt{2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)} + \sqrt{\frac{\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})}{m}}\epsilon^2\|f\|_\rho, 2\sqrt{\epsilon^2\|f\|_\rho^2 + 2\nu^2 + \kappa_{s,1}^2(\boldsymbol{c}^\star)}\right\} + \kappa_{s,1}(\boldsymbol{c}^\star) \end{aligned} \tag{21}$$

■

C.2. Proof of Corollary 6

Proof [Proof of Corollary 6] For the Corollary, since $\boldsymbol{c}^\star$ is s-sparse, $\kappa_{s,p} = 0, \forall p$ and $\boldsymbol{c}_s^\star - \boldsymbol{c}^\star = \mathbf{0}$. We can bound $\left|f^\sharp_{\mathcal{S}^\sharp}(\boldsymbol{z}) - f^\star(\boldsymbol{z})\right|^2$ using almost the same way as equation 12, but with a tighter constant. Since both $\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}$ and $\boldsymbol{c}^\star$ are s-sparse, $|\mathcal{S}^\sharp \bigcup \mathcal{S}^\star| \leqslant 2s$ and $[\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}^\star]_i = 0, \forall i \notin \mathcal{S}^\sharp \bigcup \mathcal{S}^\star$. Then,

$$\begin{aligned}\left|f^\sharp_{\mathcal{S}^\sharp}(\boldsymbol{z}) - f^\star(\boldsymbol{z})\right|^2 &= \left|[\phi(\boldsymbol{z};\boldsymbol{\omega}_1),\ldots,\phi(\boldsymbol{z};\boldsymbol{\omega}_N)](\boldsymbol{c}^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp})\right|^2 \\ &= \left|[\phi(\boldsymbol{z};\boldsymbol{\omega}_i)]_{i\in\mathcal{S}^\sharp\bigcup\mathcal{S}^\star}([\boldsymbol{c}^\star]_{i\in\mathcal{S}^\sharp\bigcup\mathcal{S}^\star} - [\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}]_{i\in\mathcal{S}^\sharp\bigcup\mathcal{S}^\star})\right|^2 \\ &\leqslant 2s\|\boldsymbol{c}^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|^2\end{aligned} \tag{22}$$

Hence, the bound in Lemma 7 (with $\eta = \sqrt{2(\epsilon^2\|f\|_\rho + E^2)}$) changes to

$$\|\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}^\star\|_2 \leqslant 3C\eta = 3C\sqrt{2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)} := C\sqrt{\epsilon^2\|f\|_\rho^2 + 4\nu^2} \tag{23}$$

after redefining C.

Furthermore, we can bound the difference in v from (91) in Hashemi et al. (2021) by

$$\begin{aligned}|\nu(\boldsymbol{z}_k) - \nu(\tilde{\boldsymbol{z}_k})| &\leqslant \frac{1}{m}\left|\left|f^\sharp_{\mathcal{S}^\sharp}(\boldsymbol{z}_k) - f^\star(\boldsymbol{z}_k)\right|^2 - \left|f^\sharp_{\mathcal{S}^\sharp}(\tilde{\boldsymbol{z}}_k) - f^\star(\tilde{\boldsymbol{z}}_k)\right|^2\right| \\ &\leqslant \frac{4s}{m}\|\boldsymbol{c}^\star - \boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp}\|_2^2 = \frac{4sC^2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)}{m}\end{aligned} \tag{24}$$

where $\tilde{\boldsymbol{z}_k}$ results from perturbing $\boldsymbol{z}_k$ at the k^{th} coordinate, and

$$t = \frac{4sC^2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)}{m}\sqrt{\frac{m}{2}\log\left(\frac{1}{\delta}\right)} = sC^2(\epsilon^2\|f\|_\rho^2 + 4\nu^2)\sqrt{\frac{8}{m}\log\left(\frac{1}{\delta}\right)}. \tag{25}$$

Therefore, with probability exceeding $1-\delta$,

$$\begin{aligned}\sqrt{\int_{\mathbb{R}^d}\left|f^\sharp_{\mathcal{S}^\sharp}(\boldsymbol{x}) - f^\star(\boldsymbol{x})\right|^2 d\mu} &\leqslant m^{-\frac{1}{2}}\sqrt{\sum_{k=1}^m\left|f^\sharp_{\mathcal{S}^\sharp}(\boldsymbol{z}_k) - f^\star(\boldsymbol{z}_k)\right|^2} + \left(\frac{8}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}} s^{\frac{1}{2}}C\sqrt{\epsilon^2\|f\|_\rho^2 + 4\nu^2} \\ &\leqslant 2\|\boldsymbol{c}^\sharp|_{\mathcal{S}^\sharp} - \boldsymbol{c}^\star\|_2 + \left(\frac{8}{m}\log\left(\frac{1}{\delta}\right)\right)^{\frac{1}{4}} s^{\frac{1}{2}}C\sqrt{\epsilon^2\|f\|_\rho^2 + 4\nu^2} \\ &\leqslant C\left(2 + 8^{\frac{1}{4}}s^{\frac{1}{2}}m^{-\frac{1}{4}}\log^{\frac{1}{4}}(1/\delta)\right)\sqrt{\epsilon^2\|f\|_\rho^2 + 4\nu^2}\end{aligned} \tag{26}$$

With order-q features, we have $\eta = \sqrt{2\epsilon^2\binom{d}{q}|||f|||^2 + 2E^2}$, where $|||f||| = \frac{1}{K}\sum_{j=1}^K\|g_j\|_\rho$. Hence, the bound reduces to

$$\sqrt{\int_{\mathbb{R}^d} \left| f^\sharp_{\mathcal{S}^\sharp}(\boldsymbol{x}) - f^\star(\boldsymbol{x}) \right|^2 d\mu} \leqslant C\Big(2 + 8^{\frac{1}{4}} s^{\frac{1}{2}} m^{-\frac{1}{4}} \log^{\frac{1}{4}}(1/\delta)\Big) \sqrt{\epsilon^2 \binom{d}{q} |||f|||^2 + E^2} \\ := \mathcal{O}\left(\left(1 + C' s^{\frac{1}{2}} m^{-\frac{1}{4}} \log^{\frac{1}{4}}(\frac{1}{\delta})\right) \sqrt{\epsilon^2 \binom{d}{q} |||f|||^2 + E^2} \right) \tag{27}$$

■

C.3. Proof of Proposition 7

Proof [Proof of Proposition 7] Let $X_\ell \in \mathbb{C}^m$ be the ℓth row of $\boldsymbol{A}^*$ for $\ell \in [N]$, i.e.,

$$X_\ell = [\overline{\phi(x_1, \boldsymbol{\omega}_\ell)}, \ldots, \overline{\phi(x_m, \boldsymbol{\omega}_\ell)}] \\ = [\overline{\phi(x_1, \boldsymbol{\omega}_\ell)}, \ldots, \overline{\phi(x_m, \boldsymbol{\omega}_\ell)}].$$

We can decompose $\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*$ into the following sum of rank-1 matrices:

$$\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^* = \frac{1}{N}\sum_{\ell=1}^N X_\ell^* X_\ell = \frac{1}{N}\sum_{\ell=1}^N [\phi(x_1, \boldsymbol{\omega}_\ell), \ldots, \phi(x_m, \boldsymbol{\omega}_\ell)]^T [\overline{\phi(x_1, \boldsymbol{\omega}_\ell)}, \ldots, \overline{\phi(x_m, \boldsymbol{\omega}_\ell)}]. \tag{28}$$

For a fixed ℓ, each component of $X_\ell^* X_\ell$ takes the form $\exp(i\langle \boldsymbol{x}_j - \boldsymbol{x}_k, \boldsymbol{\omega}_\ell \rangle)$.

Here, we quantify the effect of the conditioning of the linear system $\boldsymbol{A}\boldsymbol{A}^\star$ (overparameterized setting) as $N \to m^+$. Since $\min(m, N) = m$, we will bound $\lambda_m\left(\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*\right)$ and $\lambda_1\left(\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*\right)$.

The Rayleigh quotient can be bounded by

$$\lambda_m\left(\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*\right) \leqslant \frac{1}{N}\langle \boldsymbol{A}^*\boldsymbol{v}, \boldsymbol{A}^*\boldsymbol{v}\rangle. \tag{29}$$

for all unit vectors $\boldsymbol{v} \in \mathbb{C}^m$. The set $\mathcal{X} = \{X_1, \ldots, X_{m-1}\}$ forms a subspace of $\mathbb{C}^m$ of dimension at most $m-1$, thus there exists a unit vector $\boldsymbol{z} \in \mathbb{C}^m$ orthogonal to $span(\mathcal{X})$. And

$$\begin{aligned} \lambda_m\left(\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*\right) &\leqslant \frac{1}{N}\langle \boldsymbol{z}, \boldsymbol{A}\boldsymbol{A}^*\boldsymbol{z}\rangle \\ &\leqslant \frac{1}{N}\sum_{\ell=m}^N \boldsymbol{z}^* X_\ell^* X_\ell \boldsymbol{z} \\ &= \frac{1}{N}\sum_{\ell=m}^N \sum_{j,k=1}^m \overline{\boldsymbol{z}_j}\, \boldsymbol{z}_k \exp(i\langle \boldsymbol{x}_j - \boldsymbol{x}_k, \boldsymbol{\omega}_\ell\rangle) \\ &\leqslant \frac{N-m+1}{N} + \frac{1}{N}\sum_{\ell=m}^N \sum_{j\neq k} \overline{\boldsymbol{z}_j}\, \boldsymbol{z}_k \exp(i\langle \boldsymbol{x}_j - \boldsymbol{x}_k, \boldsymbol{\omega}_\ell\rangle). \end{aligned} \tag{30}$$

The vector $\boldsymbol{z}$ is independent of $\boldsymbol{\omega}_\ell$ for $\ell \geqslant m$, therefore, applying expectations lead to:

$$\begin{aligned}
\mathbb{E}\lambda_m\left(\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*\right) &\leqslant \frac{N-m+1}{N} + \frac{1}{N}\,\mathbb{E}\sum_{\ell=m}^{N}\sum_{j\neq k}\overline{\boldsymbol{z}_j}\,\boldsymbol{z}_k\exp(i\langle \boldsymbol{x}_j-\boldsymbol{x}_k,\boldsymbol{\omega}_\ell\rangle)\\
&= \frac{N-m+1}{N} + \frac{1}{N}\,\mathbb{E}_{\boldsymbol{x}}\,\mathbb{E}_{\boldsymbol{\omega}_1,\ldots,\boldsymbol{\omega}_{m-1}}\sum_{\ell=m}^{N}\sum_{j\neq k}\overline{\boldsymbol{z}_j}\boldsymbol{z}_k\mathbb{E}_{\boldsymbol{\omega}_\ell}[\exp(i\langle \boldsymbol{x}_j-\boldsymbol{x}_k,\boldsymbol{\omega}_\ell\rangle)]\\
&= \frac{N-m+1}{N} + \frac{1}{N}\,\mathbb{E}_{\boldsymbol{x}}\,\mathbb{E}_{\boldsymbol{\omega}_1,\ldots,\boldsymbol{\omega}_{m-1}}\sum_{\ell=m}^{N}\sum_{j\neq k}\overline{\boldsymbol{z}_j}\boldsymbol{z}_k\exp\left(-\frac{\sigma^2}{2}\|\boldsymbol{x}_k-\boldsymbol{x}_j\|_2^2\right)\\
&\leqslant \frac{N-m+1}{N} + \frac{1}{N}\sum_{\ell=m}^{N}\mathbb{E}_{\boldsymbol{x}}\sqrt{\sum_{j\neq k}\exp\left(-\sigma^2\|\boldsymbol{x}_k-\boldsymbol{x}_j\|_2^2\right)}\\
&\leqslant \frac{N-m+1}{N} + \frac{(N-m+1)\sqrt{m^2-m}}{N}\left(4\gamma^2\sigma^2+1\right)^{-\frac{q}{4}}
\end{aligned} \tag{31}$$

using Holder's and Jensen's inequalities (noting $\|\boldsymbol{z}\|_2^4 = 1$). Repeating for the maximum eigenvalue:

$$\lambda_1\left(\frac{1}{N}\boldsymbol{A}\boldsymbol{A}^*\right) \geqslant \frac{1}{N}\langle \boldsymbol{z}, \boldsymbol{A}\boldsymbol{A}^*\boldsymbol{z}\rangle$$

and setting the unit vector to $z = \frac{1}{\sqrt{N}}X_1$ yields

$$\begin{aligned}
\lambda_1\left(\frac{1}{N}AA^*\right) &\geqslant \frac{1}{N^2}\sum_{\ell=1}^{N}X_1^*X_\ell X_\ell^*X_1\\
&= \frac{1}{N^2}\left(N^2 + \sum_{\ell=2}^{N}X_1^*X_\ell X_\ell^*X_1\right)\\
&= 1 + \frac{1}{N^2}\sum_{\ell=2}^{N}\sum_{j,k=1}^{m}\exp(i\langle \boldsymbol{x}_k-\boldsymbol{x}_j,\boldsymbol{\omega}_1-\boldsymbol{\omega}_\ell\rangle)\\
&= 1 + \frac{(N-1)m}{N^2} + \frac{1}{N^2}\sum_{\ell=2}^{N}\sum_{\substack{j,k=1\\ j\neq k}}^{m}\exp(i\langle \boldsymbol{x}_k-\boldsymbol{x}_j,\boldsymbol{\omega}_1-\boldsymbol{\omega}_\ell\rangle),
\end{aligned} \tag{32}$$

and thus

$$\mathbb{E}\lambda_1\left(\frac{1}{N}AA^*\right) \geqslant 2 - \frac{(N-1)m}{N^2} + \frac{(N-1)(m^2-m)}{N^2}\left(4\gamma^2\sigma^2+1\right)^{-\frac{q}{4}}. \tag{33}$$

Consider a linear scaling $N = cm$ for $c > 1$, then

$$\begin{aligned}
\mathbb{E}\lambda_m\left(\frac{1}{N}AA^*\right) &\leqslant \frac{(c-1)m+1}{cm} + \frac{((c-1)m+1)\sqrt{m^2-m}}{cm}\left(4\gamma^2\sigma^2+1\right)^{-\frac{q}{4}}\\
&\leqslant \frac{c-1}{c} + \frac{1}{m} + \frac{c-1}{c}m\left(4\gamma^2\sigma^2+1\right)^{-\frac{q}{4}} + \left(4\gamma^2\sigma^2+1\right)^{-\frac{q}{4}}.
\end{aligned} \tag{34}$$

■

Appendix D. Further Discussion

In this section, we provide further discussion and future research directions on how SHRIMP connects to previous work on ℓ_0-regularized problems and random features approximations.

D.1. Connection to ℓ_0-regularized Problems

We first note the similarities of SHRIMP with the SINDy (Zhang and Schaeffer, 2019) algorithm. SINDy prunes all features whose magnitude lies below a fixed λ. Thus, SHRIMP can be considered as an adaptive form of SINDy, where the λ is chosen adaptively at each step depending on the data. Zhang and Schaeffer (2019) prove under certain conditions that SINDy converges to a fixed point, which is a local minimizer of the non-convex ℓ_0-regularized regression problem with regularization parameter λ. However, the proof requires the matrix $\boldsymbol{A}$ to be full column-rank, which in our case, is not guaranteed; in fact, one of the advantages of SHRIMP is that it can often move from the overparameterized to the underparameterized setting.

A more general perspective on the minimizers of the ℓ_0-regularized regression problem is given in Nikolova (2013). Remark 2 in Nikolova (2013) indicates that SHRIMP solves an ℓ_0-regularized minimization problem at each step. Additionally, Theorem 3.2 indicates that when $\boldsymbol{A}$ has full rank with probability one and when SHRIMP prunes in the underparameterized setting, each iterate is a *strict* local minimum of the regularized regression problem. Thus, SHRIMP can be seen as solving for local minimizers in a data-dependent sequence of ℓ_0-regularized regression problems.

However, there are still a few gaps in this direction. First, we focus on generalization performance of pruned models, while most work on ℓ_0-regularization focus on sparse recovery. Moreover, the results in Nikolova (2013) indicate that solutions of problems that SHRIMP is solving are local minimizers of the regularized regression problem for *any* regularization parameter greater than zero, but it is hard to compare which local minimizer is the best regarding test performance.

D.2. The variance of the random feature weights

In our experiments in Section 4, we set the variance of the $\mathbf{w}$ to be $1/q$, or the inverse of the order of the function assuming oracle access. This is motivated by two reasons: first, in deep learning initialization (He et al., 2015; Kumar, 2017) and corresponding theory (Ba et al., 2019), the variance of the weights in a layer is the inverse of the input dimension or the number of weights in the layer. Second, consider the bound for kernel approximation given in Rahimi and Recht (2008c), which is

$$\Pr\left[\sup_{\mathbf{x},\mathbf{y}\sim\mathcal{M}} |z(\mathbf{x})'z(\mathbf{y}) - k(\mathbf{x},\mathbf{y})| \geqslant \epsilon\right] \leqslant 2^8 \left(\frac{\sigma \operatorname{diam}(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{N\epsilon^2}{4(d+2)}\right), \tag{35}$$

where $\mathcal{M}$ is a compact set.

If the data comes from a Gaussian distribution, with high probability they lie in a compact set ($\mathcal{M}$). When the dimension (d) increases, the diameter $\operatorname{diam}(\mathcal{M})$ grows as $\mathcal{O}(\sqrt{d})$. If we use low-order features to approximate, the effective dimension is q, so the diameter grows as $\mathcal{O}(\sqrt{q})$. Thus, setting $\sigma = 1/\sqrt{q}$ mitigates this increase by allowing the numerator to be $\mathcal{O}(1)$.

However, the theory with respect to basis pursuit requires a variance that *increases* with q. Our experiments in Section B.4 corroborate this theoretical gap. While smaller variance may suffice to learn smooth or low-frequency functions, setting the variance to be small may not allow enough

high-frequency weights to be sampled to learn higher frequency functions. One possible explanation is that these higher frequency functions are better represented by functions in the RKHS corresponding to kernel parameter matching that of the larger variances in the random features. However, as shown in equation 35, the kernel approximation is much worse when σ is large. Thus, understanding this behavior is an interesting future direction.