Efficiently Learning Fourier Sparse Set Functions

Andisheh Amrollahi [*]
ETH Zurich
Zurich, Switzerland
`amrollaa@ethz.ch`

Amir Zandieh [*]
EPFL
Lausanne, Switzerland
`amir.zandieh@epfl.ch`

Michael Kapralov[†]
EPFL
Lausanne, Switzerland
`michael.kapralov@epfl.ch`

Andreas Krause
ETH Zurich
Zurich, Switzerland
`krausea@ethz.ch`

November 29, 2019

Abstract

Learning set functions is a key challenge arising in many domains, ranging from sketching graphs to black-box optimization with discrete parameters. In this paper we consider the problem of efficiently learning set functions that are defined over a ground set of size n and that are sparse (say k-sparse) in the Fourier domain. This is a wide class, that includes graph and hypergraph cut functions, decision trees and more. Our central contribution is the first algorithm that allows learning functions whose Fourier support only contains low degree (say degree $d = o(n)$) polynomials using $O(kd \log n)$ sample complexity and runtime $O(kn \log^2 k \log n \log d)$. This implies that sparse graphs with k edges can, for the first time, be learned from $O(k \log n)$ observations of cut values and in linear time in the number of vertices. Our algorithm can also efficiently learn (sums of) decision trees of small depth. The algorithm exploits techniques from the sparse Fourier transform literature and is easily implementable. Lastly, we also develop an efficient robust version of our algorithm and prove ℓ_2/ℓ_2 approximation guarantees without any statistical assumptions on the noise.

1 Introduction

How can we learn the structure of a graph by observing the values of a small number of cuts? Can we learn a decision tree efficiently by observing its evaluation on a few samples? Both of these important applications are instances of the more general problem of learning set functions.
Consider a set function which maps subsets of a ground set V of size n to real numbers, $x : 2^V \to \mathbb{R}$. Set functions that arise in applications often exhibit structure, which can be effectively captured in the Fourier (also called Walsh-Hadamard) basis. One common studied structure for set functions is *Fourier sparsity* [GL89]. A k-Fourier-sparse set function contains no more than k nonzero Fourier coefficients. A natural example for k-Fourier-sparse set functions are cut functions of graphs with k edges or evaluations of a decision tree of depth d [SK12, KM93, Man94]. The cut function of a graph only contains polynomials of degree at most two in the Fourier basis and in the general case, thecut function of a hypergraph of degree d only contains polynomials of degree at most d in

[*]The first two authors contributed equally
[†]Supported by ERC Starting Grant SUBLINEAR.

the Fourier basis [SK12]. Intuitively this means that these set functions can be written as sums of terms where each term depends on at most d elements in the ground set. Also a decision tree of depth d only contains polynomials of degree at most d in the Fourier basis [KM93][Man94]. Learning such functions has recently found applications in neural network hyper-parameter optimization [HKY18]. Therefore, the family of Fourier sparse set functions whose Fourier support only contains low order terms is a natural and important class of functions to consider.

Related work One approach for learning Fourier sparse functions uses Compressive Sensing (CS) methods [SK12]. Suppose we know that the Fourier transform of our function $\widehat{x}$ is k-sparse i.e. $|\text{supp}(\widehat{x})| \leq k$, and $\text{supp}(\widehat{x}) \subseteq P$ for some known set P of size p. In [SK12] it is shown that recovery of $\widehat{x}$ is possible (with high probability) by observing the value of x on $O(k \log^4 p)$ subsets chosen independently and uniformly at random. They utilize results from [RV08, Ver10] which prove that picking $O(k \log^4 p)$ rows of the Walsh-Hadamard matrix independently and uniformly at random results in a matrix satisfying the RIP which is required for recovery. For the case of graphs $p = \binom{n}{2} = O(n^2)$ and one can essentially learn the underlying graph with $O(k \log^4 n)$ samples. In fact this result can be further improved, and $O(k \log^2 k \log n)$ samples suffice[HR17]. Computationally, for the CS approach, one may use matching pursuit which takes $\Omega(kp)$ time and thus results in runtime of $\Omega(kn^d)$ for k Fourier sparse functions of order d. This equals $\Omega(kn^2)$ for graphs, where $d = 2$. In [SK12], proximal methods are used to optimize the Lagrangian form of the ℓ_1 norm minimization problem. Optimization is performed on p variables which results in $\Omega(n^2)$ runtime for graphs and to $\Omega(n^d)$ time for the general order d sparse recovery case. Hence, these algorithms scale *exponentially* with d and have *at least quadratic dependence on n* even in the simple case of learning graph cut functions.

There is another line of work on this problem in the sparse Fourier transform literature. [SHV15] provides a non-robust version of the sparse Walsh Hadamard Transform (WHT). This algorithm makes restrictive assumptions on the signal, namely that the k non-zero Fourier coefficients are chosen uniformly at random from the Fourier domain. This is a strong assumption that does not hold for the case of cut functions or decision trees. This work is extended in [HR17] to a robust sparse WHT coined as SPRIGHT. In addition to the the random uniform support assumption, [HR17] further presumes that the Fourier coefficients are finite valued and the noise is Gaussian. Furthermore, all existing sparse WHT algorithms are unable to exploit low-degree Fourier structure.

Our results We build on techniques from the sparse Fourier transform literature [HIKP12, IKP14, GL89] and develop an algorithm to compute the Walsh-Hadamard transform (WHT) of a k-Fourier-sparse signal whose Fourier support is constrained to low degree frequencies (low degree polynomials). For recovering frequencies with low degree we utilize ideas that are related to compressive sensing over finite fields [DV13]. We show that if the frequencies present in the support of $\widehat{x}$ are of low order then there exists an algorithm that computes WHT in $O(kn \log^2 k \log n \log d)$ time using $O(kd \log n)$ samples. As opposed to [SHV15], we avoid distributional assumptions on the support using hashing schemes. Our approach is the first one to achieve the sampling complexity of $O(kd \log n)$. Moreover its running time scales *linearly* in n and there is no exponential dependence on d. For the important special case of graphs, where $d = 2$, our sampling complexity is near optimally $O(k \log n)$ and our runtime is $O(kn \log^2 k \log n)$ which is strictly better than CS methods which take at least quadratic time in n. This allows us to learn sparse graphs which have in the range of 800 vertices in ≈ 2 seconds whereas the previous methods [SK12] were constrained to the range of 100 for similar runtimes.

For the case where $\widehat{x}$ is not exactly k-sparse, we provide novel robust algorithms that recover the k dominant Fourier coefficients with provable ℓ_2/ℓ_2 approximation guarantees. We provide a robust algorithm using appropriate hashing schemes and a novel analysis. We further develop a robust recovery algorithm that uses $O(kd \log n \log(d \log n))$ samples and runs in time $O\left(nk \log^3 k + nk \log^2 k \log n \log(d \log n) \log d\right)$.

2 Problem Statement

Here we define the problem of learning set functions. Consider a set function which maps subsets of a ground set $V \triangleq \{1, \ldots, n\} = [n]$ of size n to real numbers, $x : 2^V \to \mathbb{R}$. We assume oracle access to this function, that is, we can observe the function value $x(A)$ for any subset A that we desire. The goal is to learn the function, that is to be able to evaluate it for all subsets $B \subseteq V$. A problem which has received considerable interest is learning *cut functions* of sparse (in terms of edges) graphs [SK12]. Given a weighted undirected graph $G = (V, E, w)$, the cut function associated to G is defined as $x(A) = \sum_{s \in A, t \in V \setminus A} w(s,t)$, for every $A \subseteq V$.

Note that we can equivalently represent each subset $A \subseteq V$ by a vector $t \in \mathbb{F}_2^n$ which is the indicator of set A. Here $\mathbb{F}$ denotes the finite field with 2 elements. Hence the set function can be viewed as $x : \mathbb{F}_2^n \to \mathbb{R}$. We denote the Walsh-Hadamard transform of $x : \mathbb{F}_2^n \to \mathbb{R}$ by $\widehat{x} : \mathbb{F}_2^n \to \mathbb{R}$. It is defined as:

$$\widehat{x}_f = \frac{1}{\sqrt{N}} \sum_{t \in \mathbb{F}_2^n} x_t \cdot (-1)^{\langle f,t \rangle} \quad , f \in \mathbb{F}_2^n.$$

The inner product $\langle f, t \rangle$ throughout the paper is performed modulo 2.

The Fourier transform of the graph cut function $\widehat{x}$ is the following,

$$\widehat{x}_f = \begin{cases} \frac{1}{2} \sum_{s,t \in V} w(s,t) & \text{if } f = (0, \ldots, 0) \\ -w(s,t)/2 & \text{if } f_s = f_t = 1 \text{ and } f_i = 0 \ \forall i \neq s,t \ . \\ 0 & \text{otherwise} \end{cases}$$

It is clear that the Fourier support of the cut function for graph G contains only $|E| + 1$ nonzero elements (and hence it is *sparse*). Furthermore, the nonzero Fourier coefficients correspond to frequencies with hamming weights at most 2.

One of the classes of set functions that we consider is that of **exactly low order Fourier sparse** functions. Under this model we address the following problem:

$$\begin{aligned} \textbf{Input:} \quad & \text{oracle access to } x : \mathbb{F}_2^n \to \mathbb{R} \\ & \text{such that } \|\widehat{x}\|_0 \leq k \text{ and } |f| \leq d \text{ for all } f \in \text{support}(\widehat{x}) \qquad (1) \\ \textbf{Output:} \quad & \text{nonzero coefficients of } \widehat{x} \text{ and their corresponding frequencies} \end{aligned}$$

where $|f|$ denotes the *Hamming weight* of f.

We also consider the **robust** version of problem (1) where we only have access to noisy measurements of the input set function. We make no assumption about the noise, which can be chosen adversarially. Equivalently one can think of a general set function whose spectrum is well approximated by a low order sparse function which we refer to as *head*. *Head* of $\widehat{x}$ is just the top k Fourier coefficients $\widehat{x}_f$ such that the frequency has low Hamming weight $|f| \leq d$. We refer to the noise spectrum as *tail*.

Definition 1 (Head and Tail norm)**.** For all integers n, d, and k we define the *head* of $\widehat{x} : \mathbb{F}_2^n \to \mathbb{R}$ as,

$$\widehat{x}_{head} := \arg \min_{\substack{y:\mathbb{F}_2^n \to \mathbb{R} \\ \|y\|_0 \le k \\ |j| \le d \text{ for all } j \in \mathrm{supp}(y)}} \|\widehat{x} - y\|_2.$$

The *tail norm* of $\widehat{x}$ is defined as, $\mathrm{Err}(\widehat{x}, k, d) := \|\widehat{x} - \widehat{x}_{head}\|_2^2$.

Since the set function to be learned is only *approximately* in the low order Fourier sparse model, it makes sense to consider the *approximate* version of problem (1). We use the well known ℓ_2/ℓ_2 *approximation* to formally define the **robust** version of problem (1) as follows,

$$\begin{aligned} \textbf{Input:} \quad & \text{oracle access to } x : \mathbb{F}_2^n \to \mathbb{R} \\ \textbf{Output:} \quad & \text{function } \widehat{\chi} : \mathbb{F}_2^n \to \mathbb{R} \\ & \text{such that } \|\widehat{\chi} - \widehat{x}\|_2^2 \le (1+\delta)\mathrm{Err}(\widehat{x}, k, d), \\ & |f| \le d \text{ for all } f \in \mathrm{support}(\widehat{\chi}) \end{aligned} \tag{2}$$

Note that no assumptions are made about the function x and it can be any general set function.

3 Algorithm and Analysis

In this section we present our algorithm and analysis. We use techniques from the sparse FFT literature [HIKP12, IKP14, GL89]. Our main technical novelty is a new primitive for estimating a low order frequency –i.e., $|f| \le d$, efficiently using optimal number of samples $O(d \log n)$ given in Section 3.1. This primitive relies heavily on the fact that a low order frequency is constrained on a subset of size $\binom{n}{d}$ as opposed to the whole universe of size 2^n. We show that problem (1) can be solved quickly and using a few samples from the function x by proving the following theorem,

Theorem 2. *For any integers n, k, and d, the procedure* ExactSHT *solves problem* (1) *with probability* 9/10*. Moreover the runtime of this algorithm is $O\left(kn \log^2 k \log n \log d\right)$ and the sample complexity of this procedure is $O\left(kd \log n\right)$.*

We also show that problem (2) can be solved efficiently by proving the following theorem in Section 4,

Theorem 3. *For any integers n, k, and d, the procedure* RobustSHT *solves problem* (2) *with probability* 9/10*. Moreover the runtime of this procedure is $O\left(nk \log^3 k + nk \log^2 k \log n \log(d \log n) \log d\right)$ and the sample complexity of the procedure is, $O\left(kd \log n \log(d \log n)\right)$.*

Remark: This theorem proves that for any arbitrary input signal, we are able to achieve the ℓ_2/ℓ_2 guarantee using $O\left(kd \cdot \log n \cdot \log(d \log n)\right)$ samples. Using the techniques of [PW11] one can prove that the sample complexity is optimal up to $\log(d \log n)$ factor. Note that it's impossible to achieve this sample complexity without exploiting the low degree structure of the Fourier support.

3.1 Low order frequency recovery

In this section we provide a novel method for recovering a frequency $f \in \mathbb{F}_2^n$ with bounded Hamming weight $|f| \le d$, from measurements $\langle m_i, f \rangle$ $i \in [s]$ for some $s = O(d \log n)$. The goal of this section is to design a measurement matrix $M \in \mathbb{F}_2^{s \times n}$ with small s, such that for any $f \in \mathbb{F}_2^n$ with $|f| \le d$

the following system of constraints, with constant probability, has a unique solution $j = f$ and has an efficient solver,

$$j \in \mathbb{F}_2^n \text{ such that } \begin{cases} Mj = Mf \\ |j| \le d \end{cases} .$$

To design an efficient solver for the above problem with optimal s, we first need to have an optimal algorithm for recovering frequencies with weight one $|f| \le 1$. In this case, we can locate the index of the nonzero coordinate of f optimally via binary search using $O(\log n)$ measurements and runtime.

Definition 4 (Binary search vectors)**.** For any integer n, the ensemble of vectors $\{v^l\}_{l=0}^{\lceil \log_2 n \rceil} \subseteq \mathbb{F}_2^n$ corresponding to binary search on n elements is defined as follows. Let $v^0 = \{1\}^n$. For every $l \in \{1, \cdots, \lceil \log_2 n \rceil\}$ and every $j \in [n]$, $v_j^l = \left\lfloor \frac{(j \mod 2^l)}{2^{l-1}} \right\rfloor$.

Lemma 5. *There exists a set of measurements $\{m_i\}_{i=1}^s$ for $s = \lceil \log_2 n \rceil + 1$ together with an algorithm such that for every $f \in \mathbb{F}_2^n$ with $|f| \le 1$ the algorithm can recover f from the measurements $\langle f, m_i \rangle$ in time $O(\log_2 n)$.*

Proof. Let us denote by $\tilde{f}$ what the algorithm recovers from the measurements. Let the measurements correspond to the binary search vectors v^l for $l = 0, 1, \cdots \lceil \log_2 n \rceil$ (Definition 4). Note that for any f with $|f| \le 1$ and any vector $m \in \mathbb{F}_2^n$, if $\langle f, m \rangle = 1$ then $\text{supp}(f) \subseteq \text{supp}(m)$ and otherwise $\text{supp}(f) \subseteq \text{supp}(\bar{m})$ where $\bar{m}$ is the entrywise complement of m. Therefore the following algorithm can recover f,

```
1: $\tilde{f} \leftarrow \{0\}^n$.
2: **if** $\langle f, v^0 \rangle = 1$ **then**
3:     $index \leftarrow \{0\}^{\lceil \log_2 n \rceil}$, a $\lceil \log_2 n \rceil$ bits number initialized at 0.
4:     For every $l = 1, 2, \cdots, \lceil \log_2 n \rceil$, if $\langle f, v^l \rangle = 1$ then $[index]_l \leftarrow 1$, $l^{th}$ bit of $index$ sets to 1.
5:     $\tilde{f}(index) \leftarrow 1$, the nonzero coordinate of $\tilde{f}$ is positioned at $index$.
```

The proof of why the above algorithm works is by induction and fairly straightforward. Above algorithm runs in time $O(\log n)$. □

To recover a frequency f with Hamming weight d, we hash the coordinates of f randomly into $O(d)$ buckets. In expectation a constant fraction of nonzero elements of f get isolated in buckets, and hence the problem reduces to the weight one recovery. We know how to solve this using binary search as shown in Lemma 5 in time $O(\log n)$ and with sample complexity $O(\log n)$. We recover a constant fraction of the nonzero indices of f and then we subtract those from f and recurse on the residual. We decrease the number of buckets we hash the coordinates into by a factor of $\frac{1}{2}$ in each step. We expect the sparsity of the frequency (the hamming weight of the frequency) to go down by a factor of $\frac{1}{4}$ in each step. The recovery procedure is presented in Algorithm 1.

Lemma 6. *For any integers n and d , any power of two integer $D \ge 128d$, and any frequency $f \in \mathbb{F}_2^n$ with $|f| \le d$, the procedure* RecoverFrequency *given in Algorithm 1 outputs f with probability at least $7/8$, if the following holds,*

1. *For every $r = 0, 1, \cdots, \log_4 D$, the hash function $h_r : [n] \to [D/2^r]$ is an instance from a pairwise independent hash family.*

2. *For every $l = 0, 1, \cdots, \lceil \log_2 n \rceil$ and every $r = 0, 1, \cdots, \log_4 D$, the measurements $\phi_r^l(i)$ are equal to $\phi_r^l(i) = \sum_{j \in h_r^{-1}(i)} f_j \cdot v_j^l$ for every $i \in [D/2^r]$.*

Moreover, the runtime of this procedure is $O(D \log D \log n)$ and the number of measurements is $O(D \log n)$.

Algorithm 1 RecoverFrequency

input: power of two integer D, hash functions $h_r : [n] \to [D/2^r]$ for every $r \in \{0, 1, \cdots, \log_4 D\}$, measurement vectors $\phi_r^l \in \mathbb{F}_2^{D/2^r}$ for every $l = 0, 1, \cdots \lceil \log_2 n \rceil$ and every $r = 0, 1, \cdots, \log_4 D$.
output: recovered frequency $\tilde{f}$.

```
1: Let $\{v^l\}_{l=0}^{\lceil \log_2 n \rceil}$ be the binary search vectors on $n$ elements (Definition 4).
2: $T \leftarrow \log_4 D$.
3: $\tilde{f}^{(0)} \leftarrow \{0\}^n$.
4: for $r = 0$ to $T$ do
5:     $w \leftarrow \{0\}^n$.
6:     for $i = 1$ to $D/2^r$ do
7:         if $\phi_r^0(i) - \sum_{j \in h_r^{-1}(i)} \tilde{f}_j^{(r)} \cdot v_j^0 = 1$ then
8:             $index \leftarrow \{0\}^{\lceil \log_2 n \rceil}$, a $\lceil \log_2 n \rceil$ bits pointer.
9:             for $l = 1$ to $\lceil \log_2 n \rceil$ do
10:                if $\phi_r^l(i) - \sum_{j \in h_r^{-1}(i)} \tilde{f}_j^{(r)} \cdot v_j^l = 1$ then
11:                    $[index]_l \leftarrow 1$, set $l^{th}$ bit of $index$ to 1.
12:            $w(index) \leftarrow 1$, set the coordinate of $w$ positioned at $index$ to 1.
13:    $\tilde{f}^{(r+1)} \leftarrow \tilde{f}^{(r)} + w$.
14: return $\tilde{f}^{(T+1)}$.
```

Proof. The proof is by induction on the iteration number $r = 0, 1, \cdots, T$. We denote by $\mathcal{E}_r$ the event $|f - \tilde{f}^{(r)}| \le \frac{d}{4^r}$, that is the sparsity goes down by a factor of 4 in every iteration up to r^{th} iteration. The inductive hypothesis is

$$\Pr[\mathcal{E}_{r+1} | \mathcal{E}_r] \ge 1 - \frac{1}{16 \cdot 2^r}.$$

Conditioning on $\mathcal{E}_r$ we have that $|f - \tilde{f}^{(r)}| \le \frac{d}{4^r}$. For every $i \in [D/2^r]$ and every $l \in \{0, 1, \cdots, \lceil \log_2 n \rceil\}$ it follows from the definition of ϕ_r^l that,

$$\phi_r^l(i) - \sum_{j \in h_r^{-1}(i)} \tilde{f}_j^{(r)} \cdot v_j^l = \sum_{j \in h_r^{-1}(i)} \left(f_j - \tilde{f}_j^{(r)} \right) \cdot v_j^l.$$

Let us denote by S the support of vector $f - \tilde{f}^{(r)}$,

$$S = \operatorname{supp}\left(f - \tilde{f}^{(r)} \right).$$

From the pairwise independence of the hash function h_r the following holds for every $a \in S$,

$$\begin{aligned} \Pr[h_r(a) \in h_r(S \setminus \{a\})] &\le 2^r \cdot \frac{|S|}{D} \\ &\le 2^r \cdot \frac{1}{128 \cdot 4^r} \\ &\le \frac{1}{128 \cdot 2^r}. \end{aligned}$$

This shows that for every $a \in S$, with probability $1 - \frac{1}{128 \cdot 2^r}$, the bucket $h_r(a)$ contains no other element of S. Because the vector $f - \tilde{f}^{(r)}$ restricted to the elements in bucket $h_r^{-1}(h_r(a))$ has Hamming weight one, hence for every $a \in S$,

$$\Pr\left[\left| \left(f - \tilde{f}^{(r-1)} \right)_{h_r^{-1}(h_r(a))} \right| = 1 \right] \ge 1 - \frac{1}{128 \cdot 2^r}.$$

If the above condition holds then it is possible to find the index of the nonzero element via binary search as in Lemma 5. The for loop in line 9 of Algorithm 1 implements this. Therefore with probability $1 - \frac{1}{16 \cdot 2^r}$ by Markov's inequality a $1 - 1/8$ fraction of the support elements, S, gets recovered correctly and at most $1/8$ fraction of elements remain unrecovered and possibly result in false positive. Since the algorithm recovers at most one element per bucket, the total number of falsely recovered indices is no more than the number of non-isolated buckets which is at most $1/8 \cdot |S|$. Therefore with probability $1 - \frac{1}{16 \cdot 2^r}$, the residual at the end of rth iteration has sparsity $1/8 \cdot |S| + 1/8 \cdot |S| = 1/4 \cdot |S|$,

$$\left| f - \tilde{f}^{(r+1)} \right| \le \frac{|S|}{4} \le \frac{d}{4^{r+1}}.$$

This proves the inductive step.

It follows from the event $\mathcal{E}_T$ for $T = \log_4 D$ that $\tilde{f}^{(T)} = f$, where $\tilde{f}^{(T)}$ is the output of Algorithm 1. The inductive hypothesis along with union bound implies that,

$$\begin{aligned} \Pr\left[\bar{\mathcal{E}}_T\right] &\le \sum_{r=1}^{T} \Pr\left[\bar{\mathcal{E}}_r | \mathcal{E}_{r-1}\right] + \Pr\left[\bar{\mathcal{E}}_0\right] \\ &\le \sum_{r=0}^{T} \frac{1}{16 \cdot 2^r} \\ &\le 1/8. \end{aligned}$$

Runtime: the algorithm has three nested loops and the total number of repetitions of all loops together is $O(D \log n)$. The recovered frequency $\tilde{f}^{(r)}$ always has at most $O(D)$ nonzero entries therefore the time to calculate $\sum_{j \in h_r^{-1}(i)} \tilde{f}_j^{(r-1)} \cdot v_j^l$ for a fixed r and a fixed l and all $i \in [D/2^r]$ is $O(D)$. Therefore the total runtime is $O(D \log D \log n)$.

Number of measurements: the number of measurements is the total size of the measurement vectors ϕ_r^l which is $O(D \cdot \log n)$.

$\square$

3.2 Signal reduction

The main tool for estimating a sparse signal is the HASH2BINS primitive. If we hash the frequencies of a k-sparse signal into $O(k)$ buckets then we expect each bucket of contain at most one of the elements of the support of our signal. Next definition shows how to compute the hashing of a signal in the time domain.

Definition 7. For every $n, b \in \mathbb{N}$, every $a \in \mathbb{F}_2^n$, and every $\sigma \in \mathbb{F}_2^{n \times b}$ and every $x : \mathbb{F}_2^n \to \mathbb{R}$, we define the hashing of $\widehat{x}$ as $u_\sigma^a : \mathbb{F}_2^b \to \mathbb{R}$ given by,

$$u_\sigma^a(t) = \sqrt{\frac{2^n}{2^b}} \cdot x_{\sigma t + a},$$

for every $t \in \mathbb{F}_2^b$.

We denote by $B = 2^b$ the number of buckets of the hash function. In the next claim we show that the Fourier transform of u_σ^a corresponds to hashing $\widehat{x}$ into B buckets.

Claim 8. *For every $j \in \mathbb{F}_2^b$,*

$$\widehat{u}_\sigma^a(j) = \sum_{f \in \mathbb{F}_2^n : \sigma^\top f = j} \widehat{x}_f \cdot (-1)^{\langle a, f \rangle}$$

Proof. We know that $x_t = \frac{1}{\sqrt{N}} \sum\limits_{f \in \mathbb{F}_2^n} \widehat{x}_f \cdot (-1)^{\langle f,t \rangle}$. Hence it follows that:

$$x_{\sigma t+a} = \frac{1}{\sqrt{N}} \sum_{f \in \mathbb{F}_2^n} \widehat{x}_f \cdot (-1)^{\langle f, \sigma t + a\rangle} = \frac{1}{\sqrt{N}} \sum_{f \in \mathbb{F}_2^n} \widehat{x}_f \cdot (-1)^{\langle \sigma^\top f, t\rangle + \langle f, a\rangle} \tag{3}$$

By definition of the WHT we have that:

$$\widehat{u}_\sigma^a(j) = \frac{1}{\sqrt{B}} \sqrt{\frac{2^n}{2^b}} \sum_{t \in \mathbb{F}_2^b} x_{\sigma t+a} \cdot (-1)^{\langle j,t\rangle} = \frac{1}{B} \sum_{t \in \mathbb{F}_2^b} x_{\sigma t+a} \cdot (-1)^{\langle j,t\rangle} \tag{4}$$

Inserting Equation (3) into Equation (4) yields:

$$\widehat{u}_\sigma^a(j) = \frac{1}{B} \sum_{f \in \mathbb{F}_2^n} \widehat{x}_f (-1)^{\langle a,f\rangle} \sum_{t \in \mathbb{F}_2^b} (-1)^{\langle \sigma^\top f + j, t\rangle}$$

The second summation is zero $\sigma^\top f \neq j$ and equal to $2^b = B$ otherwise. Hence:

$$\widehat{u}_\sigma^a(j) = \sum_{f \in \mathbb{F}_2^n : \sigma^\top f = j} \widehat{x}_f \cdot (-1)^{\langle a,f\rangle}$$

□

Let $h(f) \triangleq \sigma^\top f$. For every $j \in \mathbb{F}_2^b$, $\widehat{u}_\sigma^a$ is the sum of $\widehat{x}_f \cdot (-1)^{\langle a,f\rangle}$ for all frequencies $f \in \mathbb{F}_2^n$ such that $h(f) = j$, hence $h(f)$ can be thought of as the bucket that f is hashed into. In next claim we show that if the matrix σ is chosen uniformly at random then the hash function $h(\cdot)$ is pairwise independent.

Claim 9. *For any $n, b \in \mathbb{N}$, if the hash function $h : \mathbb{F}_2^n \to \mathbb{F}_2^b$ is defined as $h(\cdot) = \sigma^\top(\cdot)$, where $\sigma \in \mathbb{F}_2^{n \times b}$ is a random matrix whose entries are distributed independently and uniformly at random on $\mathbb{F}_2$, then for any $f \neq f' \in \mathbb{F}_2^n$ it holds that*

$$\Pr[h(f) = h(f')] = \frac{1}{B},$$

where the probability is over picking $n \cdot b$ random bits of σ.

Proof. The difference of $h(f)$ and $h(f')$ is equal to the summation of the columns in $\sigma^\top$ corresponding to the bits that f and f' differ in. Since f and f' differ in at least one bit it follows that $\Pr[h(f) = h(f')] = \frac{1}{2^b} = \frac{1}{B}$. □

Algorithm 2 HASH2BINS

```
input: signal $x \in \mathbb{R}^{2^n}$, signal $\widehat{\chi} \in \mathbb{R}^{2^n}$, integer $b$, binary matrix $\sigma \in \mathbb{F}_2^{n \times b}$, shift vector $a \in \mathbb{F}_2^n$.
output: hashed signal $\widehat{u}_\sigma^a$.
1: Compute $\widehat{u}_\sigma^a = \mathsf{FHT}\left(\sqrt{\frac{2^n}{2^b}} \cdot x_{\sigma(\cdot)+a}\right)$.    ▷ FHT is the fast Hadamard transform algorithm
2: $\widehat{u}_\sigma^a(j) \leftarrow \widehat{u}_\sigma^a(j) - \sum_{f \in \mathbb{F}_2^n : \sigma^\top f = j} \widehat{\chi}_f \cdot (-1)^{\langle a,f\rangle}$ for every $j \in \mathbb{F}_2^b$.
3: return $\widehat{u}_\sigma^a$.
```

The HASH2BINS primitive computes the Fourier coefficients of the residue signal that are hashed to each of the buckets. We denote by $\widehat{\chi}$ the estimate of $\widehat{x}$ in each iteration. As we will see in Section 3.3, the recovery algorithm is iterative in the sense that we iterate over $\widehat{x} - \widehat{\chi}$ (the residue) whose sparsity is guaranteed to decrease by a constant factor in each step.

Claim 10. *For any signals $x, \widehat{\chi} : \mathbb{F}_2^n \to \mathbb{R}$, integer b, matrix $\sigma \in \mathbb{F}_2^{n\times b}$, and vector $a \in \mathbb{F}_2^n$ the procedure* Hash2Bins$(x, \widehat{\chi}, b, \sigma, a)$ *given in Algorithm 2 computes the following using $O(B)$ samples from x in time $O(Bn \log B + \|\widehat{\chi}\|_0 \cdot n \log B)$*

$$\widehat{u}_\sigma^a(j) = \sum_{f\in\mathbb{F}_2^n : \sigma^\top f = j} (\widehat{x-\chi})_f \cdot (-1)^{\langle a,f\rangle}.$$

3.3 Exact Fourier recovery

In this section we present our algorithm for solving the exact low order Fourier sparse problem defined in (1) and prove Theorem 2. Let

$$S = \mathrm{supp}(\widehat{x}) = \{f \in \mathbb{F}_2^n | \widehat{x}_f \neq 0\}.$$

Problem (1) implies that $|S| \leq k$ and also for every $f \in S$, $|f| \leq d$. The recovery algorithm hashes the frequencies into $B = 2^b$ buckets using Algorithm 2. Every frequency in the support $f \in S$ is recoverable if no other frequency from the support collides with it in the hashed signal. The collision event is formally defined below,

Definition 11 (Collision)**.** For any frequency $f \in \mathbb{F}_2^n$ and every sparse signal $\widehat{x}$ with support $S = \mathrm{supp}(\widehat{x})$ we define the collision event $E_{coll}(f)$ corresponding to the hash function $h(f) = \sigma^\top f$ as,

- $E_{coll}(f)$: holds iff $h(f) \in h(S \setminus \{f\})$.

Claim 12 (Probability of collision)**.** *For every $f \in \mathbb{F}_2^n$, if the hash function $h : \mathbb{F}_2^n \to \mathbb{F}_2^b$ is defined as $h(\cdot) = \sigma^\top(\cdot)$, where $\sigma \in \mathbb{F}_2^{n\times b}$ is a random matrix whose entries are distributed independently and uniformly at random on $\mathbb{F}_2$ then the collision probability (see Definition 11) satisfies,*

$$\Pr[E_{coll}(f)] \leq \frac{k}{B}.$$

The above probability is over the randomness of matrix σ.

Proof. The probability of collision is equal to the probability that at least one of the elements of $S \setminus \{f\}$ get hashed into the bucket $h(f)$. Hence, Claim 9 implies that $\Pr[E_{coll}(f)] \leq \frac{|S|}{B} \leq \frac{k}{B}$. □

If the hash function $h(\cdot) = \sigma^\top(\cdot)$ is such that the collision event $E_{coll}(f)$ doesn't occur for a frequency f, then it follows from Claim 8 and Definition 11 that for every $a \in \mathbb{F}_2^n$,

$$\widehat{u}_\sigma^a(h(f)) = \widehat{x}_f \cdot (-1)^{\langle a,f\rangle}.$$

Therefore under this condition, the problem reduces to one sparse recovery. If $a = \{0\}^n$ then, $\widehat{u}_\sigma^a(h(f)) = \widehat{x}_f$. Hence for any $m \in \mathbb{F}_2^n$, one can learn the inner product $\langle m, f\rangle$ by comparing the sign of $\widehat{u}_\sigma^m(h(f)) = \widehat{x}_f \cdot (-1)^{\langle m,f\rangle}$ and $\widehat{u}_\sigma^a(h(f))$. If the signs are the same then $(-1)^{\langle m,f\rangle} = 1$ meaning that $\langle m, f\rangle = 0$ and if the signs are different then $\langle m, f\rangle = 1$. In previous section we gave an algorithm for learning a low order frequency $|f| \leq d$ from measurements of the form $\langle m, f\rangle$ so putting these together gives the inner subroutine for our sparse fast Hadamard transform which performs one round of hashing, presented in Algorithm 3.

Algorithm 3 SHTInner

```
input: signal $x \in \mathbb{R}^{2^n}$, signal $\widehat{\chi} \in \mathbb{R}^{2^n}$, failure probability $p$, integer $b$, integer $d$.
output: recovered signal $\widehat{\chi}'$.
 1: Let $\{v^l\}_{l=0}^{\lceil \log_2 n \rceil} \subset \mathbb{F}_2^n$ be the binary search vectors on $n$ elements (Definition 4).
 2: $D \leftarrow$ smallest power of two integer such that $D \geq 128d$.
 3: $R \leftarrow \lceil 2\log_2(1/p) \rceil$.
 4: For every $r \in \{0, 1, \cdots, \log_4 D\}$ and every $s \in [R]$, let $h_r^s : [n] \to [D/2^r]$ be an independent
    copy of a pairwise independent hash function.
 5: For every $r \in \{0, 1, \cdots, \log_4 D\}$, every $s \in [R]$, and every $j \in [D/2^r]$ let $w_{r,s}^j \in \mathbb{F}_2^n$ be the binary
    indicator vector of the set $h_r^s(j)^{-1}$.
 6: For every $s \in [R]$, every $r \in \{0, 1, \cdots, \log_4 D\}$ and every $l \in \{0, 1, \cdots, \lceil \log_2 n \rceil\}$ and every
    $j \in [D/2^r]$, add $w_{r,s}^j \cdot v^l$ to set $A_s$.
 7: Let $\sigma \in \mathbb{F}_2^{n \times b}$ be a random matrix. Each entry is independent and uniform on $\mathbb{F}_2$.
 8: For every $a \in \cup_{s \in [R]} A_s$ compute $\widehat{u}_\sigma^a = \textsc{Hash2Bins}(x, \widehat{\chi}, b, \sigma, a)$.
 9: for $j = 1$ to $B$ do
10:     Let $L$ be an empty multi-set.
11:     for $s \in [R]$ do
12:         for every $r \in \{0, 1, \cdots, \log_4 D\}$, every $i \in [D/2^r]$, and every $l \in \{0, 1, \cdots, \lceil \log_2 n \rceil\}$ do
13:             if $\widehat{u}_\sigma^c(j) \neq 0$, where $c = \{0\}^n$ then
14:                 if $\widehat{u}_\sigma^c(j)$ and $\widehat{u}_\sigma^{w_{r,s}^i \cdot v^l}(j)$ have same sign then
15:                     $\phi_r^l(i) \leftarrow 0$.
16:                 else
17:                     $\phi_r^l(i) \leftarrow 1$.
18:         $\tilde{f} \leftarrow \textsc{RecoverFrequency}\left(D, \{h_r^s\}_{r=0}^{\log_2 D}, \left\{\{\phi_r^l\}_{r=0}^{\log_4 D}\right\}_{l=0}^{\lceil \log_2 n \rceil}\right)$.
19:         Append $\tilde{f}$ to multi-set $L$.
20:     $f \leftarrow$ majority$(L)$
21:     $\widehat{\chi}'_f \leftarrow \widehat{u}^c(j)$, where $c = \{0\}^n$.
22: return $\widehat{\chi}'$.
```

Lemma 13. *For all integers b and d, every signals $x, \widehat{\chi} \in \mathbb{R}^{2^n}$ such that $|\xi| \leq d$ for every $\xi \in supp(\widehat{x - \chi})$, and any parameter $p > 0$, Algorithm 3 outputs a signal $\widehat{\chi}' \in \mathbb{R}^{2^n}$ such that $|supp(\widehat{\chi}')| \leq |supp(\widehat{x - \chi})|$ and also for every frequency $f \in supp(\widehat{x - \chi})$, if the collision event $E_{coll}(f)$ doesn't hold then,*

$$\Pr\left[\widehat{\chi}'_f = (\widehat{x - \chi})_f\right] \geq 1 - p.$$

Moreover the sample complexity of this procedure is $O(Bd \log n \log \frac{1}{p})$ and also its time complexity is $O\left(B \log B(n + d \log n \log \frac{1}{p}) + nB \log n \log d \log \frac{1}{p} + \|\widehat{\chi}\|_0 \cdot n(\log B + \log n \log d \log \frac{1}{p})\right)$.

Proof. Because of the if condition in line 13 of the algorithm the empty buckets don't contribute to nonzero elements in the recovered signal $\widehat{\chi}'$. Hence, $|\text{supp}(\widehat{\chi}')| \leq |\text{supp}(\widehat{x - \chi})|$. Assume that $E_{coll}(f)$ does not hold for some arbitrary $f \in \text{supp}(\widehat{x - \chi})$. Therefore, for every $a \in \mathbb{F}_2^n$,

$$\widehat{u}_\sigma^a(h(f)) = (\widehat{x - \chi})_f \cdot (-1)^{\langle a, f \rangle}.$$

Fix one $s \in [R]$. For every $r \in \{0, 1, \cdots, \log_4 D\}$, every $i \in [D/2^r]$ and every $l \in \{0, 1, \cdots, \lceil \log_2 n \rceil\}$, the algorithm compares the sign of $\widehat{u}_\sigma^c(j)$ and $\widehat{u}_\sigma^{w_{r,s}^i \cdot v^l}(j)$, where $c = \{0\}^n$. If they have the same sign then it means that $\langle f, w_{r,s}^i \cdot v^l \rangle = 0$ otherwise $\langle f, w_r^i \cdot v^l \rangle = 1$. Hence, $\phi_r^l(i) = \langle f, w_r^i \cdot v^l \rangle = \sum_{j \in h_r^{-1}(i)} f_j \cdot v_j^l$. Lemma 6 implies that $\textsc{RecoverFrequency}\left(D, \{h_r\}_{r=0}^{\log_2 D}, \left\{\{\phi_r^l\}_{r=0}^{\log_4 D}\right\}_{l=0}^{\lceil \log_2 n \rceil}\right)$ recovers f with probability $1 - 1/8$ in each iteration of the for loop over $s \in [R]$. The algorithm repeats this independently for every $s \in [R]$ and then takes a majority vote over all the outputted frequencies by RecoverFrequency. Frequency f in line 20 of the algorithm is the frequency which appears the most in the output of RecoverFrequency. The probability of failing to recover f is the following,

$$\binom{R}{R/2} \cdot (1/8)^{R/2} \leq (1/2)^{R/2} \leq p.$$

Then the algorithm estimates the value of $\widehat{\chi}'_f$ in line 21 as $\widehat{\chi}'_f = \widehat{u}^c(h(f)) = (\widehat{x - \chi})_f$ which is correct with probability one.

Runtime: Computing the hashing $\widehat{u}_\sigma^a$ with all the different shift parameters $a \in \cup_{s \in [R]} A_s$ is one of the most expensive operations in this procedure. For a fixed $a \in \mathbb{F}_2^n$, in order to compute $x_{\sigma t + a}$ we need B time samples, one for each $t \in \mathbb{F}_2^b$. The computation of the indices σt and all $t \in \mathbb{F}_2^b$ is upper bounded by $O(nB \log B)$ operations. Given that we have computed σt for all $t \in \mathbb{F}_2^b$ and stored it in memory, for a fixed a computing $\sigma t + a$ for all t takes $O(B\|a\|_0)$ operation. Note that vectors $a \in \cup_{s \in [R]} A_s$ are sparse because for every r, s, j the vector $w_{r,s}^j$ in line 5 of Algorithm 3 has only $2^r n/D$ non-zero entries. Therefore, the total complexity of forming the reduced signals $x_{\sigma t + a}$ for all $a \in \cup_{s \in [R]} A_s$ is $O(nB \log B + nB \log n \log d \log(1/p))$. The computational complexity of a fast Walsh Hadamard transform on $x_{\sigma t + a}$ is equal to $B \log_2 B$. Hence, the computational complexity of computing the hashings $\widehat{u}_\sigma^a$ is $O(B \log B(d \log n \log(1/p) + n) + nB \log n \log d \log(1/p))$.
We also need to subtract off the current estimate χ. For each frequency $f \in \mathrm{supp}(\widehat{\chi})$ we compute $j = \sigma^\top f$. This takes $O(\|\widehat{\chi}\|_0 n \log B)$ time in total. Next for each frequency $f \in \mathrm{supp}(\widehat{\chi})$ and each $a \in \cup_{s \in [R]} A_s$ the inner product $\langle f, a \rangle$ needs to be computed. This takes total time of $O(\|\widehat{\chi}\|_0 \cdot n \log n \log d \log(1/p))$. Hence the total runtime for this part is:

$$O\left(B \log B(n + d \log n \log(1/p)) + nB \log n \log d \log(1/p) + \|\widehat{\chi}\|_0 \cdot n(\log B + \log n \log d \log(1/p))\right)$$

By Lemma 6, time to run RecoverFrequency on all the buckets is $O(Bd \log n \log d \log(1/p))$, hence the total runtime of Algorithm 3 is,

$$O\left(B \log B\left(n + d \log n \log \frac{1}{p}\right) + nB \log n \log d \log \frac{1}{p} + \|\widehat{\chi}\|_0 \cdot n\left(\log B + \log n \log d \log \frac{1}{p}\right)\right)$$

Sample complexity: Samples are only consumed for computing the hashings $\widehat{u}_\sigma^a$ for all $a \in \cup_{s \in [R]} A_s$. Hence, the total sample complexity is $O(Bd \log n \log \frac{1}{p})$. □

Lemma 14. *For any parameter $p > 0$, all integers k, d, and $b \geq \log_2(k/p)$, every signals $x, \widehat{\chi} \in \mathbb{R}^{2^n}$ such that $\|\widehat{x - \chi}\|_0 \leq k$ and $|\xi| \leq d$ for every $\xi \in supp(\widehat{x - \chi})$, the output of* $\textsc{SHTInner}(x, \widehat{\chi}, p, b, d)$, *$\widehat{\chi}'$ satisfies the following with probability $1 - 32p$,*

$$\|\widehat{x} - \widehat{\chi} - \widehat{\chi}'\|_0 \leq k/8.$$

Proof. The number of buckets is $B = 2^b \geq \frac{k}{p}$, hence, for every $f \in \mathrm{supp}(\widehat{x} - \widehat{\chi})$ the probability of the collision event $E_{coll}(f)$ is at most $k/B \leq p$. Hence, from Lemma 13 along with a union bound it follows that, for every $f \in \mathrm{supp}(\widehat{x - \chi})$,

$$\Pr[\widehat{\chi}'_f = (\widehat{x - \chi})_f] \geq 1 - 2p.$$

Therefore by Markov's inequality it follows that with probability $1-32p$,

$$\left|\left\{f \in \operatorname{supp}(\widehat{x-\chi}) : \widehat{\chi}'_f \neq (\widehat{x-\chi})_f\right\}\right| \leq k/16,$$

It also follows from Lemma 13 that $\|\widehat{\chi}'\|_0 \leq k$, hence, $\|\widehat{x} - \widehat{\chi} - \widehat{\chi}'\|_0 \leq k/8$. □

Our sparse Hadamard transform algorithm iteratively calls the primitive SHTInner to reduces the sparsity of the residual signal by a constant factor hence it terminates in $O(\log k)$ iteration. See Algorithm 4.

Algorithm 4 ExactSHT

```
input: signal $x \in \mathbb{R}^{2^n}$, failure probability $q$, sparsity $k$, integer $d$.
output: estimate $\widehat{\chi} \in \mathbb{R}^{2^n}$.
 1: $p^{(1)} \leftarrow q/32$.
 2: $b^{(1)} \leftarrow \lceil \log_2 \frac{64k}{q} \rceil$.
 3: $T \leftarrow \lceil \log_8 k \rceil$.
 4: $w^{(0)} \leftarrow \{0\}^{2^n}$.
 5: for $r = 1$ to $T$ do
 6:     $\widetilde{\chi} \leftarrow$ SHTInner$(x, w^{(r-1)}, p^{(r)}, b^{(r)}, d)$
 7:     $w^{(r)} \leftarrow w^{(r-1)} + \widetilde{\chi}$.
 8:     $p^{(r+1)} \leftarrow p^{(r)}/2$.
 9:     $b^{(r+1)} \leftarrow b^{(r)} - 2$.
10: $\widehat{\chi} \leftarrow w^{(T)}$.
11: return $\widehat{\chi}$.
```

Proof of Theorem 2: The proof is by induction. We denote by $\mathcal{E}_r$ the event corresponding to $\|\widehat{x} - w^{(r)}\|_0 \leq \frac{k}{8^r}$. The inductive hypothesis is that,

$$\Pr[\mathcal{E}_r | \mathcal{E}_{r-1}] \geq 1 - 16p^{(r)}.$$

Conditioned on $\mathcal{E}_{r-1}$ we have that $\|\widehat{x} - w^{(r-1)}\|_0 \leq \frac{k}{8^{r-1}}$. The number of buckets in iteration r of the algorithm is $B^{(r)} = 2^{b^r} \geq \frac{64k}{4^{r-1} \cdot q}$. Hence, it follows from Lemma 14, that with probability $1 - 32p^{(r)}$,

$$\|\widehat{x} - w^{(r)}\|_0 \leq \frac{k}{8^r}.$$

This proves the inductive step.

Runtime and Sample complexity: In iteration $r \in [\lceil \log_8 k \rceil]$, the size of the bucket $B^{(r)} = 2^{b^{(r)}} = \frac{64k}{q \cdot 4^r}$ and the error probability $p^{(r)} = \frac{q}{32 \cdot 2^r}$. Moreover at most $\sum_r B^{(r)}$ elements are added to $\widehat{\chi}$, hence we can assume that $\|\widehat{\chi}\|_0 \leq \frac{128k}{q}$. From Lemma 13 it follows that the runtime at iteration r is:

$$O\left(\frac{k}{4^r} \log \frac{k}{4^r} \left(n + d \log n \log 2^r\right) + \frac{k}{4^r} n \log n \log d \log 2^r + kn \left(\log \frac{k}{4^r} + \log n \log d \log 2^r\right)\right).$$

Summing over all $r = 1, \ldots, \lceil \log_8 k \rceil$, the total runtime is $O\left(kn \log^2 k \log n \log d\right)$.

The sample complexity of iteration r is $O\left(\frac{kd}{2^r} \log n \log 2^r\right)$ hence the total sample complexity is dominated by the sample complexity of the first iteration which is equal to $O\left(kd \log n\right)$. □

4 Robust recovery

In this section we present an algorithm to solve the robust set function learning, Problem (2). We also analyze our algorithm and prove Theorem 2. We show that our robust recovery algorithm achieves the well studied ℓ_2/ℓ_2 sparse recovery guarantee without making any assumptions on the input signal. Any general signal can be decomposed into two parts, *head* and *tail* which are defined in Definition 1. The *head* is basically the top k coefficients of $\widehat{x}_f$ such that their corresponding frequencies, f, have Hamming weight at most d.

We use hashing techniques similar to our exact recovery algorithm to solve the robust Problem (2). We can achieve the ℓ_2/ℓ_2 guarantee for general signals using $O(kd \log_2 n \log_2(d \log_2 n))$ samples which heavily relies on our novel and sample optimal primitive for recovery of low Hamming weight frequencies presented in Section 3.1.

Note that the sample complexity of our robust algorithm in comparison to the known lower bound for this problem is only off by a $\log(d \log n)$ [PW11]. The reason for this extra factor is that when we hash a general and non-sparse signal $\widehat{x}$ into $O(k)$ buckets, the noise in each bucket might add up constructively and dominate the *head* of signal, $\widehat{x}_{head}$. Therefore we need to repeat the hashing a number of times and use median trick to get and accurate estimate to the *head* elements up to noise level.

4.1 Preliminaries

We consider general signals which are not necessarily sparse nor their support contains low order frequencies only. We first need to define the notion of *approximate support* for a signal. We repeat Definition 1 here.

Definition 15 (Head and Tail norm)**.** For all integers n, d, and k we define the *head* of $\widehat{x} : \mathbb{F}_2^n \to \mathbb{R}$ as,

$$\widehat{x}_{head} := \arg \min_{\substack{y:\mathbb{F}_2^n \to \mathbb{R} \\ \|y\|_0 \le k \\ |j| \le d \text{ for all } j \in \mathrm{supp}(y)}} \|\widehat{x} - y\|_2.$$

The *tail norm* of $\widehat{x}$ is defined as, $\mathrm{Err}(\widehat{x}, k, d) := \|\widehat{x} - \widehat{x}_{head}\|_2^2$.

For ease of notation, we state the next definitions and claims in terms of the residual signal $\widehat{x}' \triangleq \widehat{x} - \widehat{\chi}$ since we will be working with this signal in all iterations.

Definition 16 (Covered frequencies)**.** The average tail norm per bucket is denoted by ρ,

$$\rho := \frac{\mathrm{Err}(\widehat{x}', k, d)}{B}.$$

The set of covered frequencies is defined as,

$$S_\alpha := \{f \in \mathbb{F}_2^n : |f| \le d \text{ and } |\widehat{x}'_f|^2 \ge \alpha \cdot \rho\}$$

for some $\alpha > 1$.

S_α contains those frequencies that are recoverable by our algorithm. Intuitively, it contains all the frequencies that have large Fourier coefficients compared to average noise in each bucket. We show how recovering the frequencies in S_α provides a good ℓ_2/ℓ_2 approximation.

It follows from the definition of the set of covered frequencies S_α that if we let $\widehat{x}'_{S_\alpha}$ be the residual signal $\widehat{x}'$ restricted to set S_α then the following holds,

$$\begin{aligned}
\|\widehat{x}' - \widehat{x}'_{S_\alpha}\|_2^2 &= \|(\widehat{x}'_{head} + \widehat{x}'_{tail}) - (\widehat{x}'_{head\cap S_\alpha} + \widehat{x}'_{tail\cap S_\alpha})\|_2^2 \\
&= \|\widehat{x}'_{head} - \widehat{x}'_{head\cap S_\alpha}\|_2^2 + \|\widehat{x}'_{tail} - \widehat{x}'_{tail\cap S_\alpha}\|_2^2 \\
&\leq |head \setminus S_\alpha| \cdot (\alpha\rho) + \|\widehat{x}'_{tail}\|_2^2 \\
&\leq \left(1 + \alpha \cdot \frac{k}{B}\right) \cdot \mathrm{Err}(\widehat{x}', k, d). \qquad (5)
\end{aligned}$$

It also follows that,

$$|S_\alpha| \leq \frac{B}{\alpha} + k. \qquad (6)$$

The reason is that out of the k elements in the support of $\widehat{x}'_{head}$, S_α can contains all of them and out of the components in the tail it can not contain more than $\frac{B}{\alpha}$ elements since this would imply the energy present in these components is at least $\frac{B}{\alpha} \cdot (\alpha\rho) = \mathrm{Err}(\widehat{x}', k, d)$ which is a contradiction. We provide an algorithm that recovers S_α with constant probability. We face a trade-off when picking the value of α. If α is small, we are guaranteed a better approximation however the number of elements of S_α increase meaning extra runtime.

Definition 17 (Collision and Large noise events)**.** For any frequency $f \in \mathbb{F}_2^n$ and every residual signal $\widehat{x}' \in \mathbb{R}^{\mathbb{F}_2^n}$ we define two types of bad events, $E_{noise}(f)$ and $E_{coll}(f)$ corresponding to the hash function $h(f) = \sigma^\top f$,

1. Large noise, $E_{noise}(f)$: holds iff $\|\widehat{x}'_{h^{-1}(h(f))\setminus S_\alpha}\|_2^2 \geq \beta \cdot \rho$ for some $\beta > 1$.
2. Collision, $E_{coll}(f)$: holds iff $h(f) \in h(S_\alpha \setminus \{f\})$.

If the large noise event happen, the noise might dominate the frequency $f \in S_\alpha$. The collision event happens if at least two elements of S_α are hashed to the same bucket.

Claim 18 (Probability of collision and large noise)**.** *For every $f \in \mathbb{F}_2^n$ the collision probability (see Definition 17) is upper bounded as follows,*

$$\Pr[E_{coll}(f)] \leq \frac{1}{\alpha} + \frac{k}{B}.$$

Also the probability of large noise (see Definition 17) is bounded as follows,

$$\Pr[E_{noise}(f)] \leq \frac{1 + \alpha(k/B)}{\beta}.$$

The above probabilities are over the randomness of matrix σ which is used for hashing.

Proof. The probability of collision is equal to the probability that at least one of the elements of $S_\alpha \setminus \{f\}$ get hashed into the bucket $h(f)$. Hence, it follows from Claim 9 along with (6) that,

$$\Pr[E_{coll}(f)] \leq \frac{|S_\alpha|}{B} \leq \frac{1}{\alpha} + \frac{k}{B}.$$

To calculate the probability of large noise, first note that (5) implies the following for the expected noise energy per bucket (the randomness is over the hashing),

$$\begin{aligned}
\mathbb{E}\left[\|\widehat{x}'_{h^{-1}(h(f))\setminus S_\alpha}\|_2^2\right] &= \sum_{j\in\mathbb{F}_2^n\setminus S_\alpha} |\widehat{x}'_j|^2 \cdot \Pr[j \in h^{-1}(h(f))] \\
&= \frac{1}{B}\cdot \|\widehat{x}' - \widehat{x}'_{S_\alpha}\|_2^2 \\
&\le \left(1+\alpha\cdot\frac{k}{B}\right)\cdot\frac{\mathrm{Err}(\widehat{x}',k,d)}{B} \\
&= \left(1+\alpha\cdot\frac{k}{B}\right)\cdot\rho
\end{aligned}$$

Hence, by Markov's inequality we have,

$$\Pr\left[\|\widehat{x}'_{h^{-1}(h(f))\setminus S_\alpha}\|_2^2 \ge \beta\cdot\rho\right] \le \frac{1+\alpha\cdot(k/B)}{\beta}.$$

□

4.2 Location

In this section we design a primitive to locate any covered frequency $f \in S_\alpha$ with constant probability (Algorithm 5). More precisely, for any covered frequency $f \in S_\alpha$ if neither of the bad events $E_{coll}(f)$ and $E_{noise}(f)$ occur then f can be recovered with constant probability. Moreover, the events $E_{coll}(f)$ and $E_{noise}(f)$ also occur with constant probability by Claim 18.

Algorithm 5 Locate

input: signal $x \in \mathbb{R}^{2^n}$, signal $\widehat{\chi} \in \mathbb{R}^{2^n}$, failure probability p, integer b, integer d.
output: list of covered frequencies L.

```
1: Let $\{v^l\}_{l=0}^{\lceil \log_2 n \rceil} \subset \mathbb{F}_2^n$ be the binary search vectors on $n$ elements (Definition 4).
2: $D \leftarrow$ smallest power of two integer such that $D \geq 128d$.
3: $R \leftarrow \lceil 6 \log_2(1/p) \rceil$.
4: Let $A$ be a set of $\Theta(\log(D \log n))$ iid uniform random vectors in $\mathbb{F}_2^n$.
5: For every $r \in \{0, 1, \cdots, \log_4 D\}$ and every $s \in [R]$ let $h_r^s : [n] \to [D/2^r]$ be an independent copy
   of a pairwise independent hash function.
6: For every $r \in \{0, 1, \cdots, \log_4 D\}$, every $s \in [R]$, and every $j \in [D/2^r]$ let $w_{r,s}^j \in \mathbb{F}_2^n$ be the binary
   indicator vector of the set $h_r^s(j)^{-1}$.
7: For every $a \in A$, every $r \in \{0, 1, \cdots, \log_4 D\}$, every $s \in [R]$, every $l \in \{0, 1, \cdots, \lceil \log_2 n \rceil\}$, and
   every $j \in [D/2^r]$, add $a + w_{r,s}^j \cdot v^l$ to set $A'_s$.
8: Let $\sigma \in \mathbb{F}_2^{n \times b}$ be a random matrix. Each entry is independent and uniform on $\mathbb{F}_2$.
9: For every $a \in A \cup \left( \cup_{s \in [R]} A'_s \right)$ compute $\widehat{u}_\sigma^a = \textsc{Hash2Bins}(x, \widehat{\chi}, b, \sigma, a)$.
10: $L \leftarrow \emptyset$.
11: for $j = 1$ to $B$ do
12:     $S \leftarrow$ Empty multi-set.
13:     for $s \in [R]$ do
14:         for every $r \in \{0, 1, \cdots, \log_4 D\}$, every $i \in [D/2^r]$, and every $l \in \{0, 1, \cdots, \lceil \log_2 n \rceil\}$ do
15:             $C \leftarrow 0$.
16:             for every $a \in A$ do
17:                 if $\widehat{u}_\sigma^a(j)$ and $\widehat{u}_\sigma^{a + w_{r,s}^i \cdot v^l}(j)$ have same sign then
18:                     $C \leftarrow C + 1$.
19:             if $C \geq \frac{|A|}{2}$ then
20:                 $\phi_r^l(i) \leftarrow 0$.
21:             else
22:                 $\phi_r^l(i) \leftarrow 1$.
23:         $\tilde{f} \leftarrow \textsc{RecoverFrequency}\left( D, \{h_r^s\}_{r=0}^{\log_2 D}, \left\{ \{\phi_r^l\}_{r=0}^{\log_4 D} \right\}_{l=0}^{\lceil \log_2 n \rceil} \right)$.
24:         Append $\tilde{f}$ to multi-set $S$.
25:     $f^* \leftarrow \text{majority}(S)$.
26:     $L \leftarrow L \cup \{f^*\}$.
27: return $L$.
```

Lemma 19. *For any positive integers n, b, d, every signals $x, \widehat{\chi} : \mathbb{F}_2^n \to \mathbb{R}$, every $p > 0$ and every covered frequency $f \in S_\alpha$, assuming that neither $E_{coll}(f)$ nor $E_{noise}(f)$ hold, if $\alpha \geq 10\beta$ then the procedure* Locate *(Algorithm 5) returns a list L of size $|L| \leq B$ such that,*

$$\Pr[f \in L] \geq 1 - p.$$

Moreover the runtime of this procedure is

$$O\left(nB \log B + (d \log B + n) B \log n \log(d \log n) \log d \log \frac{1}{p} + n \|\widehat{\chi}\|_0 (\log B + \log n \log(d \log n) \log d \log \frac{1}{p}) \right)$$

and the number of accesses to the signal x is $O\left(Bd \log n \log(d \log n) \log(1/p)\right)$.

Proof. Let j be the bucket that f is hashed into, $j = h(f) = \sigma^\top f$. By the assumption of the lemma, $E_{coll}(f)$ doesn't hold and hence $j \notin h(S_\alpha \setminus \{f\})$. Also it is assumed that $E_{noise}(f)$ doesn't hold, hence,

$$\|\widehat{x}'_{h^{-1}(j)\setminus\{f\}}\|_2^2 < \beta\rho.$$

Therefore, for a uniformly random $a \in \mathbb{F}_2^n$ we have,

$$\begin{aligned}\mathbb{E}_a\left[\left|\widehat{u}_\sigma^a(j) - \widehat{x}'_f \cdot (-1)^{\langle a,f\rangle}\right|^2\right] &= \mathbb{E}_a\left[\left|\sum_{f'\in h^{-1}(j)\setminus\{f\}} \widehat{x}'_{f'} \cdot (-1)^{\langle a,f'\rangle}\right|^2\right] \\ &= \sum_{f'\in h^{-1}(j)\setminus\{f\}} |\widehat{x}'_{f'}|^2 \\ &< \beta\rho.\end{aligned}$$

By Markov's inequality,

$$\Pr\left[\left|\widehat{u}_\sigma^a - \widehat{x}'_f \cdot (-1)^{\langle a,f\rangle}\right|^2 < 10\beta\rho\right] \geq 9/10.$$

By the assumption $\alpha \geq 10\beta$, the above implies that $\widehat{u}^a$ and $\widehat{x}'_f \cdot (-1)^{\langle a,f\rangle}$ have the same sign with probability 9/10. Now, fix one $s \in [R]$. Similar to the above argument, for every $r \in \{0, 1, \cdots, \log_4 D\}$, every $i \in [D/2^r]$, and every $l \in \{0, 1, \cdots, \lceil \log_2 n \rceil\}$, $\widehat{u}^{a+w_{r,s}^i \cdot v^l}$ and $\widehat{x}'_f \cdot (-1)^{\langle a+w_{r,s}^i \cdot v^l, f\rangle}$ have the same sign with probability 9/10. Therefore for every $a \in A$, every $r \in \{0, 1, \cdots, \log_4 D\}$, every $i \in [D/2^r]$, and every $l \in \{0, 1, \cdots, \lceil \log_2 n \rceil\}$, line 17 of Algorithm 5 correctly determines the inner product $\langle w_{r,s}^i \cdot v^l, f\rangle$ with probability 8/10. Then Algorithm 5 uses the median trick to boost the success probability. The failure probability after taking the median over all elements of A is,

$$\binom{|A|}{|A|/2} \cdot (2/10)^{|A|/2} \leq (\frac{4}{5})^{|A|/2} \leq O\left(\frac{1}{D\log n}\right).$$

By a union bound over all r, l, and i, the algorithm can determine the inner products $\langle w_{r,s}^i \cdot v^l, f\rangle$ simultaneously for all r, l, and i with probability $1 - 1/16$. By Lemma 6, the procedure RecoverFrequency$\left(D, \{h_r^s\}_{r=0}^{\log_2 D}, \left\{\{\phi_r^l\}_{r=0}^{\log_4 D}\right\}_{l=0}^{\lceil \log_2 n\rceil}\right)$ outputs f correctly with probability $1 - 1/8$. Hence, by union bound, the frequency f gets recovered with probability $1 - 3/16$ in each iteration of the for loop over $s \in [R]$. The algorithm repeats this independently for every $s \in [R]$ and then takes a majority vote over all the outputted frequencies by RecoverFrequency. Frequency f^* in line 25 of Algorithm 5 is the frequency which appears the most in the output of RecoverFrequency. The probability of failing to recover f is the following,

$$\binom{R}{R/2} \cdot (3/16)^{R/2} \leq (3/4)^{R/2} \leq p.$$

Runtime: Computing the hashing $\widehat{u}_\sigma^a$ with all the different shift parameters $a \in A \cup (\cup_{s\in[R]} A'_s)$ dominates the runtime of this procedure. For a fixed $a \in \mathbb{F}_2^n$, in order to compute $x_{\sigma t+a}$ we need B time samples, one for each $t \in \mathbb{F}_2^b$. The computation of the indices σt and all $t \in \mathbb{F}_2^b$ is upper bounded by $O(nB\log B)$ operations. Given that we have computed σt for all $t \in \mathbb{F}_2^b$ and stored it in memory, for a fixed a' computing $\sigma t + a'$ for all t takes $O(Bn)$ operation. Note that vectors $a \in \cup_{s\in[R]} A_s$ can be written as $a = a' + w_{r,s}^j \cdot v^l$ for some r, s, j, l and some $a' \in A$. We can compute $\sigma t + a'$ for

all $t \in \mathbb{F}_2^b$ and all $a' \in A$ and stored it in memory in time $O(nB(\log B + \log(d \log n)))$. Note that for every r, s, j the vector $w_{r,s}^j$ in line 6 of Algorithm 5 has only $2^r n/D$ non-zero entries. Therefore, given that $\sigma t + a'$ is computed and stored in memory for all $t \in \mathbb{F}_2^b$ and all $a' \in A$, we can compute $\sigma t + a$ for all $t \in \mathbb{F}_2^b$ and $a \in A \cup (\cup_{s\in[R]} A_s)$, in time $O(Bn \log n \log(d \log n) \log d \log \frac{1}{p})$. Hence, the total complexity of forming the reduced signals $x_{\sigma t+a}$ for all $a \in A \cup (\cup_{s\in[R]} A_s)$ is $O(nB(\log B + \log n \log(d \log n) \log d \log \frac{1}{p}))$. The computational complexity of a fast Walsh Hadamard transform on $x_{\sigma t+a}$ is equal to $B \log_2 B$. Hence, the computational complexity of computing the hashings $\widehat{u}_\sigma^a$ is $O(B \log B(d \log n \log(d \log n) \log(1/p) + n) + nB \log n \log(d \log n) \log d \log(1/p))$.
We also need to subtract off the current estimate χ. For each frequency $f \in \text{supp}(\widehat{\chi})$ we compute $j = \sigma^\top f$. This takes $O(\|\widehat{\chi}\|_0 n \log B)$ time in total. Next for each frequency $f \in \text{supp}(\widehat{\chi})$ and each $a \in A \cup (\cup_{s\in[R]} A_s)$ the inner product $\langle f, a\rangle$ needs to be computed. This takes total time of $O(\|\widehat{\chi}\|_0 \cdot n \log n \log(d \log n) \log d \log(1/p))$. Hence the total runtime for this part is:

$$O\left(nB \log B + (d \log B + n)B \log n \log(d \log n) \log d \log \frac{1}{p} + n\|\widehat{\chi}\|_0(\log B + \log n \log(d \log n) \log d \log \frac{1}{p})\right)$$

Sample complexity: The sample complexity is the number of samples taken to form the hashings. The algorithm hashes the signal with $\left|A \cup \left(\cup_{s\in[R]} A'_s\right)\right|$ different shift parameters each of which requiring B samples hence the total sample complexity is $O(Bd \log n \log(d \log n) \log \frac{1}{p})$. □

Lemma 20. *For every $f \in S_\alpha$, the output L of the procedure* Locate *satisfies,*

$$\Pr[f \in L] \geq 1 - \left(p + \frac{1}{\alpha} + \frac{k}{B} + \frac{1 + \alpha(k/B)}{\beta}\right).$$

Proof. Note that for every $f \in S_\alpha$, by Claim 18,

$$\Pr\left[E_{coll}(f) \text{ or } E_{noise}(f)\right] \leq \frac{1}{\alpha} + \frac{k}{B} + \frac{1 + \alpha(k/B)}{\beta}.$$

Therefore the preconditions of Lemma 19 hold with probability $1 - \left(\frac{1}{\alpha} + \frac{k}{B} + \frac{1+\alpha(k/B)}{\beta}\right)$. By Lemma 19 given that its preconditions hold we have,

$$\Pr[f \in L \,|\, E_{coll} \text{ and } E_{noise} \text{ not holding}] \geq 1 - p.$$

Hence the lemma follows by a union bound. □

Lemma 21.

$$\mathbb{E}\left[\|\widehat{x}'_{S_\alpha \setminus L}\|_2^2\right] \leq \left(p + \frac{1}{\alpha} + \frac{k}{B} + \frac{1 + \alpha(k/B)}{\beta}\right) \cdot \|\widehat{x}'\|_2^2.$$

Proof. By Lemma 20 we have,

$$\begin{aligned}
\mathbb{E}\left[\|\widehat{x}'_{S_\alpha \setminus L}\|_2^2\right] &= \sum_{f \in S_\alpha} |\widehat{x}'_f|^2 \cdot \Pr[f \notin L] \\
&\leq \sum_{f \in S_\alpha} |\widehat{x}'_f|^2 \cdot \left(2p + \frac{1}{\alpha} + \frac{k}{B} + \frac{1 + \alpha(k/B)}{\beta}\right) \\
&\leq \left(2p + \frac{1}{\alpha} + \frac{k}{B} + \frac{1 + \alpha(k/B)}{\beta}\right) \cdot \|\widehat{x}'\|_2^2.
\end{aligned}$$

□

If we let $\beta = \frac{10}{\delta q}$ and $\alpha = \beta$ and $B = 10\alpha k$ and run the procedure LOCATE with failure probability $p = \delta q/10$, then by Markov's inequality the following holds,

$$\Pr\left[\|\widehat{x}'_{S_\alpha \setminus L}\|_2^2 \le \delta \cdot \|\widehat{x}'\|_2^2\right] \ge 1 - q.$$

4.3 Estimation

In this section we use the hashing technique to estimate the values of the signal $\widehat{x}$ at frequencies $f \in L$ for some set of locations $L \subset \mathbb{F}_2^n$. This is done in Algorithm 6.

Algorithm 6 ESTIMATE

```
input: signal $x \in \mathbb{R}^{2^n}$, signal $\widehat{\chi} \in \mathbb{R}^{2^n}$, failure probability $p$, integer $b$, list of frequencies $L$, integer $k$, parameter $\gamma$.
output: estimated signal $\widehat{\chi}'$.
1: $B \leftarrow 2^b$.
2: $T \leftarrow O(\log \frac{B}{p\gamma k})$.
3: for $r = 1$ to $T$ do
4:     Let $a_r$ be a uniformly random vector in $\mathbb{F}_2^n$.
5:     Let $\sigma_r \in \mathbb{F}_2^{n\times b}$ be a random matrix. Each entry is independent and uniform on $\mathbb{F}_2$.
6:     Compute $\widehat{u}_{\sigma_r}^{a_r} = $ HASH2BINS$(x, \widehat{\chi}, b, \sigma_r, a_r)$.
7: $\widehat{w} \leftarrow \{0\}^{2^n}$.
8: for $f \in L$ do
9:     $\widehat{w}_f \leftarrow \text{median}_{r\in[T]}\left(\widehat{u}_{\sigma_r}^{a_r}(\sigma_r^\top f)\cdot(-1)^{\langle f, a_r\rangle}\right)$.
10: $J \leftarrow \arg\max_{J:|J|=2k} \|\widehat{w}_J\|_2^2$
11: $\widehat{\chi}' \leftarrow \widehat{w}_J$.
12: return $\widehat{\chi}'$.
```

Lemma 22 (Estimation guarantee for a single frequency)**.** *For any $L \subset \mathbb{F}_2^n$ with size $|L| \le B$, any $\gamma > 0$, and any $f \in L$, if $\beta \ge 40$ and $\alpha \ge 4\beta$ and $B \ge 10\alpha k$ then*

1. $\Pr\left[\left|\widehat{x}'_f - \widehat{w}_f\right|^2 \ge 6\beta\rho\right] \le p\gamma\frac{k}{B}$*, where $\widehat{x}' = \widehat{x} - \widehat{\chi}$ is the input signal to the estimation procedure.*
2. $\Pr\left[Err(\widehat{x}'_L - \widehat{\chi}', \gamma k) \le Err(\widehat{x}'_L, k) + 24\beta k\rho\right] \ge 1 - p.$

Moreover the runtime of this procedure is $O\left(Bn\log(B/p\gamma k)\log B + \|\widehat{\chi}\|_0 \cdot n\log(B/p\gamma k)\log B\right)$ and the number of accesses to the signal x is $O\left(B\cdot\log(B/p\gamma k)\right)$.

Proof. For every $f \in L$ and every r, let $j = \sigma_r^\top f$ be the bucket that f is hashed into by the r^{th} hash function. Note that for every r, with probability $1 - \left(1/\alpha + k/B + \frac{1+\alpha(k/B)}{\beta}\right)$, neither of the events $E_{coll}(f)$ and $E_{noise}(f)$ hold. Conditioning on $E_{coll}(f)$ and $E_{noise}(f)$ not holding, we have,

$$\begin{aligned}\mathbb{E}_{a_r}\left[\left|\widehat{x}'_f - \widehat{u}_{\sigma_r}^{a_r}\cdot(-1)^{\langle f,a_r\rangle}\right|^2\right] &= \mathbb{E}\left[\left|\sum_{f'\in h^{-1}(j)\setminus\{f\}} \widehat{x}'_{f'}\cdot(-1)^{\langle f,a_r\rangle}\right|^2\right]\\ &\le \sum_{f'\in h^{-1}(j)\setminus\{f\}} |\widehat{x}'_{f'}|^2\\ &\le \beta\rho.\end{aligned}$$

Therefore by Markov's inequality and a union bound,

$$Pr\left[\left|\widehat{x}'_f - \widehat{u}^{a_r}_{\sigma_r}\cdot(-1)^{\langle f,a_r\rangle}\right|^2 \geq 6\beta\rho\right] \leq 1/6 + 1/\alpha + k/B + \frac{1+\alpha(k/B)}{\beta} \leq 1/5.$$

Hence when we take the median of $\widehat{u}^{a_r}\cdot(-1)^{\langle f,a_r\rangle}$ for all r the error probability goes down as follows,

$$\Pr\left[\left|\widehat{x}'_f - \widehat{w}_f\right|^2 \geq 6\beta\rho\right] \leq \binom{T}{T/2}\cdot(\frac{1}{5})^{T/2} \leq (\frac{4}{5})^{T/2} \leq p\gamma k/(2B).$$

This proves the first claim of the lemma.

Let $U = \left\{f \in L : \left|\widehat{x}'_f - \widehat{w}_f\right|^2 \geq 6\beta\rho\right\}$. It follows from the first claim of the lemma along with Markov's inequality that $|U| \leq |L|\cdot\gamma k/B \leq \gamma k$ with probability $1-p/2$. Conditioning on this happening we have,

$$\|(\widehat{x}' - \widehat{w})_{L\setminus U}\|_\infty^2 \leq 6\beta\rho.$$

Let T denote the top k coordinates of $\widehat{w}_{L\setminus U}$. It follows from the above that,

$$\begin{aligned}\|\widehat{x}'_{L\setminus U} - \widehat{w}_T\|_2^2 &\leq \mathrm{Err}(\widehat{x}'_{L\setminus U}, k) + (3k)\cdot(6\beta\rho)\\ &\leq \mathrm{Err}(\widehat{x}'_L, k) + 18\beta k\rho. \end{aligned} \tag{7}$$

Because J is the top $2k \geq (1+\gamma)k$ coordinates of $\widehat{w}$, $T \subseteq J \setminus U$. Let $J' = J \setminus (T \cup U)$, so $|J'| \leq k$. Therefore,

$$\begin{aligned}\mathrm{Err}(\widehat{x}_L - \widehat{\chi}', \gamma k) &\leq \|\widehat{x}'_{L\setminus U} - \widehat{\chi}'_{J\setminus U}\|_2^2\\ &= \|\widehat{x}_{L\setminus(U\cup J')} - \widehat{\chi}'_T\|_2^2 + \|(\widehat{x} - \widehat{\chi}')_{J'}\|_2^2\\ &\leq \|\widehat{x}_{L\setminus U} - \widehat{\chi}'_T\|_2^2 + |J'|\cdot\|(\widehat{x}-\widehat{\chi}')_{J'}\|_\infty^2\\ &\leq \mathrm{Err}(\widehat{x}_L, k, d) + 18\beta k\rho + 6\beta k\rho\\ &= \mathrm{Err}(\widehat{x}_L, k, d) + 24\beta k\rho.\end{aligned}$$

Third inequality above follows from (7).

Runtime: The runtime is dominated by the time to compute the hashings $\widehat{u}^{a_r}_{\sigma_r}$ for every $r \in [T]$. By Claim 10, this computation takes a total of $O(Bn\log_2(B/kp\gamma)\log B + \|\widehat{\chi}\|_0\cdot n\log_2(B/kp\gamma)\log B)$ operations.

Sample complexity: The sample complexity is the number of samples taken to form the hashings $u^{a_r}_{\sigma_r}$. The algorithm uses $O(\log(B/kp\gamma))$ hashes each of which requiring B samples and hence the total sample complexity is $O(B\log(B/kp\gamma))$. □

4.4 Reduce SNR

In this section we put the primitives Locate and Estimate together to design a procedure which reduces the norm of the head of the signal $\|\widehat{x}_{head}\|_2$ by a large constant factor while the norm of tail $\mathrm{Err}(\widehat{x}, k)$ only mildly increases hence the signal to noise ratio decreases by a constant factor. Algorithm 7 does this task. We also show that the residual $\widehat{x} - \widehat{\chi}'$ after running this procedure is sparser than the original signal $\widehat{x} - \widehat{\chi}$.

Algorithm 7 ReduceSNR

```
input: signal $x \in \mathbb{R}^{2^n}$, signal $\widehat{\chi} \in \mathbb{R}^{2^n}$, parameter $\delta > 0$, failure probability $q$, sparsity $k$, integer $d$, parameter $\gamma$.
output: refined estimate $\widehat{\chi}'$.
1: $p \leftarrow q\gamma\delta/10$.
2: $b \leftarrow \left\lceil \log_2(\frac{10k}{p\delta}) \right\rceil$.
3: $L \leftarrow$ Locate$(x, \widehat{\chi}, p, b, d)$.
4: $L' \leftarrow \{f \in L : |f| \le d\}$.
5: $\widetilde{\chi} \leftarrow$ Estimate$(x, \widehat{\chi}, p, b, L', k, d, \gamma)$.
6: $\widehat{\chi}' \leftarrow \widehat{\chi} + \widetilde{\chi}$.
7: return $\widehat{\chi}'$.
```

Lemma 23. *For all integers n, k, and d, every parameters $0 < \gamma < 1/2$, $0 < \delta < 0.1$, and $0 < q < 1/2$, every signals $x, \widehat{\chi} : \mathbb{F}_2^n \to \mathbb{R}$, the procedure* ReduceSNR *outputs a signal $\widehat{\chi}' : \mathbb{F}_2^n \to \mathbb{R}$ such that the following holds,*

$$\Pr\left[Err(\widehat{x} - \widehat{\chi}', 2\gamma k, d) \le (1 + 4\delta) \cdot Err(\widehat{x} - \widehat{\chi}, k, d)\right] \ge 1 - q.$$

Moreover, the runtime of this procedure is

$$O\left(\log\frac{1}{q\gamma\delta}\left(\frac{k}{q\gamma\delta^2}(n\log\frac{k}{q\gamma\delta} + (d\log\frac{k}{q\gamma\delta} + n)\log n\log(d\log n)\log d) + n\|\widehat{\chi}\|_0(\log\frac{k}{q\gamma\delta} + \log n\log(d\log n)\log d)\right)\right)$$

and the sample complexity of the procedure is, $O\left(\frac{kd}{q\gamma\delta^2} \cdot \log n \cdot \log(d\log n) \cdot \log\frac{1}{q\gamma\delta}\right)$.

Proof. Let $\beta = 1/p$, $\alpha = 10\beta$, and $B = (\alpha/\delta) \cdot k = \frac{10k}{p\delta}$. Let S_α be the set of covered frequencies of $\widehat{x}' = \widehat{x} - \widehat{\chi}$. Let us denote the output of Estimate$(x, \widehat{\chi}, p, b, L', k, \gamma)$ by $\widetilde{\chi}$. The support of $\widetilde{\chi}$ is denoted by $J = \text{supp}(\widetilde{\chi})$. Also let L be the output of the procedure Locate$(x, \widehat{\chi}, p, b, d)$. Since by definition of covered frequencies, $|f| \le d$ for every $f \in S_\alpha$, by Lemma 20 we have,

$$\Pr[f \in L' | f \in S_\alpha] = \Pr[f \in L | f \in S_\alpha] \ge 1 - q\gamma\delta/2.$$

The above along with $|S_\alpha| \le (1 + 1/\delta)k$, which follows from (6), and Markov's inequality gives the following,

$$\Pr[|S_\alpha \setminus L'| \le \gamma k] \ge 1 - 2q/3.$$

Therefore, conditioning on the above event,

$$\begin{aligned}
\text{Err}(\widehat{x}' - \widehat{x}'_{L'}, \gamma k, d) &\le \|\widehat{x}' - \widehat{x}'_{(L' \cup S_\alpha)}\|_2^2 \\
&\le \text{Err}(\widehat{x}' - \widehat{x}'_{(L' \cup S_\alpha)}, k, n) + k \cdot \|\widehat{x}' - \widehat{x}'_{(L' \cup S_\alpha)}\|_\infty^2 \\
&\le \text{Err}(\widehat{x}' - \widehat{x}'_{L'}, k, n) + k \cdot \|\widehat{x}' - \widehat{x}'_{S_\alpha}\|_\infty^2 \\
&\le \text{Err}(\widehat{x}' - \widehat{x}'_{L'}, k, d) + k \cdot \alpha\rho,
\end{aligned}$$

where the first inequality follows because $\forall f \in S_\alpha \setminus L'$, $|f| \le d$. Note that $\text{supp}(\widetilde{\chi}) \subseteq L$, hence,

$$\begin{aligned}
\text{Err}(\widehat{x} - \widehat{\chi}', 2\gamma k, d) &= \text{Err}(\widehat{x}' - \widetilde{\chi}, 2\gamma k, d) \\
&\le \text{Err}(\widehat{x}'_{L'} - \widetilde{\chi}, \gamma k, d) + \text{Err}(\widehat{x}' - \widehat{x}'_{L'}, \gamma k, d) \\
&\le \text{Err}(\widehat{x}'_{L'} - \widetilde{\chi}, \gamma k, d) + \text{Err}(\widehat{x}' - \widehat{x}'_{L'}, k, d) + \alpha k \rho
\end{aligned}$$

By second part of Lemma 22, $\mathrm{Err}(\widehat{x}'_{L'} - \widetilde{\chi}, \gamma k, d) \leq \mathrm{Err}(\widehat{x}'_{L'}, k, d) + 24\beta k\rho$ with probability $1-p$, therefore,

$$\begin{aligned}
\mathrm{Err}(\widehat{x} - \widehat{\chi}', 2\gamma k, d) &\leq \mathrm{Err}(\widehat{x}'_{L'}, k, d) + 24\beta k\rho + \mathrm{Err}(\widehat{x}' - \widehat{x}'_{L'}, k, d) + \alpha k\rho \\
&\leq \mathrm{Err}(\widehat{x}'_{L'}, k, d) + \mathrm{Err}(\widehat{x}' - \widehat{x}'_{L'}, k, d) + 34\beta k\rho \\
&\leq \mathrm{Err}(\widehat{x}', k, d) + 34\beta k\rho \\
&\leq (1+4\delta)\mathrm{Err}(\widehat{x}', k, d).
\end{aligned}$$

By a union bound over the randomness of Locate and Estimate primitives the above holds with probability at least $1-q$.

Runtime and Sample complexity: The runtime and sample complexity follow from invoking Lemma 19 and Lemma 22 with $B = O(\frac{k}{q\delta^2\gamma})$ and $p = O(q\gamma\delta)$.

□

4.5 Iterative Peeling

Here we present an iterative algorithm which reduces the SNR of the input signal as well as its sparsity by a constant factor in each iteration and hence terminates in $O(\log_2 k)$ rounds.

Algorithm 8 RobustSHT

```
input: signal $x \in \mathbb{R}^{2^n}$, parameter $\delta > 0$, failure probability $q$, sparsity $k$, integer $d$.
output: estimate $\widehat{\chi} \in \mathbb{R}^{2^n}$.
 1: $\gamma \leftarrow 1/64$.
 2: $T \leftarrow \lceil \log_{1/\gamma} k \rceil$.
 3: $q^{(1)} \leftarrow q/2$.
 4: $\delta^{(1)} \leftarrow \delta/20$.
 5: $k^{(1)} \leftarrow k$.
 6: $w^{(0)} \leftarrow \{0\}^{2^n}$.
 7: for $r = 1$ to $T$ do
 8:     $w^{(r)} \leftarrow$ ReduceSNR$(x, w^{(r-1)}, \delta^{(r)}, q^{(r)}, k^{(r)}, d, \gamma/2)$.
 9:     $\delta^{(r+1)} \leftarrow \delta^{(r)}/2$.
10:     $q^{(r+1)} \leftarrow q^{(r)}/2$.
11:     $k^{(r+1)} \leftarrow \gamma \cdot k^{(r)}$.
12: $\widehat{\chi} \leftarrow w^{(T)}$.
13: return $\widehat{\chi}$.
```

Proof of Theorem 3: The proof is by induction. The induction hypothesis is that for every iteration r the followings hold,

$$\Pr\left[\mathcal{E}_r | \mathcal{E}_{r-1}\right] \geq 1 - q^{(r)},$$

where event $\mathcal{E}_r$ for every r is defined as the following,

$$\mathcal{E}_r = \begin{cases} 1)\ |\mathrm{supp}(w^{(r)})| \leq \sum_{t=0}^{r} 2k^{(t)} \\ 2)\ \mathrm{Err}(\widehat{x} - w^{(r)}, \gamma k^{(r)}, d) \leq \prod_{t=0}^{r}\left(1 + 3\delta^{(t)}\right) \cdot \mathrm{Err}(\widehat{x}, k, d) \end{cases}.$$

It follows from Lemma 23 that the inductive hypothesis holds for every r. Therefore by a union bound we have the following,

$$\begin{aligned}\Pr\left[\bar{\mathcal{E}}_T\right] &\leq \sum_{r=1}^{T} \Pr\left[\bar{\mathcal{E}}_r|\mathcal{E}_{r-1}\right] + \Pr\left[\bar{\mathcal{E}}_0\right]\\ &\leq \sum_{r=1}^{T} q^{(r)}\\ &\leq q.\end{aligned}$$

Conditioning on $\mathcal{E}_T$ happening which occurs with probability $1-q$ the following holds,

$$|\mathrm{supp}(\widehat{\chi})| \leq \sum_{r=1}^{T} 2k^{(r)} \leq 3k.$$

And also,

$$\begin{aligned}\mathrm{Err}(\widehat{x}-\widehat{\chi},0) &= \|\widehat{x}-\widehat{\chi}\|_2^2\\ &\leq \prod_{r=1}^{T}\left(1+4\delta^{(r)}\right)\cdot \mathrm{Err}(\widehat{x},k)\\ &\leq (1+\delta)\cdot \mathrm{Err}(\widehat{x},k).\end{aligned}$$

This concludes the first and third claims of the theorem. Second claim of the theorem follows from line 4 of Algorithm 7.

Runtime and Sample complexity: At iteration r of the algorithm the values of the input parameters to ReduceSNR primitive are as follows, $\delta^{(r)} = O(\delta/2^r)$, $q^{(r)} = O(q/2^r)$, $k^{(r)} = O(k/64^r)$ and $\gamma = O(1)$ and also it follows from the inductive proof that $\|w^{(t)}\|_0 = O(k)$. Hence by Lemma 23, the runtime of this iteration is the following,

$$O\left(\log 4^r\left(\frac{k}{8^r\delta^2}(n\log k + (d\log k + n)\log n\log(d\log n)\log d) + nk(\log k + \log n\log(d\log n)\log d)\right)\right).$$

Therefore the total runtime is $O\left(nk\log^3 k + nk\log^2 k\log n\log(d\log n)\log d\right)$. Also the sample complexity of the r^{th} iteration is $O\left(\frac{kd}{8^r\delta^2}\log n\log(d\log n)\log 4^r\right)$ which leads to a total sample complexity of $O\left(\frac{kd}{\delta^2}\log n\log(d\log n)\right)$. □

5 Experiments

We test our ExactSHT algorithm for graph sketching on a real world data set. We utilize the autonomous systems dataset from the SNAP data collection.[1] In order to compare our methods with [SK12] we reproduce their experimental setup. The dataset consists of 9 snapshots of an autonomous system in Oregon on 9 different dates. The goal is detect which edges are added and removed when comparing the system on two different dates. As a pre-processing step, we find the common vertices that exist on all dates and look at the induced subgraphs on these vertices. We take the symmetric differences (over the edges) of dates 7 and 9. Results for other date combinations can be found in the supplementary material. This results in a sparse graph (sparse in the number of edges). Recall that the running time of our algorithm is $O(kn\log^2 k\log n\log d)$ which reduces to $O(nk\log^2 k\log n)$ for the case of cut functions where $d=2$.

[1]snap.stanford.edu/data/

Table 1: Sampling and computational complexity

No. of vertices	CS method		Our method	
	Runtime	Samples	Runtime	Samples
70	1.14	767	0.85	6428
90	1.88	812	0.92	6490
110	3.00	850	0.82	6491
130	4.31	880	1.01	7549
150	5.34	905	1.16	7942
170	6.13	927	1.22	7942
190	7.36	947	1.18	7271
210	8.24	965	1.28	7271
230	$*$	$*$	1.38	7942
250	$*$	$*$	1.38	7271
300	$*$	$*$	1.66	8051
400	$*$	$*$	2.06	8794
500	$*$	$*$	2.42	8794
600	$*$	$*$	3.10	9646
700	$*$	$*$	3.35	9646
800	$*$	$*$	3.60	9646

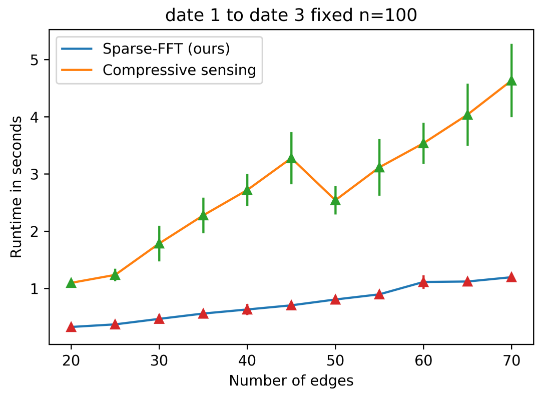
5.1 Sample and time complexities as number of vertices varies

In the first experiment depicted in Figures 1b-4b we order the vertices of the graph by their degree and look at the induced subgraph on the n largest vertices in terms of degree where n varies. For each n we pick $e = 50$ edges uniformly at random. The goal is to learn the underlying graph by observing the values of cuts. We choose parameters of our algorithm such that the probability of success is at least 0.9. The parameters tuned in our algorithm to reach this error probability are the initial number of buckets the frequencies are hashed to and the ratio at which they reduce in each iteration. We plot running times as n varies. We compare our algorithm with that of [SK12] which utilizes a CS approach. We fine-tune their algorithm by the only tuneable parameter which is sampling complexity. Both algorithms are run in a way such that each sample (each observation of a cut value) takes the same time. As one can see our algorithm scales *linear* in n whereas the CS approach scales *quadratically.* Our algorithm continues to work in a reasonable amount of time for vertex sizes as much as 900 in under 2 seconds. The error bars show the standard deviation of the running times.

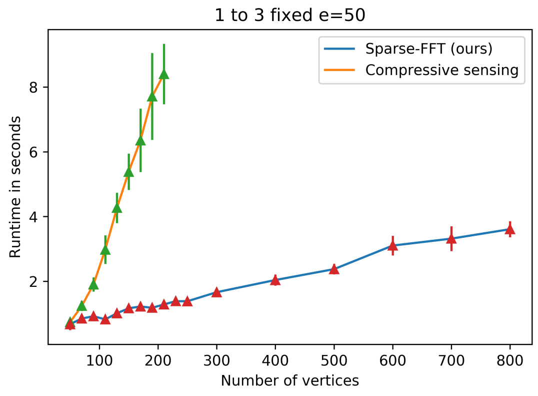
In Table 1 we include both sampling complexities (number of observed cuts) and running times as n varies. Our sampling complexity is equal to $O(k \log n)$. In practice they perform around a constant factor of 10 worse than the compressive sensing method, which are not provably optimal (see Section 1) but perform well in practice. In terms of computational cost, however, the CS approach quickly becomes intractable, taking large amounts of time on instance sizes around 200 and larger [SK12]. Asterisks in Table 1 refer to experiments that have taken too long to be feasible to run.

5.2 Time complexities as number of edges varies

Here we fix the number of vertices to $n = 100$ and consider the induced subgraph on these vertices. We randomly pick e edges to include in the graph. We plot computational complexities. Our running time provably scales linearly in the number of edges as can be seen in Figures 1a-4a.

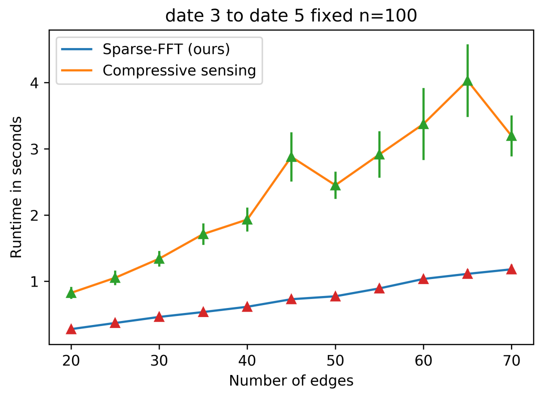


(a) Avg. time vs. no. edges

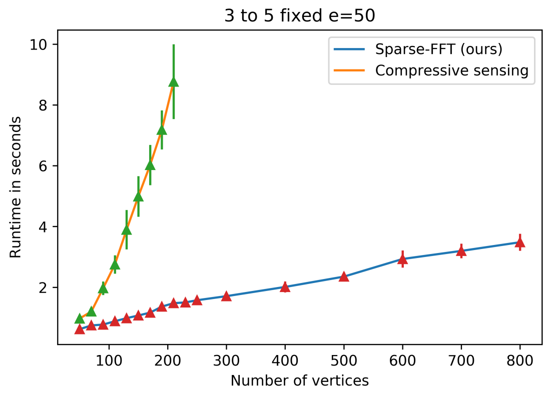


(b) Avg. time vs. no. vertices

Figure 1: Comparison of scaling of runtimes of our algorithm vs CS approaches

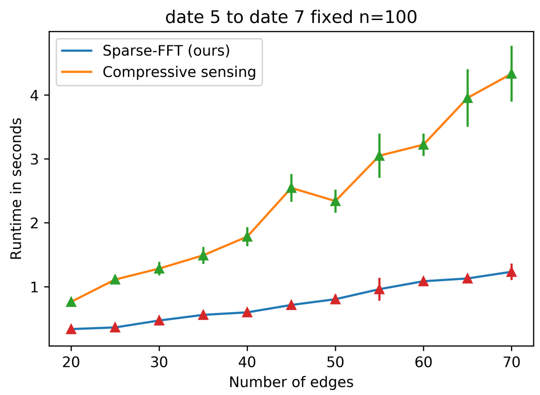


(a) Avg. time vs. no. edges

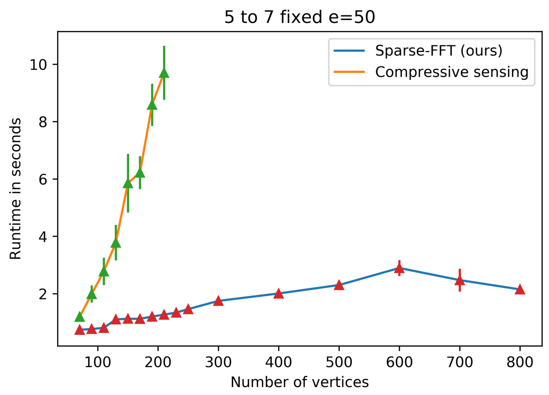


(b) Avg. time vs. no. vertices

Figure 2: Comparison of scaling of runtimes of our algorithm vs CS approaches

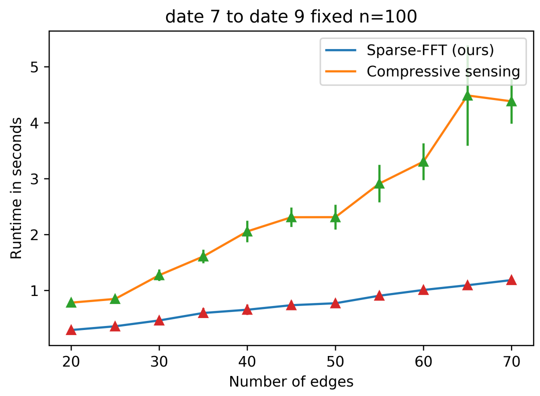


(a) Avg. time vs. no. edges

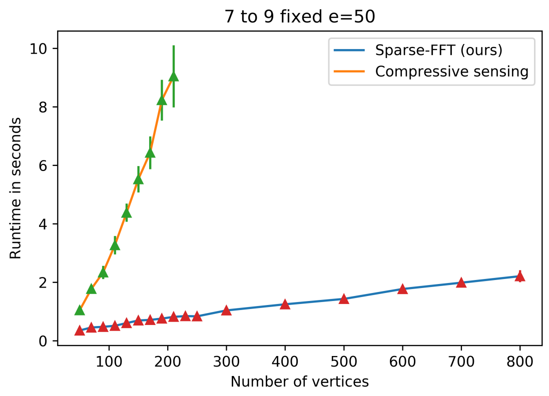


(b) Avg. time vs. no. vertices

Figure 3: Comparison of scaling of runtimes of our algorithm vs CS approaches



(a) Avg. time vs. no. edges



(b) Avg. time vs. no. vertices

Figure 4: Comparison of scaling of runtimes of our algorithm vs CS approaches

References

[DV13] Abhik Kumar Das and Sriram Vishwanath. On finite alphabet compressive sensing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5890–5894. IEEE, 2013.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

[HIKP12] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 563–578. ACM, 2012.

[HKY18] Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: a spectral approach. In *International Conference on Learning Representations*, 2018.

[HR17] Ishay Haviv and Oded Regev. The restricted isometry property of subsampled fourier matrices. In *Geometric Aspects of Functional Analysis*, pages 163–179. Springer, 2017.

[IKP14] Piotr Indyk, Michael Kapralov, and Eric Price. (nearly) sample-optimal sparse fourier transform. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 480–499. Society for Industrial and Applied Mathematics, 2014.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[Man94] Yishay Mansour. Learning boolean functions via the fourier transform. In *Theoretical advances in neural computation and learning*, pages 391–424. Springer, 1994.

[PW11] Eric Price and David P. Woodruff. (1 + eps)-approximate sparse recovery. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 295–304, Washington, DC, USA, 2011. IEEE Computer Society.

[RV08] Mark Rudelson and Roman Vershynin. On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 61(8):1025–1045, 2008.

[SHV15] Robin Scheibler, Saeid Haghighatshoar, and Martin Vetterli. A fast hadamard transform for signals with sublinear sparsity in the transform domain. *IEEE Transactions on Information Theory*, 61(4):2115–2132, 2015.

[SK12] Peter Stobbe and Andreas Krause. Learning fourier sparse set functions. In *Artificial Intelligence and Statistics*, pages 1125–1133, 2012.

[Ver10] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv e-prints*, page arXiv:1011.3027, Nov 2010.