Learning Non-parametric Choice Models with Discrete Fourier Analysis

HAOYU SONG, Purdue University, USA, song522@purdue.edu
HAI NGUYEN, Purdue University, USA, nguye245@purdue.edu
THANH NGUYEN, Purdue University, USA, nguye161@purdue.edu

Choice modeling provides important information for many economic and operational decisions. Recently, non-parametric choice models, such as decision forests and random rankings, receive a great deal of attention due to their universality. However, a major drawback is that the parameter space is factorial, and as a result practitioners often rely on heuristics to estimate non-parametric choice models. These heuristics lack theoretical guarantee on sample/computational efficiency and can also be far from optimal in practice. To address this challenge, we propose a solution using discrete Fourier analysis. We demonstrate that any choice function can be approximated with a small number of parameters in the Fourier domain. Using this concise representation, we design a provably sample-efficient, active-learning algorithm to estimate those choice models. The algorithm does not require an explicit description of the targeted choice model and needs at most \( \text{poly}(\log n, \frac{1}{\varepsilon}) \) queries of data to estimate any choice function up to \( \varepsilon \) accuracy. We also complement the theoretical results with computational studies. The experimental results show that on average, Fourier methods obtain a reduction of RMSE (root mean squared error) by more than 20 percent and MAPE (mean absolute percentage error) by more than 10 percent when compared against MNL and two common heuristics for non-parametric choice estimation.
1 INTRODUCTION

Choice modeling provides key information for the three basic operational and economic decisions: inventory, assortment, and prices. However, estimating choice functions is often challenging when consumers’ preferences are substitutes and/or complements across multiple products. The literature on modeling and learning customer preferences is vast and ever-expanding. Besides the classical choice model of multinomial logit [McFadden et al., 1973], Markov chain model [Blanchet et al., 2016], and other rational choice models (see [Ben-Akiva and Bierlaire, 1999] for a survey), in recent years, there has been expanding literature on sophisticated and even irrational choice-based models (e.g., [Chen and Mišić, 2022]) thanks to the rise of business analytics and data-driven approaches.

Among existing choice models, those non-parametric models receive great attention recently. The most prominent ones are the random rankings model [Farias et al., 2013] and the decision forest model [Chen and Mišić, 2022]. Their greatest appeal is generality. It has been known that all rational choice functions (RUM), which till far form the most important category of choice models, can be expressed as a distribution over rankings [Luce, 1959]. More impressively, [Chen and Mišić, 2022] shows that any choice function, whether rational or not, can be written as a decision forest.

Unfortunately, the generality of non-parametric models also comes with a price. Since the parameter space is factorial, practitioners often resort to heuristics such as random sampling and heuristic choice generation when estimating non-parametric choice models. These heuristics lack theoretical guarantee and can also be far from optimal.

Our paper shows that discrete Fourier analysis gives succinct representation for non-parametric choice models and helps speed up learning algorithms. Specifically, we show that any choice function can be expressed with a small number of parameters in the discrete Fourier domain, a property denoted as Fourier sparsity. Here an important technical ingredient is the use of $L_1$ Fourier norm, since it has been known that a small $L_1$ norm automatically implies Fourier sparsity. We show that regardless of how many trees are involved, a decision forest’s $L_1$ Fourier norm is bounded by the maximum number of leaves in a tree. Then we use the characterization of choice functions as binary choice forests to show that any choice function is $\epsilon$-close to a decision forest with leaf-size $\log(\frac{1}{\epsilon})$ and therefore has a concise Fourier representation.

Our second contribution is to design an efficient active-learning algorithm estimating parameters in the Fourier domain tailored for choice functions. There are two significant features of our algorithm. First, it does not depend on the explicit description of the choice model (say the number of trees), but rather on natural Fourier properties of the underlying choice function, which can be proved to be small. Second, the sample complexity of the learning algorithm scales with $\log(n)$ rather than $n$, where $n$ is the number of products. A straightforward solution is to extend the well-known Goldreich-Levin algorithm [O’Donnell, 2014] to the high-dimensional choice setting, which has sample complexity $\text{poly}(n, s, \frac{1}{\epsilon})$, where $s$ is the number of parameters in the Fourier domain. However, we consider this approach non-satisfactory because of the scaling with $n$, which can be large in many managerial settings. To overcome this issue, we observe that the choice functions we consider have Fourier degree $d$ much smaller than $n$, another nice Fourier property we will introduce later. Then combining Goldreich-Levin with hashing and the linear measurement technique in [Amrollahi et al., 2019], we obtain an algorithm with sample complexity $\text{poly}(d, \log n, s, \frac{1}{\epsilon})$. To the best of our knowledge, no previous work has considered merging Goldreich-Levin with low Fourier-degree before.

The remaining of the paper is organized as follows. We next discuss related work. Section 2 provides some basics of discrete Fourier analysis and how it is applied to choice functions and our learning model. Section 3 shows any choice function can be approximated with a small number of parameters in the Fourier domain. Section 4 gives the algorithm learning the parameters of those
functions in the Fourier domain. Section 5 gives numerical results showing that Fourier-based methods have an edge over MNL and common heuristics used to learn non-parametric models. And missing proofs can be found in the appendix.

1.1 Related works

Our paper contributes to two separate strands of literature: non-parametric choice estimation and Fourier-based learning. In the choice modeling literature, the works most closely related to ours are [Farias et al., 2013], which gives the random rankings model, and [Chen and Mišić, 2022], which gives the decision forest model. Given the difficulty of working with factorial number of parameters, the authors focuses on smaller set of models, such as trees/rankings with small depth. However, the number of trees/rankings there (O(n^d)) is still too big even for moderately size n. Therefore, they propose a number of heuristics such as heuristic column generation and random sampling, which are easy to implement. However, these heuristics lack theoretical guarantees and can be far from optimal in practice and this is why we turn to Fourier.

The second contribution is a technical one regarding our methodology to Fourier-based learning. There is a vast literature on employing discrete Fourier to learn one-dimensional boolean functions with range \{0, 1\} or \(\mathbb{R}\). For example, [Haviv and Regev, 2017, Negahban and Shah, 2012, Stobbe and Krause, 2012] studied learning Boolean functions using random samples, [Blum, 1994, Blum and Langley, 1997, Kolountzakis et al., 2005, Mossel et al., 2003] considered learning junta Boolean functions\(^1\), [Hassanieh et al., 2012, Indyk et al., 2014] studied sparse fast Fourier transform. There have been a lot of other works on discrete Fourier-based learning for the various important classes of Boolean functions over the uniform distribution [Bshouty et al., 1999, Jackson, 1997, Jackson et al., 2002, Kucera et al., 1994, Linial et al., 1993].

But none of them consider the high-dimensional choice setting. We fill in this gap by giving an efficient, tailored algorithm, which combines the idea of the Goldreich-Levin[Goldreich and Levin, 1989, Kushilevitz and Mansour, 1993] with hashing of sparse and low-degree functions. This technique is related to [Amrollahi et al., 2019], which uses both approximate sparsity and low-degree, but their method does not deal with multi-dimensional settings nor work under the sample oracle mode which will be discussed in section 2.3. [Amrollahi et al., 2019] requires a close estimate of \(f(x)\) at each query point \(x\). Unfortunately, it often happens that practitioners can only obtain a small number of samples drawn from \(f(x)\), which can be insufficient to derive a close estimate. By contrast, our algorithm work even if we only get one sample drawn from \(f(x)\) as a consequence of law of total expectation. More discussion can be found in section 4 and appendix.

2 PRELIMINARIES

2.1 Choice functions

In this section, we define some important classes of choice functions which will be discussed in more detail in later sections.

Let \(N\) be the set of products, and let \(n = |N|\), a choice function \(f\) is a mapping from a subset of the products (an assortment) to a distribution over the items being purchased (including the option of not purchasing any item). Then each choice set can be expressed by a vector \(x \in \{0, 1\}^n\) where \(x_i = 1\) means that the \(i\)-th item is offered in the choice set. And for each \(x\), \(f_i(x)\) gives the probability that the \(i\)-th item is selected. Thus a choice function can be formally defined as follows.

**Definition 1.** \(f : \{0, 1\}^n \rightarrow \mathbb{R}^n\) is a choice function if the following conditions are satisfied: (1) \(f_i(x) = 0\) whenever \(x_i = 0\) (2) \(f_i(x) \geq 0\) (3) \(\sum_i f_i(x) \leq 1\).

\(^1\)A Boolean function \(f : \{0, 1\}^n \rightarrow \{0, 1\}\) is a junta if it depends on a constant number of variables.
Random utility maximization model (RUM) is the most common model of choice functions. The model is often just called the "rational model," since it matches the common sense that a rational agent will choose the option maximizing its utility. Indeed, RUM till far is arguably the most important choice model since rationality is a common assumption in many settings.

**Definition 2.** In the random utility model, each item $i$ is associated with a fixed utility $u_i$ and a randomized error term $\varepsilon_i$. An item $i$ is selected if its randomized utility $u_i + \varepsilon_i$ is greater than all other items in the set. In other words,

$$
    f_i(x) = \begin{cases} 
    \Pr[\forall x_j > 0: u_i + \varepsilon_i \geq u_j + \varepsilon_j] & \text{if } x_i > 0 \\
    0 & \text{otherwise} 
    \end{cases}
$$

This formulation of RUM often fails to yield a closed-form formula. Therefore, sometimes it is easier to work with an equivalent characterization—in random ranking. The equivalence is first proved in [Block and Marshak, 1959].

**Definition 3.** For each permutation $\sigma$, there is a corresponding ranking function $R_\sigma$ where

$$
    [R_\sigma(x)]_i = \begin{cases} 
    1 & \text{if } \forall x_j > 0: \sigma(i) < \sigma(j) \\
    0 & \text{otherwise} 
    \end{cases}
$$

And a random ranking model $f$ is associated with a distribution over all possible rankings i.e.

$$
    f(x) = \sum_{\sigma \in S_n} c_\sigma R_\sigma(x),
$$

where $S_n$ denotes the set of all rankings and $\sum_{\sigma \in S_n} c_\sigma = 1$.

Recently [Chen and Mišić, 2022] generalized the RUM to the decision forest model to capture irrational choice behavior. To describe their model, we start with a single decision tree. Given a decision tree $T$, an agent will make their decision as follows. The agent starts from the root node. At each node $i$, the agent checks whether the $i$-th product is provided. If it is, the agent will move to the left child node and otherwise to the right child node. This process will continue unless a leaf node $\ell$ is reached. Then the agent will select the $\ell$-th item.

Figure 1 visualizes an example of a decision tree. If the choice set is $S = \{1, 2, 3\}$, then the customer will choose product 2; if $S = \{1, 4\}$, she will choose no purchase option 0. To formulate the choice function corresponding to a decision tree as a function $f$ mapping from $\{0, 1\}^n$ to $\mathbb{R}^n$, we give an alternative representation of the decision tree as follows (refer to Figure 2 for an example). Each non-leaf node is labeled by a variable $x_i$. Each leaf node is labeled by a vector $e_i \in \mathbb{R}^n$, where $e_0$ denotes a vector of zeros, and $e^{(i)}$ denotes the standard basis vector in $\mathbb{R}^n$. Every edge going from a parent node to a left child node is labeled by 1. Other edges are labeled by 0. So given a
choice set expressed as $x \in \{0, 1\}^n$, the customer’s decision can be determined from the decision tree. For instance, if $x = 1110$ (corresponding to the case $S = \{1, 2, 3\}$), $f(x) = e_2$ means that the customer will choose product 2 with probability 1.

![Figure 2. Our formulation of the decision forest](image)

There are two common parameters measuring the size of a decision tree. One is leaf-size, which is the number of leaves. Another is depth, which is the maximum number of nodes a customer will encounter in a path from the root to a leaf. A decision forest is a choice model that can be expressed as a distribution of decision tree models.

**Definition 4.** A choice function $f : \{0, 1\}^n \rightarrow \mathbb{R}^n$ is a decision forest if

$$f = \sum_{T \in T_n} c_T T,$$

where $T_n$ denotes the set of all trees with depths smaller than $n$ and $\sum_{T \in T_n} c_T = 1$.

And we say that a forest is with leaf-size $s$ if every tree with non-zero probability is of leaf-size smaller than $s$. The rationale is the same when we say a forest is of depth $d$.

### 2.2 Discrete Fourier Analysis

This section presents the basics of discrete Fourier analysis which we shall use as the technical tools for our learning algorithms. We start with a simple case of real-valued functions.

**Fourier basics of real-valued functions.** Let $f, g : \{0, 1\}^n \rightarrow \mathbb{R}$ be two real-valued functions. The inner product of two functions $f$ and $g$ is defined as

$$\langle f, g \rangle = \frac{1}{2^n} \sum_{x \in \{0, 1\}^n} f(x) \cdot g(x) = \sum_{x \in \{0, 1\}^n} [f(x) \cdot g(x)].$$

For each $S \subseteq [n]$, the characteristic function $\chi_S(x) = (-1)^{\sum_{i \in S} x_i}$ is a linear function that computes the parity of the bits $(x_i)_{i \in S}$ of $x$. The set of all $\chi_S$ forms an orthonormal basis for the space of real-valued functions on $\{0, 1\}^n$. For any $S \subseteq [n]$, the Fourier coefficient of $f$ at $S$ is defined as $\hat{f}(S) = \langle f, \chi_S \rangle$. Any function $f$ can be uniquely expressed as $f = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S$ which is called Fourier expansion of $f$. The Fourier weight of $f$ on a set $S \subseteq [n]$ is defined to be $\hat{f}(S)^2$. Parseval’s identity says that $\sum_{x \in \{0, 1\}^n} f(x)^2 = \sum_{S \subseteq [n]} \hat{f}(S)^2$. We abuse notation and shall use interchangeably $S$ as a subset of $[n]$, or as a vector in $\{0, 1\}^n$. We denote $|S|$ as the size of the set $S$ (also the same as the weight of the vector $S$). We refer the readers to [O’Donnell, 2014] for more details on Fourier analysis.

**Fourier basics of vector-valued functions.** Fourier analysis is naturally extended to functions mapping from $\{0, 1\}^n$ to $\mathbb{R}^d$. The inner product of any two functions $f, g : \{0, 1\}^n \rightarrow \mathbb{R}^d$ is defined as:

$$\langle f, g \rangle := \sum_{x \in \{0, 1\}^n} [f(x) \cdot g(x)],$$
where \( f(x) \cdot g(x) \) denotes the dot product of the two vectors \( f(x), g(x) \in \mathbb{R}^n \). For any vector \( x \in \mathbb{R}^n \), the \( p \)-norm of \( x \) is defined as \( \|x\|_p := (x_1^p + x_2^p + \ldots + x_n^p)^{1/p} \). For any function \( f: \{0,1\} \rightarrow \mathbb{R}^n \), the \( L_p \) norm of \( f \) is defined as
\[
\|f\|_p := \left( \mathbb{E}_{x \in \{0,1\}^n} \|f(x)\|_p \right)^{1/p} = \left( \mathbb{E}_{x \in \{0,1\}^n} \sum_{i=1}^d |f(x)_i|^p \right)^{1/p}.
\]

Also, given any two functions \( f, g \), their \( L_p \) distance is defined as
\[
L_p(f,g) = \|f - g\|_p.
\]

And when \( f \) is the target function, we will refer this as the \( L_p \) error of \( g \).

As same as in the one-dimensional case, the characteristic function \( \chi_S: \{0,1\}^n \rightarrow \mathbb{R} \) is defined as \( \chi_S(x) = (-1)^{\sum_{i=1}^n x_i} \). The Fourier coefficient at \( S \subseteq [n] \) is \( \hat{f}(S) = \mathbb{E}_{x \in \{0,1\}^n} f(x) \chi_S(x) \). It follows immediately from the uniqueness of Fourier expansion in one-dimensional output space that the Fourier expansion in multi-dimensional output space is also unique. That is, for any \( S \subseteq [n] \), any function \( f: \{0,1\}^n \rightarrow \mathbb{R}^n \) can be uniquely represented as
\[
f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x),
\]
where \( \hat{f}(S) \) is a vector in \( \mathbb{R}^n \) for every \( S \subseteq [n] \), and \( \chi_S(x) = (-1)^{\sum_{i \in S} x_i} \). The Plancherel’s identity says that \( \langle f, g \rangle = \sum_{S \subseteq [n]} (\hat{f}(S) \cdot \hat{g}(S)) \). In particular, when \( f = g \), we have

**FACT 1 (Paserval’s identity).** \( \|f\|_2^2 = \sum_{S \subseteq [n]} \|\hat{f}(S)\|_2^2 \).

Intuitively, the identity says that the \( L_2 \) norm of a function is equal to the sum of all Fourier weights of the function. Also, for any function \( f: \{0,1\}^n \rightarrow \mathbb{R}^n \), we define the \( L_1 \) Fourier norm of \( f \) as \( \|f\|_1 = \sum_{S \subseteq [n]} \|\hat{f}(S)\|_1 \).

The following proposition states some properties of the \( L_1 \) Fourier norm which will be useful for proving Fourier properties of decision trees. It follows directly from the definition of \( L_1 \) Fourier norm and the triangle inequality.

**PROPOSITION 1.** Let \( f, g: \{0,1\}^n \rightarrow \mathbb{R}^n \). Then, the following hold.

1. \( \|f + g\|_1 \leq \|f\|_1 + \|g\|_1 \).
2. \( \|cf\|_1 = c \|f\|_1 \) for any real value \( c \geq 0 \).

**DEFINITION 5.** We define the degree of a function \( f: \{0,1\}^n \rightarrow \mathbb{R}^n \) to be \( \max\{|S| : \hat{f}(S) \neq 0\} \).

**DEFINITION 6.** A function \( f: \{0,1\}^n \rightarrow \mathbb{R}^n \) is said to be \((s, \varepsilon)\)-sparse if there exists a function \( g: \{0,1\}^n \rightarrow \mathbb{R}^n \) s.t. (1) \( g \) is \( \varepsilon \)-close to \( f \) and (2) the number of non-zero Fourier coefficients is at most \( s \). If \( f \) is also of degree smaller than \( d \), we say that the function is \((s, d, \varepsilon)\)-sparse.

### 2.3 Data Access Model

Given a choice function \( f \), we are allowed to query choice sets \( x \in \{0,1\}^n \) and use the returned data to find a \( \hat{f} \) that minimizes the \( L_2 \) error, which corresponds to the uniform distribution as discussed in 2.2. The choice is standard in the Fourier literature [Amrollahi et al., 2019, O’Donnell, 2014] etc. and has also been previously employed in management science literature [Chen and Mišić, 2022] to derive theoretical bounds. We would like to comment that while our theoretical guarantee is obtained with \( L_2 \) error, our method can be combined with other heuristics to deliver good performance under other metrics and distributions, as demonstrated in section 5.
We adopt what [Chierichetti et al., 2018] defined as a Sample Oracle. Each time we query a \( x \in \{0,1\}^n \), the oracle will return an element \( t \) randomly drawn according to \( f(x) \). This mode is practically meaningful because in many cases there might not be enough transactions for a choice set \( x \in \{0,1\}^n \) to obtain a close estimate of \( f(x) \). To the best of our knowledge, the majority of work in the field of choice modelling does not provide a theoretical guarantee under this setting.

### 3 EFFICIENT FOURIER REPRESENTATION OF NON-PARAMETRIC CHOICE MODELS

In this section, we first show that any decision forest with small leaf-size can be approximated by a small number of Fourier bases, irregardless of the number of trees involved. Then we use this result and the characterization of choice functions as binary choice forests to prove that any choice function has a concise Fourier representation.

#### 3.1 Decision Forest with Bounded Leaf-size

We begin with recalling our alternative way of representing the decision tree in the decision forest model. This new formulation shall help us employ discrete Fourier analysis nicely. The formulation is inspired by the way of representing a Boolean function \( f: \{0,1\}^n \rightarrow \{0,1\} \) as a decision tree (see section 3.2 in [O’Donnell, 2014] for example); however, an important difference is that when a decision tree is considered as a choice function, the leaves are not practical meaningful because in many cases there might not be enough transactions for a choice function has a concise Fourier representation.

Consider the path \( P \) from the root to a leaf. The decision tree is considered as a choice function, the leaves are not necessarily offers. Every decision tree has a concise Fourier representation.

**Example 1.** To better see how a path can be represented in this way, let us go back to Figure 1. Consider the path \( \{x_1, \text{left}, x_2, \text{right}, x_4, \text{left}, \text{leaf} = x_1\} \). Here the leaf is \( x_1 \) and we will use the vector \( e_1 \) to denote it. This corresponds to the fact that item 1 is the only possible output choice for this path. For the agent to follow this path, he/she will first check whether item 1 is offered. If it is not, then this path will not be followed and the output of this path will be zero. This corresponds to the indicator function \( 1(x_1 = 1) \). Then the agent arrives at \( x_2 \) and checks whether item 2 is not offered because he/she is now going rightwards. This corresponds to the indicator function \( 1(x_2 = 0) \). The logic is the same for \( x_4 \). Since the agent will arrive at the leaf only if all those conditions are met, the three indicators need to be multiplied together. Thus we have the representation

\[
1(x_1 = 1, x_2 = 0, x_4 = 1) \cdot e_1.
\]

Now observe that a decision tree can be expressed in terms of path functions corresponding to various paths from the root to the leaves as:

\[
f(x) = \sum_{\text{Path } P} 1_P(x)f(P),
\]

where \( f(P) \) is the label on the leaf when the function \( f \) takes the path \( P \) in its decision tree. Every indicator function of a path in a decision tree enjoys some nice Fourier properties as follows.

**Lemma 1.** Let \( P \) be a path of a decision tree \( T: \{0,1\}^n \rightarrow \mathbb{R}^n \). Suppose there are \( d \) variables in the path \( P \). Then, it holds that (1) \( \deg(1_P) = d \) and (2) \( \|1_P\|_1 = 1 \).
Proof. Suppose \( x_1, x_2, \ldots, x_d \) are the variables in the path \( P \) and \( i_1, i_2, \ldots, i_d \) are the labels of edges in the path. Observe that each indicator function \( 1(x_1 = i_1) \) can be written in terms of the characteristic function \( \chi_{x_i} : \{0, 1\}^n \rightarrow \{-1, 1\} \) as follows.

\[
1(x_1 = i_1) = \frac{(-1)^{i_1} \chi_{x_1}(y) + 1}{2}
\]

Therefore, we have

\[
1_P(x) = 1(x_1 = i_1) \cdot 1(x_2 = i_2) \cdots 1(x_d = i_d) \cdot e_i
\]

\[
= \frac{(-1)^{i_1} \chi_{x_1}(y) + 1}{2} \cdot \frac{(-1)^{i_2} \chi_{x_2}(y) + 1}{2} \cdots \frac{(-1)^{i_d} \chi_{x_d}(y) + 1}{2} \cdot e_i.
\]

Expanding the first \( d \) terms and using the homomorphism property of the characteristic functions \( \chi_{S \cup T} = \chi_S \cdot \chi_T \) implies that there are \( 2^d \) non-zero Fourier coefficients, each is of magnitude \( 1/2^d \). Thus, \( 1_P \) has degree \( d \) and \( \|1_P\|_1 = 1 \).

Next, we show that a decision tree is \( L_1 \) norm bounded.

Lemma 2. If \( f : \{0, 1\}^n \rightarrow \mathbb{R}^n \) is a decision tree with leaf-size \( s \), then \( \|f\|_1 \leq s \).

Proof. Recall that \( f(x) = \sum_{P \in \mathcal{P}} 1_P(x) f(P) \). By definition of choice function, it holds that \( \sum_i f(P)_i \leq 1 \). By Proposition 1, we have

\[
\|f\|_1 = \left\| \sum_{P \in \mathcal{P}} 1_P \cdot f(P) \right\|_1 \leq \sum_{P \in \mathcal{P}} \left\| 1_P \cdot f(P) \right\| \leq \sum_{P \in \mathcal{P}} \|1_P\| \cdot \sum_i f(P)_i \leq \sum_{P \in \mathcal{P}} \|1_P\| \leq s,
\]

which completes the proof.

Now we demonstrate that trees with small leaf-size can be approximated by low-degree bases. The intuition is that for each long path \( 1_P \) with length \( \ell \), the probability that \( 1_P(x) \neq 0 \) is as small as \( \frac{1}{\ell^2} \). So we can safely cut down a portion of the path without significantly affecting the result.

Lemma 3. Given a decision tree \( T \) with leaf-size \( s \), there exists a decision tree \( T' \) with depth at most \( \log \left( \frac{s^2}{\varepsilon} \right) \) such that \( \|T - T'\|_2^2 \leq \varepsilon \).

Proof. Let \( P = \{x_1 = i_1, x_2 = i_2, \ldots, x_k = i_k, e_l\} \) be an arbitrary path of \( T \). If \( k \leq \log(\frac{s^2}{\varepsilon}) \), we keep \( P \) as it is. Otherwise, let \( k' = \log(\frac{s^2}{\varepsilon}) \). Then form a path \( P' \) by removing all internal nodes after the \( k' \)-th one and output the same value \( e_l \). Then, \( T(x) \neq T'(x) \) only when \( x \) takes a path of length greater than \( \log(s^2/\varepsilon) \) in \( T \). This happen with probability \( 2^{\log(s^2/\varepsilon)} = \varepsilon/s^2 \). So we have

\[
\|1_P - 1_{P'}\|_2^2 = \Pr_{x \in \{0, 1\}^n} [T(x) \neq T'(x)] \leq \varepsilon/s^2
\]

Now we can apply triangle inequality to obtain

\[
\|T - T'\|_2 \leq \sum_P \|1_P - 1_{P'}\|_2 \leq s \cdot \sqrt{\frac{2}{s^2}} = \sqrt{\varepsilon}.
\]

This implies that \( \|T - T'\|_2 \leq \varepsilon \), as desired.

As a consequence of Lemma 3, a decision forest can be approximated by a low-degree decision forest as following.

Theorem 1. For any decision forest \( f : \{0, 1\}^n \rightarrow \mathbb{R}^n \) with leaf-size \( s \), and \( \varepsilon > 0 \), there exists a decision forest \( g : \{0, 1\}^n \rightarrow \mathbb{R}^n \) such that \( (1) \|f - g\|_2^2 \leq \varepsilon \) and \( (2) \deg(g) = \log(s^2/\varepsilon) \).
This completes the proof. □

Now we are ready to establish the approximate Fourier sparsity of decision forests. The key insight is that any decision forests with small leaf-size has a small Fourier $L_1$ norm, which then implies that the function can be approximated by a sparse function.

**Lemma 4.** Let $f : \{0, 1\}^n \to \mathbb{R}^n$ satisfies $\|f\|_1 \leq L$, and $\epsilon > 0$. Then, there exists a function $g : \{0, 1\}^n \to \mathbb{R}^n$ such that (1) $\|f - g\|_2^2 \leq \epsilon$, and (2) $g$ is $L_2^2/\epsilon$-sparse.

**Proof.** The main idea is that a small $L_1$ norm allows us to throw away many insignificant Fourier coefficients. Let us define $S = \{S \subseteq \{0, 1\}^n : \|\hat{f}(S)\|_1 \geq \frac{\epsilon}{L}\}$, and $g = \sum_{S \in S} \hat{f}(S)\chi_S$.

It is clear that $|S| \leq \frac{L^2}{\epsilon}$. This implies that $g$ contains at most $L^2/\epsilon$ non-zero Fourier coefficients. Using Parseval’s identity, we have

$$\|f - g\|_2^2 = \sum_{S \in S} \|\hat{f}(S)\|_2^2 \leq \max_{S \notin S} \|\hat{f}(S)\|_2 \sum_{S \notin S} \|\hat{f}(S)\|_2$$

$$\leq \max_{S \notin S} \|\hat{f}(S)\|_1 \sum_{S \notin S} \|\hat{f}(S)\|_1 \leq \max_{S \notin S} \|\hat{f}(S)\|_1 \left(\sum_{S \in S} \|\hat{f}(S)\|_1^2 + \sum_{S \notin S} \|\hat{f}(S)\|_1^2\right)$$

$$\leq \frac{\epsilon}{\|f\|_1} \cdot \|f\|_1 \leq \frac{\epsilon}{L} \cdot L = \epsilon$$

This completes the proof. □

Then using the triangle inequality for Fourier $L_1$ norm, we get the following surprising result that regardless of the number of trees, the Fourier sparsity of any decision forests is always upper-bounded in terms of $s$ and $\epsilon$. This argument crucially depends on the fact that we are working on choice functions so that the sum of coefficients of tree is bounded by 1 and thus the $L_1$ norm will it blow up with the number of trees.

**Theorem 2.** Let $f : \{0, 1\}^n \to \mathbb{R}^n$ be a decision forest with leaf-size $s$ and $\epsilon > 0$. Then, there exists a decision forest $f' : \{0, 1\}^n \to \mathbb{R}^n$ such that (1) $\|f - f'\|_2^2 \leq 2\epsilon$, (2) $\deg(f') \leq \log(\frac{s^2}{\epsilon})$, and (3) $f'$ is $L_1^2/\epsilon$-sparse.

**Proof.** Let $f = \sum_T c_T T$. As in Theorem 1, we can cut down each $T$ to be of depth at most $\log(\frac{s^2}{\epsilon})$. Since cutting down paths will not increase leaf-size, every resulting tree $T'$ has a leaf-size smaller than $s$ and therefore its $L_1$ Fourier norm is smaller than $s$. Define $f_{\text{cut}} = \sum_T c_T T'$. By Theorem 1, $f_{\text{cut}}$ is within $\epsilon$ of $f$. Then using triangle inequality for $L_1$ Fourier norm, we get

$$\|f_{\text{cut}}\|_1 \leq \sum_T c_T \|T'\|_1 \leq \max_T \|T'\|_1 \leq s.$$

Then by Lemma 4, $f_{\text{cut}}$ can be approximated within $\epsilon$ by a function $f'$ consisting of at most $\frac{s^2}{\epsilon}$ non-zero Fourier coefficients. According to Lemma 4, Fourier bases of $f'$ are all bases of $f_{\text{cut}}$, which are of degree less than $\log(\frac{s^2}{\epsilon})$. This completes the proof. □
3.2 Choice Models as Binary Choice Forests

In this section, we utilize the characterization of choice models as binary choice forests [Chen et al., 2019] to show that any choice function can be approximated by a decision forest with small leaf-size and therefore enjoys a sparse, low-degree Fourier representation.

**Definition 7.** A binary choice tree $f$ is a decision tree of $n$ nodes $(x_i, \sigma_i, o_i)$ with $c_i \in [n]$, $\sigma_i \in \{0, 1\}$, $o_i \in [n]$. Given $x \in \{0, 1\}^n$, $f(x)$ is computed as follows: starting from $i = 0$, $f$ outputs $o_i$ if $x_i = \sigma_i$ and proceeds to the $(i + 1)$-th node otherwise. A binary choice forest is a distribution over binary choice trees.

**Remark 1.** Just like ranking, there is only one node at each level in a binary choice tree. However, there are two differences. First, in a ranking it must be the case that $c_i = o_i$. Second, in a ranking $\sigma_i = 1$ for all $i$; in other words, we proceed only when $x_i = 0$. A binary choice tree gets rid of these two restrictions and therefore can be viewed as a general form of ranking.

![Fig. 3. An example of a binary choice tree](image_url)

**Theorem 3 ([Chen et al., 2019]).** Every choice function can be expressed as a binary choice forest with at most $n \times 2^{n-1} + 1$ binary choice trees.

Since a binary choice tree is a tree with leaf-size $n$, naively applying Theorem 2 yields a sparsity bound $\frac{n^2}{\varepsilon}$ and a degree bound $\log(\frac{n^2}{\varepsilon})$. However, resorting to the special structure of ranking trees, we can obtain a degree/sparsity bound completely dependent only on $\varepsilon$.

As seen in the figure above, an important observation is that there is exactly one leaf at each level, except the lowest level. Therefore, if we can cut down the tree to a smaller depth, the leaf size will also be reduced.

**Lemma 5.** Let $R : \{0, 1\}^n \rightarrow \mathbb{R}^n$ be an arbitrary binary choice tree. Then, there exists a decision tree $R' : \{0, 1\}^n \rightarrow \mathbb{R}^n$ such that (1) $\|R - R'\|_2^2 \leq \varepsilon$, (2) the leaf size of $R'$ is smaller than $\log(1/\varepsilon)$.

**Proof.** Suppose $R$ is given by $\{(x_1, \sigma_1, o_1), (x_2, \sigma_2, o_2), ..., (x_n, \sigma_n, o_n)\}$. Then we obtain $R'$ by cutting down all nodes after $x_{\log(1/\varepsilon)}$. This implies that $\deg(R') \leq \log(1/\varepsilon)$. Since there is only one leaf at each level, the leaf size of $R'$ is also $\log(1/\varepsilon)$.

Then by the definition of binary choice tree, $x_{i+1}$ will be accessed only if $x_j \neq \sigma_j$ for all $1, 2, ..., i$. It follows

$$\|R - R'\|_2^2 \leq 1 \cdot \Pr[x_1 \neq \sigma_1, x_2 \neq \sigma_2, ..., x_{\log(1/\varepsilon)} \neq \sigma_{\log(1/\varepsilon)}]$$

$$= \left(\frac{1}{2}\right)^{\log(1/\varepsilon)} = \varepsilon,$$

which completes the proof. □
Example 2. To better illustrate the idea of this lemma, consider the following example. We use an example of ranking for convenience. Suppose an agent has a ranking $\{1 \rightarrow 2 \rightarrow 3 \ldots \rightarrow 100\}$. Suppose we want to estimate the ranking function to the precision of $\frac{1}{16}$. Now we cut down the ranking to $\{1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5\}$. For the two rankings to produce different results, it must be the case that items 1,2,3,4,5 are all not available in the choice set. And this probability is $\left(\frac{1}{2}\right)^5 = \frac{1}{32} \leq \frac{1}{16}$ and by Parseval this is also the distance between the two rankings. And the new ranking’s leaf size is only five. This matches the common sense that given a long ranking, only the first few items are really relevant.

Combining lemma 5 and theorem 2, we can conclude that every binary choice forest, and therefore every choice function, has a concise Fourier representation.

Theorem 4. For any choice function $f$, there exist a function $g$: $\{0,1\}^n \rightarrow \mathbb{R}^n$ such that (1) $\|f - g\|_2^2 \leq \varepsilon$, (2) $g$ is $\frac{\log^2(1/\varepsilon)}{\varepsilon}$-sparse, and (3) $\deg(g) \leq \log(1/\varepsilon)$.

4 FOURIER-BASED LEARNING

This section presents two Fourier-based learning algorithms. The one in Section 4.1 is a straightforward extension of the well-known Goldreich-Levin algorithm [O’Donnell, 2014] to learn a Fourier-sparse function with range $\mathbb{R}^n$. We include it here mainly because the subroutine for estimating Fourier coefficients is needed in Section 4.2. However, we want to point out that to avoid an unnecessary blow-up of factor of $n$, the analysis is much more involved in the multi-dimensional setting with the use of vectorized concentration inequalities. We put the detail of the algorithm and the analysis in the appendix. We also show in the appendix that the algorithm is robust to bounded noise (and therefore supports the sample-oracle mode), and the argument extends to the algorithm in Section 4.2.

The algorithm in Section 4.2 is our main contribution. We are the first to utilize the low-degree structure to improve the sample efficiency of Goldreich-Levin. In particular, the algorithm requires a novel combination of Goldreich-Levin, hashing and the linear measurement technique in [Amrollahiet al., 2019]. Since we have shown that all choice functions are approximately Fourier-sparse with low degree, the algorithm can then be applied to learn them efficiently.

4.1 Learning Sparse Functions

Now we give an efficient algorithm that learns any function $f$: $\{0,1\}^n \rightarrow \mathbb{R}^n$ that is close to a sparse function.

Theorem 5. Let $f$: $\{0,1\}^n \rightarrow \mathbb{R}^n$ be a $(s, \varepsilon)$-sparse function. Then $f$ can be efficiently learned with sample complexity $O\left(\frac{ns^3}{\varepsilon^2} \log(ns/\varepsilon)\right)$ and time complexity $\text{poly}(n, s, 1/\varepsilon)$ with high probability.

Before giving the algorithm, let us first observe that the weight of a $(s, \varepsilon)$-sparse function must concentrate on a small set of Fourier coefficients (Claim 1). Then the main problem is to identify the location of those Fourier coefficients.

Claim 1. Let $f$: $\{0,1\}^n \rightarrow \mathbb{R}^n$ be a $(s, \varepsilon)$-sparse function. Define $S = \{S \subseteq [n]: \|\hat{f}(S)\|_2^2 \geq \frac{s}{3}\}$. Then $|S| \leq 2s$ and $\|f - \sum_{S \in S} \hat{f}(S) \chi_S\|_2^2 \leq 2\varepsilon$.

From now on, we will call $S$ the set of significant Fourier coefficients.

At a high-level idea, the learning algorithm consists of two steps. The first step is to find the location of all large Fourier coefficients of the function (Theorem 6) and ignore all small Fourier coefficients, that is, find a list $L = \{S \subseteq [n]: \|\hat{f}(S)\|_2 \geq \tau\}$ for some appropriate threshold $\tau \geq 0$. Theorem 6.
Then, we approximately estimate every Fourier coefficient in $L$ using random samples (Lemma 6), which, in turn, gives a good approximation of the original function. In the second step, we use Azuma-like concentration bounds [Hayes, 2005] for random vectors, while Chernoff-Hoeffding bounds were used in the one-dimensional case.

**Theorem 6.** Let $\mathcal{N}(\gamma)$ denote a $\gamma$-subGaussian, mean-zero noise which only depends on the current queried point. Given query access with noise $\mathcal{N}(\gamma)$ to a function $f : \{0, 1\}^n \rightarrow \mathbb{R}^n$ as well as input $0 \leq \tau \leq 1$, there is a randomized algorithm that with high probability outputs a list $L$ of subsets of $[n]$ satisfying

1. if $\left\lVert \hat{f}(U) \right\rVert_2^2 \geq \tau$, then $U \in L$, and
2. if $U \in L$, then $\left\lVert \hat{f}(U) \right\rVert_2^2 \geq \tau/4$.

Furthermore, the running time is polynomial in $(n, 1/\tau)$, and sample complexity $O\left(\frac{(1+\gamma)n\log(n/\tau)}{\epsilon^3}\right)$.

**Remark 2.** It is not hard to see that the output of a sample-oracle model has at most $\mathcal{N}(2)$-noise. Since $y \in f(x)$ is an indicator vector, it follows from triangle inequality that $|y - f(x)|_2 \leq |y|_2 + |f(x)|_2 \leq 2$.

**Lemma 6.** Given access to random samples of a function $f : \{0, 1\}^n \rightarrow \mathbb{R}^n$ with noise $\mathcal{N}(\gamma)$, there is a randomized algorithm which takes as input $S \subseteq [n], 0 < \epsilon, \delta \leq 1$ and outputs an estimate $\hat{f}(S)$ for $\hat{f}(S)$ such that

$$\left\lVert \hat{f}(S) - \hat{f}(S) \right\rVert_2^2 \leq \epsilon$$

except with probability at most $\delta$. Furthermore, the number of samples needed is $\frac{(2+\gamma)\log(4\epsilon^2/\delta)}{\epsilon^2}$.

We provide the proofs of Theorem 5, Theorem 6, and Lemma 6 in the appendix.

**Algorithm 1: Estimate**

**Input:** Random access to $f$, confident parameter $\delta$, accuracy parameter $\epsilon$, and a subset $S \subseteq [n]$  

**Output:** $\hat{f}(S)$ such that $\left\lVert \hat{f}(S) - \hat{f}(S) \right\rVert_2^2 \leq \epsilon$

$\hat{f}(S) = 0, m = \frac{\log(1/\delta)}{\epsilon}$

for $i = 1$ to $m$

Sample $x$ uniformly at random from $\{0, 1\}^n$, 

$\hat{f}(S) = \hat{f}(S) + \frac{1}{m} \cdot f(x) \cdot \chi_S(x)$

end

Return $\hat{f}(S)$

**Comparison with a naive approach.** A naive approach would learn function $f : \{0, 1\}^n \rightarrow \mathbb{R}^n$ by learning each $f_i : \{0, 1\}^n \rightarrow \mathbb{R}$ separately using Goldreich-Levin, where $f = (f_1, f_2, \ldots, f_n)$. This implies that the number of samples needed to approximately learn $f$ with the same error $\epsilon$ and confident parameter $\delta$ is scaled up by a factor of $n$. We provide more details in the appendix.

### 4.2 Learning Sparse and Low Degree Functions

In this section, we consider a smaller class of functions, that is, the set of all functions mapping from $\{0, 1\}^n$ to $\mathbb{R}^n$ that are close to a sparse and low degree function. We shall show that this class of functions can be learned more efficiently.

**Theorem 7.** Let $f : \{0, 1\}^n \rightarrow \mathbb{R}^n$ be a function that is $\epsilon$-close to a $s$-sparse degree-$d$ function. Then $f$ can be efficiently learned using $O\left((ds^2 \log(n)(sd \log n))/\epsilon^2\right)$ samples with time complexity $\text{poly}(n, s, 1/\epsilon)$. 


Compared to Theorem 5, the sample complexity is reduced by a multiplicative factor of \( n/(d \log n) \), which is significant when \( d \log n \ll n \). Similar to the learning algorithm in the previous section, our algorithm has 2 steps. We develop a new algorithm for the first step locating all significant Fourier coefficients, while the second step is the same as the one in the previous section.

4.2.1 Locating significant Fourier coefficients. Recently, [Amrollahi et al., 2019] presented an efficient algorithm that learns a sparse and low degree function mapping from \( \{0, 1\}^n \) to \( \mathbb{R} \). Their algorithm extensively utilizes the low-degree structure. In particular, they showed that there exists a set of measurement vectors \( \{v(i)\} \) such that any \( S \in \{0, 1\}^n \) can be recovered from the linear measurements \( \langle v(i), S \rangle \).

**Lemma 7 (Amrollahi et al., 2019).** For any integers \( n, d \), there exists a set of measurements vectors \( \{v(i)\}_{i=0}^m \) for \( m = d \log n \) such that, every \( S \in \{0, 1\}^n \) with \( |S| \leq d \) can be recovered given the linear measurements \( \langle v(i), S \rangle \) for all \( 1 \leq i \leq m \).

Inspired from this idea and the GL/KM algorithm, we develop an efficient algorithm for locating the set of all significant Fourier coefficients. Let us introduce some notation before presenting the algorithm. Let \( V = \{v(1), v(2), \ldots, v(m)\} \) be the set of vectors as defined in Lemma 7. For any matrix \( \sigma \in \{0, 1\}^{n \times k} \) and \( b \in \{0, 1\}^k \), we denote \( H_{\sigma}^b = \{x \in \{0, 1\}^n : \sigma^T x = b\} \). For each \( v(i) \in V \), we define \( \sigma_i = \sigma_{ij}/v_{ij} \). So \( H_{\sigma_i}^{b,j} = \{x \in \{0, 1\}^n : \sigma_i^T x = (b, j)\} \).

**Algorithm 2: Locate**

**Input:** Query access to \( f \), degree \( d \), sparsity parameter \( s \), weight threshold \( \tau \).  
**Output:** A list \( L = \{S \subseteq [n] : \|\mathcal{F}(S)\|_2 \geq \tau\} \)

1. Sample \( k = \log s + 6 \) vectors in \( \{0, 1\}^n \) uniformly randomly and form \( \sigma \)
2. Get measurement vectors \( V = \{v(1), v(2), \ldots, v(d \log n)\} \)
3. For each \( v(i) \in V \)
    - Estimate weight \( (H_{\sigma}^{b}) \) simultaneously with \( \frac{\log(ad \log n)}{16\tau^2} \) samples for all \( b \in \{0, 1\}^k \)
    - Estimate weight \( (H_{\sigma_i}^{b,j}) \) simultaneously with \( \frac{\log(sd \log n)}{16\tau^2} \) samples for all \( b \in \{0, 1\}^k \)
4. For each \( b \in \{0, 1\}^k \)
    - If weight \( (H_{\sigma}^{b}) \geq \frac{\tau}{2} \) then
    - For each \( v(i) \in V \)
        - If weight \( (H_{\sigma_i}^{b,j}) \geq \frac{\tau}{2} \) then
            - Set \( \langle v(i), \cdot \rangle = 0 \) and add the measurement to \( M_b \)
        - Else
            - Set \( \langle v(i), \cdot \rangle = 1 \) and add the measurement to \( M_b \)
    - End
    - End
5. Recover some \( S_b \) from measurements in \( M_b \) and add \( S_b \) to \( L \)
6. Return \( L \)

The algorithm first selects \( m = O(\log(s)) \) random vectors from \( \{0, 1\}^n \) and then concatenate them to form a matrix \( \sigma \in \{0, 1\}^{m \times n} \). Then all Fourier coefficients are divided into \( 2^m \) buckets \( B_j = \{S \in \{0, 1\}^n : \sigma \cdot S = j\} \) for each \( j \in \{0, 1\}^m \). It will later be shown that if the entries of
\(\sigma\) are generated uniformly i.i.d, with high probability bucket \(B_j\) contains only one significant Fourier coefficient \(\hat{f}(S)\) such that \(\|\hat{f}(S)\|_2^2 \geq \tau\) and \(\text{weight}(B_j \setminus S) \leq \frac{\tau}{16}\). Next, concatenate \(\sigma\) with \(u_i\) to form a new matrix \(\sigma_i = \left[ \begin{array}{c} \sigma \\ u_i \end{array} \right]\). So we have formed a new bucket system with twice the number of buckets. That is, each bucket \(B_j = \{S \in \{0, 1\}^n : \sigma \cdot S = j\}\) is split into two buckets \(B_{j,0} = \{S \in \{0, 1\}^n : \sigma \cdot S = j, v(i) \cdot S = 0\}\), \(B_{j,1} = \{f \in \{0, 1\}^n : \sigma \cdot S = j, v(i) \cdot S = 1\}\). We estimate the weight of each bucket \(B_{j,0}\). If \(\text{weight}(B_{j,0}) \geq \tau\), we can conclude that \(\langle v(i), S \rangle = 0\) and \(\langle v(i), S \rangle = 0\) otherwise. In this way, we can use \(L_2\) estimations (sum of Fourier weight in the bucket) to obtain all measurements and consequently recover \(S\).

In [Amrollahi et al., 2019], the linear measurements \(\langle v(i), S \rangle\) are estimated via a random hashing technique, and the estimations of Fourier coefficients. They hash Fourier coefficients into buckets. If the number of buckets is large enough, each bucket contains only one large Fourier coefficient with high probability. Then, they use the connection between the Fourier coefficient of the hashed function and the Fourier coefficient (of the original function) in each bucket to derive the linear measurements. On the other hand, we compute these linear measurements through random hashing and \(L_2\) estimation of Fourier weights, which are intuitively square of Fourier coefficients, of the buckets. The main advantage of using \(L_2\) estimations is that we can handle both multi-dimensional output space and the noise on the output. To the best of our knowledge, it is not evident to extend their technique to this setting. We note that GL/KM algorithm also uses \(L_2\) estimations of Fourier weights but neither the random hashing technique nor the low-degree structure.

For the rest of this section, we shall prove that the locate algorithm is correct and efficient.

**Fourier Hashing.** Let \(f : \{0, 1\}^n \to \mathbb{R}^n\) be a function. Let \(H\) be a subspace of \(\{0, 1\}^n\). Let \(H^\perp := \{x \in \{0, 1\}^n : x \cdot h = 0\ for\ every\ h \in H\}\) be the dual of \(H\). Given \(a \in H\), the coset \(a+H\) is defined by \(a+H := \{a+h : h \in H\}\). We define the weight of a coset \(a+H\) as \(\text{weight}(a+H) := \sum_{S \in a+H} \|\hat{f}(S)\|_2^2\). The following fact about the Fourier weight of a coset will be useful for our learning algorithm.

**Claim 2.** For any function \(f : \{0, 1\}^n \to \mathbb{R}^n\), it holds that

\[
\sum_{S \in a+H} \|\hat{f}(S)\|_2^2 = \mathbb{E}_{x \in \{0, 1\}^n, z \in H^\perp} \chi_a(z) \langle f(x), f(x+z) \rangle
\]

It implies that the Fourier weight of a coset can be estimated using query access to \(f\).

**Claim 3.** For \(0 < \varepsilon, \delta < 1\), the Fourier weight of the coset \(a+H\) can be estimated within an error of \(\varepsilon\) with probability at least \(1 - \delta\) using \(\frac{(1+\sqrt{n})^2}{\varepsilon^2}\) queries access to \(f\) with noise \(N(\gamma)\).

Hashing is an important technique in machine learning and theoretical computer science. In our context, a Hashing function \(h : \{0, 1\}^n \to \{0, 1\}^b\) is associated with a matrix \(\sigma\) where all entries are generated uniformly i.i.d.

We want to emphasize some nice properties of Hashing which will be important for our algorithm. The first one is pairwise independence.

**Fact 2.** Let \(\sigma\) be a random hashing matrix. Then for any \(S, T \in \{0, 1\}^n\),

\[
\Pr(\exists \alpha \in \{0, 1\}^n \ s.t. \ S \in \alpha + \text{null}(\sigma), T \in \alpha + \text{null}(\sigma)) = \frac{1}{2^m}.
\]

Here \(2^m\) gives the number of cosets. This essentially says that given any two vectors in \(\{0, 1\}^n\), they will be hashed to different cosets w.h.p if the number of cosets is big enough. This allows us to obtain linear measurements for each of them without being influenced by the other.
Another key observation is that when \( \sigma \) is full rank, every coset is a bucket (defined in section 3.2.1) and vice versa. And we will show that if \( \sigma \) is generated with uniformly i.i.d entries, full-rankness is satisfied with extremely high probability. Our algorithm crucially relies on this duality. The bucket structure allows us to perform splitting and thereby obtain linear measurements, while the coset structure allows us to estimate weights efficiently.

The following claims are needed for the proof of Theorem 7.

**Claim 4.** Let \( f : \{0, 1\}^n \rightarrow \mathbb{R}^n \) be a function that is \( \epsilon \)-close to a s-sparse function. Let \( S \subseteq [n] \) be such that \( \| \hat{f}(S) \|_2^2 \geq \frac{\epsilon}{s} \). Suppose a sub-space \( H \) of co-dimension 16s is drawn uniformly at random and \( S \in a + H \). Then the following events happen with probability at least \( \frac{3}{5} \):

1. there does not exist another \( T \) with \( \| \hat{f}(T) \|_2^2 \geq \frac{\epsilon}{s} \) such that \( T \in a + H \).
2. \( \sum_{U \neq S, U \in a + H} \| \hat{f}(U) \|_2^2 \leq \frac{\epsilon}{16s} \).

Intuitively, this lemma says that large Fourier coefficients do not collide into the same coset and small Fourier coefficients inside each coset will not overshadow the significant one.

We also need to show that \( \sigma_i \) is with high probability full-rank so that the weight of buckets can be obtained by sampling.

**Claim 5.** Given a fixed set of vectors \( v_1, v_2, ..., v_{d \log n} \in F_2^n \). Suppose we sample \( 2 \log(s) \) vectors uniformly randomly from \( F_2^n \) and form matrix \( \sigma \in F_2^{n \times s} \). Then it happens with negligible probability that for any \( v_i \), \( \sigma_i \) is full-rank.

**Claim 6.** Suppose that \( f \) is \( \epsilon \)-close to being s sparse. Let \( \tau \geq \frac{\epsilon}{s} \) be given. For any \( S \subseteq [n] \) such that \( \| \hat{f}(S) \|_2^2 \geq \tau \). The algorithm Locate can recover \( S \) with probability at least \( \frac{1}{2} \). The sample complexity of the procedure is \( O\left(\frac{d \log(n)}{\tau^2}\right) \) and the time complexity is \( O\left(\frac{nd \log(n)}{\tau^2}\right) \).

### 4.2.2 Learning Algorithm

We present our algorithm for learning sparse and low-degree functions as follows.

**Algorithm 3:** LearningSparseLowDegree

**Input:** Random access to \( f \), confident parameter \( \delta \), accuracy parameter \( \epsilon \)

**Output:** \( g : \{0, 1\}^n \rightarrow \mathbb{R}^n \) such that \( \| g - f \|_2 \leq \epsilon \) with constant probability

\[
L = \emptyset
\]

for \( i = 0, 1, ..., \log(s) \) do

\[
L_i = \text{Locate}(f, d, s, \epsilon/s)
\]

\[
L = L \cup L_i
\]

end

for each \( S \in L \) do

\[
g(S) = \text{Estimate}(f, \delta, \epsilon, S)
\]

end

Let \( g = \sum_{S \in L} g(S) \chi_S(x) \).

Return \( g \).

Observe that as long as we repeat the Locate procedure sufficiently number of times, we can recover all significant Fourier coefficients.

**Proof of Theorem 7.** Let \( S \in S \) be the location a given significant Fourier coefficient and use \( 1_{S,t} \) to denote the random vector indicating whether \( S \) is already successfully recovered at round \( t \). Since i.i.d random vectors are produced for hashing in each round, \( 1_{S,t} \) is independent across
different rounds. By Claim 6, \( 1_{S,t} \geq \frac{1}{2} \) for any \( t \). Then by independence, the probability that \( S \) fails to be recovered and included in \( L \) for \( T \) rounds is \( \frac{1}{2^T} \). And by union bounds, the probability that there exists one significant Fourier fails to be recovered is \( \frac{s}{2^T} \). Then setting \( T = \log(\frac{s}{\delta}) \) gives the desire confidence parameter. Then as for time/sample complexity, we plug \( \tau = \varepsilon \) into Claim 6 and scale the results by \( \log(\frac{s}{\delta}) \).

Now with the bounds given in Theorem 4, it immediately follows that any non-parametric choice model can be estimated efficiently with Fourier, irregardless of the number of trees involved.

**Theorem 8.** Let \( f \) be an arbitrary choice function. For every \( 0 < \varepsilon < 1 \), \( f \) is learnable with error at most \( \varepsilon \) with time complexity \( O(n, \frac{1}{\varepsilon}) \) and sample complexity \( \tilde{O}(\log \frac{n}{\varepsilon^2}) \).

## 5 IMPLEMENTATION AND EXPERIMENTS

### 5.1 Implementation for Ranking Model

The algorithm in 4.2 identifies a set of Fourier bases \( S \) and coefficients with provably small \( L_2 \) error, which corresponds to the uniform distribution and squared loss. In this section we aim to demonstrate that the method can also be adapted to obtain superior performance under other metrics for the ranking model. Suppose we have already obtained a set of Fourier bases \( S \) and now we are given a new loss function \( L \) and a dataset \( X_P \) generated from another distribution \( \mathcal{P} \), a straightforward extension is to solve the following optimization problem:

\[
\min_{\hat{f}_S, \hat{f}_0} \sum_{(x, y) \in X_P} L(y, \sum_{S \in S, i \in [n]} \hat{f}_{S,i}(x)) \]

s.t. \[
\sum_{S \in S, i \in [n]} |\hat{f}_{S,i}| \leq d
\]

where \( d = \max |S| \) and \( \chi_{S,i} \) denotes the coordinate vector of \( \chi_S \) corresponding to item \( i \).

The \( L_1 \) norm constraint is cumbersome to work with. To avoid this problem, we make the observation that the Fourier coefficients of a ranking model has a fixed pattern.

**Observation 1.** For any ranking model, \( \hat{f}_{S,i} \) has the same sign as \( (-1)^{|S|} \) if \( i \notin S \) and has the same sign as \( (-1)^{|S|+1} \) if \( i \in S \).

We now define \( \tilde{\chi}_{S,i}(x) = \begin{cases} (-1)^{|S|+1} \chi_{S,i}(x), & \text{if } i \in S \\ (-1)^{|S|}, & \text{otherwise} \end{cases} \). Then we can scale all coefficients to have sum 1 and rewrite the optimization problem as

\[
\min_{\hat{f}_S, \hat{f}_0} \sum_{(x, y) \in X_P} L(y, \hat{f}_0 \cdot \vec{0} + \sum_{S \in S, i \in [n]} \hat{f}_{S,i} \cdot \frac{\tilde{\chi}_{S,i}(x)}{d})
\]

s.t. \[
\sum_{S \in S, i \in [n]} \hat{f}_{S,i} = 1,
\]

\[
\hat{f}_0, \hat{f}_S \geq 0 \quad \forall S \in S
\]

where \( d = \max_{S \in S} |S| \) and \( \chi_{S,i} \) denotes the coordinate of \( \chi_S \) corresponding to item \( i \).

\[\text{The precise bound is } O(\log^3 (1/\varepsilon) \cdot \log n \cdot \log((\log^3 (1/\varepsilon) \log n)/\varepsilon^4))\]
In case that $S$ is not sufficient, we can also start from $S$ and adopt a column generation strategy similar to [Jagabathula and Venkataraman, 2022], whose objective is to find a distribution over rankings that best fits data. They make the observation that since the parameter space is a convex hull over rankings, at each point the steepest direction of descent must correspond to a single ranking. Similarly, since we require the sum of coefficients in our formulation to be 1, the steepest direction corresponds to a single Fourier base. In practice, it is sufficient to just find a descending direction at each step rather than the steepest one and halt once no further improvements can be made. We refer the readers to [Jagabathula and Venkataraman, 2022] for a more detailed discussion of column-generation strategies for non-parametric choice models.

5.2 Computational Studies

5.2.1 Experimental Set-up. We use the ranking model as the ground-truth and follow a set-up similar to [Francisca et al., 2022]. There are $n$ products and $K$ major rankings, which correspond to major customer groups, with weights added up to 0.9. And there are 100 "noisy" rankings with total weight 0.1. The "no-purchase" threshold is set to be $d = 4$. We select $K = 10, 15$ and $n = \{20, 25, 30, 35, 40, 45, 50\}$. Major rankings and "noisy rankings" are all generated uniformly and the coefficients are drawn from the Dirichlet distribution.

To demonstrate the effectiveness of the Fourier-based approach across different distributions, for each $n, K$ we create 5 product distributions, where the mean probability of each item is generated uniformly in $[0, 1]$. For each distribution $\mathcal{P}$, we generate 500,000 transactions (2,000 choice sets and 250 transactions for each) as training samples.

For each choice function $f$, we first use our algorithm to obtain a set of Fourier bases $S$ with $L_1$ weight greater than 0.003. We limit the number of queries to be 100,000. Then we apply the procedure as outlined in section 5.1 on training samples generated from $\mathcal{P}$.

5.2.2 Benchmarks. Apart from the standard MNL model, we also compare our approach against two common heuristics used when estimating ranking-based models. The first one is random sampling (RS) as proposed in [Farias et al., 2013], which first selects a uniformly random collection of rankings $R$ and then find the distribution on $R$ that best fits the data. We set $|T| = 1000$. The second one is heuristic column generation (HCG) [Chen and Mišić, 2022], which starts from a small set of rankings (often single-item rankings) and then iteratively incorporates new ones to the set.

5.2.3 Results. We use two common metrics for comparisons: root mean squared error (RMSE) and mean absolute percentage error (MAPE). In the plots below, we can see that as $n$ varies, the Fourier-based method consistently outperforms MNL and the two ranking heuristics. We notice that as $n$ increases, the performance of two ranking heuristics deteriorates in the sense that the error becomes close to, and sometimes even worse than, the vanilla MNL model. This is not surprising since when the parameter size becomes big, it is much more likely that the randomly-selected rankings for RS and the starting set for HCG is far from the true model. On average, the Fourier-based method obtains a reduction of RMSE by more than 20 percent and a reduction of MAPE by more than 10 percent, clearly demonstrating the potential advantage of adopting Fourier-based methods.
(a) RMSE Error

(b) MAPE Error

Fig. 4. Comparison of the Fourier-based method with MNL and the best of HCG/RS under $K = 10$

(a) RMSE Error

(b) MAPE Error

Fig. 5. Comparison of the Fourier-based method with MNL and the best of HCG/RS under $K = 15$

REFERENCES


APPENDIX

A OMITTED PROOFS IN SECTION 4.1

A.1 Proof of Theorem 5

First, by Theorem 6, the algorithm runs the extended Goldreich-Levin algorithm on $f$ with threshold $\tau = \frac{\epsilon}{2s}$ to obtain a list $L$ of significant Fourier coefficients. Next, approximate each $\hat{f}(S)$ for $S \in L$ by $\tilde{f}(S)$ to within error $\alpha$ (to be chosen later) by Lemma 6. Define

$$ g(x) = \sum_{S \in L} \tilde{f}(S) \chi_S(x). $$

Haoyu Song, Hai Nguyen, and Thanh Nguyen


APPENDIX

A OMITTED PROOFS IN SECTION 4.1

A.1 Proof of Theorem 5

First, by Theorem 6, the algorithm runs the extended Goldreich-Levin algorithm on $f$ with threshold $\tau = \frac{\epsilon}{2s}$ to obtain a list $L$ of significant Fourier coefficients. Next, approximate each $\hat{f}(S)$ for $S \in L$ by $\tilde{f}(S)$ to within error $\alpha$ (to be chosen later) by Lemma 6. Define

$$ g(x) = \sum_{S \in L} \tilde{f}(S) \chi_S(x). $$
We shall use the following concentration bound for the proof. 

\[ \|f - g\|_2^2 \leq \varepsilon. \]

We have

\[
\|f - g\|_2^2 = \sum_{S \subseteq [n]} \left\| \hat{f}(S) - \hat{g}(S) \right\|_2^2
\]

(Parseval’s identity)

\[
= \sum_{S \subseteq L} \left\| \hat{f}(S) - \hat{g}(S) \right\|_2^2 + \sum_{S \not\subseteq L} \left\| \hat{f}(S) \right\|_2^2
\]

\[
\leq \alpha^2 |L| + \sum_{S \not\subseteq L} \left\| \hat{f}(S) \right\|_2^2
\]

We upper bound the sum over \( S \not\subseteq L \) by

\[
\sum_{S \not\subseteq L} \left\| \hat{f}(S) \right\|_2^2 \leq \max_{S \subseteq L} \left\| \hat{f}(S) \right\|_2 \cdot \sum_{S \not\subseteq L} \left\| \hat{f}(S) \right\|_2 \leq \frac{\varepsilon}{2s} \cdot \sum_{S \not\subseteq L} \left\| \hat{f}(S) \right\|_2 \leq \frac{\varepsilon}{2s} \cdot s = \varepsilon/2.
\]

Choosing \( \alpha = \sqrt{1/(2\varepsilon|L|)} \) gives \( \|f - g\|_2^2 \leq \varepsilon. \)

Next, we compute the number of queries needed. By Claim 1, \( |L| \leq 2s \). Finding the set \( L \) requires \( O\left(\frac{n^3}{\varepsilon^2} \log(ns/\varepsilon)\right) \). For each \( S \subseteq L \), we approximate \( \hat{f}(S) \) to within \( \alpha \) with except probability \( \gamma \) (chosen later). This requires \( O\left(\frac{\log(1/\gamma)}{\alpha^2} \right) \) samples. If we choose \( \gamma = 1/10|L| \), then by union bound the probability that we estimate all Fourier coefficients in \( L \) to within accuracy \( \alpha \) with except probability \( 1 - |L| \cdot \gamma = 1/10, \) and the number of random samples needed is \( O\left(\frac{n^3}{\varepsilon^2} \log(s^2/\varepsilon)\right) \). Therefore, the sample complexity is \( O\left(\frac{n^3}{\varepsilon^2} \log(ns/\varepsilon)\right) + O\left(\frac{n^3}{\varepsilon^2} \log(s^2/\varepsilon)\right) = O\left(\frac{n^3}{\varepsilon^2} \log(ns/\varepsilon)\right) \), and the running time is polynomial in \( (n, s, 1/\varepsilon) \).

### A.2 Proof of Lemma 6

We shall use the following concentration bound for the proof.

**Definition 8.** Let \( X = (X_1, X_2, \ldots, X_m) \) be a sequence of random vector such that \( X_i : \Omega \to \mathbb{R}^d \), \( X_0 = 0 \), and for every \( j < i, \mathbb{E}[||X_i||] < \infty \) and \( \mathbb{E}[X_i | X_j] = X_j \). Then, we call \( X \) a weak martingale sequence.

**Theorem 9 ([Hayes, 2005]).** Let \( X = (X_1, X_2, \ldots, X_m) \) be a weak martingale sequence taking value in \( \mathbb{R}^d \) such that \( X_0 = 0 \) and for every \( i, ||X_i - X_{i-1}||_2^2 \leq 1 \). Then for every \( \varepsilon > 0 \),

\[
\Pr[||X_m||_2^2 \geq \varepsilon] \leq 2e^2e^{-\varepsilon/2m}.
\]

**Fact 3.** For any non-empty \( S \subseteq [n] \), it holds that \( \mathbb{E}_{x \in (0,1)^n} \chi_S(x) = 0 \).

**Proof of Lemma 6.** First we show how to estimate \( \hat{f}(S) \) using query access to \( f \) without noise.

Recall that \( \hat{f}(S) = \mathbb{E}[f(x)\chi_S(x)] \). The main idea is to use \( m \) (chosen later) independent samples of \( f \), say \( f(x_1), f(x_2), \ldots, f(x_m) \), to empirically approximate \( \hat{f}(S) \).

Let \( \hat{f}(S) = \frac{1}{m} \sum_{i=1}^{m} f(x_i)\chi_S(x_i) \). Let \( Y_i = f(x_i)\chi_S(x_i) - \hat{f}(S) \). First observe that \( \mathbb{E}[Y_i] = 0 \) for every \( i \). Let \( X_0 = 0, X_i = Y_1 + Y_2 + \ldots + Y_i \) for every \( i > 0 \). It is easy to see that, for every \( j < i, \mathbb{E}[X_j | X_i] = X_j \). Therefore, \( X = (X_0, X_1, \ldots, X_m) \) is a martingale sequence. Furthermore, we have

\[
||X_i - X_{i-1}||_2 = ||Y_i||_2 = ||f(x_i)\chi_S(x_i) - \hat{f}(S)||_2 \leq ||f(x_i)\chi_S(x_i)||_2 + ||\hat{f}(S)||_2
\]

\[
\leq ||f(x_i)\chi_S(x_i)||_2 + \mathbb{E}[f(x)\chi_S(x)]_2 \leq ||f(x_i)\chi_S(x_i)||_1 + \mathbb{E}[f(x)\chi_S(x)]_1 \leq 1 + 1 = 2,
\]
since \( \|f(x)\chi_S(x)\|_1 = \|f(x)\|_1 \leq 1 \) for every \( x \). Applying Theorem 9 for the sequence \( X/2 = (X_1/2, X-2/2, \ldots, X_m/2) \), we have, for every \( \epsilon > 0 \),
\[
\Pr[\|X_m/2\|_2^2 \geq \epsilon] \leq 2e^{2\epsilon^{-\epsilon/2m}}.
\]
It implies that
\[
\Pr\left[ \left\| \sum_{i=1}^m f(x_i)\chi_S(x_i) - m\hat{f}(S) \right\|_2^2 \geq \epsilon \right] \leq 4e^{2\epsilon^{-\epsilon/2m}}.
\]
Equivalently,
\[
\Pr\left[ \left\| \frac{1}{m} \sum_{i=1}^m f(x_i)\chi_S(x_i) - \hat{f}(S) \right\|_2^2 \geq \epsilon \right] \leq 4e^{2\epsilon^{-\epsilon/2}}.
\]
Now, let \( \hat{f}(S) = \frac{1}{m} \sum_{i=1}^m f(x_i)\chi_S(x_i) \), and choose \( m = \frac{2\log(4\epsilon^2/\delta)}{\epsilon} = O\left( \frac{\log(1/\delta)}{\epsilon} \right) \), we have
\[
\Pr\left[ \left\| \hat{f}(S) - \tilde{f}(S) \right\|_2^2 \geq \epsilon \right] \leq \delta
\]

**Adding noise to the outputs.** In this setting, on input \( x \in \{0, 1\}^n \), the output is \( f(x) + \rho \), where \( \rho \) is independently drawn from a distribution \( \mu \) such that \( \mathbb{E}[\rho] = 0 \) and \( \rho \) is bounded by \( \gamma \). Define \( Y_i = (f(x_i) + \sigma)\chi_S(x_i) - \hat{f}(S) \), and the same for \( X_i \). Then, it is still the case that \( X \) is a martingale sequence since \( \mathbb{E}[\rho_i] = 0 \).
\[
\mathbb{E}[Y_i] = \mathbb{E}\left[ (f(x_i) + \sigma)\chi_S(x_i) - \hat{f}(S) \right] = \mathbb{E}\left[ f(x_i)\chi_S(x_i) - \hat{f}(S) \right] + \mathbb{E}[\sigma_i\chi_S(x_i)] = 0 + \mathbb{E}[\sigma_i] \mathbb{E}[\chi_S(x_i)]
\]
If \( S \neq \emptyset \) then \( \mathbb{E}[\chi_S(x_i)] = 0 \). If \( S = 0 \), then \( \mathbb{E}[Y_i] = \mathbb{E}[\rho_i] = 0 \). So it always holds that \( \mathbb{E}[Y_i] = 0 \). The only difference from no noise case is that
\[
\|X_i - X_{i-1}\|_2 = \|Y_i\|_2 \leq \|(f(x_i) + \rho_i)\chi_S(x_i)\|_2 + \|\hat{f}(S)\|_2
\]
is bounded by a larger number but smaller than \( 2 + \gamma \). Effectively, the number of samples needed is still \( \frac{(2+\gamma) \log(4\epsilon^2/\delta)}{\epsilon} \).

**Comparison with a naive approach.** Suppose we learn \( \hat{f}(S) \) by approximating each coordinate separately. Let \( f = (f_1, f_2, \ldots, f_d) \) where each \( f_i : \{0, 1\}^n \rightarrow \mathbb{R} \) for every \( 1 \leq i \leq n \). For every \( S \subseteq [n] \), if we use \( m \) independent samples to estimate \( \hat{f}_i(S) \) by \( \tilde{f}_i(S) \), then
\[
\Pr\left[ \left\| \hat{f}_i(S) - \tilde{f}_i(S) \right\|_2^2 \geq \epsilon \right] \leq 2e^{-2me^4/4}
\]
A naive approach would require that \( \left\| \hat{f}_i(S) - \tilde{f}_i(S) \right\|_2^2 \geq \epsilon / d \) for every \( i \) to ensure that \( \left\| \hat{f}(S) - \tilde{f}(S) \right\|_2 \geq \epsilon \). Therefore, we have
\[
\Pr\left[ \left\| \hat{f}(S) - \tilde{f}(S) \right\|_2^2 \geq \epsilon \right] \leq \Pr \left[ \exists i : \left\| \hat{f}_i(S) - \tilde{f}_i(S) \right\|_2^2 \geq \epsilon / d \right]
\]
\[
\leq \sum_{i=1}^n \Pr \left[ \left\| \hat{f}_i(S) - \tilde{f}_i(S) \right\|_2^2 \geq \epsilon / \sqrt{d} \right] \tag{union bound}
\]
\[
\leq \sum_{i=1}^n 2e^{-2me^4/4d}
\]
\[
= 2n \cdot e^{-2me^4/4d}
\]
To achieve δ confidence, one needs to choose $m = O\left( \frac{d \log(1/\delta)}{\varepsilon^2} \right)$.

### A.3 Proof of Theorem 6

This section extends the Golreich-Levin/KM algorithm to the vector-valued functions. To prove the theorem we need the following results.

**THEOREM 10 (HOEFFDING’S BOUND).** Let $X_1, X_2, \ldots, X_n$ be independent variables such that $X_i \in [a, b]$. Let $\bar{X} = \mathbb{E}[X_i]$. Then for any $\varepsilon > 0$, the following bound holds

$$\Pr[|\bar{X} - \mathbb{E}[^{\bar{X}}]| \geq \varepsilon] \leq 2 \exp(-2n\varepsilon^2/(b-a)^2).$$

**PROPOSITION 2.** Let $f : \{0, 1\}^n \rightarrow \mathbb{R}^d$ be a function in $\mathcal{H}$. Let $J \subseteq [n]$. Given query access to $f$ with some random noise $\sigma$ such that $\mathbb{E}[^{\sigma}] = 0$ and $\|\sigma\|_2 \leq \beta$ and only depends on the current queried point $x$. For any $S \subseteq J$, the quantity $\sum_{T \subseteq J} \left\| \hat{f}(S \cup T) \right\|_2^2 = \mathbb{E}_z \left\| \hat{f}_{J|z}(S) \right\|_2^2$ can be estimate with error at most $\varepsilon$ (except with probability at most $\delta$) using $\frac{2(1+\delta)^2 \log(2/\delta)}{\varepsilon^2}$ queries.

**Proof.** Basically, treating each coordinate separately for each restriction function and then adding things up give us the following formula.

$$\sum_{T \subseteq J} \left\| \hat{f}(S \cup T) \right\|_2^2 = \mathbb{E}_z \left\| \hat{f}_{J|z}(S) \right\|_2^2$$

$$= \mathbb{E}_z \left\| \sum_{i=1}^n \hat{f}_{J|z}(S)_i \right\|_2^2$$

$$= \sum_{i=1}^n \mathbb{E}_z \left\| \hat{f}_{J|z}(S)_i \right\|_2^2$$

$$= \sum_{i=1}^n \mathbb{E}_z \mathbb{E}_y \left[ f(y, z)_i \chi_S(y) \right]$$

Next, we will show that $f(y, z) \cdot f(y', z) \chi_S(y) \chi_S(y')$ is a bounded real value.

$$|f(y, z) \cdot f(y', z) \chi_S(y) \chi_S(y')| = |f(y, z) \cdot f(y', z)|$$

$$\leq \|f(y, z)\|_2 \|f(y', z)\|_2$$

(Cauchy-Schwartz inq.)

Thus, $f(y, z) \cdot f(y', z) \chi_S(y) \chi_S(y')$ is a bounded random variable that can be sampled from using query access to $f$. By Hoeffding’s bound, we can estimate $\mathbb{E}_z \left\| \hat{f}_{J|z}(S) \right\|_2^2$ with error at most $\varepsilon$ except with probability at most $\delta$ using $2 \frac{\log(2/\delta)}{\varepsilon^2}$ queries access to $f$. 
Adding noise. Even in the present of the noise, it still holds that
\[ E \left[ \sum_{S \subseteq [n]} \left( f(y, z) + \sigma \cdot (f(y', z) + \sigma') \chi_S(y) \chi_S(y') \right) \right] = \sum_{T \subseteq [n]} \left\| \tilde{f}(S \cup T) \right\|_2^2. \]

This is an easy consequence of the law of iterated expectation. Once we fix \( y, y', z \), the only random variables are \( \sigma, \sigma' \) with \( E[\sigma] = E[\sigma'] = 0 \). Given that \( y, y', z \) are now constants, we only need to worry about \( E[\sigma \cdot \sigma'] \). Since \( \sigma, \sigma' \) are mutually independent, it follows that \( E[\sigma \cdot \sigma'] = 0 \) and the desired equality is obtained.

The random variable \( (f(y, z) + \sigma) \cdot (f(y', z) + \sigma') \chi_S(y) \chi_S(y') \) is bounded if \( \|\sigma\|_2, \|\sigma'\|_2 \) are bounded. Once again, it only changes the constant in the bigO notation.

\[
|(f(y, z) + \sigma) \cdot (f(y', z) + \sigma') \chi_S(y) \chi_S(y')| = |(f(y, z) + \sigma) \cdot (f(y', z) + \sigma')| \\
\leq \|f(y, z) + \sigma\|_2 \|f(y', z) + \sigma'\|_2 \\
\leq (\|f(y, z)\|_2 + \|\sigma\|_2)(\|f(y', z)\|_2 + \|\sigma'\|_2) \\
\leq (1 + \beta)^2
\]

By Hoeffding’s bound, we can estimate \( E_x \left\| \tilde{f}_{|z}(S) \right\|_2^2 \) with error at most \( \epsilon \) except with probability at most \( \delta \) using \( \frac{2(1 + \beta)^2 \log(2/\delta)}{\epsilon^2} \) queries access to \( f \). □

Proof of Theorem 6. We begin with an overview of how the algorithm works.

**Initialization:** all \( 2^n \) possible subsets of \([n]\) is put in a single bucket. The algorithm then repeats the following loop:
1. Select any bucket \( B \) (containing \( 2^m \) sets for some \( m \geq 1 \)).
2. Split \( B \) into two buckets \( B_1 \) and \( B_2 \) of \( 2^{m-1} \) sets each.
3. Estimate \( \sum_{U \in B_i} \left\| \tilde{f}(U) \right\|_2^2 \) for each \( i = 1, 2 \).
4. Discard \( B_1 \) or \( B_2 \) if its weight estimate is at most \( \tau/2 \).

The algorithm stops when all buckets contain only 1 set, and then outputs the list of these sets.

**Bucket system.** Next, we describe the bucketing system. For \( 1 \leq k \leq n \) and \( S \subseteq [n] \), we define
\[ B_{k,S} = \{ S \cup T : T \subseteq \{k+1, k+2, \ldots, n\} \}. \]

The initial bucket is \( B_{0,\emptyset} \). The buckets at the end of the algorithm have the form \( B_{k,S} = \{S\} \). Note that \( |B_{k,S}| = 2^{n-k} \). The algorithm always split the bucket \( B_{k,S} \) into two buckets \( B_{k,1}, S \) and \( B_{k+1,S \cup \{k+1\}} \).

The weight of bucket \( B_{k,S} \) is exactly \( \sum_{|T| \leq \{k+1, \ldots, n\}} \left\| \tilde{f}(S \cup T) \right\|_2^2 \).

**Correctness.** Any set \( U \) with \( \left\| \tilde{f}(U) \right\|_2^2 \geq \tau \) is never be discarded since it always contributes at least \( \tau/2 \) to the bucket it’s in. On the other hand, any set \( U \) with \( \left\| \tilde{f}(U) \right\|_2^2 \leq \tau/2 \) is always be discarded since \( \tau/4 \leq \tau/2 \). Therefore, as long as the estimation is accurate within \( \pm \tau/4 \), the algorithm is correct.

**Running time.** Any non-discarded bucket has weight at least \( \tau/2 \) even assuming that the weight estimate are only accurate within \( \pm \tau/4 \). The total weight of all non-discarded bucket is at most \( \sum_{S \subseteq [n]} \left\| \tilde{f}(S) \right\|_2^2 \leq 1 \). Thus, at anytime, there are at most \( 2/\tau \) non-discarded buckets. Note that each bucket can be split at most \( n \) times. Therefore, the algorithm repeats at most \( 2n/\tau \) loops. In each loop, the algorithm will need to estimate two buckets to accuracy \( \pm \tau/4 \) and confidence \( 1 - \delta \), each requires \( O(\log(1/\delta)/\tau^2) \) queries. The algorithm overall needs to make at most \( 4n/\tau \) weighings. This
implies that we need $O(n \log(1/\delta)/\tau^3)$ queries. By union bound, the probability that all weighings are accurate to within $\tau/4$ is at least $1 - 4n/\tau \cdot \delta$. Choose $\delta = \tau/40n$, we get that the probability is at least $9/10$. \hfill \square

**B OMITTED PROOF IN SECTION 4.2**

**B.1 Proof of Claim 1**

Suppose $f$ is $\varepsilon$-close to a $s$-sparse function $g$. Then define $\mathcal{S}_g = \{S \in [n] | \hat{g}(S) \neq 0\}$. Obviously $|\mathcal{S} \cap \mathcal{S}_g| \leq s$. Now consider $\mathcal{S} \cap \mathcal{S}_g^c$. Then by Parseval,

$$\sum_{S \in \mathcal{S} \cap \mathcal{S}_g^c} \left\| \hat{f}(S) \right\|_2^2 = \sum_{S \in \mathcal{S} \cap \mathcal{S}_g^c} \left\| \hat{f}(S) - g(S) \right\|_2^2 \leq \sum_{S \in [n]} \left\| \hat{f}(S) - g(S) \right\|_2^2 \leq \varepsilon.$$ 

Then it follows $|\mathcal{S} \cap \mathcal{S}_g^c| \leq \frac{\varepsilon}{\tau^2} = s$ and $|\mathcal{S}| \leq 2s$.

Now we need to give the accuracy of the approximation. We can easily calculate

$$\sum_{S \in \mathcal{S} \cap \mathcal{S}_g} \left\| \hat{f}(S) \right\|_2^2 \leq \frac{\varepsilon}{s} \cdot s = \varepsilon$$

$$\sum_{S \in \mathcal{S} \cap \mathcal{S}_g^c} \left\| \hat{f}(S) \right\|_2^2 \leq \sum_{S \in \mathcal{S}} \left\| \hat{f}(S) \right\|_2^2 = \sum_{S \in \mathcal{S} \cap \mathcal{S}_g} \left\| \hat{f}(S) - g(S) \right\|_2^2 \leq \varepsilon.$$ 

Now merging the two inequalities above and applying Parseval yields

$$\left\| f - \sum_{S \in \mathcal{S}} \hat{f}(S) \chi_S \right\|_2^2 = \sum_{S \in \mathcal{S}^c} \left\| \hat{f}(S) \right\|_2^2 \leq 2\varepsilon$$

and the proof is complete.

**B.2 Proof of Claim 3**

**Proof.** Now noise is considered. We need to estimate the weight with

$$E_{x,z,\rho,\rho'} [X_a(z) \langle f(x) + \rho, f(x + z) + \rho' \rangle],$$

where $x \sim \{0,1\}^n$, $z \in H^\perp$. Using the fact that $\rho, \rho'$ are mutually independent and independent from $x, x+z$, we can obtain

$$E_{x,z,\rho,\rho'} [X_a(z) \langle f(x) + \rho, f(x + z) + \rho' \rangle] = E_{x,z,\rho,\rho'} [X_a(z) \langle f(x), f(x + z) \rangle] = \text{weight}(a + H).$$

Thus the estimation is unbiased.

Now we show that the random variable is bounded. By triangle inequality,

$$\left\| f(x) + \rho \right\|_2 \leq \left\| f(x) \right\|_2 + \left\| \rho \right\|_2 \leq 1 + \sqrt{\lambda}$$

Then we can apply Cauchy-Schwartz inequality to bound the inner product

$$\left| X_a(z) \langle f(x) + \rho, f(x + z) + \rho' \rangle \right| = \left| \langle f(x) + \rho, f(x + z) + \rho' \rangle \right| \leq \left\| f(x) + \rho \right\|_2 \left\| f(x) + \rho \right\|_2 \leq (1 + \sqrt{\lambda})^2$$

Using this bound with Hoeffding will yield the desired result. \hfill \square
B.3 Proof of Claim 4

First show (1). Since hashing is pairwise independent, the probability of \( S \) hashed into the same bucket with another significant Fourier coefficient is \( \frac{1}{2^{2^\log(s)+10}} = \frac{1}{64} \). Since there are a total of \( s \) large Fourier coefficients, by union bound, collision happens with probability at most \( \frac{1}{64} \).

Now let us show (2). We need to analyze the insignificant Fourier coefficients hashed into \( a + H \). Then by Parseval and claim 1, we have

\[
\sum_{U: \|\hat{f}(U)\|_2 \leq \frac{\varepsilon}{s}} \left\| \hat{f}(U) \right\|_2^2 = \left\| f - \sum_{S \in S} \hat{f}(S) \chi_S \right\|_2^2 \leq \varepsilon.
\]

Let \( \mathcal{V} = \{ U : U \in a + H, \|\hat{f}(U)\|_2 \leq \varepsilon/s \} \). We can calculate

\[
E \left[ \sum_{U \in \mathcal{V}} \left\| \hat{f}(U) \right\|_2^2 \right] = E \left[ \sum_{U: \|\hat{f}(U)\|_2 \leq \frac{\varepsilon}{s}} \left\| \hat{f}(U) \right\|_2^2 \right] \cdot E [1(U \in a + H)]
\]

\[
= \sum_{U: \|\hat{f}(U)\|_2 \leq \frac{\varepsilon}{s}} \left\| \hat{f}(U) \right\|_2^2 \cdot E [1(U \in a + H)]
\]

\[
\leq \frac{1}{64s} \cdot \sum_{U: \|\hat{f}(U)\|_2 \leq \frac{\varepsilon}{s}} \left\| \hat{f}(U) \right\|_2^2 \cdot \Pr(U \in a + H)
\]

\[
\leq \frac{1}{64s} \cdot \varepsilon
\]

Then it follows from Markov’s inequality that

\[
\Pr \left[ \sum_{U \in \mathcal{V}} \left\| \hat{f}(U) \right\|_2^2 \right] \leq \frac{1}{4}.
\]

Now by union bound (1), (2) fails with probability at most \( \frac{2}{5} \).

B.4 Proof of Claim 5

Let us only consider only a single \( v_i \) and then the lemma follows from union bound. If we select a \( h_1 \) uniformly randomly from \( F_2^n \), then since \( |\text{span}(v)| = 2 \) and the probability of \( h_1 \notin \text{span}(v) \) is \( \frac{2^n - 2^1}{2^n} \). Now we select \( h_2 \) uniformly from \( F_2^n \). Then since \( |\text{span}(v, h_1)| = 2^2 \), the probability of \( h_2 \notin \text{span}(v, h_1) \) is \( \frac{2^n - 2^2}{2^n} \). Continuing in this fashion, we can calculate the probability that \( \sigma_i \) is full-rank is

\[
\frac{2^n - 2^1}{2^n} \cdot \frac{2^n - 2^2}{2^n} \cdots \frac{2^n - 2^{\log(s)+10}}{2^n},
\]

which is extremely close to 1 since \( 2^n \gg 2^{\log(s)+10} \) when \( n \) is big enough.
B.5 Proof of Claim 6

Let us first show correctness. For the recovery to be successful, three conditions must hold. Two conditions given in lemma Claim 4 hold with probability at least $\frac{4}{5}$. Another condition is that $\sigma_i$ is full rank for every $i$ so that we can estimate $\text{weight}(H_{\sigma_i}^{b,0})$, $\text{weight}(H_{\sigma_i}^{b,1})$. Claim 5 has shown that this holds with extremely high probability.

Suppose the three conditions hold. Then $S$ is the single Fourier coefficient with weight $\geq \tau$ hashed into some bucket $H_{\sigma_i}^{b,0}$. If $<v, S> = 0$, then $S \in H_{\sigma_i}^{b,0}$ and therefore $\text{weight}(H_{\sigma_i}^{b,0}) \geq \tau$. And since with high probability our estimation of the weight is accurate within $\frac{\tau}{4}$, the estimation is greater than $\frac{\tau}{2}$. If $<v, S> = 1$, then $H_{\sigma_i}^{b,0}$ contains only insignificant Fourier coefficients and the sum of weight cannot be more than $\frac{\tau}{16}$. Then the estimation must be smaller than $\frac{\tau}{2}$. In this way the algorithm can always correctly obtain all measurements $<v, S>$.

The sample complexity and time complexity are straightforward. We need $d \log n$ linearly measurements for successful recover, and for each measurement $\frac{\log(sd \log n)}{16\tau^2}$ queries are used to obtain the desired accuracy. Therefore, the sample complexity is $\frac{d \log(n)}{16\tau^2}$. And the time complexity is scaled by a $n$ since we need to perform tasks such as calculating the inner product between two vectors, which takes time $n$. 