Learning Set Functions that are Sparse in Non-Orthogonal Fourier Bases

Chris Wendler, Andisheh Amrollahi, Bastian Seifert, Andreas Krause, Markus Püschel
Department of Computer Science, ETH Zurich, Switzerland
wendlerc@ethz.ch, amrollaa@ethz.ch, baseifert@ethz.ch, krausea@ethz.ch, pueschel@inf.ethz.ch

Abstract

Many applications of machine learning on discrete domains, such as learning preference functions in recommender systems or auctions, can be reduced to estimating a set function that is sparse in the Fourier domain. In this work, we present a new family of algorithms for learning Fourier-sparse set functions. They require at most $nk - k\log_2 k + k$ queries (set function evaluations), under mild conditions on the Fourier coefficients, where n is the size of the ground set and k the number of non-zero Fourier coefficients. In contrast to other work that focused on the orthogonal Walsh-Hadamard transform (WHT), our novel algorithms operate with recently introduced *non-orthogonal Fourier transforms* that offer different notions of Fourier-sparsity. These naturally arise when modeling, e.g., sets of items forming substitutes and complements. We demonstrate effectiveness on several real-world applications.

Introduction

Numerous problems in machine learning on discrete domains involve learning *set functions*, i.e., functions $s : 2^N \to \mathbb{R}$ that map subsets of some ground set N to the real numbers. In recommender systems, for example, such set functions express diversity among sets of articles and their relevance w.r.t. a given need (Sharma, Harper, and Karypis 2019; Balog, Radlinski, and Arakelyan 2019); in sensor placement tasks, they express the informativeness of sets of sensors (Krause, Singh, and Guestrin 2008); in combinatorial auctions, they express valuations for sets of items (Brero, Lubin, and Seuken 2019). A key challenge is to estimate s from a small number of observed evaluations. Without structural assumptions an exponentially large (in $n = |N|$) number of queries is needed. Thus, a key question is which families of set functions can be efficiently learnt, while capturing important applications. One key property is *sparsity in the Fourier domain* (Stobbe and Krause 2012; Amrollahi et al. 2019).

The Fourier transform for set functions is classically known as the orthogonal Walsh-Hadamard transform (WHT) (De Wolf 2008; Li and Ramchandran 2015; Cheraghchi and Indyk 2017). Using the WHT, it is possible to learn functions with at most k non-zero Fourier coefficients with $O(nk)$ evaluations (Amrollahi et al. 2019). In this paper, we consider an alternative family of *non-orthogonal* Fourier transforms, recently introduced in the context of discrete signal processing on set functions (DSSP) (Püschel 2018; Püschel and Wendler 2020). In particular, we present the *first efficient algorithms* which (under mild assumptions on the Fourier coefficients), efficiently learn k-Fourier-sparse set functions requiring at most $(n+1)k - k\log_2 k$ evaluations. In contrast, naively computing the Fourier transform requires 2^n evaluations and $n2^{n-1}$ operations (Püschel and Wendler 2020).

Importantly, sparsity in the WHT domain does *not* imply sparsity in the alternative Fourier domains we consider, or vice versa. Thus, we significantly expand the class of set functions that can be efficiently learnt. One natural example of set functions, which are sparse in one of the non-orthogonal transforms, but not for the WHT, are certain preference functions considered by Djolonga, Tschiatschek, and Krause (2016) in the context of recommender systems and auctions. In recommender systems, each item may cover the set of needs that it satisfies for a customer. If needs are covered by several items at once, or items depend on each other to provide value there are substitutability or complementarity effects between the respective items, which are precisely captured by the new Fourier transforms (Püschel and Wendler 2020). Hence, a natural way to learn such set functions is to compute their respective sparse Fourier transforms.

Contributions. In this paper we develop, analyze, and evaluate novel algorithms for computing the sparse Fourier transform under various notions of Fourier basis:

1. We are the first to introduce an efficient algorithm to compute the sparse Fourier transform for the recent notions of non-orthogonal Fourier basis for set functions (Püschel and Wendler 2020). In contrast to the naive fast Fourier transform algorithm that requires 2^n queries and $n2^{n-1}$ operations, our sparse Fourier transform requires at most $nk - k\log_2 k + k = O(nk - k\log k)$ queries and $O(nk^2)$ operations to compute the k non-zero coefficients of a Fourier-sparse set function. The algorithm works in all cases up to a null set of pathological set functions.
2. We then further extend our algorithm to handle an even larger class of Fourier-sparse set functions with $O(n^2k - nk\log k)$ queries and $O(n^2k + k^2n)$ operations using filtering techniques.
3. We demonstrate the effectiveness of our algorithms in three

real-world set function learning tasks: learning surrogate objective functions for sensor placement tasks, learning facility locations functions for water networks, and preference elicitation in combinatorial auctions. The sensor placements obtained by our learnt surrogates are indistinguishable from the ones obtained using the compressive sensing based WHT by Stobbe and Krause (2012). However, our algorithm does not require prior knowledge of the Fourier support and runs significantly faster. The facility locations function learning task shows that certain set functions are sparse in the non-orthogonal basis while being dense in the WHT basis. In the preference elicitation task also only half as many Fourier coefficients are required in the non-orthogonal basis as in the WHT basis, which indicates that bidders' valuation functions are well represented in the non-orthogonal basis.

Because of the page limit, all proofs are in the supplementary material.

Fourier Transforms for Set Functions

We introduce background and definitions for set functions and associated Fourier bases, following the discrete-set signal processing (DSSP) introduced by (Püschel 2018; Püschel and Wendler 2020). DSSP generalizes key concepts from classical signal processing, including shift, convolution, and Fourier transform to the powerset domain. The approach follows a general procedure that derives these concepts from a suitable definition of the shift operation (Püschel and Moura 2008).

Set functions. We consider a ground set $N = \{x_1, \dots, x_n\}$. An associated set function maps each subset of N to a real value:

$$s : 2^N \to \mathbb{R}; A \mapsto s(A). \tag{1}$$

Each set function can be identified with a 2^n-dimensional vector $\mathbf{s} = (s(A))_{A \subseteq N}$ by fixing an order on the subsets. We choose the lexicographic order on the set indicator vectors.

Shifts. Classical convolution (e.g., on images) is associated with the translation operator. Analogously, DSSP considers different versions of "set translations", each yielding a different DSSP model, numbered 1–5. One choice is

$$\text{(model 4)} \quad T_Q s(A) = s(A \cup Q), \text{ for } Q \subseteq N. \tag{2}$$

The shift operators T_Q are parameterized by the powerset monoid $(2^N, \cup)$, since the equality $T_Q(T_R s) = T_{Q \cup R} s$ holds for all $Q, R \subseteq N$, and $s \in \mathbb{R}^{2^N}$.

Convolutional filters. The corresponding linear, shift-equivariant convolution in model 4 is given by

$$(h * s)(A) = \sum_{Q \subseteq N} h(Q)\, s(A \cup Q). \tag{3}$$

Namely, $(h * T_R s)(A) = T_R(h * s)(A)$, for all $R \subseteq N$. Convolving with h is a linear mapping called a *filter* and h is also a set function. In matrix notation we have $\mathbf{h} * \mathbf{s} = H\mathbf{s}$, where H is the filter matrix.

Fourier transform and convolution theorem. The Fourier transform (FT) simultaneously diagonalizes all filters, i.e., the matrix FHF^{-1} is diagonal for all filter matrices H, where F denotes the matrix form of the Fourier transform. Thus, different definitions of set shifts yield different notions of Fourier transform. For the shift in (2) the Fourier transform of s takes the form

$$\widehat{s}(B) = \sum_{A \subseteq N : A \cup B = N} (-1)^{|A \cap B|} s(A) \tag{4}$$

with the inverse

$$s(A) = \sum_{B \subseteq N : A \cap B = \emptyset} \widehat{s}(B). \tag{5}$$

As a consequence we obtain the convolution theorem

$$\widehat{(h * s)}(B) = \overline{h}(B)\, \widehat{s}(B). \tag{6}$$

Interestingly, $\overline{h}$ (the so-called frequency response) is computed differently than $\widehat{s}$, namely as

$$\overline{h}(B) = \sum_{A \subseteq N : A \cap B = \emptyset} h(A). \tag{7}$$

In matrix form, with respect to the chosen order of $\mathbf{s}$, the Fourier transform and its inverse are

$$F = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes n} \quad \text{and} \quad F^{-1} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{\otimes n}, \tag{8}$$

respectively, in which $M^{\otimes n} = M \otimes \cdots \otimes M$ denotes the n-fold Kronecker product of the matrix M. Thus, the Fourier transform $\widehat{\mathbf{s}} = F\mathbf{s}$ and its inverse $\mathbf{s} = F^{-1}\widehat{\mathbf{s}}$ can be computed in $n2^{n-1}$ operations.

The columns of F^{-1} form the Fourier basis and can be viewed as indexed by $B \subseteq N$. The B-th column is given by $\mathbf{f}_A^B = \iota_{A \cap B = \emptyset}$, where $\iota_{A \cap B = \emptyset} = 1$ if $A \cap B = \emptyset$ and $\iota_{A \cap B = \emptyset} = 0$ otherwise. The basis is not orthogonal as can be seen from the triangular structure in (8).

Example and interpretation. We consider a special class of preference functions that, e.g., model customers in a recommender system (Djolonga, Tschiatschek, and Krause 2016). Preference functions naturally occur in machine learning tasks on discrete domains such as recommender systems and auctions, in which, for example, they are used to model complementary- and substitution effects between goods. Goods complement each other when their combined utility is greater than the sum of their individual utilities. Analogously, goods substitute each other when their combined utility is smaller than the sum of their individual utilities. Formally, a preference function is given by

$$\begin{aligned} p(A) = &\sum_{i \in A} u_i + \sum_{\ell=1}^{L} \left(\max_{i \in A} r_{\ell i} - \sum_{i \in A} r_{\ell i} \right) \\ &- \sum_{k=1}^{K} \left(\max_{i \in A} a_{ki} - \sum_{i \in A} a_{ki} \right). \end{aligned} \tag{9}$$

Equation (9) is composed of a so-called modular term parametrized by $u \in \mathbb{R}^n$, a repulsive term parametrized by $r \in \mathbb{R}_{\geq 0}^{L \times n}$, with $L \in \mathbb{N}$, and an attractive term parametrized by $a \in \mathbb{R}_{\geq 0}^{K \times n}$, with $K \in \mathbb{N}$. The repulsive term captures substitution and the attractive term complementary effects.

Example 1 (Running example)**.** *Consider the ground set $\{x_1, x_2, x_3\}$, in which x_1 represents a tablet, x_2 a laptop and x_3 a tablet pen. Now, we create a preference function with a substitution effect between the laptop and the tablet which is expressed in the repulsive term $r = (2, 2, 0)$ and a complementary effect between the tablet and the tablet pen which is expressed in the attractive term $a = (1, 0, 1)$. The individual values of the items are expressed in the modular term $u = (2, 2, 1)$. As a result we get the following preference function p and also show its Fourier transform $\widehat{p}$, where x_{12} is short for $\{x_1, x_2\}$:*

	$\emptyset$	x_1	x_2	x_{12}	x_3	x_{13}	x_{23}	x_{123}
p	0	2	2	2	1	4	3	4
$\widehat{p}$	4	-1	0	-2	-2	1	0	0

Namely, as desired, $p(\{x_1, x_2\}) < p(\{x_1\}) + p(\{x_2\})$ and $p(\{x_1, x_3\}) > p(\{x_1\}) + p(\{x_3\})$. Note that $\widehat{p}$ is sparse. Next, we show this is always the case.

Lemma 1. *Preference functions of the form* (9) *are* Fourier-sparse w.r.t. model 4 *with at most $1 + n + Ln + Kn$ non-zero Fourier coefficients.*

Motivated by Lemma 1, we call set functions that are Fourier-sparse w.r.t. model 4 *generalized preference functions*. Formally, a generalized preference function is defined in terms of a collection of distinct subsets $N = \{S_1, \ldots, S_n\}$ of some universe $U : S_i \subseteq U$, $i \in \{1, \ldots, n\}$, and a weight function $w : U \to \mathbb{R}$. The weight of a set $S \subseteq U$ is $\mathbf{w}(S) = \sum_{u \in S} w(u)$. Then, the corresponding *generalized preference function* is

$$s : 2^N \to \mathbb{R}; A \mapsto \mathbf{w}\left(\bigcup_{S_i \in A} S_i\right). \tag{10}$$

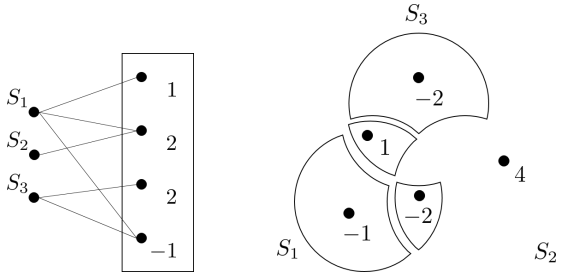
For non-negative weights s is called a *weighted coverage function* (Krause and Golovin 2014), but here we allow general (signed) weights. Thus, generalized preference functions are *generalized coverage functions* as introduced in (Püschel and Wendler 2020). Generalized coverage functions can be visualized by a bipartite graph, see Fig. 1a. In recommender systems, S_i could model the customer-needs covered by item i. Then, the score that a customer associates to a set of items corresponds to the needs covered by the items in that set. Substitution as well as complementary effects occur if the needs covered by items overlap (e.g., $S_i \cap S_j \neq \emptyset$).

Concretely, we observe in Fig. 1a that in our running example the tablet and the laptop share one need with a positive sign which yields the substitution effect, and the tablet and the tablet pen share a need with a negative sign which yields the complementary effect.

Interestingly, the Fourier coefficients for model 4 in (4) of a generalized coverage function are

$$\widehat{s}(B) = \begin{cases} \sum_{u \in U} w(u), & \text{if } B = \emptyset, \\ -\mathbf{w}\left(\bigcap_{S_i \in B} S_i \setminus \bigcup_{S_i \in N \setminus B} S_i\right), & \text{otherwise,} \end{cases} \tag{11}$$

which corresponds to the weights of the fragments of the Venn-diagram of the sets S_i (Fig. 1b). If the universe U contains fewer than 2^n items, some fragments will have weight zero, i.e., are Fourier-sparse. We refer to Section VIII of Püschel and Wendler (2020) for an in-depth discussion and interpretation of the spectrum (11).



(a) Coverage function (b) Fourier transform in (4)

Figure 1: Preference function p from Example 1 as generalized coverage function visualized as bipartite graph (a) and as Venn diagram (b). The Fourier coefficients are the weights of the fragments. Here, three are zero.

	shift $T_Q s(A)$	F^{-1} (sum) : $s_A =$
model 3	$s(A \setminus Q)$	$\sum_{B \subseteq A} (-1)^{\lvert B \rvert} \widehat{s}(B)$
model 4	$s(A \cup Q)$	$\sum_{\substack{B \subseteq N: \\ A \cap B = \emptyset}} \widehat{s}(B)$
model 5	$s(A \setminus Q \cup Q \setminus A)$	$\frac{1}{2^n} \sum_{B \subseteq N} (-1)^{\lvert A \cap B \rvert} \widehat{s}(B)$

Table 1: Shifts and Fourier expansions.

Other shifts and Fourier bases. As mentioned before, Püschel and Wendler (2020) considers 5 types of shift, i.e. DSSP models, each with its respective shift-equivariant convolution, associated Fourier basis, and thus notion of Fourier-sparsity. Model 5 is the classical definition that yields the WHT and model 4 the version introduced above. Table 1 collects the key concepts, also including model 3.

The notions of Fourier-sparsity differ substantially between models. For example, consider the coverage function for which there is only one element in the universe U and this element is covered by all sets $S_1, \ldots, S_n$. Then, $\widehat{s}(\emptyset) = 1$, $\widehat{s}(N) = -1$ and $\widehat{s}(B) = 0$ for $\emptyset \subset B \subset N$ w.r.t. model 4, and $\widehat{s}(\emptyset) = 2^n - 1$ and $\widehat{s}(B) = -1$ for all $\emptyset \subset B \subseteq N$ w.r.t. the WHT.

More generally, one can show that preference functions in (9) with at least one row of pairwise distinct values in either the repulsive or attractive part are dense w.r.t. the WHT basis.

The Fourier bases have appeared in different contexts before. For example, (4) can be related to the W-transform, which has been used by Chakrabarty and Huang (2012) to test coverage functions.

Learning Fourier-Sparse Set Functions

We now present our algorithm for learning Fourier-sparse set functions w.r.t. model 4. One of our main contributions is that the derivation and algorithm are general, i.e., they also apply

to the other models. We derive the variants for *models 3 and 5* from Table 1 in the supplementary material.

Definition 1. *A set function s is called k-Fourier-sparse if*

$$\operatorname{supp}(\widehat{s}) = \{B : \widehat{s}(B) \neq 0\} = \{B_1, \ldots, B_k\}, \quad (12)$$

where we assume that k is significantly smaller than 2^n.

Thus, exactly learning a k-Fourier-sparse set function is equivalent to computing its k non-zero Fourier coefficients and associated support. Formally, we want to solve:

Problem 1 (Sparse FT)**.** *Given oracle access to query a k-Fourier-sparse set function s, compute its Fourier support and associated Fourier coefficients.*

Sparse FT with Known Support

First, we consider the simpler problem of computing the Fourier coefficients if the Fourier support $\operatorname{supp}(\widehat{s})$ (or a small enough superset $\mathcal{B} \supseteq \operatorname{supp}(\widehat{s})$) is known. In this case, the solution boils down to selecting queries $\mathcal{A} \subseteq 2^N$ such that the linear system of equations

$$\mathbf{s}_{\mathcal{A}} = F^{-1}_{\mathcal{A}\mathcal{B}} \widehat{\mathbf{s}}_{\mathcal{B}}, \quad (13)$$

admits a solution. Here, $\mathbf{s}_{\mathcal{A}} = (s(A))_{A \in \mathcal{A}}$ is the vector of queries, $F^{-1}_{\mathcal{A}\mathcal{B}}$ is the submatrix of F^{-1} obtained by selecting the rows indexed by $\mathcal{A}$ and the columns indexed by $\mathcal{B}$, and $\widehat{\mathbf{s}}_{\mathcal{B}}$ are the unknown Fourier coefficients we want to compute.

Theorem 1 (Püschel and Wendler (2020))**.** *Let s be k-Fourier-sparse with $\operatorname{supp}(\widehat{s}) = \{B_1, \ldots, B_k\} = \mathcal{B}$. Let $\mathcal{A} = \{N \setminus B_1, \ldots, N \setminus B_k\}$. Then $F^{-1}_{\mathcal{A}\mathcal{B}}$ is invertible and s can be perfectly reconstructed from the queries $\mathbf{s}_{\mathcal{A}}$.*

Consequently, we can solve Problem 1 if we have a way to discover a $\mathcal{B} \supseteq \operatorname{supp}(\widehat{s})$, which is what we do next.

Sparse FT with Unknown Support

In the following we present our algorithm to solve Problem 1. As mentioned, the key challenge is to determine the Fourier support w.r.t. (4). The initial skeleton is similar to the algorithm *Recover Coverage* by Chakrabarty and Huang (2012), who used it to test coverage functions. Here we take the novel view of Fourier analysis to expand it to a sparse Fourier transform algorithm for all set functions. Doing so creates challenges since here the weight function in (10) is not guaranteed to be positive. Using the framework in Section we will analyze and address them.

Let $M \subseteq N$, and consider the associated restriction of a set function s on N:

$$s \downarrow_{2^M} \colon 2^M \to \mathbb{R}; A \mapsto s(A) \quad (14)$$

The Fourier coefficients of s and the restriction can be related as (proof in supplementary material):

$$\widehat{s \downarrow_{2^M}}(B) = \sum_{A \subseteq N \setminus M} \widehat{s}(A \cup B). \quad (15)$$

We observe that, if the Fourier coefficients on the right hand side of (15) do not cancel, knowing $\widehat{s \downarrow_{2^M}}$ contains information about the sparsity of $\widehat{s \downarrow_{2^{M \cup \{x\}}}}$, for $x \in N \setminus M$. To be precise, if there are no cancellations, the relation

$$\widehat{s \downarrow_{2^M}}(B) = \widehat{s \downarrow_{2^{M \cup \{x\}}}}(B) + \widehat{s \downarrow_{2^{M \cup \{x\}}}}(B \cup \{x\}) \quad (16)$$

implies that both $\widehat{s \downarrow_{2^{M \cup \{x\}}}}(B)$ and $\widehat{s \downarrow_{2^{M \cup \{x\}}}}(B \cup \{x\})$ must be zero whenever $\widehat{s \downarrow_{2^M}}(B)$ is zero. As a consequence, we can construct

$$\mathcal{B} = \bigcup_{B \in \operatorname{supp}(\widehat{s \downarrow_{2^M}})} \{B, B \cup \{x\}\}, \quad (17)$$

with $\operatorname{supp}(\widehat{s \downarrow_{2^{M \cup \{x\}}}}) \subseteq \mathcal{B}$, from (16), and then compute $\widehat{s \downarrow_{2^{M \cup \{x\}}}}$ with Theorem 1 in this case.

As a result we can solve Problem 1 with our algorithm **SSFT**, under mild conditions on the coefficients that guarantee that cancellations do not occur, by successively computing the non-zero Fourier coefficients of restricted set functions along the chain

$$s \downarrow_{2^\emptyset} = \widehat{s \downarrow_{2^\emptyset}}, \widehat{s \downarrow_{2^{\{x_1\}}}}, \widehat{s \downarrow_{2^{\{x_1, x_2\}}}}, \ldots, \widehat{s \downarrow_{2^N}} = \widehat{s}. \quad (18)$$

For example, **SSFT** works with probability one if all non-zero Fourier coefficients are sampled from independent continuous probability distributions:

Lemma 2. *With probability one **SSFT** correctly computes the Fourier transform of Fourier-sparse set functions s with $\operatorname{supp}(\widehat{s}) = \mathcal{B}$ and randomly sampled Fourier coefficients, that satisfy*

1. *$\widehat{s}(B) \sim P_B$, where P_B is a continuous probability distribution with density function p_B, for $B \in \operatorname{supp}(\widehat{s})$,*
2. *$p_{\mathcal{B}} = \prod_{B \in \mathcal{B}} p_B$.*

Algorithm. We initialize **SSFT** with $M_0 = \emptyset$ and $s \downarrow_{2^\emptyset}(\emptyset) = s(\emptyset)$ (lines 1–2). Then, in each iteration of the for loop (line 3), we grow our set $M_i = M_{i-1} \cup \{x_i\}$ by adding the next element (line 4), determine the superset $\mathcal{B}$ of $\operatorname{supp}(\widehat{s \downarrow_{2^{M_i}}})$ based on the Fourier coefficients from the previous iteration $\widehat{s \downarrow_{2^{M_{i-1}}}}$ (lines 5–8) and solve the resulting known-support-problem using Theorem 1 (lines 9–12). After n iterations we end up with the Fourier coefficients of s.

We depict an execution of **SSFT** on p from Example 1 in Fig. 2. We process the elements x_1, x_2, x_3 in reverse order, which results in the final Fourier coefficients being lexicographically ordered. Note that in this example applying **SSFT** naively would fail, as $\widehat{p \downarrow_{2^\emptyset}}(\emptyset) = p(\emptyset) = 0$ **SSFT** would falsely conclude that both $\widehat{p \downarrow_{2^{\{x_3\}}}}(\emptyset)$ and $\widehat{p \downarrow_{2^{\{x_3\}}}}(\{x_3\})$ are zero resulting in them and their children being pruned. Instead, we initialize the algorithm with $\widehat{p \downarrow_{\{x_3\}}}$. The levels of the tree correspond to the elements of the chain (18). When computing the Fourier coefficients of a level, the ones of the previous level determine the support superset $\mathcal{B}$. Whenever, **SSFT** encounters a Fourier coefficient that is equal to zero, all of its children are pruned from the tree. For example, $\widehat{p \downarrow_{2^{\{x_2, x_3\}}}}(\{x_2, x_3\}) = 0$, thus, $\{x_2, x_3\}$ and $\{x_1, x_2, x_3\}$ are not included in the support superset $\mathcal{B}$ of $\widehat{p \downarrow_N} = \widehat{p}$. While pruning in our small example only avoids the computation of $\widehat{p \downarrow_{2^N}}(\{x_2, x_3\})$ and $\widehat{p \downarrow_{2^N}}(\{x_1, x_2, x_3\})$, it avoids an exponential amount of computation in larger examples (for $|N| = n$, when $\widehat{s \downarrow_{2^{M_{i-1}}}}(B) = 0$ the two subtrees of height $n - i$ containing its children are pruned).

SSFT Sparse set function Fourier transform of s

1: $M_0 \leftarrow \emptyset$
2: $\widehat{s \downarrow_{2^{M_0}}}(\emptyset) \leftarrow s(\emptyset)$
3: **for** $i = 1, \ldots, n$ **do**
4: $\quad M_i \leftarrow M_{i-1} \cup \{x_i\}$
5: $\quad \mathcal{B} \leftarrow \emptyset, \mathcal{A} \leftarrow \emptyset$
6: $\quad$ **for** $B \in \operatorname{supp}(\widehat{s \downarrow_{2^{M_{i-1}}}})$ **do**
7: $\quad\quad \mathcal{B} \leftarrow \mathcal{B} \cup \{B, B \cup \{x_i\}\}$
8: $\quad\quad \mathcal{A} \leftarrow \mathcal{A} \cup \{M_i \setminus B, M_i \setminus (B \cup \{x_i\})\}$
9: $\quad \mathbf{s}_{\mathcal{A}} \leftarrow (s(A))_{A \in \mathcal{A}}$
10: $\quad \mathbf{x} \leftarrow$ solve $\mathbf{s}_{\mathcal{A}} = F^{-1}_{\mathcal{A}\mathcal{B}}\mathbf{x}$ for $\mathbf{x}$
11: $\quad$ **for** $B \in \mathcal{B}$ with $\mathbf{x}_B \neq 0$ **do**
12: $\quad\quad \widehat{s \downarrow_{2^{M_i}}}(B) \leftarrow \mathbf{x}_B$
13: **return** $\widehat{s \downarrow_{2^{M_n}}}$

SSFT+ Filtering based **SSFT** of s

1: *// Sample random coefficients.*
2: $h(\emptyset) = 1$
3: **for** $x \in \{x_1, \ldots, x_n\}$ **do**
4: $\quad h(\{x\}) \leftarrow c \sim \mathcal{N}(0, 1)$
5: *// Fourier transform of filtered set function.*
6: $\widehat{h * s} \leftarrow$**SSFT**$(h * s)$
7: *// Compute the original coefficients.*
8: **for** $B \in \operatorname{supp}(\widehat{h * s})$ **do**
9: $\quad \widehat{s}_B \leftarrow \widehat{(h * s)}(B) / \overline{h}(B)$
10: **return** $\widehat{s}$

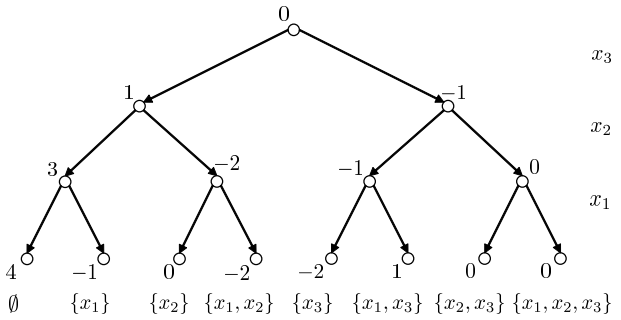


Figure 2: The Fourier coefficients of the restricted versions of p from Example 1. Depth 0 (root) corresponds to the Fourier transform of $p \downarrow_{2^\emptyset}$, depth 1 to the one of $p \downarrow_{2^{\{x_3\}}}$, depth 2 to the one of $p \downarrow_{2^{\{x_2, x_3\}}}$ and depth 3 to the one of p.

Note that for practical reasons we only process up to $k_{\max}$ subsets in line 6. In line 11, we consider a Fourier coefficient $|\mathbf{x}_B| < \epsilon$ (a hyperparameter) as zero.

Analysis. We consider set functions s that are k-Fourier-sparse (but not $(k-1)$-Fourier-sparse) with support $\operatorname{supp}(\widehat{s}) = \{B_1, \ldots, B_k\} = \mathcal{B}$, i.e., $\{s : 2^N \to \mathbb{R} : \widehat{s}(B) \neq 0 \text{ iff } B \in \mathcal{B}\}$, which is isomorphic to

$$\mathcal{S} = \{\widehat{\mathbf{s}} \in \mathbb{R}^k : \widehat{\mathbf{s}}_i \neq 0 \text{ for all } i \in \{1, \ldots, k\}\}. \tag{19}$$

Let λ denote the Lebesgue measure on $\mathbb{R}^k$. Let $\mathcal{P}_C^{M_i} = \{B \in \mathcal{B} : B \cap M_i = C\}$.

Pathological set functions. SSFT fails to compute the Fourier coefficients for which $\widehat{s \downarrow_{2^{M_i}}}(C) = 0$ despite $\mathcal{P}_C^{M_i} \neq \emptyset$. Thus, the set of pathological set functions $\mathcal{D}_1$, i.e., the set of set functions for which **SSFT** fails, can be written as the finite union of kernels

$$\mathcal{K}_1(M_i, C) = \{\widehat{\mathbf{s}} \in \mathbb{R}^k : \widehat{s \downarrow_{2^{M_i}}}(C) = 0\} \tag{20}$$

intersected with $\mathcal{S}$.

Theorem 2. *Using prior notation, the set of pathological set functions for **SSFT** is given by*

$$\mathcal{D}_1 = \bigcup_{i=0}^{n-1} \bigcup_{C \subseteq M_i : \mathcal{P}_C^{M_i} \neq \emptyset} \mathcal{K}_1(M_i, C) \cap \mathcal{S}, \tag{21}$$

and has Lebesgue measure zero, i.e., $\lambda(\mathcal{D}_1) = 0$.

Complexity. By reusing queries and computations from the $(i-1)$-th iteration of **SSFT** in the i-th iteration, we obtain:

Theorem 3. ***SSFT** requires at most $nk - k\log_2 k + k$ queries and $O(nk^2)$ operations.*

Shrinking the Set of Pathological Fourier Coefficients

According to Theorem 2, the set of pathological Fourier coefficients for a given support has measure zero. However, unfortunately, this set includes important classes of set functions including graph cuts (in the case of unit weights) and hypergraph cuts.[1]

Solution. The key idea to exclude these and further narrow down the set of pathological cases is to use the convolution theorem (6), i.e., the fact that we can modulate Fourier coefficients by filtering. Concretely, we choose a random filter h such that **SSFT** works for $h * s$ with probability one. $\widehat{s}$ is then obtained from $\widehat{h * s}$ by dividing by the frequency response $\overline{h}$. We keep the associated overhead in $O(n)$ by choosing a one-hop filter, i.e., $h(B) = 0$ for $|B| > 1$. Motivated by the fact that, e.g., the product of a Rademacher random variable (which would lead to cancellations) and a normally distributed random variable is again normally distributed, we sample our filtering coefficients i.i.d. from a normal distribution. By filtering our signal with such a random filter we aim to end up in a situation similar to Lemma 2. We call the resulting algorithm **SSFT+**, shown above.

[1] As an example, consider the cut function c associated with the graph $V = \{1, 2, 3\}$, $E = \{\{1, 2\}, \{2, 3\}\}$ and $w_{12} = w_{23} = 1$, using $\widehat{\mathbf{c}} = (0, 1, 2, -2, 1, 0, -2, 0)^T$. c maps every subset of V to the weight of the corresponding graph cut.

Algorithm. In **SSFT+** we create a random one-hop filter h (lines 2–4), apply **SSFT** to the filtered signal $h * s$ (line 6) and compute $\widehat{s}$ based on $\widehat{h * s}$ (lines 8–9).

Analysis. Building on the analysis of **SSFT**, recall that $\mathcal{S}$ denotes the set of k-Fourier-sparse (but not $(k-1)$-Fourier-sparse) set functions and $\mathcal{P}_C^{M_i}$ are the elements $B \in \operatorname{supp}(\widehat{s})$ satisfying $B \cap M_i = C$. Let

$$\mathcal{K}_2(M_i, C) = \Big\{ \widehat{\mathbf{s}} \in \mathbb{R}^k : \widehat{s \downarrow_{2^{M_i}}}(C) = 0 \text{ and } \widehat{s \downarrow_{2^{M_i \cup \{x_j\}}}}(C) = 0 \text{ for } j \in \{i+1, \dots, n\} \Big\}. \quad (22)$$

Theorem 4. *With probability one with respect to the randomness of the filtering coefficients, the set of pathological set functions for **SSFT+** has the form (using prior notation)*

$$\mathcal{D}_2 = \bigcup_{i=0}^{n-2} \bigcup_{C \subseteq M_i : \mathcal{P}_C^{M_i} \neq \emptyset} \mathcal{K}_2(M_i, C) \cap \mathcal{S}. \quad (23)$$

Theorem 4 shows that **SSFT+** correctly processes $\widehat{s \downarrow_{2^{M_i}}}(C) = 0$ with $\mathcal{P}_C^{M_i} \neq \emptyset$, iff there is an element $x \in \{x_{i+1}, \dots, x_n\}$ for which $\widehat{s \downarrow_{2^{M_i \cup \{x\}}}}(C) \neq 0$.

Theorem 5. *If $\mathcal{D}_1$ is non-empty, $\mathcal{D}_2$ is a proper subset of $\mathcal{D}_1$. In particular, $\mathcal{K}_1(M_i, C) \cap S \neq \emptyset$ implies $\mathcal{K}_2(M_i, C) <_{\mathbb{R}} \mathcal{K}_1(M_i, C)$, for all $C \subseteq M_i \subseteq N$ with $\mathcal{P}_C^{M_i} \neq \emptyset$.*

Complexity. There is a trade-off between the number of non-zero filtering coefficients and the size of the set of pathological set functions. For example, for the one-hop filters used, computing $(h * s)(A)$ requires $1 + n - |A|$ queries.

Theorem 6. *The query complexity of **SSFT+** is $O(n^2 k - nk \log k)$ and the algorithmic complexity is $O(n^2 k + nk^2)$.*

Related Work

We briefly discuss related work on learning set functions.

Fourier-sparse learning. There is a substantial body of research concerned with learning Fourier/WHT-sparse set functions (Stobbe and Krause 2012; Scheibler, Haghighatshoar, and Vetterli 2013; Kocaoglu et al. 2014; Li and Ramchandran 2015; Cheraghchi and Indyk 2017; Amrollahi et al. 2019). Recently, Amrollahi et al. (2019) have imported ideas from the hashing based sparse Fourier transform algorithm (Hassanieh et al. 2012) to the set function setting. The resulting algorithms compute the WHT of k-WHT-sparse set functions with a query complexity $O(nk)$ for general frequencies, $O(kd \log n)$ for low degree ($\leq d$) frequencies and $O(kd \log n \log(d \log n))$ for low degree set functions that are only approximately sparse. To the best of our knowledge this latest work improves on previous algorithms, such as the ones by Scheibler, Haghighatshoar, and Vetterli (2013), Kocaoglu et al. (2014), Li and Ramchandran (2015), and Cheraghchi and Indyk (2017), providing the best guarantees in terms of both query complexity and runtime. E.g., Scheibler, Haghighatshoar, and Vetterli (2013) utilize similar ideas like hashing/aliasing to derive sparse WHT algorithms that work under random support (the frequencies are uniformly distributed on 2^N) and random coefficient (the coefficients are samples from continuous distributions) assumptions. Kocaoglu et al. (2014) propose a method to compute the WHT of a k-Fourier-sparse set function that satisfies a so-called unique sign property using queries polynomial in n and 2^k.

In a different line of work, Stobbe and Krause (2012) utilize results from compressive sensing to compute the WHT of k-WHT-sparse set functions, for which a superset $\mathcal{P}$ of the support is known. This approach also can be used to find a k-Fourier-sparse approximation and has a theoretical query complexity of $O(k \log^4 |\mathcal{P}|)$. In practice, it even seems to be more query-efficient than the hashing based WHT (see experimental section of Amrollahi et al. (2019)), but suffers from the high computational complexity, which scales at least linearly with $|\mathcal{P}|$. Regrading coverage functions, to our knowledge, there has not been any work in the compressive sensing literature for the non-orthogonal Fourier bases which do not satisfy RIP properties and hence lack sparse recovery and robustness guarantees.

In summary, all prior work on Fourier-based methods for learning set functions was based on the WHT. Our work leverages the broader framework of signal processing with set functions proposed by Püschel and Wendler (2020), which provides a larger class of Fourier transforms and thus new types of Fourier-sparsity.

Other learning paradigms. Other lines of work for learning set functions include methods based on new neural architectures (Dolhansky and Bilmes 2016; Zaheer et al. 2017; Weiss, Lubin, and Seuken 2017), methods based on backpropagation through combinatorial solvers (Djolonga and Krause 2017; Tschiatschek, Sahin, and Krause 2018; Wang et al. 2019; Vlastelica et al. 2019), kernel based methods (Buathong, Ginsbourger, and Krityakierne 2020), and methods based on other succinct representations such as decision trees (Feldman, Kothari, and Vondrák 2013) and disjunctive normal forms (Raskhodnikova and Yaroslavtsev 2013).

Empirical Evaluation

We evaluate the two variants of our algorithm (**SSFT** and **SSFT+**) for model 4 on three classes of real-world set functions. First, we approximate the objective functions of sensor placement tasks by Fourier-sparse functions and evaluate the quality of the resulting surrogate objective functions. Second, we learn facility locations functions (which are preference functions) that are used to determine cost-effective sensor placements in water networks (Leskovec et al. 2007). Finally, we learn simulated bidders from a spectrum auctions test suite (Weiss, Lubin, and Seuken 2017).

Benchmark learning algorithms. We compare our algorithm against three state-of-the-art algorithms for learning WHT-sparse set functions: the compressive sensing based approach **CS-WHT** (Stobbe and Krause 2012), the hashing based approach **H-WHT** (Amrollahi et al. 2019), and the robust version of the hashing based approach **R-WHT** (Amrollahi et al. 2019). For our algorithm we set $\epsilon = 0.001$ and $k_{\text{max}} = 1000$. **CS-WHT** requires a superset $\mathcal{P}$ of the (unknown) Fourier support, which we set to all $B \subseteq N$ with $|B| \leq 2$ and the parameter for expected sparsity to 1000. For **H-WHT** we used the exact algorithm without low-degree as-

sumption and set the expected sparsity parameter to 2000. For **R-WHT** we used the robust algorithm without low-degree assumption and set the expected sparsity parameter to 2000 unless specified otherwise.

Sensor Placement Tasks

We consider a discrete set of sensors located at different fixed positions measuring a quantity of interest, e.g., temperature, amount of rainfall, or traffic data, and want to find an informative subset of sensors subject to a budget constraint on the number of sensors selected (e.g., due to hardware costs). To quantify the informativeness of subsets of sensors, we fit a multivariate normal distribution to the sensor measurements (Krause, Singh, and Guestrin 2008) and associate each subset of sensors $A \subseteq N$ with its information gain (Srinivas et al. 2010)

$$G(A) = \frac{1}{2} \log |I_{|A|} + \sigma^{-2}(K_{ij})_{i,j \in A}|, \qquad (24)$$

where $(K_{ij})_{i,j \in A}$ is the submatrix of the covariance matrix K that is indexed by the sensors $A \subseteq N$ and $I_{|A|}$ the $|A| \times |A|$ identity matrix. We construct two covariance matrices this way for temperature measurements from 46 sensors at Intel Research *Berkeley* and for velocity data from 357 sensors deployed under a highway in *California*.

The information gain is a submodular set function and, thus, can be approximately maximized using the greedy algorithm by Nemhauser, Wolsey, and Fisher (1978): $A^* \approx \arg\max_{A \subseteq N : |A| \leq d} G(A)$ to obtain informative subsets. We do the same using Fourier-sparse surrogates s of G: $A^+ \approx \arg\max_{A \subseteq N : |A| \leq d} s(A)$ and compute $G(A^+)$. As a baseline we place d sensors at random A_{rand} and compute $G(A_{\text{rand}})$. Figure 3 shows our results. The x-axes correspond to the cardinality constraint used during maximization and the y-axes to the information gain obtained by the respective informative subsets. In addition, we report next to the legend the execution time and number of queries needed by the successful experiments.

Interpretation of results. H-WHT only works for the *Berkeley* data. For the other data set it is not able to reconstruct enough Fourier coefficients to provide a meaningful result. The likely reason is that the target set function is not exactly Fourier-sparse, which can cause an excessive amount of collisions in the hashing step. In contrast, **CS-WHT** is noise-robust and yields sensor placements that are indistinguishable from the ones obtained by maximizing the true objective function in the first task. However, for the *California* data, **CS-WHT** times out. In contrast, **SSFT** and **R-WHT** work well on both tasks. In the first task, **SSFT** is on par with **CS-WHT** in terms of sensor placement quality and significantly faster despite requiring more queries. On the *California* data, **SSFT** yields sensor placements of similar quality as the ones obtained by **R-WHT** while requiring orders of magnitude fewer queries and time.

Learning Preference Functions

We now consider a class of preference functions that are used for the cost-effective contamination detection in water networks (Leskovec et al. 2007). The networks stem from the Battle of Water Sensor Networks (BSWN) challenge (Ostfeld et al. 2008). The junctions and pipes of each BSWN network define a graph. Additionally, each BSWN network has dynamic parameters such as time-varying water consumption demand patterns, opening and closing valves, and so on.

$\|N\|$		α	queries	k	$\|\mathbf{p} - \mathbf{p}'\|/\|\mathbf{p}\|$
20	**SSFT**		**734**	**102**	**0**
	WHT			2^9	0.000143
				2^{14}	0.000078
				2^{19}	0.000001
50	**SSFT**		**10K**	**648**	**0**
	R-WHT	1	*2, 103K*	*1, 380*	*0.001744*
		2	*4, 192K*	*2, 739*	*0.000847*
		4	*8, 370K*	*5, 054*	*0.000129*
		8	*16, 742K*	*9, 547*	*0.000108*
100	**SSFT**		**76K**	**2,308**	**0**
	R-WHT	1	16,544K	2,997	0.000546
		2	33,100K	6,466	0.000380
200	**SSFT**		**494K**	**7,038**	**0**
300	**SSFT**		**1,644K**	**16,979**	**0**
400	**SSFT**		**3,859K**	**28,121**	**0**
500	**SSFT**		**7,218K**	**38,471**	**0**

Table 2: Comparison of model 4 sparsity (SSFT) against WHT sparsity (R-WHT) of facility locations functions in terms of reconstruction error $\|\mathbf{p} - \mathbf{p}'\|/\|\mathbf{p}\|$ for varying $|N|$; The italic results are averages over 10 runs.

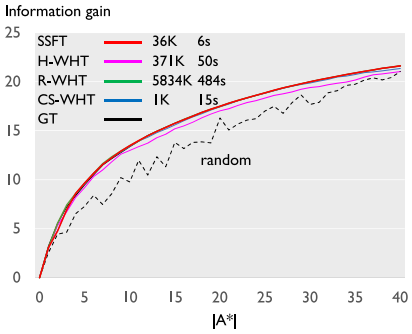
To determine a cost-effective subset of sensors (e.g., given a maximum budget), Leskovec et al. (2007) make use of facility locations functions of the form

$$p : 2^N \to \mathbb{R}; A \mapsto \sum_{\ell=1}^{L} \max_{i \in A} r_{\ell i}, \qquad (25)$$

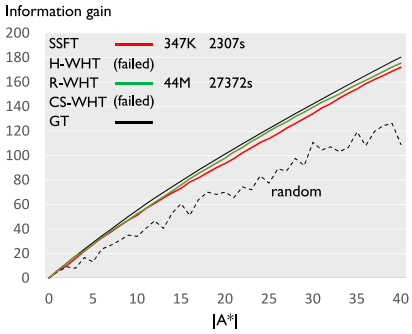
where r is a matrix in $\mathbb{R}_{\geq 0}^{L \times n}$. Each row corresponds to an event (e.g., contamination of the water network at any junction) and the entry $r_{\ell i}$ quantifies the utility of the i-th sensor in case of the ℓ-th event. It is straightforward to see that (25) is a preference function with $a = 0$ and $u_i = \sum_{\ell=1}^{L} r_{\ell i}$. Thus, they are sparse w.r.t. model 4 and dense w.r.t. WHT (see Lemma 1).

Leskovec et al. (2007) determined three different utility matrices $r \in \mathbb{R}^{3424 \times 12527}$ that take into account the fraction of events detected, the detection time, and the population affected, respectively. The matrices were obtained by costly simulating millions of possible contamination events in a 48 hour timeframe. For our experiments we select one of the utility matrices and obtain subnetworks by selecting the columns that provide the maximum utility, i.e., we select the $|N| = n$ columns j with the largest $\max_\ell r_{\ell j}$.

In Table 2 we compare the sparsity of the corresponding facility locations function in model 4 against its sparsity in the WHT. For $|N| = 20$, we compute the full WHT and select the k largest coefficients. For $|N| > 20$, we compute the k largest WHT coefficients using **R-WHT**. The model 4 coefficients are always computed using **SSFT**. If the facility locations



(a) Berkeley, $n = 46$



(b) California, $n = 357$

Figure 3: Comparison of learnt surrogate objective functions on submodular maximization tasks subject to cardinality constraints (x-axis); On the y-axis we plot the information gain achieved by the informative subset obtained by the respective method. We report the number of queries and execution time in seconds next to the legend or indicate failure.

	number of queries (in thousands)			Fourier coefficients recovered			relative reconstruction error		
	SSFT	**SSFT+**	**H-WHT**	**SSFT**	**SSFT+**	**H-WHT**	**SSFT**	**SSFT+**	**H-WHT**
L	3 ± 4	229 ± 73	781 ± 0	118 ± 140	303 ± 93	675 ± 189	0.566 ± 0.490	0 ± 0	0 ± 0
R	20 ± 1	646 ± 12	781 ± 0	659 ± 32	813 ± 36	$1,779 \pm 0$	0.012 ± 0.007	0 ± 0	0 ± 0
N	71 ± 0	$3,305 \pm 1$	781 ± 0	$1,028 \pm 3$	$1,027 \pm 6$	$4,170 \pm 136$	0.012 ± 0.001	0.015 ± 0.009	0.268 ± 0.212

Table 3: Multi-region valuation model ($n = 98$). Each row corresponds to a different bidder type.

function is k sparse w.r.t. model 4 for some $|N| = n$, we set the expected sparsity parameter of **R-WHT** to different multiples αk up to the first α for which the algorithm runs out of memory. We report the number of queries, number of Fourier coefficients k, and relative reconstruction error. For **R-WHT** experiments that require less than one hour we report average results over 10 runs (indicated by italic font). For $|N| > 20$, the relative error cannot be computed exactly and thus is obtained by sampling 100,000 sets $\mathcal{A}$ uniformly at random and computing $\|\mathbf{p}_{\mathcal{A}} - \mathbf{p'}_{\mathcal{A}}\|/\|\mathbf{p}_{\mathcal{A}}\|$, where p denotes the real facility locations function and p' the estimate.

Interpretation of results. The considered facility locations functions are indeed sparse w.r.t. model 4 and dense w.r.t. the WHT. As expected, **SSFT** outperforms **R-WHT** in this scenario, which can be seen by an error of exactly zero and the lower number of queries, which also lead to a proportional speedup, for the SSFT. This experiment shows certain classes of set functions of practical relevance are better represented in the model 4 basis than in the WHT basis.

Preference Elicitation in Auctions

In combinatorial auctions a set of goods $N = \{x_1, \ldots, x_n\}$ is auctioned to a set of m bidders. Each bidder j is modeled as a set function $b_j : 2^N \to \mathbb{R}$ that maps each bundle of goods to its subjective value for this bidder. The problem of learning bidder valuation functions from queries is known as the preference elicitation problem (Brero, Lubin, and Seuken 2019). Our experiment sketches an approach under the assumption of Fourier sparsity.

As common in this field, we resort to simulated bidders. Specifically, we use the multi-region valuation model (MRVM) from the spectrum auctions test suite (Weiss, Lubin, and Seuken 2017). In MRVM, 98 goods are auctioned off to 10 bidders of different types (3 local, 4 regional, and 3 national). We learn these bidders using the prior Fourier-sparse learning algorithms, this time including **SSFT+**, but excluding **CS-WHT**, since $\mathcal{P}$ is not known in this scenario. Table 3 shows the results: means and standard deviations of the number of queries required, number of Fourier coefficients recovered, and relative error (estimated using 10,000 samples) taken over the bidder types and 25 runs.

Interpretation of results. First, we note that **SSFT+** can indeed improve over **SSFT** for set functions that are relevant in practice. Namely, **SSFT+** consistently learns sparse representations for local and regional bidders, while **SSFT** fails. **H-WHT** also achieves perfect reconstruction for local and regional bidders. For the remaining bidders none of the methods achieves perfect reconstruction, which indicates that those bidders do not admit a sparse representation. Second, we observe that, for the local and regional bidders, in the non-orthogonal model 4 basis only half as many coefficients are required as in the WHT basis. Third, **SSFT+** requires less queries than **H-WHT** in the Fourier-sparse cases.

Conclusion

We introduced an algorithm for learning set functions that are sparse with respect to various generalized, non-orthogonal Fourier bases. In doing so, our work significantly expands the set of efficiently learnable set functions. As we explained, the new notions of sparsity connect well with preference functions in recommender systems and the notions of complementarity and substitutability, which we consider an exciting avenue for future research.

Ethics Statement

Our approach is motivated by a range of real world applications, including modeling preferences in recommender systems and combinatorial auctions, that require the modeling, processing, and analysis of set functions, which is notoriously difficult due to their exponential size. Our work adds to the tool set that makes working with set functions computationally tractable. Since the work is of foundational and algorithmic nature we do not see any immediate ethical concerns. In case that the models estimated with our algorithms are used for making decisions (such as recommendations, or allocations in combinatorial auctions), of course additional care has to be taken to ensure that ethical requirements such as fairness are met. These questions are complementary to our work.

References

Amrollahi, A.; Zandieh, A.; Kapralov, M.; and Krause, A. 2019. Efficiently Learning Fourier Sparse Set Functions. In *Advances in Neural Information Processing Systems*, 15094–15103.

Balog, K.; Radlinski, F.; and Arakelyan, S. 2019. Transparent, Scrutable and Explainable User Models for Personalized Recommendation. In *Proc. Conference on Research and Development in Information Retrieval (ACM SIGIR)*, 265–274.

Brero, G.; Lubin, B.; and Seuken, S. 2019. Machine Learning-powered Iterative Combinatorial Auctions. *arXiv preprint arXiv:1911.08042*.

Buathong, P.; Ginsbourger, D.; and Krityakierne, T. 2020. Kernels over Sets of Finite Sets using RKHS Embeddings, with Application to Bayesian (Combinatorial) Optimization. In *International Conference on Artificial Intelligence and Statistics*, 2731–2741.

Chakrabarty, D.; and Huang, Z. 2012. Testing Coverage Functions. In *International Colloquium on Automata, Languages, and Programming*, 170–181. Springer.

Cheraghchi, M.; and Indyk, P. 2017. Nearly optimal deterministic algorithm for sparse Walsh-Hadamard transform. *ACM Transactions on Algorithms (TALG)* 13(3): 1–36.

De Wolf, R. 2008. A brief introduction to Fourier analysis on the Boolean cube. *Theory of Computing* 1–20.

Djolonga, J.; and Krause, A. 2017. Differentiable Learning of Submodular Models. In *Advances in Neural Information Processing Systems*, 1013–1023.

Djolonga, J.; Tschiatschek, S.; and Krause, A. 2016. Variational Inference in Mixed Probabilistic Submodular Models. In *Advances in Neural Information Processing Systems*, 1759–1767.

Dolhansky, B. W.; and Bilmes, J. A. 2016. Deep Submodular Functions: Definitions and Learning. In *Advances in Neural Information Processing Systems*, 3404–3412.

Feldman, V.; Kothari, P.; and Vondrák, J. 2013. Representation, Approximation and Learning of Submodular Functions Using Low-rank Decision Trees. In *Conference on Learning Theory*, 711–740.

Hassanieh, H.; Indyk, P.; Katabi, D.; and Price, E. 2012. Nearly Optimal Sparse Fourier Transform. In *Proc. ACM Symposium on Theory of Computing*, 563–578.

Kocaoglu, M.; Shanmugam, K.; Dimakis, A. G.; and Klivans, A. 2014. Sparse Polynomial Learning and Graph Sketching. In *Advances in Neural Information Processing Systems*, 3122–3130.

Krause, A.; and Golovin, D. 2014. Submodular Function Maximization. In *Tractability: Practical Approaches to Hard Problems*, 71–104. Cambridge University Press.

Krause, A.; Singh, A.; and Guestrin, C. 2008. Near-optimal Sensor Placements in Gaussian processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research* 9: 235–284.

Leskovec, J.; Krause, A.; Guestrin, C.; Faloutsos, C.; VanBriesen, J.; and Glance, N. 2007. Cost-effective Outbreak Detection in Networks. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 420–429.

Li, X.; and Ramchandran, K. 2015. An Active Learning Framework using Sparse-Graph Codes for Sparse Polynomials and Graph Sketching. In *Advances in Neural Information Processing Systems*, 2170–2178.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions — I. *Mathematical programming* 14(1): 265–294.

Ostfeld, A.; Uber, J. G.; Salomons, E.; Berry, J. W.; Hart, W. E.; Phillips, C. A.; Watson, J.-P.; Dorini, G.; Jonkergouw, P.; Kapelan, Z.; et al. 2008. The Battle of the Water Sensor Networks (BWSN): A Design Challenge for Engineers and Algorithms. *Journal of Water Resources Planning and Management* 134(6): 556–568.

Püschel, M. 2018. A Discrete Signal Processing Framework for Set Functions. In *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4359–4363. IEEE.

Püschel, M.; and Moura, J. M. 2008. Algebraic signal processing theory: Foundation and 1-D time. *IEEE Trans. on Signal Processing* 56(8): 3572–3585.

Püschel, M.; and Wendler, C. 2020. Discrete Signal Processing with Set Functions. *arXiv preprint arXiv:2001.10290*.

Raskhodnikova, S.; and Yaroslavtsev, G. 2013. Learning pseudo-Boolean k-DNF and Submodular Functions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1356–1368.

Scheibler, R.; Haghighatshoar, S.; and Vetterli, M. 2013. A Fast Hadamard Transform for Signals with Sub-linear Sparsity. In *Proc. Annual Allerton Conference on Communication, Control, and Computing*, 1250–1257. IEEE.

Sharma, M.; Harper, F. M.; and Karypis, G. 2019. Learning from Sets of Items in Recommender Systems. *ACM Trans. on Interactive Intelligent Systems (TiiS)* 9(4): 1–26.

Srinivas, N.; Krause, A.; Kakade, S. M.; and Seeger, M. 2010. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proc. International Conference on Machine Learning (ICML)*, 1015–1022.

Stobbe, P.; and Krause, A. 2012. Learning Fourier Sparse Set Functions. In *Artificial Intelligence and Statistics*, 1125–1133.

Tschiatschek, S.; Sahin, A.; and Krause, A. 2018. Differentiable Submodular Maximization. In *Proc. International Joint Conference on Artificial Intelligence*, 2731–2738.

Vlastelica, M.; Paulus, A.; Musil, V.; Martius, G.; and Rolínek, M. 2019. Differentiation of Blackbox Combinatorial Solvers. *arXiv preprint arXiv:1912.02175* .

Wang, P.-W.; Donti, P. L.; Wilder, B.; and Kolter, Z. 2019. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. *arXiv preprint arXiv:1905.12149* .

Weiss, M.; Lubin, B.; and Seuken, S. 2017. SATS: A Universal Spectrum Auction Test Suite. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 51–59.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep Sets. In *Advances in Neural Information Processing Systems*, 3391–3401.