Recoverabilty Conditions for Rankings Under Partial Information

Srikanth Jagabathula **Devavrat Shah** *

Abstract

We consider the problem of *exact* recovery of a function, defined on the space of permutations (or, the symmetric group) over n elements, from a given partial set of its Fourier series coefficients; we focus on non-negative functions. As the main result, we derive sufficient conditions not only in terms of the structural properties of the functions, but also in terms of a bound on the sparsity of the functions that can be recovered. We also propose an algorithm for recovery and prove that it finds the function with the sparsest support, which could be obtained through ℓ_0 optimization. We also show that ℓ_1 optimization, however, fails to recover the function.

1 Introduction

Functions over permutations serve as rich tools for modeling uncertainty in several important practical applications; however, they are limited because their size has a factorial blow-up. In order to overcome this difficulty, they are often approximated using a small subset of their Fourier coefficients. Such an approximation raises the natural question: which functions can be "well approximated" by a partial set of Fourier coefficients? This setup has several applications; as an example consider the *Identity Management Problem*, where the goal is to track the identities of n objects from noisy measurements of identities and positions. This problem is typically modeled using a distribution over permutations, which is often approximated using a partial set of Fourier coefficients. Recent work by [1, 2] deals with updating the distribution with new observations in the Fourier domain. Since only a partial set of Fourier coefficients are maintained, the final distribution must be recovered from only partial information.

To fix ideas, let S_n denote the set of permutations of n elements and $f : S_n \to \mathbb{R}_+$ denote a non-negative function. Our goal is to determine f from a partial set of Fourier series coefficients. Variants of this problem were considered before. One approach [3] was to obtain lower bounds on the energy contained in subsets of Fourier Transform coefficients to obtain an approximation of a function with the "minimum" energy. This approach, however, does not naturally extend to the case of exact recovery. The problem of exact recovery, on the other hand, has been widely studied in the context of discrete-time functions. In particular, this problem received much attention in the area of *compressive sensing*, which recently has gained immense popularity. In the compressive sensing literature (see [4]), the basic question pertains to the *design* of an $m \times n$ "sensing" matrix A so that based on $y = Ax$ (or its noisy version), one can recover x. Our setup is different because in our case, the corresponding matrix A is *given* rather than a *design* choice. In addition, the matrix A does not satisfy the Restricted Isometry Property (RIP) [5], which is crucial for the approaches in the compressive sensing literature to work.

*Both authors are supported by NSF CAREER project CNS 0546590 and AFOSR Complex Networks program. They are with the Laboratory of Information and Decision Systems and Department of EECS, MIT. Emails: {jskanth, devavrat } @ mit.edu

2 Notation and Problem Statement

In this section we describe the setup and the precise problem statement. As mentioned before, our interest is in learning non-negative valued functions $f : S_n \to \mathbb{R}_+$, where $\mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$. The support of f is defined as $\text{supp}(f) = \{\sigma \in S_n : f(\sigma) \neq 0\}$, and its cardinality, $|\text{supp}(f)|$, is called the *sparsity* of f and denoted by K. We also call the sparsity of f its ℓ_0 norm and denote it by $|f|_0$.

In this paper, we wish to learn f from a partial set of Fourier coefficients. In order to define the Fourier transform of a function over the permutation group, we introduce the following notation. Consider a partition of n, i.e. an ordered tuple $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$, such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r \geq 1$ and $n = \lambda_1 + \lambda_2 + \ldots + \lambda_r$; for example, $\lambda = (n-1, 1)$ is a partition of n. Now consider a partition of the n elements, $\{1, \ldots, n\}$, as per the λ partition, i.e. divide n elements into r bins with ith bin having λ_i elements. It is easy to see that n elements can be divided as per the λ partition in $D_\lambda = n!/(\lambda_1!\lambda_2! \ldots \lambda_r!)$ distinct ways. Let the distinct partitions be denoted by $t_i, 1 \leq i \leq D_\lambda$[1]. For example, for $\lambda = (n-1, 1)$ there are $D_\lambda = n!/(n-1)! = n$ distinct partitions given by $t_i \equiv \{1, \ldots, i-1, i+1, \ldots, n\}\{i\}$, $1 \leq i \leq n$.

Now, given a permutation $\sigma \in S_n$, its action on t_i is defined through its action on the n elements of t_i, resulting in a λ partition with the n elements permuted. In the above example with $\lambda = (n-1, 1)$, σ acts on t_i to give $t_{\sigma(i)}$, i.e. $\sigma : t_i \equiv \{1, \ldots, i-1, i+1, \ldots, n\}\{i\} \to t_{\sigma(i)} \equiv \{1, \ldots, \sigma(i)-1, \sigma(i)+1, \ldots, n\}\{\sigma(i)\}$. Now, for a given partition λ and a permutation $\sigma \in S_n$, define a 0/1 valued $D_\lambda \times D_\lambda$ matrix $M^\lambda(\sigma)$ as $M^\lambda_{ij}(\sigma) = 1$ if and only if $\sigma(t_j) = t_i$ for all $1 \leq i, j \leq D_\lambda$. The matrix $M^\lambda(\sigma)$ corresponds to a degree D_λ representation of the permutation group.

The partial information we consider in this paper is the Fourier transform coefficient of f at the representation M^λ for each λ. The motivation for considering Fourier coefficients at representations M^λ is two fold: first, as we shall see, Fourier coefficients at representations M^λ have natural interpretations, which makes it easy to collect in practical applications; second, each representation M^λ contains all the lower-order irreducible representations; thus, for each λ, M^λ conveniently captures the information contained in all the lower-order Fourier coefficients up to λ. We now define the Fourier coefficient of f at representation M^λ, which we call λ-partial information.

Definition 1 (λ-Partial Information)**.** *Given a function $f : S_n \to \mathbb{R}_+$ and partition λ, the Fourier Transform coefficient at representation M^λ, which we call the λ-partial information, is denoted by $\hat{f}(\lambda)$ and is defined as $\hat{f}(\lambda) = \sum_{\sigma \in S_n} f(\sigma) M^\lambda(\sigma)$.*

Recall the example of $\lambda = (n-1, 1)$ with f as a probability distribution on S_n. Then, $\hat{f}(\lambda)$ is an $n \times n$ matrix with the (i, j)th entry being the probability of element j mapped to element i under f. That is, $\hat{f}(\lambda)$ corresponds to the *first order* marginal of f in this case.

It is helpful to think about the partial information we have in terms of a weighted bipartite graph. In particular, given λ-partial information, we think of each permutation σ as a complete matching in a $D_\lambda \times D_\lambda$ bipartite graph with the left nodes labeled from t_1 to t_{D_λ}, the right nodes labeled from t_1 to t_{D_λ}, and with an edge between t_i on left and t_j on right if and only if $\sigma(t_i) = t_j$. We refer to this representation as the λ-bipartite graph of σ. We can now think of the λ-partial information as a weighted bipartite graph obtained by the superposition of individual λ-bipartite graphs of permutations in the support of the function, weighted by their respective function values; we call this the λ-weighted bipartite graph. For brevity, we also use the terms bipartite graphs and weighted bipartite graphs when the partition λ they correspond to is clear from the context .

Our goal is to recover f from the given partial information. Of course, since only partial information is available, such recovery is not possible in general. Therefore, our goal is to: (a) characterize the class of functions that can be recovered exactly from only partial information, and (b) design a procedure to perform recovery.

[1]To keep notation simple, we use t_i instead of t_i^λ that takes explicit dependence on λ into account.

3 Main Results

In this section, we provide our main results. Before we provide a characterization of the class of functions that can be recovered from only partial information, it is helpful to think about the problem we are trying to solve as a game between Alice and Bob in which Alice picks a function $f\colon S_n \to \mathbb{R}_+$, constructs $\hat{f}(\lambda)$ and gives it to Bob, while Bob tries to correctly guess the function f used by Alice. In order to understand what kind of functions can be recovered, we first take a few examples.

Example 1. *For any $n \geq 4$, consider the four permutations $\sigma_1 = (1,2)$, $\sigma_2 = (3,4)$, $\sigma_3 = (1,2)(3,4)$ and $\sigma_4 = \mathrm{id}$, where* id *is the identity permutation. Let $f\colon S_n \to \mathbb{R}_+$ be a function such that $f(\sigma_1) = p_1, f(\sigma_2) = p_2, f(\sigma_3) = p_3$ and $f(\sigma) = 0$ for $\sigma \neq \sigma_1, \sigma_2, \sigma_3$. When $\lambda = (n-1,1)$, it is easy to see that $M^\lambda(\sigma_1) + M^\lambda(\sigma_2) = M^\lambda(\sigma_3) + M^\lambda(\sigma_4)$. Without loss of generality, let $p_1 \leq p_2$. Then, $\hat{f}(\lambda) = \sum_{i=1}^{3} p_i M^\lambda(\sigma_i) = (p_2 - p_1)M^\lambda(\sigma_2) + (p_3 + p_1)M^\lambda(\sigma_3) + p_1 M^\lambda(\sigma_4)$. Thus, function g with $g(\sigma_2) = p_2 - p_1$, $g(\sigma_3) = p_3 + p_1$, $g(\sigma_4) = p_1$ and $g(\sigma) = 0$ for all other $\sigma \in S_n$ is such that $\hat{g}(\lambda) = \hat{f}(\lambda)$; moreover, $\|g\|_0 = 3 = \|f\|_0$.*

It is easy to see from Example 1 that, without any additional information, it is impossible to recover the underlying function uniquely from the given information. We can imagine recovering the function by decomposing the λ-weighted bipartite graph into corresponding λ-bipartite graphs with weights. Note that the bipartite graph of σ_1 is completely contained in the superimposition of the bipartite graphs of σ_2 and σ_3. Thus, we can decompose the weighted bipartite graph either by"peeling-off" σ_1 resulting in the function f, or by "absorbing" σ_1 into σ_2 and σ_3 resulting in the function g. We have this confusion because the permutations in the support of f don't have their respective unique "signatures" or "witnesses" in the data; the "signature" of σ_1 is swamped by those of σ_2 and σ_3. Therefore, in order to be able to recover f we require that each permutation possess a unique "signature" or "witness" in the data that can be used to identify it. In particular, we require that each permutation in the support of the underlying function should have at least one edge in its bipartite graph that is not contained in the bipartite graph of any other permutation in the support; this edge serves as the "witness" or the "signature" of this permutation. This condition ensures that the bipartite graph of no permutation is completely contained in the superposition of the bipartite graphs of other permutations.

It turns out that the "witness" we imposed above is not sufficient for exact recovery. To see why, consider *any* four permutations $\sigma_1, \sigma_2, \sigma_3$ and σ_4, possibly satisfying the witness condition. We construct four new permutations by appending two new elements $n+1, n+2$ as follows: $\sigma_1' = \sigma(n+1)(n+2), \sigma_2' = \sigma_2(n+1)(n+2), \sigma_3' = \sigma_3(n+1, n+2), \sigma_4' = \sigma_4(n+1, n+2)$, where we assume that all permutations are represented using the cycle notation. Now suppose we form a function f by respectively assigning the four permutations weights p_1, p_2, p_3 and p_4 such that $p_1 + p_2 = p_3 + p_4$. It is easy to see that given the λ-partial information corresponding to $\lambda = (n-1,1)$, we can also decompose it into the function g with support $\sigma_1'' = \sigma(n+1, n+2), \sigma_2' = \sigma_2(n+1, n+2), \sigma_3' = \sigma_3(n+1)(n+2), \sigma_4' = \sigma_4(n+1)(n+2)$. Thus, it is impossible to recover f given only $\hat{f}(\lambda)$. This example suggests that the function values must not only all be distinct, but the sum of the function values corresponding to any one subset of permutations should be distinct from the sum of function values over any other subset.

As we show shortly, the above two conditions are, in fact, sufficient for exact recovery. Formally, the conditions we impose on f are as follows.

Condition 1 (Sufficiency Conditions)**.** Let f satisfy the following:

- *Unique Witness*: for any $\sigma \in \mathrm{supp}\,(f)$, there exists $1 \leq i_\sigma, j_\sigma \leq D_\lambda$ such that $M^\lambda_{i_\sigma j_\sigma}(\sigma) = 1$, but $M^\lambda_{i_\sigma j_\sigma}(\sigma') = 0$, for all $\sigma' (\neq \sigma) \in \mathrm{supp}\,(f)$.
- *Linear Independence*: for any collection of integers $c_1, \ldots, c_K$ taking values in $\{-K, \ldots, K\}$, $\sum_{k=1}^{K} c_k p_k \neq 0$, unless $c_1 = \ldots = c_K = 0$.

Note that as shown through the above examples, if either of these conditions is not satisfied, then, in general, it is not possible to recover the function even if we impose a bound on the sparsity of the function. Thus, in that sense, these conditions are necessary.

We prove that Condition 1 is sufficient for exact recovery by proposing an algorithm called the sparsest-fit algorithm (the rationale for the name will become apparent shortly) and proving that the algorithm recovers f from $\hat{f}(\lambda)$ whenever f satisfies Condition 1. The description of the algorithm is given in [6]. For reasons that will become apparent soon, we discuss the complexity of the algorithm towards the end of the section. We now have the following theorem.

Theorem 1. *When f satisfies Condition 1, the sparsest-fit algorithm recovers f exactly from $\hat{f}(\lambda)$. Thus, Condition 1 is sufficient for exact recovery of f from $\hat{f}(\lambda)$.*

The characterization we have provided above is in terms of the structural properties of the underlying function. The next natural issue that arises is how Condition 1 translates into the "complexity" of the functions that can be recovered. A natural measure of complexity is the sparsity of the function. Therefore, we try to understand how Condition 1 translates into a bound on the sparsity of the functions that can be recovered. In order to understand this question, let's go back to the game between Alice and Bob. Clearly, if Alice is adversarial, even if Alice and Bob agree on a bound on the sparsity of f, Bob has no chance of winning (cf. Example 1). However, suppose Alice is not adversarial and chooses a function with a given sparsity bound randomly. Then, it is reasonable to wonder if Bob can guess the function correctly with a high probability. Put differently, we know that recovery, even with a sparsity bound, is not possible in the worst-case; however, is it possible in the average case? In order to answer this question, we first propose a natural random generative model for the function with a given sparsity. Then, for a given partition λ, we obtain a sparsity bound $K(\lambda)$ so that if $K < K(\lambda)$ then recovery of f generated according to the generative model from $\hat{f}(\lambda)$ is possible with high probability. Note that this characterization is similar to the characterization for functions in the compressive sensing literature. We discuss this connection in detail in the next section. We now describe the random model and then provide our results on the sparsity bound $K(\lambda)$ for all partitions λ.

Definition 2 (Random Model)**.** *Given $K \in \mathbb{Z}_+$ and an interval $\mathscr{C} = [a, b]$, $0 < a < b$, a random function f with sparsity K and values in $\mathscr{C}$ is generated as follows: choose K permutations from S_n independently and uniformly at random [2], say $\sigma_1, \ldots, \sigma_K$; select K values from $\mathscr{C}$ uniformly at random, say $p_1, \ldots, p_K$; then function f is defined as $f(\sigma) = p_i$ for $\sigma = \sigma_i$, $1 \leq i \leq K$ and $f(\sigma) = 0$ otherwise. We will denote this model as $R(K, \mathscr{C})$.*

In what follows, we spell out the result starting with few specific cases so as to better explain the dependency of $K(\lambda)$ on D_λ.

Case 1: $\lambda = (n-1, 1)$. Here $D_\lambda = n$ and $\hat{f}(\lambda)$ provides the *first order* marginal information. As stated next, for this case the achievable recoverability threshold $K(\lambda)$ scales[3] as $n \log n$.

Theorem 2. *A function f generated as per Definition 2 can be recovered from $\hat{f}(\lambda)$ by the sparsest-fit algorithm with probability $1 - o(1)$ as long as $K \leq (1-\varepsilon) n \log n$ for any fixed $\varepsilon > 0$.*

Case 2: $\lambda = (n-m, m)$ with $1 < m = O(1)$. Here $D_\lambda = \Theta(n^m)$ and $\hat{f}(\lambda)$ provides the *mth order* marginal information. As stated next, for this case we find that $K(\lambda)$ scales at least as $n^m \log n$.

Theorem 3. *A function f generated as per Definition 2 can be recovered from $\hat{f}(\lambda)$ by the sparsest-fit algorithm for $\lambda = (n-m, m), m = O(1)$, with probability $1 - o(1)$ as long as $K \leq \frac{(1-\varepsilon)}{m!} n^m \log n$ for any fixed $\varepsilon > 0$.*

In general, for any λ with $\lambda_1 = n - m$ and $m = O(1)$, arguments of Theorem 3 can be adapted to show that $K(\lambda)$ scales as $n^m \log n$. Theorems 2 and 3 suggest that the recoverability threshold scales $D_\lambda \log D_\lambda$ for $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 = n - m$ for $m = O(1)$.

[2] Throughout, we will assume that the random selection is done *with* replacement.

[3] Throughout this paper, by log we mean the natural logarithm, i.e. $\log_e$, unless otherwise stated.

Next, we consider the case of more general λ.

Case 3: $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 = n - O\left(n^{2/9-\delta}\right)$ for any $\delta > 0$. As stated next, for this case, the recoverability threshold $K(\lambda)$ scales at least as $D_\lambda \log\log D_\lambda$.

Theorem 4. *A function f generated as per Definition 2 can be recovered from $\hat{f}(\lambda)$ by the sparsest-fit algorithm for $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 = n - n^{2/9-\delta}$ for any $\delta > 0$, with probability $1 - o(1)$ as long as $K \leq (1-\varepsilon) D_\lambda \log\log D_\lambda$ for any fixed $\varepsilon > 0$.*

Case 4: Any $\lambda = (\lambda_1, \ldots, \lambda_r)$. The results stated thus far suggest that the threshold is essentially D_λ, ignoring the logarithm term. For general λ, we establish a bound on $K(\lambda)$ as stated in Theorem 5 below. Before stating the result, we introduce some notation. For given λ, define $\alpha = (\alpha_1, \ldots, \alpha_r)$ with $\alpha_i = \lambda_i/n,\ 1 \leq i \leq r$. Let $H(\alpha) = -\sum_{i=1}^r \alpha_i \log \alpha_i$ and $H'(\alpha) = -\sum_{i=2}^r \alpha_i \log \alpha_i$.

Theorem 5. *Given $\lambda = (\lambda_1, \ldots, \lambda_r)$, a function f generated as per Definition 2 can be recovered from $\hat{f}(\lambda)$ by the sparsest-fit algorithm with probability $1 - o(1)$ as long as $K \leq C\, D_\lambda^{\gamma(\alpha)}$ where*

$$\gamma(\alpha) = \frac{M}{M+1}\left[1 - O(1)\frac{H(\alpha) - H'(\alpha)}{H(\alpha)}\right], \quad \textit{with} \quad M = \left\lfloor \frac{1}{1-\alpha_1} \right\rfloor$$

and $0 < C < \infty$ is a constant.

At a first glance, the above result seems very different from the crisp formulas of Theorems 2-4. Therefore, let us consider a few special cases. First, observe that as $\alpha_1 \uparrow 1$, $M/(M+1) \to 1$. Further, as stated in Lemma 1, $H'(\alpha)/H(\alpha) \to 1$. Thus, we find that the bound on sparsity essentially scales as D_λ. Note that the cases 1, 2 and 3 fall squarely under this scenario since $\alpha_1 = \lambda_1/n = 1 - o(1)$. Thus, this general result contains the results of Theorems 2-4 (ignoring the logarithm terms). Next, consider the other extreme of $\alpha_1 \downarrow 0$. Then, $M \to 1$ and again by Lemma 1, $H'(\alpha)/H(\alpha) \to 1$. Therefore, the bound on sparsity scales as $\sqrt{D_\lambda}$. This ought to be the case because for $\lambda = (1, \ldots, 1)$ we have $\alpha_1 = 1/n \to 1$, $D_\lambda = n!$, and unique signature property holds only up to $o(\sqrt{D_\lambda}) = o(\sqrt{n!})$ due to the standard Birthday paradox. In summary, Theorem 5 appears reasonably tight for the general form of partial information λ. We now state the Lemma 1 used above.

Lemma 1. *Consider any $\alpha = (\alpha_1, \ldots, \alpha_r)$ with $1 \geq \alpha_1 \geq \cdots \geq \alpha_r \geq 0$ and $\sum_{i=1}^r \alpha_r = 1$. Then, $\lim_{\alpha_1 \uparrow 1} H'(\alpha)/H(\alpha) = 1$ and $\lim_{\alpha_1 \downarrow 0} H'(\alpha)/H(\alpha) = 1$.*

This finishes all the results on sparsity bounds. Now, coming back to the complexity of the sparsest-fit algorithm, for $K = O(D_\lambda \log D_\lambda)$, the algorithm has a running time complexity of $O\left(\exp(\log^2 D_\lambda)\right)$ with a high probability under the random model $R(K, \mathscr{C})$. Thus, the complexity is quasi-polynomial in the input size, which is "close" to $\text{poly}(D_\lambda)$, the best running-time complexity we can hope for since the input size is D_λ^2. The worst-case time complexity is, however, $O(2^K)$; note that this is exponential in sparsity and not input size.

4 The sparsest solution

In this section, we explore the connections of our problem setup with the area of compressive sensing. As mentioned before, the typical question in the area of compressive sensing pertains to the design of the sensing matrix A such that it is possible to recover x from partial measurements $y = Ax$. The typical result in this context corresponds to a bound on the sparsity of x for which exact recovery is possible, provided the matrix A satisfies some conditions (viz. RIP conditions). Our problem can be cast in this form by, with some abuse of notation, imagining the function f as an $n! \times 1$ vector and the data $\hat{f}(\lambda)$ as the $D_\lambda^2 \times 1$ vector. Then $\hat{f}(\lambda) = Af$, where A is an $D_\lambda^2 \times n!$ matrix and each column corresponds to the matrix $M^\lambda(\sigma)$ for a certain permutation σ. As mentioned before, however, the matrix A in our setup is given rather than a design choice.

The typical approach in compressive sensing is to recover x by finding the sparsest solution (through ℓ_0 optimization) consistent with the given information y. Since finding the sparsest solution is, in general, computationally hard, its convex relaxation, ℓ_1 optimization, is considered. The main results, then, correspond to conditions under which the two approaches yield the correct solution. Such conditions, and the sparsity bounds we obtained above, raise the natural question of whether the two approaches will work in our setup. We answer the first question in the affirmative and the second one in the negative. In particular, suppose the ℓ_0 optimization problem we solve is

$$\begin{aligned} &\text{minimize} && \|g\|_0 && \text{over} && g : S_n \to \mathbb{R}_+ \\ &\text{subject to} && \hat{g}(\lambda) = \hat{f}(\lambda). \end{aligned} \tag{1}$$

Then, we have the following theorem.

Theorem 6. *Consider a function f that satisfies Condition 1. Then, f is the unique solution to the ℓ_0 optimization problem* (1)*.*

It now follows from Theorem 1 that the sparsest-fit algorithm indeed identifies the sparsest solution. Note that Condition 1 is essentially necessary for f to be the unique sparsest solution. In particular, the examples given above Condition 1 indicate that when either of the conditions is not satisfied, then f may not be the unique sparsest solution.

Now, ℓ_1 minimization ((1) with the objective replaced by $\|g\|_1$), on the other hand, fails to recover the function, as stated in the following theorem.

Theorem 7. *Consider a function f randomly generated as per Definition 2 with sparsity $K \geq 2$. Then, with probability $1 - o(1)$ the solution to ℓ_1 minimization is not unique.*

5 Conclusion

In summary, we considered the problem of *exactly* recovering a non-negative function over the space of permutations from a given partial set of Fourier coefficients. We derived sufficient conditions not only in terms of the structural properties, but also in terms of the sparsity bounds for functions that can be recovered. If either of the conditions we imposed in not satisfied, then, in general, it is not possible to recovery the underlying function even if we impose The conditions we propose. We also proposed the sparsest-fit algorithm that recovers f whenever Condtion 1 is satisfied. We also show that the sparsest-fit algorithm indeed finds the sparsest solution. Moreover, we show that ℓ_1 optimization fails to recover the function and propose a simple iterative algorithm to perform recovery under the recoverability conditions. Natural next steps include extending our approach to other forms of partial information and to the scenario where the underlying function is not exactly, but only approximately, sparse.

References

[1] J. Huang, C. Guestrin, and L. Guibas. Efficient inference for distributions on permutations. *Advances in Neural Information Processing Systems*, 20:697–704, 2008.

[2] R. Kondor, A. Howard, and T. Jebara. Multi-object tracking with representations of the symmetric group. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, 2007.

[3] K.L. Kueh, T. Olson, D. Rockmore, and K.S. Tan. Nonlinear approximation theory on compact groups. *Journal of Fourier Analysis and Applications*, 7(3):257–281, 2001.

[4] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2005.

[5] S. Jagabathula and D. Shah. Inferring rankings under constrained sensing. In *NIPS*, pages 7–1, 2008.

[6] S. Jagabathula and D. Shah. Inferring rankings using constrained sensing. *http://arxiv.org/abs/0910.0063*, 2009.