Ranking with kernels in Fourier space

Risi Kondor
Center for the Mathematics of Information
California Institute of Technology
risi@caltech.edu

Marconi Barbosa
NICTA, Australian National University
marconi.barbosa@nicta.com.au

Abstract

In typical ranking problems the total number n of items to be ranked is relatively large, but each data instance involves only $k << n$ items. This paper examines the structure of such partial rankings in Fourier space. Specifically, we develop a kernel–based framework for solving ranking problems, define some canonical kernels on permutations, and show that by transforming to Fourier space, the complexity of computing the kernel between two partial rankings can be reduced from $O((n-k)!^2)$ to $O((2k)^{2k+3})$.

1 Introduction

The word "ranking" covers a wide array of problems from learning a preference function from lists, through fusing the results of multiple search engines to methods for evaluating the results of elections. A closely related problem is "permutation learning" which attempts to learn an assignment between two sets of objects, rather than an ordering of a single set (Helmbold & Warmuth, 2009). On the theory side, "rank aggregation", i.e., finding a single ranking that best represents a collection of individual rankings according to some metric, is known to be hard, but recently some tractable approximations to this problem have emerged (Ailon et al., 2005).

The common feature of all these problems is that they involve aggregating information from a large set of diverse ranking instances. Typically, the individual instances, rather than being complete rankings of all n items under consideration, are partial rankings that only involve $k << n$ items. For example, in the search engine case, we might only take the top ten results from each search engine, giving us a sequence of partially overlapping lists. Similarly, in a movie recommendation system no user will have seen all the movies in our database, and even if they have, they would find it very difficult to give a single linear ordering of a large number of them. Instead, in many situations, including social surveys and election schemes, people are asked to rank order just a small subset of the n items under consideration. An extreme case of this are sports tournaments, such as in chess, where a single global ranking must be established from a large collection of game outcomes, each of which involves only two players.

Ultimately, ranking is a problem of inference on the group of permutations of n objects, called the symmetric group. However, for various mathematical and computational reasons, formulating it as such is difficult. A number of approaches are popular that circumvent this combinatorial problem by reducing ranking to a sequence of binary decisions (Ailon & Mohri, 2008; Balcan et al., 2008), estimating a parametric distribution (Lebanon & Lafferty, 2002), or estimating a scoring function. Depending on the problem at hand, these heuristics might be more or less appropriate. For example, in the search engine case it is reasonable to assume that the relevance of each web page can be described by a real valued score, and, in fact, formulating the ranking problem as learning the score might be a good way to reduce the complexity of the hypothesis space. On the other hand, in the collaborative filtering example the scoring heuristic is not so helpful, since there is no single "best movie". As a more extreme case, in social surveys there is a lot of interest not just in finding a single consensus ranking, but in finding clusters, trends, and principal axes of variation in the data, which really do force us to treat individual instances as permutations or sets of permutations and take into account the combinatorial structure of the symmetric group.

The algorithmic framework that we use in this paper is that of kernel methods. There are a number of reasons why this is attractive for ranking: (1) kernel methods are flexible in that once we define an appropriate kernel between any two (partial) rankings they allow us to handle diverse data involving a mixture of ranking instances of different types (partial rankings of different structure and order); (2) by the representer theorem they allow us to circumvent the problem that the symmetric group is of size $n!$; (3) by the input/output kernels formalism they allow us to predict rankings as well as learn from rankings, in fact, there might not even be a distinction between input and output spaces; (4) by the same token, they make it easy to introduce additional features (correlates) that are not rankings; (5) finally, there is a wealth of algorithms to choose from to suit the learning problem at hand.

Despite these potential advantages of kernel methods, they have not been used much in the ranking domain, and to the best of our knowledge the issue of defining generic kernels on the symmetric group has only been addressed in (Kondor, 2008). We believe that one of the main reasons for this is that it is difficult to marginalize kernels to the types of partial rankings that actually occur in data. The main practical conclusion of the present paper is that by a careful analysis in Fourier space, heavily exploiting the underlying algebra of the symmetric group, it is possible to compute such marginalized kernels quite fast.

Our focus in this paper is on structure and computational complexity rather than algorithms, so we will mostly remain agnostic about which kernel algorithm to employ. As a short menu of options we suggest the following.

In the unsupervised setting one can use kernel density estimation or some other similar method to estimate a distribution over rankings. By the representer theorem the resulting distribution takes the form of a linear combination $p(\sigma) = \sum_i \alpha_i K(R_i, \sigma)$, and we can find the "best consensus ranking" by finding $\arg\max_{\sigma\in\mathbb{S}_n} \langle \boldsymbol{p}, \boldsymbol{\sigma}\rangle$. In political data, to find factors underlying people's votes one can use kernel PCA. Similarly, to find representative groups of individuals, we would use kernel clustering.

In a more predictive setting, a typical type of question is "Given that user j ranked the movies $x_{i_1}, \ldots, x_{i_k}$ in the order $x_{i_1} \succ \ldots \succ x_{i_k}$, how would she rank $x_{i'_1}, \ldots, x_{i'_\ell}$?" A natural algorithm for this sort of problem is a multi–class SVM that minimizes a regularized risk of the form $\sum_j L(f, R_j) + \langle \boldsymbol{f}, \boldsymbol{f}\rangle_K^2$, where the loss is the multi-class hinge loss comparing the correct ranking of $x_{i'_1}, \ldots, x_{i'_\ell}$ conditional on $x_{i_1} \succ \ldots \succ x_{i_k}$ to the highest scoring incorrect ranking,

$$L(f, R_j) = \max\big[0, 1 - \big(\langle \boldsymbol{f}, \boldsymbol{R}_j^e\rangle_K - \max_{\tau\in\mathbb{S}_{k'}\setminus\{e\}} \langle \boldsymbol{f}, \boldsymbol{R}_j^\tau\rangle_K\big)\big], \tag{1}$$

$\langle\,\cdot\,,\,\cdot\,\rangle_K$ is the RKHS inner product, $\boldsymbol{R}_j^\tau$ stands for $(x_{i_1} \succ \ldots \succ x_{i_k},\ x_{i'_{\tau^{-1}(\ell)}} \succ \ldots \succ x_{i'_{\tau^{-1}(1)}})$, which is the j'th training example, the "outputs" of which have been permuted by τ, and e is the identity permutation. The exact meaning of some of these notations will become clear later in the paper.

This paper has several goals: to define a canonical class of kernels on permutations (Section 3), show how these kernels can be efficiently evaluated in Fourier space (Section 4), and give a detailed analysis of the complexity of the computations involved (Section 5). Our main result is that the inner product between any two k'th order partial rankings can be computed in at most $(2k)^{2k+3}$ operations, irrespective of the number of items to be ranked (Theorem 13).

To get to this result we need a range of tools from algebra, which we try to present in a way that is as concise yet intuitive as possible. Unfortunately, page limitations prevented us from making the paper entirely self–contained. Background information from representation theory and most proofs had to be relegated to a supplement, which will be published under separate cover.

2 Partial rankings, multirankings, and the group algebra

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be a set of items to be ranked. We will use $x \succ x'$ to denote that x is ranked higher than x' according to some ranking R. A **total ranking** of $\mathcal{X}$ is then a statement of the form

$$x_{i_1} \succ x_{i_2} \succ \ldots \succ x_{i_n}, \tag{2}$$

where $i_1, \ldots, i_n$ are distinct indices in $[1, n]$ (in this paper $[i, j]$ denotes $\{i, i+1, \ldots, j\}$).

In contrast, a **partial ranking** is a statement of the form $X_1 \succ X_2 \succ \ldots \succ X_k$, where $X_1, X_2, \ldots, X_k$ are disjoint subsets of $\mathcal{X}$, and the semantics is that for $i < j$ anything in X_i is ranked above anything in X_j. If a pair of items x and y are both in the same X_i, or one or both of them are not in any of the X_i's at all, then their relative order is indeterminate. In other words, neither $x \succ y$, nor $y \succ x$.

Partial rankings can be of different types. For example, in an election example, the vote of each person corresponds to a partial ranking of the form

$$x_{i_1} \succ x_{i_2} \succ \ldots \succ x_{i_k}, \qquad k \le n, \tag{3}$$

where, to simplify notation, we abbreviated $\{x_{i_j}\}$ to just x_{i_j}. We call this an **interleaving partial ranking** of order k, because in any total ranking satisfying (3), $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ are interleaved with the other x_i's. The outcomes of individual games in a chess tournament are an example of this type of partial ranking with $k=2$. In contrast, the top k hits returned by a search engine induce what we call **top–k partial rankings**

$$x_{i_1} \succ x_{i_2} \succ \ldots \succ x_{i_k} \succ X_{\text{rest}}, \tag{4}$$

where $X_{\text{rest}} = \mathcal{X} \setminus \{x_{i_1}, \ldots, x_{i_k}\}$. Many other types of partial rankings are conceivable. For example, a survey mentioned in (Diaconis, 1988) asked respondents which of 13 characteristics that a child might have is the most desirable, next two most desirable, least desirable, and next two least desirable. This corresponds to a partial ranking of the form $X_1 \succ X_2 \succ X_3 \succ X_4 \succ X_5$, where $|X_1| = |X_5| = 1$, $|X_2| = |X_4| = 2$ and $|X_3| = 7$. While the general framework described in this paper does apply to such more complicated partial rankings, for brevity we will limit our discussion to interleaving and top–k partial rankings.

We will also encounter cases where we have to deal with the conjunction of several partial rankings. We call this a **multiranking**. For example, in a movie recommendation system we might know that a certain user ranked movies x_1, x_2 and x_3 in the order $x_1 \succ x_2 \succ x_3$. Now based on a large number of rankings of similar movies submitted by other users, we want to decide whether given her choice $x_1 \succ x_2 \succ x_3$ she is more likely to enjoy movie x_4 or x_5. This translates to predicting whether overall the multi-ranking $(x_1 \succ x_2 \succ x_3,\ x_4 \succ x_5)$ or the multi-ranking $(x_1 \succ x_2 \succ x_3,\ x_5 \succ x_4)$ is more probable. The most general form of multi-ranking is

$$(X_{1,1} \succ \ldots \succ X_{1,k_1},\ \ldots\ ,\ X_{\ell,1} \succ \ldots \succ X_{\ell,k_\ell}),$$

but in this paper we will only discuss multi-rankings of the form

$$(x_{i_{1,1}} \succ \ldots \succ x_{i_{1,k_1}},\ \ldots\ , x_{i_{\ell,1}} \succ \ldots \succ x_{i_{\ell,k_\ell}}). \tag{5}$$

For simplicity, we will loosely use the term "partial ranking" for both strict partial rankings and multirankings.

2.1 Rankings as sets of permutations

We identify a total ranking, such as (2), with the unique permutation $\sigma\colon [1,n] \to [1,n]$ that moves the index of the item ranked first into position n, the index of the item ranked second into position $n-1$, etc.. In other words,

$$x_{\sigma^{-1}(n)} \succ x_{\sigma^{-1}(n-1)} \succ \ldots \succ x_{\sigma^{-1}(1)}. \tag{6}$$

While at this point the reversal between ranks and mapping positions might seem like an unnecessary complication, it will simplify the algebra later in the paper.

Partial rankings and multirankings are identified with the set of all permutations that satisfy them. Thus, the interleaving partial ranking (3) corresponds to

$$A_{i_1,\ldots,i_k} = \{\ \sigma\ \mid\ \sigma(i_a) > \sigma(i_b) \quad \text{if} \quad a < b\ \},$$

while the top–k partial ranking (4) corresponds to

$$B_{i_1,\ldots,i_k} = \{\ \sigma\ \mid\ \sigma(i_j) = n-j+1,\ \ \forall j\ \}.$$

Multi-rankings, in general, correspond to the intersection of the sets of permutations associated with their constituents. In particular, the set corresponding to (5) is $C_{i_{1,1}\ldots i_{\ell,k_\ell}} = \bigcap_{j=1}^{\ell} A_{i_{j,1},\ldots,i_{j,k_j}}$.

2.2 Rankings as vectors in $\mathbb{C}[\mathbb{S}_n]$

One of the key ideas of the present paper is to identify each (total, partial, or multi–) ranking with a vector in a space called the group algebra of the symmetric group. Let us now clarify this statement.

If we define the product of one permutation σ_1 with another permutation σ_2 as their composition $\sigma_2 \circ \sigma_1$, then with respect to this operation the $n!$ possible permutations of $[1,n]$ form a group. This group is called the **symmetric group** of degree n, and is denoted $\mathbb{S}_n$.Since $\mathbb{S}_n$ is a **non-commutative** group, in general, $\sigma_1\sigma_2 \neq \sigma_2\sigma_1$. The identity permutation will be denoted e.

The **group algebra** of the symmetric group, denoted $\mathbb{C}[\mathbb{S}_n]$, is an $n!$–dimensional complex[1] vector space with canonical basis $\{\boldsymbol{e}_\sigma\}_{\sigma\in\mathbb{S}_n}$, equipped with a notion of vector/vector multiplication

[1]Throughout the paper we use vector spaces defined over the complex numbers. This is because several of the general results from representation theory are much easier to state over $\mathbb{C}$ than over $\mathbb{R}$. However, using the specific system of representations of $\mathbb{S}_n$ called Young's Orthogonal Representation will allow us to perform all actual computations over the reals.

inherited from the structure of the symmetric group. In particular, $\boldsymbol{e}_\sigma \boldsymbol{e}_{\sigma'} = \boldsymbol{e}_{\sigma\sigma'}$, and this is extended to the rest of the space by linearity. In other words, for general vectors $\boldsymbol{u} = \sum_{\sigma\in\mathbb{S}_n} u_\sigma \boldsymbol{e}_\sigma$ and $\boldsymbol{v} = \sum_{\sigma\in\mathbb{S}_n} v_\sigma \boldsymbol{e}_\sigma$,

$$\boldsymbol{u}\boldsymbol{v} = \sum_{\mu\in\mathbb{S}_n}\sum_{\nu\in\mathbb{S}_n} u_\mu v_\nu \boldsymbol{e}_\mu \boldsymbol{e}_\nu = \sum_{\mu,\nu\in\mathbb{S}_n} u_\mu v_\nu \boldsymbol{e}_{\mu\nu} = \sum_{\sigma\in\mathbb{S}_n} w_\sigma \boldsymbol{e}_\sigma, \tag{7}$$

where $w_\sigma = \sum_{\nu\in\mathbb{S}_n} u_{\sigma\nu^{-1}} v_\nu$. The inner product on $\mathbb{C}[\mathbb{S}_n]$ is the usual $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \sum_{\sigma\in\mathbb{S}_n} u_\sigma^* v_\sigma$, where * stands for complex conjugation. For future reference we note that as a function w is called the **convolution** of v with u, and denoted $u \star v$.

Throughout the paper, if f is a function on permutations, then $\boldsymbol{f}$ will be the corresponding group algebra vector $\sum_{\sigma\in\mathbb{S}_n} f(\sigma)\, \boldsymbol{e}_\sigma$. Similarly, if $U \subset \mathbb{S}_n$, then $\boldsymbol{U}$ will be $\sum_{\sigma\in U} \boldsymbol{e}_\sigma$. In particular, we represent the total ranking (2) by the basis vector $\boldsymbol{\sigma} \equiv \boldsymbol{e}_\sigma$, and the partial rankings (3), (4) and (5) by $\boldsymbol{A}_{i_1,\ldots,i_k} = \sum_{\sigma\in A_{i_1,\ldots,i_k}} \boldsymbol{e}_\sigma$, $\boldsymbol{B}_{i_1,\ldots,i_k} = \sum_{\sigma\in B_{i_1,\ldots,i_k}} \boldsymbol{e}_\sigma$, and $\boldsymbol{C}_{i_{1,1}\ldots i_{\ell,k_\ell}} = \sum_{\sigma\in C_{i_{1,1}\ldots i_{\ell,k_\ell}}} \boldsymbol{e}_\sigma$.

The significance of the multiplicative structure of $\mathbb{C}[\mathbb{S}_n]$ is that it allows us to express these partial ranking vectors as

$$\boldsymbol{A}_{i_1,\ldots,i_k} = \boldsymbol{\Pi}^n_k \, \boldsymbol{\mathbb{S}}_{n-k}\, \boldsymbol{\pi}_{i_1,\ldots,i_k}, \tag{8}$$

$$\boldsymbol{B}_{i_1,\ldots,i_k} = \boldsymbol{\mathbb{S}}_{n-k}\, \boldsymbol{\pi}_{i_1,\ldots,i_k}, \tag{9}$$

$$\boldsymbol{C}_{i_{1,1}\ldots i_{\ell,k_\ell}} = \boldsymbol{\Pi}^n_{m_\ell} \ldots \boldsymbol{\Pi}^{m_2}_{m_1} \, \boldsymbol{\mathbb{S}}_{n-k}\, \boldsymbol{\pi}_{i_{1,1},\ldots,i_{1,k_1},\ldots,i_{\ell,k_\ell}}, \tag{10}$$

where in the last equation $k = \sum_{i=1}^{\ell} k_i$, and $\pi_{i_1,\ldots,i_k}$, Π^ℓ_k and $\mathbb{S}_m$ are the following elementary sets of permutations.

- $\pi_{i_1,\ldots,i_k}$ is the **selector** permutation that maps $i_1,\ldots,i_k$ to positions $n, n-1, \ldots, n-k+1$. Where it maps all other numbers is presently irrelevant, so we leave it undefined.
- Π^ℓ_k is the set of **interleavings** of $[n-\ell+1,\, n-k]$ with $[n-k+1,\, n]$, i.e., the set of permutations that satisfy
$$\begin{aligned} \sigma(i) &= i && \text{if} \quad 1 \le i \le n-\ell, \\ \sigma(i) &< \sigma(j) && \text{if} \quad n-\ell+1 \le i < j \le n-k, \\ \sigma(i) &< \sigma(j) && \text{if} \quad n-k+1 \le i < j \le n. \end{aligned}$$
  This concept is the same as the **riffle shuffles** introduced in (Huang & Guestrin, 2009).
- $\mathbb{S}_m$ is the subgroup of permutations that only permute $[1, m]$ and leave $[m+1, n]$ fixed.

Intuitively, (8)–(10) describe the way that $A_{i_1,\ldots,i_k}$, $B_{i_1,\ldots,i_k}$ and $C_{i_{1,1},\ldots,i_{\ell,k_\ell}}$ can be "built up" through a sequence of operations. For example, $A_{i_1,\ldots,i_k}$ is the set of permutations that we get by performing the following operations: (1) first map $i_1 \mapsto n$, $i_2 \mapsto n-1$, etc. up to $i_k \mapsto n-k+1$; (2) then permute all the "unoccupied" positions $[1, n-k]$ in all possible ways; (3) finally interleave the numbers occupying $[n-k+1, n]$ with whatever is in positions $[1, n-k]$, while preserving their relative ordering (see Figure 1). We will find that such factorizations are key to unraveling the spectral structure of partial rankings and for efficiently computing kernels. Before describing this, however, we need to develop some general results regarding kernels on permutations.

3 Kernels on the symmetric group

In studying kernels on $\mathbb{R}^d$ a lot of attention is focused on translation invariance (or stationarity) in the sense of $k(x, x') = k(x+z, x'+z)$ for all $z \in \mathbb{R}^d$. When looking at kernels on non-commutative groups we find a similar notion of invariance, but now **right–invariance**, meaning $K(\sigma\pi, \sigma'\pi) = K(\sigma, \sigma')$ for all σ, σ' and π, and **left–invariance**, meaning $K(\pi\sigma, \pi\sigma') = K(\sigma, \sigma')$ for all σ, σ' and π, are distinct concepts.

Right–invariance is a natural requirement in ranking, since it captures the notion that if we take any permutation π, then the similarity between two rankings should not change if we relabel the underlying items $x_1, \ldots, x_n$ as $x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(n)}$. A right–invariant kernel on $\mathbb{S}_n$ can always be expressed as $K(\sigma', \sigma) = \kappa(\sigma'\sigma^{-1})$ for some $\kappa\colon \mathbb{S}_n \to \mathbb{R}$. A function $\kappa\colon \mathbb{S}_n \to \mathbb{C}$ is said to be a **positive (semi–)definite function** on $\mathbb{S}_n$ if the corresponding $K(\sigma', \sigma) = \kappa(\sigma'\sigma^{-1})$ is a positive (semi–)definite kernel. Note that in this case $\kappa(\sigma) = K(\sigma, e) = K(e, \sigma) = \kappa(\sigma^{-1})$.

Left–invariance is not desirable in ranking, since it would imply that the value of the kernel between a pair of rankings that rank a specific item in positions one and two respectively should be the same as if they ranked it in positions, say, one and twenty (assuming that the ranking of

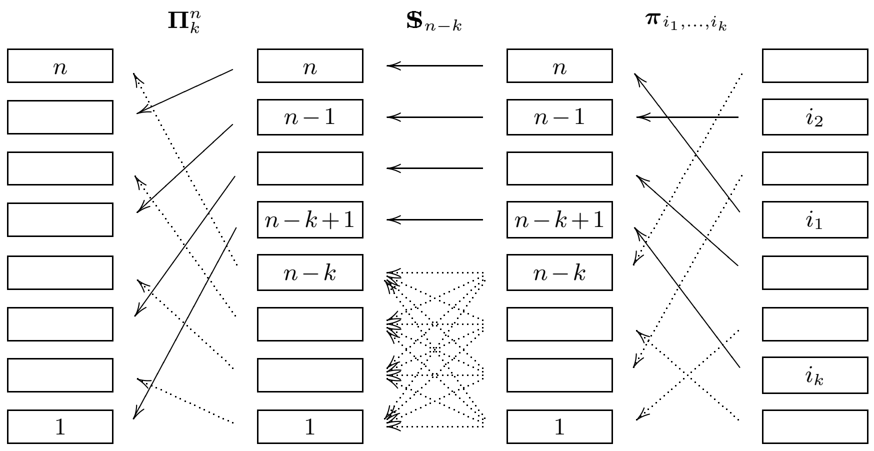


Figure 1: An illustration of the factorization $\boldsymbol{A}_{i_1,\ldots,i_k} = \mathbf{\Pi}_k^n\, \mathbb{S}_{n-k}\, \boldsymbol{\pi}_{i_1,\ldots,i_k}$. First, $\boldsymbol{\pi}_{i_1,\ldots,i_k}$ moves $(i_1, i_2, \ldots, i_k)$ into positions $(n, n–1, \ldots, n–k+1)$. Next, $\mathbb{S}_{n-k}$ permutes positions $1, 2, \ldots, n–k$ in all possible ways. Finally, $\mathbf{\Pi}_k^n$ maps $n, n-1, \ldots, n-k+1$ to $1, 2, \ldots, n$ in all possible ways that respects their relative ordering.

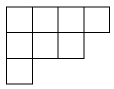
other shared items does not change). However, in permutation learning, for example when we are trying to learn the optimal assignment between aircraft and routes, left–invariance might be just as natural a requirement as right–invariance. A kernel which is both left– and right–invariant is called **bi–invariant**. In the bi–invariant case κ is a class function, which means that $\kappa(\tau^{-1}\sigma\tau) = \kappa(\sigma)$ for all τ and σ.

On $\mathbb{R}^d$ Bochner's theorem tells us that positive definite functions are characterized by the fact that their Fourier transform is pointwise positive. On finite non-commutative groups we have a similar result, except that now the ordinary **Fourier transform** must be replaced by its non-commutative generalization. In the case of the symmetric group this takes the form

$$\widehat{\kappa}(\lambda) = \sum_{\sigma \in \mathbb{S}_n} \kappa(\sigma)\, \rho_\lambda(\sigma) \qquad \lambda \vdash n, \tag{11}$$

and differs from the usual commutative Fourier transforms in two key respects:

1. Instead of frequency, the individual Fourier components are now indexed by **integer partitions** of n, by which we mean a sequence $\lambda = (\lambda_1, \ldots, \lambda_k)$ of weakly decreasing positive integers summing to n (i.e., $\lambda_1, \ldots, \lambda_k \in \mathbb{Z},\ \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k,\ \sum_{i=1}^k \lambda_i = n$). A convenient graphical representation for partitions is provided by so–called **Young diagrams**, consisting of $\lambda_1, \ldots, \lambda_k$ boxes placed in consecutive rows. For example,

   is the Young diagram of $\lambda = (4, 3, 1)$. The notation $\lambda \vdash n$ just means that λ is an integer partition of n.

2. Instead of expressions like $e^{ikx/2\pi}$, the weighting factors $\rho_\lambda(\sigma)$ appearing in (11) are complex–valued *matrices*, specifically elements of the **irreducible representation** (or **irrep**) of $\mathbb{S}_n$ corresponding to λ. For the definition of what this means and how to construct such matrices the reader is referred to the supplementary document. For now what is important to note is that the Fourier transform $(\widehat{\kappa}(\lambda))_{\lambda \vdash n}$ is a sequence of matrices of different sizes. We use d_λ to denote the dimensionality of the irrep indexed by λ, hence $\widehat{\kappa}(\lambda) \in \mathbb{C}^{d_\lambda \times d_\lambda}$. We also assume throughout the paper that the irreps are unitary, and hence $\rho_\lambda(\sigma^{-1}) = \rho_\lambda(\sigma)^\dagger$.

Aside from these two perhaps surprising features, (11) shares many important properties with ordinary Fourier transformation. In particular, with respect to the appropriate matrix norms, $\kappa \mapsto \widehat{\kappa}$ is a unitary transformation, the inverse transform being

$$\kappa(\sigma) = \frac{1}{n!} \sum_{\lambda \vdash n} d_\lambda\, \mathrm{tr}\big[\widehat{\kappa}(\lambda)\, \rho_\lambda(\sigma^{-1})\big],$$

and we also have analogs of the convolution and correlation theorems. By unitarity, the total size of the Fourier matrices must be the same as the size of the original function, that is, $\sum_{\lambda \vdash n} d_\lambda^2 = n!$. Bochner's theorem generalizes as follows. We do not claim that this result is novel to mathematics, so the attributions only serve to indicate its first appearance in the machine learning literature.

Proposition 1 *(Kondor, 2008, Thm. 4.5.4)(Fukumizu et al., 2009, Thm. 11) A function $\kappa \colon \mathbb{S}_n \to \mathbb{C}$ is positive (semi–)definite if and only if each of the $\widehat{\kappa}(\lambda)$ matrices in (11) are positive (semi–)definite.*

Much of the following discussion will revolve around moving back and forth between three different views of the same objects: (1) the function view ($f \colon \mathbb{S}_n \to \mathbb{C}$); (2) the group algebra vector view ($\boldsymbol{f} \in \mathbb{C}[\mathbb{S}_n]$); and (3) the Fourier transform view $\widehat{f}$. Therefore, we will use the symbol $\widehat{\ }$ very liberally: if it is placed over a group algebra element, it will denote the Fourier transform of the corresponding function, and if it is placed over a group element or set of group elements, it will denote the Fourier transform of the corresponding indicator function.

3.1 Diffusion kernels

While Proposition 1 gives a general characterization of right–invariant kernels on the symmetric group, it does not give us much guidance as to what specific kernel we should use for ranking. One way to derive a kernel would be to induce it from a right–invariant metric on $\mathbb{S}_n$, for example, one of the metrics described in (Diaconis, 1988). However, if we then wanted to perform operations on the kernel in Fourier space, as we will do in the next section, we would at some point have to perform an explicit Fourier transform, which is very expensive.[2]

To avoid this problem, in the present paper we use **diffusion kernels** (Kondor & Lafferty, 2002), which have strong connections to spectral theory. Recall that to define a diffusion kernel we start with an adjacency relation $\sim$, which induces a graph. The corresponding **graph Laplacian** is the matrix

$$\Delta_{\sigma',\sigma} = \begin{cases} 1 & \text{if } \sigma' \sim \sigma \\ -d_\sigma & \text{if } \sigma' = \sigma \\ 0 & \text{otherwise,} \end{cases}$$

and the diffusion kernel is $K(\sigma', \sigma) = [e^{\beta\Delta}]_{\sigma',\sigma}$ for some diffusion parameter $\beta \in \mathbb{R}$, where $e^{\beta\Delta}$ is the matrix exponential $\lim_{m\to\infty}(I + \beta\Delta/m)^m$.

A diffusion kernel on a group is right–invariant if and only if $\sim$ is a right–invariant relation, which is equivalent to saying that $\sigma' \sim \sigma \iff \sigma'\sigma^{-1} \in Q$ for some $Q \subset \mathbb{S}_n$. To ensure symmetry, Q must also be symmetric in the sense that $\pi \in Q \iff \pi^{-1} \in Q$. The set Q can be interpreted as the "elementary steps" $\sigma \mapsto \pi\sigma$ that one can take from any σ to reach its neighbors. In fact, for the graph to be connected, Q must be a generating set of $\mathbb{S}_n$. In this case the adjacency graph is known as the **Cayley graph** induced by Q.

The natural generalization of the above to weighted graphs involves setting $\Delta_{\sigma',\sigma} = q(\sigma'\sigma^{-1})$, where q is a function that must satisfy $q(\pi^{-1}) = q(\pi)$ and $\sum_{\pi \in \mathbb{S}_n} q(\pi) = 0$. The following result characterizes all possible right–invariant diffusion kernels on $\mathbb{S}_n$ and shows how to compute them in Fourier space.

Proposition 2 *If $\Delta_{\sigma',\sigma} = q(\sigma'\sigma^{-1})$ and q satisfies the above two conditions, then the diffusion kernel $K(\sigma', \sigma) = [e^{\beta\Delta}]_{\sigma',\sigma}$ is right–invariant, and $K(\sigma', \sigma) = \kappa(\sigma'\sigma^{-1})$, where $\widehat{\kappa}(\lambda) = \exp(\beta\,\widehat{q}(\lambda))$ for each $\lambda \vdash n$.*

Proposition 2 tells us that instead of exponentiating the $n!$-dimensional matrix Δ (at a cost of $O(n!^3)$), it is more efficient to compute the diffusion kernel in Fourier space, where we only have to exponentiate individual Fourier matrices (at a total cost of $O(\sum_{\lambda \vdash n} d_\lambda^3)$). The question remains as to what adjacency relation $\sim$ is appropriate for ranking problems.

A **transposition** (i, j) is a permutation that swaps i with j and leaves everything else fixed. Since transpositions are in many ways the simplest non–trivial permutations, they seem like a natural choice for Q. However, because transpositions form a **conjugacy class** (i.e., $\pi \in Q \Rightarrow \mu^{-1}\pi\mu \in$

[2] Naively, the complexity of computing a Fourier transform on $\mathbb{S}_n$ is $O(n!^2)$. Fast Fourier transforms, such as Clausen's FFT (Clausen, 1989), can bring this complexity down to $O(n^3 n!)$ or even $O(n^2 n!)$ (Maslen, 1998), but the the $n!$ factor still makes using such kernels infeasible for large n, unless we can derive the form of their Fourier transform analytically.

$Q\ \ \forall\mu$), the resulting K_{transp} will be bi–invariant, so, for the reasons we discussed above, this type of kernel is more appropriate for permutation learning than for ranking. From a purely algebraic point of view, however, K_{transp} is a canonical choice, showing some similarities with the Gaussian RBF kernel on $\mathbb{R}^d$. In fact, using a result on page 40 of (Diaconis, 1988), the Fourier transform of this kernel can be derived in closed form.

Proposition 3 *If $K_{\text{transp}}\colon \mathbb{S}_n\times\mathbb{S}_n\to\mathbb{R}$ is the diffusion kernel induced by the class of transpositions, then*

$$\widehat{\kappa}_{transp}(\lambda)=\exp\left(-\beta\tbinom{n}{2}(1-r(\lambda))\right)I_{d_\lambda},\tag{12}$$

where $r(\lambda)=\binom{n}{2}^{-1}\sum_i\binom{\lambda_i}{2}-\binom{\lambda'_i}{2}$, λ' is the transpose of the partition λ, and I_{d_λ} denotes the d_λ–dimensional identity matrix.

For ranking a better choice of kernel is K_{adj}, the diffusion kernel induced by the subset of **adjacent transpositions** $\{\tau_i := (i,i+1)\}_{i=1}^{n-1}$. This is the kernel that we use in our experiments. Unfortunately, we have no closed form expression for $\widehat{\kappa}_{\text{adj}}$. While $\widehat{q}_{\text{adj}}$ is relatively easy to compute by a direct Fourier transform since there are only $n-1$ adjacent transpositions, for large n exponentiating these matrices might be problematic. Note, however, that this is a one–time computation.

A natural variant on K_{adj} is to give different adjacent transpositions different weights. In learning from the top–k rankings returned by search engines, for example, we could give less weight to adjacent transpositions between the first few rankings than those lower down, reflecting the fact that whether something is ranked first or second is much more important than whether it is ranked, say 15th or 16th. While we do not pursue this direction further in the rest of the paper, we note that our computational results would hold for this variant of K_{adj} equally well.

4 Computing kernels in Fourier space

We started our discussion by stating that in typical ranking problems individual examples are not total rankings, but partial rankings of k out of n items. The easiest and most general way to extend the kernels of the previous section to this setting is to define the kernel between a pair of partial rankings R and R' as

$$K(R',R)=\frac{1}{|\,R'\,|\,|\,R\,|}\sum_{\sigma'\in R'}\sum_{\sigma\in R}K(\sigma',\sigma),\tag{13}$$

where, overloading notation somewhat, R and R' double as the set of all permutations satisfying the two partial rankings.

The advantage of such an averaged kernel is its flexibility: R and R' can be any type of partial rankings (interleaving, top–k, or some multiranking combining elements of the former), and even the orders (k and k') of R and R' may be different. This allows kernel–based ranking algorithms to aggregate information from a diverse array of inputs.

The problem with (13) is that it appears to be very expensive to compute: naively, its complexity is $O((n-k)!(n-k')!)$. The rest of the paper addresses this computational issue, showing that (13) can be computed efficiently in Fourier space. We begin with the following two general lemmas that both follow from the convolution theorem, which states that $\widehat{u\star v}(\lambda)=\widehat{u}(\lambda)\,\widehat{v}(\lambda)$.

Lemma 4 *If $\boldsymbol{u},\boldsymbol{v}\in\mathbb{C}[\mathbb{S}_n]$ and $\boldsymbol{w}=\boldsymbol{uv}$, then $\widehat{w}(\lambda)=\widehat{u}(\lambda)\,\widehat{v}(\lambda)$ for each $\lambda\vdash n$.*

Lemma 5 *For any $\boldsymbol{u},\boldsymbol{v}\in\mathbb{C}[\mathbb{S}_n]$,*

$$\langle\boldsymbol{u},\boldsymbol{v}\rangle=\frac{1}{n!}\sum_{\lambda\vdash n}d_\lambda\operatorname{tr}\big[\widehat{u}(\lambda)^\dagger\,\widehat{v}(\lambda)\big],\tag{14}$$

where † denotes the Hermitian conjugate.

Using these lemmas it is easy to derive the Fourier form of our kernel.

Proposition 6 *If K is right–invariant with $K(\sigma',\sigma)=\kappa(\sigma'\sigma^{-1})$, then (13) can be expressed as*

$$K(R',R)=\frac{1}{n!|R'||R|}\sum_{\lambda\vdash n}d_\lambda\operatorname{tr}\big[\,\widehat{R'}(\lambda)^\dagger\,\widehat{\kappa}(\lambda)\,\widehat{R}(\lambda)\,\big].\tag{15}$$

Proof. By right–invariance

$$\sum_{\sigma'\in R'}\sum_{\sigma\in R}K(\sigma',\sigma)=\sum_{\sigma'\in\mathbb{S}_n}\sum_{\sigma\in\mathbb{S}_n}R'(\sigma')\,K(\sigma',\sigma)\,R(\sigma)=\sum_{\sigma'\in\mathbb{S}_n}\sum_{\sigma\in\mathbb{S}_n}R'(\sigma')\,\kappa(\sigma',\sigma^{-1})\,R(\sigma)=\langle \boldsymbol{R}',\boldsymbol{\kappa}\boldsymbol{R}\rangle\,,$$

where, as before, $R(\sigma)$ is the indicator function of the set R and $\boldsymbol{R}=\sum_{\sigma\in R}\boldsymbol{e}_\sigma$ is the corresponding vector in the group algebra. Hence,

$$K(R',R)=|\,R'\,|^{-1}\,|\,R\,|^{-1}\,\langle \boldsymbol{R}',\boldsymbol{\kappa}\boldsymbol{R}\rangle\,, \tag{16}$$

and (15) follows by Lemmas 4 and 5. ■

By itself, (15) does not make our kernel any easier to evaluate, since the combined size of the matrices appearing under the sum is still $O(n!)$. The key is to additionally exploit the sparsity of $\widehat{R}$ and $\widehat{R}'$. We derive the structure of these Fourier transforms in two stages: first describing their matrix–level sparsity, and then examining the row/column–level sparsity of their individual matrix components.

4.1 Matrix level sparsity

We say that a vector $\boldsymbol{v}\in\mathbb{C}[\mathbb{S}_n]$ (equivalently, the function v or the Fourier transform $\widehat{v}$) is **bandlimited** to some subset Λ of $\{\lambda\vdash n\}$ if the only non-zero components of $\widehat{v}$ are $\{\widehat{v}(\lambda)\}_{\lambda\in\Lambda}$. Several recent papers have used bandlimited functions to approximate distributions over permutations, most notably in the context of multi–object tracking (Kondor et al., 2007; Huang et al., 2009; Huang & Guestrin, 2009).

In the present paper bandlimitedness is not an approximation, but an inherent feature of our problem. Typically, this is a sign of invariance to a subgroup, in our case, invariance to the ranking position of the $n-k$ items not involved in the partial ranking at hand. The key to unraveling this structure are the factorizations (8)–(10), and specifically, the $\mathbf{S}_{n-k}$ factors appearing in them.

Proposition 7 *The group algebra element $\mathbf{S}_{n-k}\in\mathbb{C}[\mathbb{S}_n]$ is bandlimited to the set Λ^n_{n-k} defined $\Lambda^n_{n-k}=\{\,\lambda\vdash n\mid \lambda_1\geq n-k\,\}$.*

In terms of Young diagrams, Λ^n_{n-k} is the set of diagrams of n boxes with at least $n-k$ boxes in their first row. For example, $\Lambda^{10}_7=\{(10),(9,1),(8,2),(8,1,1),(7,3),(7,2,1),\,(7,1,1,1)\}$, so the Fourier transform of $\mathbf{S}_7$ in $\mathbb{C}[\mathbb{S}_{10}]$ has just 7 non–zero components. An immediate consequence of Lemma 4 is that if $\boldsymbol{u}$ is Λ_u–bandlimited and $\boldsymbol{v}$ is Λ_v–bandlimited, then $\boldsymbol{uv}$ will be $\Lambda_u\cap\Lambda_v$–bandlimited. Thus, the k'th order partial ranking vectors (8)–(10) will all inherit the bandlimited structure of $\mathbf{S}_{n-k}$, and the summation in (15) need only extend over $\Lambda^n_{n-\min(k,k')}$, giving us the following result.

Proposition 8 *If R and R' are a pair of partial rankings of orders k and k', respectively, then the kernel (15) can be written as*

$$K(R',R)=\frac{1}{n!|R'||R|}\sum_{\lambda\in\Lambda^n_{n-\min(k,k')}}d_\lambda\,\mathrm{tr}\big[\,\widehat{R}'(\lambda)^\dagger\,\widehat{\kappa}(\lambda)\,\widehat{R}(\lambda)\,\big]. \tag{17}$$

In particular, given $\widehat{R}$, $\widehat{R}'$ and $\widehat{\kappa}$, the kernel can be computed in $\sum_{\lambda\in\Lambda^n_{n-\min(k,k')}}2\,d^3_\lambda$ operations.[3]

Several authors (Diaconis, 1988; Huang et al., 2009; Kondor et al., 2007) discuss that one possible interpretation of the Fourier matrices is that they capture detail at different scales: given a distribution p on $\mathbb{S}_n$, $(\widehat{p}(\lambda))_{\lambda\in\Lambda^n_{n-k}}$ is exactly the information needed to reconstruct p up to its k'th order marginals. In this respect it is not surprising that (17) should involve exactly these Fourier matrices, and our result fits nicely in the general theory of spectral analysis on permutations.

Unfortunately, in general, the dimensionality of the largest matrices in $(R(\lambda))_{\lambda\in\Lambda^n_{n-k}}$ grows with $O(n^k)$, so while Proposition 8 greatly reduces the number of Fourier matrices that need to be summed over, for n greater than about a dozen evaluating (17) is still problematic (see Table 1). This motivates a finer grained analysis, also taking the internal structure of the Fourier matrices into account.

[3] Throughout this paper, in line with the literature, by a single operation we mean multiplying two scalars and adding them to a third. We assume that multiplication by constants and copying information is free.

$$\widehat{\mathbb{S}}_3((5)) = (20) \qquad \widehat{\mathbb{S}}_3((4,1)) = \begin{pmatrix} 20 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\widehat{\mathbb{S}}_3((3,2)) = \begin{pmatrix} 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad \widehat{\mathbb{S}}_3((3,1,1)) = \begin{pmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 2: The non–zero Fourier matrices of $\mathbf{S}_3 \in \mathbb{C}[\mathbb{S}_5]$, relevant to any binary ranking out of 5 items. Any $\mathbf{S}_{n-2} \in \mathbb{C}[\mathbb{S}_n]$ would have the same structure except that for $n>5$ the matrices would be larger.

4.2 Row/column level sparsity

Ultimately, the sparsity of $\widehat{\mathbb{S}}_{n-k}$ is a consequence of the way that the irreps of $\mathbb{S}_n$ reduce into a direct sum of irreps of $\mathbb{S}_{n-k}$ on restriction to this subgroup. Specifically, a result called **Young's branching rule** tells us that if $\tau \in \mathbb{S}_{n-1}$, then

$$\rho_\lambda(\tau) = T^{-1} \Big[\bigoplus_{\lambda^- \in \lambda\downarrow_{n-1}} \rho_{\lambda^-}(\tau) \Big] \, T,$$

where $\lambda\downarrow_{n-1}$ is the set of all partitions of $n-1$ that we can get from λ by removing a single box, ρ_{λ^-} is the irrep of $\mathbb{S}_{n-1}$ indexed by λ^-, and T is a unitary matrix that depends on exactly what system of irreps we use. For simplicity, in the following we will assume that all irreps are from **Young's Orthogonal Representation (YOR)** (see supplement), in which case the T matrices may be dropped because they are always equal to the identity.

Now if we establish a partial order on partitions according to which $\lambda \geq \lambda'$ if and only λ' is a subdiagram of λ, recursively applying the branching rule down to $n-k$ gives that for $\lambda' \vdash n-k$ and $\tau \in \mathbb{S}_{n-k}$, $\rho_{\lambda'}(\tau)$ will be featured in the decomposition of $\rho_\lambda(\tau)$ if and only if $\lambda \geq \lambda'$. In particular, $\rho_{(n-k)}(\tau)$ appears whenever $\lambda \geq (n-k)$, which is equivalent to $\lambda \in \Lambda^n_{n-k}$. Proposition 7 follows by considering that $\sum_{\tau \in \mathbb{S}_{n-m}} \rho_{\lambda'}(\tau) = 0$ for all $\lambda' \vdash n-k$ except for $\lambda' = (n-k)$ (see the proof of Proposition 9 in the supplement).

The precise structure of $\widehat{\mathbb{S}}_{n-k}$ can be uncovered by repeating this analysis on the level of individual matrix entries. First recall that the individual rows/columns of Fourier matrices such as $\widehat{\mathbb{S}}_{n-k}(\lambda)$ are indexed by the **standard Young tableaux (SYT)** of shape λ, such as

$$t = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 4 & 5 & 7 \\ \hline 3 & 6 & 9 \\ \cline{1-3} 8 \\ \cline{1-1} \end{array} \;,$$

which is a SYT of shape $\lambda = (5,3,1)$. Just like partitions, SYT also have a natural inclusion order, for example if $t' = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 4 & 5 & 7 \\ \hline 3 & 6 \\ \cline{1-2} \end{array}$, then $t \geq t'$ because t can be constructed from t' by adding $\boxed{8}$ and $\boxed{9}$. A convenient shorthand for SYT are Yamanouchi symbols, in particular, $[\ldots]_m$ denotes $\begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & \ldots & m \\ \hline \end{array}$. Using these notations, the branching rule in YOR can be made more explicit as

$$[\rho_\lambda(\tau)]_{t,t'} = \begin{cases} [\rho_{\lambda^-}(\tau)]_{u,u'} & \text{if } u \leq t,\ u' \leq t', \text{ and } u \text{ and } u' \text{ are both of shape } \lambda^- \vdash n-1, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

Using this result we can derive the exact form of $\widehat{\mathbb{S}}_{n-k}$.

Proposition 9 *For $0 \leq k \leq n-1$ the Fourier transform of $\mathbf{S}_{n-k} \in \mathbb{C}[\mathbb{S}_n]$ (in YOR) is of the form*

$$[\widehat{\mathbb{S}}_{n-k}(\lambda)]_{t,t'} = \begin{cases} (n-k)! & \text{if } t = t' \text{ and } t \geq [\ldots]_{n-k}\,, \\ 0 & \text{otherwise.} \end{cases}$$

In simple terms $t \geq [\ldots]_{n-k}$ means that the first $m = n-k$ boxes in the first row of t must be $\boxed{1}\boxed{2}\boxed{3}\boxed{\ldots}\boxed{m}$, which is a very severe restriction. Thus, not only is $\mathbf{S}_{n-k}$ bandlimited to just a few matrix components, even those components will have few non–zero entries. As an example, Figure 2 shows the actual Fourier matrices of $\mathbf{S}_3 \in \mathbb{C}[\mathbb{S}_5]$. Taking advantage of this form of sparsity is more difficult than just taking advantage of bandlimitedness because in expressions such as (8) $\mathbf{S}_{n-k}$ is multiplied from both the left and the right. To overcome this problem, we need to introduce adjoints.

Proposition 10 *For any $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \mathbb{C}[\mathbb{S}_n]$, there is a $\boldsymbol{v}^\dagger \in \mathbb{C}[\mathbb{S}_n]$ called the **adjoint** of $\boldsymbol{v}$, such that*

$$\langle \boldsymbol{u}, \boldsymbol{v}\boldsymbol{w} \rangle = \langle \boldsymbol{v}^\dagger \boldsymbol{u}, \boldsymbol{w} \rangle \quad \textit{and} \quad \langle \boldsymbol{u}, \boldsymbol{w}\boldsymbol{v} \rangle = \langle \boldsymbol{u}\boldsymbol{v}^\dagger, \boldsymbol{w} \rangle.$$

Moreover, $v^\dagger(\sigma) = v(\sigma^{-1})^*$ for all $\sigma \in \mathbb{S}_n$, and in YOR $\widehat{(v^\dagger)}(\lambda) = \widehat{v}(\lambda)^\dagger$ for all $\lambda \vdash n$.

Substituting a pair of interleaving rankings $\boldsymbol{A}_{i_1,\ldots,i_k}$ and $\boldsymbol{A}_{i'_1,\ldots,i'_k}$ for $\boldsymbol{R}$ and $\boldsymbol{R}'$ in (16) we can now rearrange the inner product as

$$\begin{aligned}
&\big\langle \boldsymbol{A}_{i'_1,\ldots,i'_k}, \boldsymbol{\kappa}\, \boldsymbol{A}_{i_1,\ldots,i_k} \big\rangle = \\
&\big\langle \boldsymbol{\Pi}^n_k\, \mathbf{S}_{n-k}\, \boldsymbol{\pi}_{i'_1,\ldots,i'_k},\, \boldsymbol{\kappa}\, \boldsymbol{\Pi}^n_k\, \mathbf{S}_{n-k}\, \boldsymbol{\pi}_{i_1,\ldots,i_k} \big\rangle = \\
&\big\langle \mathbf{S}_{n-k}\, \boldsymbol{\pi}_{i'_1,\ldots,i'_k}\, \boldsymbol{\pi}^\dagger_{i_1,\ldots,i_k}\, \mathbf{S}^\dagger_{n-k},\, \boldsymbol{\Pi}^{n\,\dagger}_k \boldsymbol{\kappa}\, \boldsymbol{\Pi}^n_k \big\rangle = \\
&\big\langle \mathbf{S}_{n-k}\, \boldsymbol{\pi}_{i'_1,\ldots,i'_k}\, \boldsymbol{\pi}^{-1}_{i_1,\ldots,i_k}\, \mathbf{S}_{n-k},\, \boldsymbol{\Pi}^{n\,\dagger}_k \boldsymbol{\kappa}\, \boldsymbol{\Pi}^n_k \big\rangle,
\end{aligned}$$

where the last line follows from $\mathbf{S}^\dagger_{n-k} = \mathbf{S}_{n-k}$ and $\boldsymbol{\pi}^\dagger_{i_1,\ldots,i_k} = \boldsymbol{\pi}^{-1}_{i_1,\ldots,i_k}$. Now the first argument of the last inner product contains an expression sandwiched between two $\mathbf{S}_{n-k}$'s, which has the effect of zeroing out all rows and columns indexed by $t \not\geq [\ldots]_{n-k}$. This dramatically reduces the effective size of the Fourier matrices that we need to multiply together to compute the kernel. Introducing the notation $[M]_{\geq[\ldots]_{n-k}}$ for the submatrix of the Fourier matrix indexed by rows/columns indexed by SYT descended from $[\ldots]_{n-k}$, we have the following result.

Proposition 11 *If $R = \boldsymbol{A}_{i_1,i_2,\ldots,i_k}$ and $R' = \boldsymbol{A}_{i'_1,i'_2,\ldots,i'_k}$, then the kernel (13) can be expressed as*

$$K(R, R') = \frac{(n-k)!^2}{n!\,|R'|\,|R|} \sum_{\lambda \in \Lambda^n_{n-k}} d_\lambda\, \big[\widehat{\Omega}(\lambda)\big]_{\geq[\ldots]_{n-k}} \odot \big[\widehat{\overline{\kappa}}(\lambda)\big]_{\geq[\ldots]_{n-k}}, \tag{19}$$

where $\boldsymbol{\Omega} = \boldsymbol{\pi}_{i'_1,\ldots,i'_k}\, \boldsymbol{\pi}^{-1}_{i_1,\ldots,i_k}$, $\overline{\boldsymbol{\kappa}} = \boldsymbol{\Pi}^{n\,\dagger}_k \boldsymbol{\kappa}\, \boldsymbol{\Pi}^n_k$ and $\odot$ denotes the matrix inner product $A \odot B = \sum_{i,j} A_{i,j} B_{i,j}$.

The restricted matrices appearing in (19) are much smaller than the matrices that we had to multiply together in (15) (see Table 1), and what is particularly attractive is that their size is *independent of* n. To be fair, (19) also requires computing $[\widehat{\Omega}(\lambda)]_{\geq[\ldots]_{n-k}}$. The next section explains how to do that efficiently, with a complexity that does not grow with n, either.

Clearly, formulae similar to (19) also hold for the more general case $k \neq k'$, as well as for other types of partial rankings. For example, if R and R' are both top–k rankings, then all that we need to change is to set $\overline{\boldsymbol{\kappa}} = \boldsymbol{\kappa}$. If R is an interleaving ranking, but R' is a top-k ranking, then $\overline{\boldsymbol{\kappa}} = \boldsymbol{\kappa}\boldsymbol{\Pi}^n_k$, and so on.

5 Complexity

A function $f\colon \mathbb{S}_n \to \mathbb{C}$ is called **right** $\mathbb{S}_{n-k}$**–invariant** if $f(\sigma\tau) = f(\sigma)$ for all $\tau \in \mathbb{S}_{n-k}$. Clearly, the space spanned by these functions has dimension $n!/(n-k)!$. In (Kondor et al., 2009) it was argued that such functions are bandlimited to $\{\widehat{f}(\lambda)\}_{\lambda \in \Lambda^n_{n-k}}$ and that their Fourier transforms fully occupy at least one column in each of these matrices. Therefore, $\sum_{\lambda \in \Lambda^n_{n-k}} d_\lambda \leq n!/(n-k)!$, and thus, even if we assume that $\widehat{\kappa}(\lambda)$, $\widehat{R}(\lambda)$ and $\widehat{R}'(\lambda)$ have all been pre-computed, the complexity of computing (17) is

$$\sum_{\lambda \in \Lambda^n_{n-k}} 2 d^3_\lambda \leq 2(n!/(n-k)!)^3 = O(n^{3k})$$

for fixed $k = k'$. See Table 1 for the exact operation count for $n = 20$ and some small values of k.

In contrast, denoting the set of all SYT of shape λ by $\mathcal{T}^\lambda$, and denoting its subset of SYT descended from $[\ldots]_{n-k}$ by $\mathcal{T}^\lambda_{n-k}$, computing (19) only requires $\sum_{\lambda \in \Lambda^n_{n-k}} (\mathcal{T}^\lambda_{n-k})^2$ operations. Now

k	2	3	4	5	6	7
c_0	$2.0\cdot 10^7$	$1.8\cdot 10^{10}$	$7.0\cdot 10^{12}$	$4.2\cdot 10^{15}$	$2.6\cdot 10^{17}$	$2.6\cdot 10^{19}$
c_1	7	34	209	1,546	13,327	130,922
c_2	1068	$8,7\cdot 10^4$	$7.7\cdot 10^6$	$7.9\cdot 10^8$	$9.2\cdot 10^{10}$	$1.2\cdot 10^{13}$
$(2k)^{2k+2}$	4096	$1.7\cdot 10^6$	$1.0\cdot 10^9$	$1.0\cdot 10^{12}$	$1.3\cdot 10^{15}$	$2.2\cdot 10^{18}$

Table 1: Comparison of the cost of computing the kernel with different methods. Here c_0 is the cost of computing (17) using naive matrix multiplication for $n=20$. In contrast, c_1 is the cost of computing (19) given the $\widehat{\Omega}(\lambda)$ and $\widehat{\kappa}(\lambda)$ Fourier matrices (irrespective of n). The maximum number of operations required to compute $[\widehat{\Omega}(\lambda)]_{\geq[\ldots]_{n-k}}$ as an input to (19) is c_2. Finally, the last row is our (loose) upper bound on c_2.

assuming that $n\geq 2k$, imagine that we construct each $t\in\bigcup_{\lambda\in\Lambda^n_{n-k}}\mathcal{T}^\lambda_{n-k}$ in two stages: first deciding which of the numbers $[n-k+1,n]$ should appear in row one, and then placing the remaining, say ℓ, numbers in subsequent rows. As a result of this placement, rows two and higher will form a diagram λ' of their own, and the number of ways that λ' can be filled with the ℓ numbers is the same as the number of ways that it could be filled with $[1,\ell]$, i.e., $|\mathcal{T}^{\lambda'}|$. In summary, the total number of entries in the matrices appearing in (19) is

$$\sum_{\ell=0}^{k}\binom{k}{\ell}^2\sum_{\lambda'\vdash\ell}|\mathcal{T}^{\lambda'}|^2. \qquad (20)$$

By the unitarity of the Fourier transform, we know that the total size of the Fourier matrices must be the same as the size of the group, so $\sum_{\lambda'\vdash\ell}|\mathcal{T}^{\lambda'}|^2=\ell!$, giving a complexity of $\sum_{\ell=0}^{k}\binom{k}{\ell}^2\ell!=O(k^{2k+1})$ for computing all the matrix inner products in (19) (see "c_1" in Table 1).

It remains to quantify the complexity of computing $[\widehat{\Omega}(\lambda)]_{\geq[\ldots]_{n-k}}$. This hinges on the special structure of YOR, namely that if τ is an adjacent transposition, then $\rho_\lambda(\tau)$ has only two non–zero elements in each row (or column) and therefore for any $M\in\mathbb{C}^{d_\lambda\times d_\lambda}$ the product $\rho_\lambda(\tau)M$ takes only $2d_\lambda^2$ operations to compute. By a bubblesort type procedure (see, e.g., (Kondor et al., 2009) or the original $\mathbb{S}_n$ FFT paper (Clausen, 1989)) any permutation can be decomposed into a product of at most $\binom{n}{2}$ transpositions, so if $\sigma=\pi_{i'_1,\ldots,i'_k}\pi^{-1}_{i_1,\ldots,i_k}$, then $\widehat{\sigma}(\lambda)$ can be computed in $n(n-1)d_\lambda^2$ operations. Since, as we have seen, $d_\lambda=O(n^k)$, computing $\widehat{\Omega}(\lambda)$ this way would make the total complexity of kernel evaluations $O(n^{2k+2})$, which is not much better than what we had for (15).

5.1 Relabeling

To address this problem we observe that the role of $\boldsymbol{\pi}^{-1}_{i_1,\ldots,i_k}$ is to map $[n-k+1,n]$ to some item labels, and $\boldsymbol{\pi}_{i'_1,\ldots,i'_k}$ maps some of those labels back to $[n-k+1,n]$. However, when we sandwich this product between two $\mathbf{S}_{n-k}$'s, where exactly we map $[1,n-k]$, and what is mapped to $[1,n-k]$ does not matter. This lets us "relabel" our items so that the permutations only touch $[n-2k+1,n]$. More formally, we can find a pair of permutations μ and μ' that fix $[1,n-2k]$ and have the property that

$$\mathbf{S}_{n-k}\,\boldsymbol{\pi}_{i'_1,\ldots,i'_k}\,\boldsymbol{\pi}^{-1}_{i_1,\ldots,i_k}\,\mathbf{S}_{n-k}=\mathbf{S}_{n-k}\,\boldsymbol{\mu}'\,\boldsymbol{\mu}^{-1}\,\mathbf{S}_{n-k}. \qquad (21)$$

Once again, we find that in YOR the representation matrices of such permutations have a special form. If $T\subset\mathcal{T}^\lambda$, then we say that $[\rho_\lambda(\sigma)]_{T,T}$ is a **block** in $\rho_\lambda(\sigma)$ if $[\rho_\lambda(\sigma)]_{t,t'}=0$ whenever $t\in T$, but $t'\notin T$ or vice versa. By the definition of YOR, if τ_i is the adjacent transposition $(i,i+1)$ and $[\rho_\lambda(\tau_i)]_{t,t'}\neq 0$, then t and t' must differ by at most the position of i and i' in their tableaux. In particular, if $t\in\mathcal{T}^\lambda_m$, then the first row of t starts with boxes numbered $1,\ldots,m$, so if $i>m$ and $[\rho_\lambda(\tau_i)]_{t,t'}\neq 0$, then the first row of t' must also start with $1,\ldots,m$, i.e., $[\rho_\lambda(\tau_i)]_{\mathcal{T}^\lambda_m,\mathcal{T}^\lambda_m}$ is a block. Since any σ that fixes $[1,m]$ can be written as a product of such adjacent transpositions, we have the following result.

Proposition 12 *If $\sigma\in\mathbb{S}_n$ fixes $[1,m]$, then in YOR $[\rho_\lambda(\tau)]_{\mathcal{T}^\lambda_m,\mathcal{T}^\lambda_m}$ is a block in $\rho_\lambda(\tau)$ for any $\lambda\in\Lambda^n_m$.*

Letting $\sigma=\mu'\mu^{-1}$ from (21), Proposition 12 tells us that $[\rho_\lambda(\sigma)]_{\mathcal{T}^\lambda_{n-2k},\mathcal{T}^\lambda_{n-2k}}$ is a block, and clearly $[\widehat{\Omega}(\lambda)]_{\geq[\ldots]_{n-k}}$ needed by (19) is a submatrix of this block. Therefore, to compute $[\widehat{\Omega}(\lambda)]_{\geq[\ldots]_{n-k}}$ we need only construct this block and not the whole matrix $\widehat{\Omega}(\lambda)$. Using the same argument as what lead to (20) and multiplying by the $\binom{2k}{2}$ adjacent transpositions necessary we finally arrive at the following result.

Theorem 13 *Given the exponentiated kernel Fourier matrices $[\hat{\overline{\kappa}}(\lambda)]_{\geq[\ldots]_{n-k}}$, the kernel (13) can be computed in Fourier space in*

$$k(k-1)\sum_{\ell=0}^{k}\binom{2k}{\ell}^2 \ell! + \sum_{\ell=0}^{k}\binom{k}{\ell}^2 \ell! = O((2k)^{2k+3})$$

operations, including computing the $\{[\widehat{\Omega}(\lambda)]_{\geq[\ldots]_{n-k}}\}_{\lambda\in\Lambda^n_{n-k}}$ matrices appearing in (19).

It should be stressed that $(2k)^{2k+3}$ is only an upper bound on the cost of $\{[\widehat{\Omega}(\lambda)]_{\geq[\ldots]_{n-k}}\}_{\lambda\in\Lambda^n_{n-k}}$, and in practice, the size of the $[\rho_\lambda(\sigma)]_{\mathcal{T}^\lambda_{n-2k},\mathcal{T}^\lambda_{n-2k}}$ blocks does not grow as fast as implied by Theorem 13 (see "c_2" in Table 1 for an exact operations count). We have not addressed the complexity of computing $[\hat{\overline{\kappa}}(\lambda)]_{\geq[\ldots]_{n-k}}$, because these are constant matrices that can be pre–computed before any kernel evaluations take place. For top–k rankings the issue here is computing the diffusion kernel, which is significantly accelerated by Proposition 2. For interleaving rankings the $\widehat{\Pi}^n_k(\lambda)$ also have to be computed. In our experience for n around 10 all this takes just a few minutes. For much larger n, however, some additional computational tricks or approximations will have to be employed.

6 Experiments

Fourier transforms on $\mathbb{S}_n$ can be computed with the open source FFT library $\mathbb{S}_n$`ob` (Kondor, 2006). However, $\mathbb{S}_n$`ob` can only manipulate dense Fourier matrices, so to take advantage of the results described in Section 4 it had to be substantially extended.

We tested our kernel K_{adj} on the "sushi" dataset, available at `http://www.kamishima.net`, which is a dataset of 5000 total rankings of $n=10$ sushi dishes ranked by different individuals. We subsampled the data to produce 500 "3+2" multirankings, i.e., multirankings of the form $(x_{i_1}\succ x_{i_2}\succ x_{i_3}, x_{i_4}\succ x_{i_5})$ and used 80% of the dataset for training and tested on the remaining 20% to see how well we can predict whether a given individual will choose $x_{i_4}\succ x_{i_5}$ or $x_{i_5}\succ x_{i_4}$ given that she ranked the sushis $x_{i_1}, x_{i_2}, x_{i_3}$ in the order $x_{i_1}\succ x_{i_2}\succ x_{i_3}$. To solve this conditional prediction problem we used an SVM, as in the introduction, which in this particular case, because our prediction is just binary ($x_{i_4}\succ x_{i_5}$ vs. $x_{i_5}\succ x_{i_4}$), reduces to ordinary two-class classification.

As a baseline we used the same SVM with a kernel based on the correlation between partial rankings as described in (Kamishima & Akaho, 2006). Our experiments showed that the SVM trained with our method is relatively insensitive to the value of β and the regularization parameter C, attaining about a 20% error rate (with 10–fold cross-validation) in a wide range of parameter space. In contrast, the optimal performance of the baseline kernel was 42% error. While admittedly preliminary, these experiments show that the adjacent transpositions based diffusion kernel is promising for some types of applications.

Using the Fourier method and caching kernel values that have already been computed, generating the entire 1000×1000 Gram matrix on a desktop machine took less than five minutes. The bulk of this time was taken up by exponentiating the kernel and computing the "riffled kernels" $\widehat{\overline{\kappa}}(\lambda)$, both of which are pre-computations. Our kernel would have no trouble scaling to larger datasets or much larger n, provided that $\widehat{\overline{\kappa}}(\lambda)$ can be computed ahead of time or approximated in some way.

7 Conclusions

In this paper we argued that kernels methods are a powerful framework for solving a wide variety of learning tasks involving ranking and ordering. To establish such algorithms, we started by defining some canonical kernels on permutations, namely the diffusion kernels induced from transpositions and adjacent transpositions.

The difficulty with such kernels is that in most ranking problems individual training/test examples are not, in fact, total rankings of all n items under consideration, but partial rankings involving only k items, and thus, naively, each kernel evaluation involves summing over many possibilities. The main technical contribution of this paper was to address this computational issue by showing that in Fourier space the kernel can be efficiently computed.

In particular, we showed that the indicator functions of partial rankings are bandlimited, and that this reduces the complexity of kernel evaluations to $O(n^{3k})$. While this result is novel, it is in many ways a natural extension of spectral analysis on permutations developed in e.g. (Diaconis, 1988), and by itself would not be difficult to derive.

Our more surprising result is that by using techniques involving the group algebra, the complexity can be further reduced to $O((2k)^{2k+3})$, which does not involve n at all. This result applies to averaging any right–invariant kernel over partial rankings, not just diffusion kernels. The reason we elected to use diffusion kernels was (a) because we believe they are a canonical class of kernels on permutations, and (b) because by Proposition 2 their Fourier transform is easy to pre–compute.

In practice the scaling behavior is much better than what is suggested by our $(2k)^{2k+3}$ upper bound. Still, computationally our kernels are limited to partial rankings of order up to about $k = 7$. For problems where k is large, such as in merging long lists of results from different search engines, it would be better to employ a method that reduces partial rankings to binary rankings or employs a scoring function.

The strength of our method is that it is based purely on the algebra of permutations and does not employ any reduction heuristic, which might cover up some of the structure in the data. Thus, it is best suited to relatively small problems where a careful analysis of the data is required, such as the evaluation of social surveys or voting schemes.

Our paper concentrated on just the kernel, rather than any specific algorithm that it is to be plugged into. There is much room for research on the algorithms side, and on quantifying the complexity of the function classes induced by our kernels. More generally, we feel that our results on the spectral structure of partial rankings are relevant to not just the kernels approach, but to other ranking methods, as well.

Acknowledgments

R. K. would like to thank Tony Jebara for his contributions to some of the initial work on kernels on permutations. Much of the present work was done while M. B. was visiting R. K. at the Gatsby Computational Neuroscience Unit, University College London. National ICT Australia is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian National Research Council through the ICT Center of Excellence program.

References

Ailon, N., Charikar, M., & Newman, A. (2005). Aggregating inconsistent information: Ranking and clustering. *Proceedings of STOC 2005*.

Ailon, N., & Mohri, M. (2008). An efficient reduction of ranking to classification. *COLT 2008*.

Balcan, M.-F., Bansal, N., Beygelzimer, A., Coppersmith, D., Langford, J., & Sorkin, G. B. (2008). Robust reductions from ranking to classification. *Mach. Learn.*, *72*, 139–153.

Clausen, M. (1989). Fast generalized Fourier transforms. *Theor. Comput. Sci.*, 55–63.

Diaconis, P. (1988). *Group representation in probability and statistics*, vol. 11 of *IMS Lecture Series*. Institute of Mathematical Statistics.

Fukumizu, K., Sriperumbudur, B. K., Gretton, A., & Schölkopf, B. (2009). Characteristic kernels on groups and semigroups. In *NIPS 2008*, 473–480.

Helmbold, D. P., & Warmuth, M. K. (2009). Learning permutations with exponential weights. *Journal of Machine Learning Research*, *10*, 1687–1718.

Huang, J., & Guestrin, C. (2009). Riffled independence for ranked data. In *NIPS 2009*, 799–807.

Huang, J., Guestrin, C., & Guibas, L. (2009). Fourier theoretic probabilistic inference over permutations. *Journal of Machine Learning Research*, *10*, 997–1070.

Kamishima, T., & Akaho, S. (2006). Nantonac collaborative filtering: Recommendation based on multiple order responses. *Proceedings of the international workshop on data mining and statistical science*.

Kondor, R. (2006). $\mathbb{S}_n$`ob`: a C++ library for fast Fourier transforms on the symmetric group. Currently available at `http://www.its.caltech.edu/~risi/Snob/`.

Kondor, R. (2008). *Group theoretical methods in machine learning*. Ph.D. thesis, Columbia University.

Kondor, R., Howard, A., & Jebara, T. (2007). Multi-object tracking with representations of the symmetric group. *AISTATS 2007*.

Kondor, R., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. *ICML 2002*.

Kondor, R., Shervashidze, N., & Borgwardt, K. (2009). The graphlet spectrum. *ICML 2009*.

Lebanon, G., & Lafferty, J. (2002). Cranking: Combining rankings using conditional probability models on permutations. *ICML 2002*.

Maslen, D. K. (1998). The efficient computation of Fourier transforms on the symmetric group. *Mathematics of Computation*, *67*, 1121–1147.